CRYPTOGRAPHY AND NETWORK SECURITY

UNIT I

1.1 OSI Security Architecture

The OSI security architecture is useful to managers as a way of organizing the task of providing security. The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as follows:

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

Security mechanism: A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.

Security service: A processing or communication 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.

Threat

A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit vulnerability.

Attack

An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

(i) Security Attacks

A useful means of classifying security attacks, is in terms passive attacks and active attacks.

A **passive attack** attempts to learn or make use of information from the system but does not affect system resources.

An **active attack** attempts to alter system resources or affect their operation.

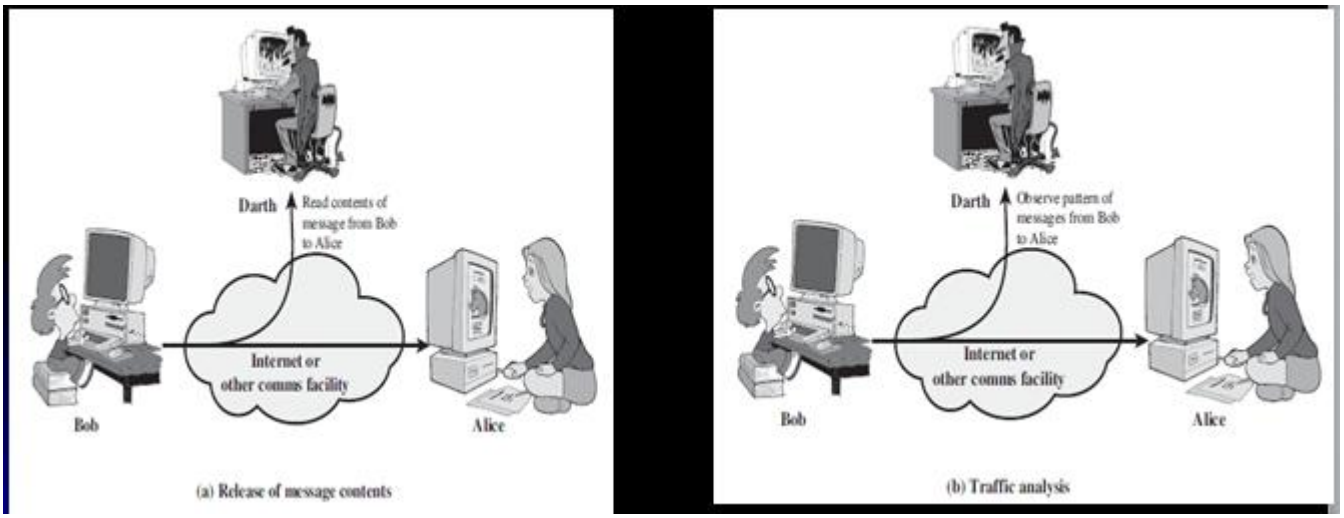
Passive Attacks

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. Two types of passive attacks are

- ✓ **release of message contents**
- ✓ **traffic analysis**

The **release of message contents** is easily understood. A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.

A second type of passive attack, **traffic analysis**, is subtler. Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption Passive attacks are very difficult to detect because they do not involve any alteration of the data.



Active Attacks

Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories:

- ✓ Masquerade
- ✓ Replay
- ✓ modification of messages
- ✓ denial of service.

A **masquerade** takes place when one entity pretends to be a different entity. A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be

captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.

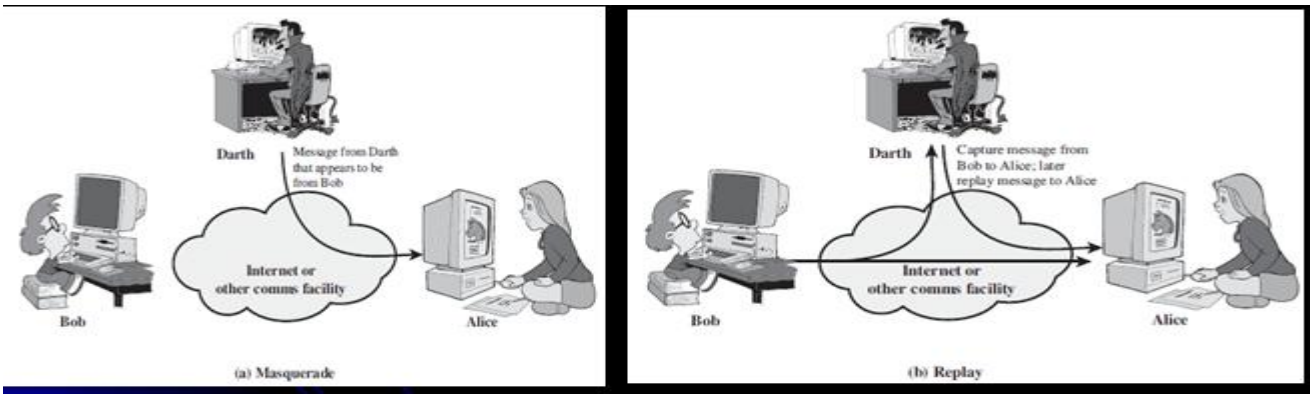
Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.

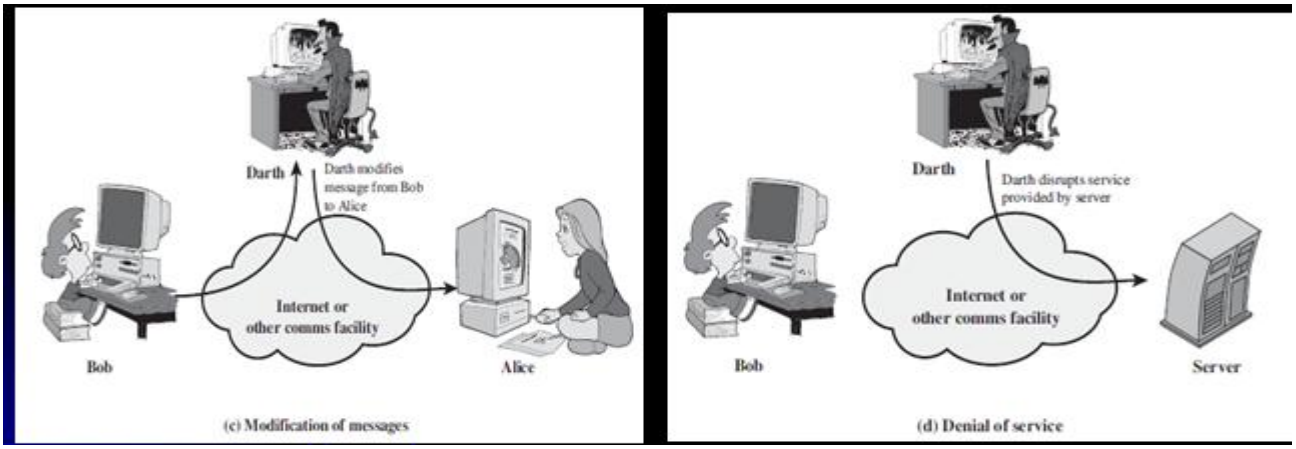
Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect.

The **denial of service** prevents or inhibits the normal use or management of communications facilities. This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service).

Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are

available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely.





(ii) Security Services

AUTHENTICATION

The assurance that the communicating entity is the one that it claims to be.

Peer Entity Authentication
Used in association with a logical connection to provide confidence in the identity of the entities connected.

Data-Origin Authentication
In a connectionless transfer, provides assurance that the source of received data is as claimed.

ACCESS CONTROL

The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).

DATA CONFIDENTIALITY

The protection of data from unauthorized disclosure.

Connection Confidentiality
The protection of all user data on a connection.

Connectionless Confidentiality
The protection of all user data in a single data block

Selective-Field Confidentiality
The confidentiality of selected fields within the user data on a connection or in a single data block.

Traffic-Flow Confidentiality
The protection of the information that might be derived from observation of traffic flows.

DATA INTEGRITY

The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).

Connection Integrity with Recovery
Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.

Connection Integrity without Recovery
As above, but provides only detection without recovery.

Selective-Field Connection Integrity
Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.

Connectionless Integrity
Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.

Selective-Field Connectionless Integrity
Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

NONREPUDIATION

Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.

Nonrepudiation, Origin
Proof that the message was sent by the specified party.

Nonrepudiation, Destination
Proof that the message was received by the specified party.

(iii) Security Mechanisms

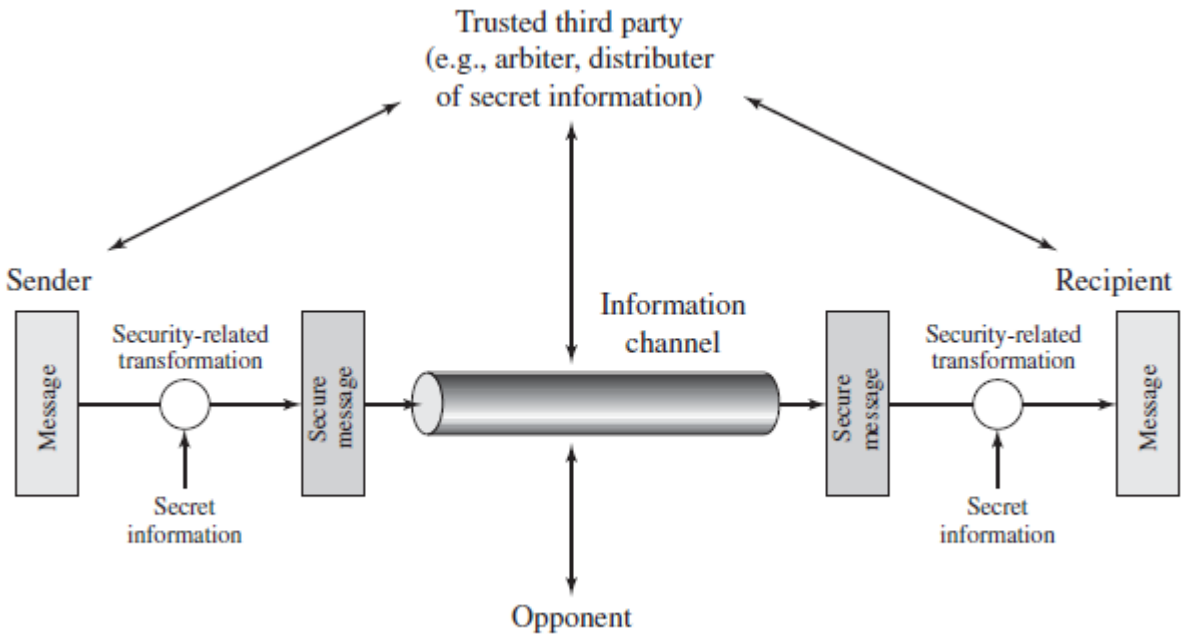
Security Mechanisms (X.800)

SPECIFIC SECURITY MECHANISMS	PERVASIVE SECURITY MECHANISMS
May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.	Mechanisms that are not specific to any particular OSI security service or protocol layer.
Encipherment The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.	**Trusted Functionality** That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).
Digital Signature Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).	**Security Label** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
Access Control A variety of mechanisms that enforce access rights to resources.	**Event Detection** Detection of security-relevant events.
Data Integrity A variety of mechanisms used to assure the integrity of a data unit or stream of data units.	**Security Audit Trail** Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.
Authentication Exchange A mechanism intended to ensure the identity of an entity by means of information exchange.	**Security Recovery** Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.
Traffic Padding The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.	
Routing Control Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.	
Notarization The use of a trusted third party to assure certain properties of a data exchange.	

Relationship Between Security Services and Mechanisms

Service	Mechanism							
	Encipherment	Digital Signature	Access Control	Data Integrity	Authentication Exchange	Traffic Padding	Routing Control	Notarization
Peer Entity Authentication	Y	Y			Y			
Data Origin Authentication	Y	Y						
Access Control			Y					
Confidentiality	Y						Y	
Traffic Flow Confidentiality	Y					Y	Y	
Data Integrity	Y	Y		Y				
Nonrepudiation		Y		Y				Y
Availability				Y	Y			

A Model for Network Security



Model for Network Security

This general model shows that there are four basic tasks in designing a particular security service: Design an algorithm for performing the security-related transformation.

1.2 Classical Encryption techniques

- ✓ Symmetric Cipher Model
  - Cryptography
  - Cryptanalysis and Brute-Force Attack

- ✓ Substitution Techniques
  - Caesar Cipher
  - Monoalphabetic Ciphers
  - Playfair Cipher
  - Hill Cipher
  - Polyalphabetic Ciphers
  - One-Time Pad
- ✓ Transposition Techniques
- ✓ Rotor Machines

An original message is known as the plaintext, while the coded message is called the cipher text. The process of converting from plaintext to cipher text is known as enciphering or encryption; restoring the plaintext from the cipher text is deciphering or decryption. The many schemes used for encryption constitute the area of study known as cryptography. Such a scheme is known as a cryptographic system or a cipher. Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of cryptanalysis. Cryptanalysis is what the layperson calls "breaking the code." The areas of cryptography and cryptanalysis together are called cryptology.

A **symmetric encryption** scheme has five ingredients

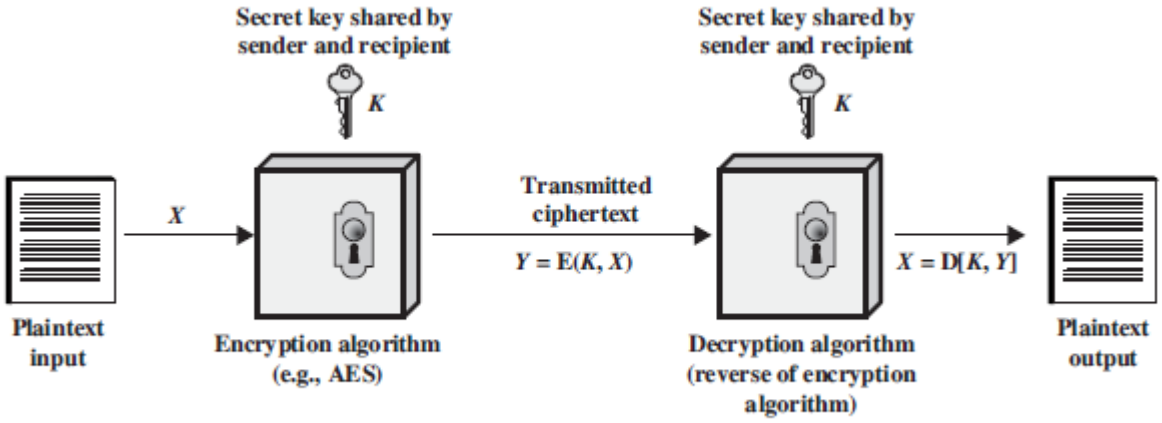
• **Plaintext**: This is the original intelligible message or data that is fed into the algorithm as input.

• **Encryption algorithm**: The encryption algorithm performs various substitutions and transformations on the plaintext.

• **Secret key**: The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.

• **Cipher text:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different cipher texts. The cipher text is an apparently random stream of data and, as it stands, is unintelligible.

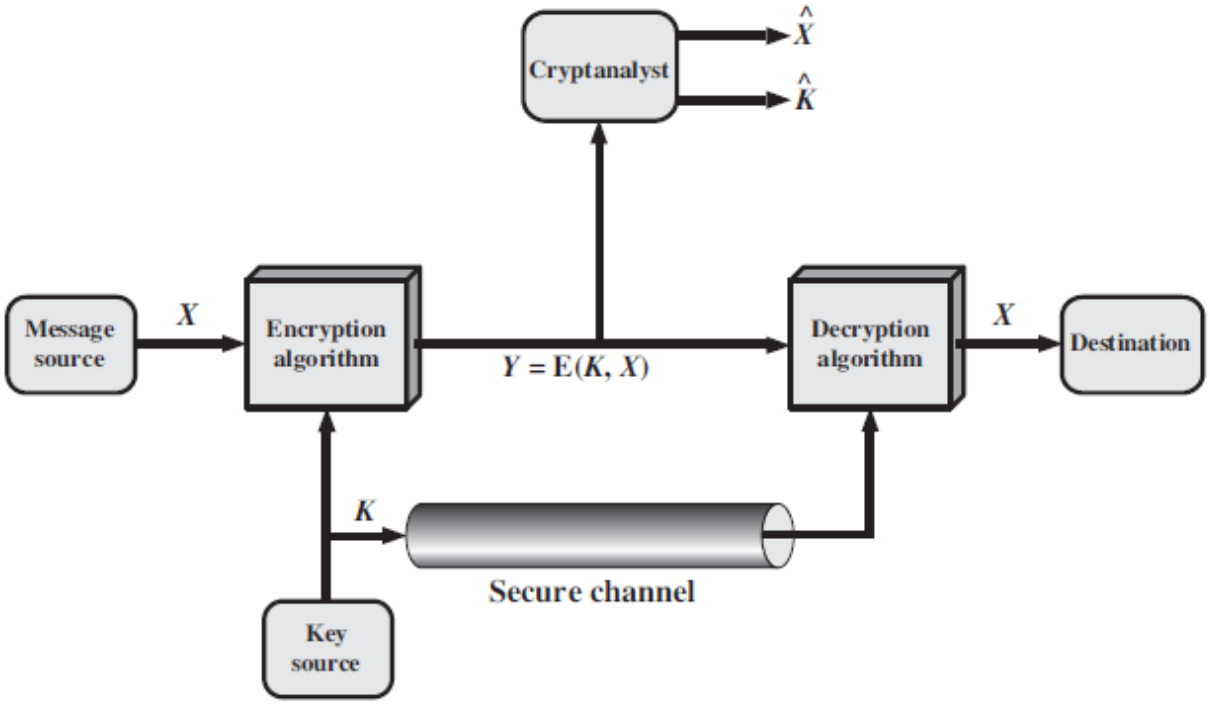
• **Decryption algorithm**: This is essentially the encryption algorithm run in reverse. It takes the cipher text and the secret key and produces the original plaintext.



Simplified Model of Symmetric Encryption

There are two requirements for secure use of conventional encryption:

1. We need a strong encryption algorithm.
2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure.



Model of Symmetric Cryptosystem

Cryptography

Cryptographic systems are characterized along three independent dimensions:

1. The type of operations used for transforming plaintext to cipher text. All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext (bit, letter, group

of bits or letters) is mapped into another element, and transposition, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (that is, that all operations are reversible). Most systems, referred to as product systems, involve multiple stages of substitutions and transpositions.

2. The number of keys used. If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver use different keys, the system is referred to as asymmetric, two-key, or public-key encryption.

3. The way in which the plaintext is processed. A block cipher processes the input one block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along.

Cryptanalysis and Brute-Force Attack

Typically, the objective of attacking an encryption system is to recover the key in use rather than simply to recover the plaintext of a single cipher text. There are two general approaches to attacking a conventional encryption scheme:

• Cryptanalysis: Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext–cipher text pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

• Brute-force attack: The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

1.2.1. Substitution Techniques

(i) Caesar Cipher

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

plain: meet me after the toga party

cipher: PHHW PH DIWHU WKH WRJD SDUWB

Note that the alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows:

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

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

Then the algorithm can be expressed as follows. For each plaintext letter , substitute the ciphertext letter :2 A shift may be of any amount, so that the general Caesar algorithm is

where takes on a value in the range 1 to 25.The decryption algorithm is simply

p = D(k, C) = (C - k) mod 26

(ii)Mono alphabetic Ciphers

The Mono alphabetic Cipher (often referred to as a cryptogram) uses a KEY which is the rearrangement of the letters of the alphabet. These different letters are then substituted for the letters in the message to create a secret message. That KEY is needed to decipher the secret message. Start by creating a key that maps each letter of the alphabet to a (possibly the same) letter of the alphabet. A sample key might be:

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

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

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

Mono alphabetic encryption is very easy to break, for two main reasons. First, commonly used letters like 'e' show up very quickly as the 'x' in the example. Second, words with repeated letters like "meet" in the example show that repetition in the cipher text. And indeed this is so weak that the daily cryptogram run by some newspapers is typically an mono alphabetic substitution.

(iii)Playfair Cipher

The best-known multiple-letter encryption cipher is the Playfair, which treats diagrams in the plaintext as single units and translates these units into cipher text diagrams. The Playfair algorithm is based on the use of a 5 × 5 matrix of letters constructed using a keyword.

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

In this case, the keyword is monarchy. 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 alphabetic order. The letters I and J count as one letter. Plaintext is encrypted two letters at a time, according to the following rules:

1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.
2. Two 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 circularly following the last. For example, ar is encrypted as RM.
3. Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, mu is encrypted as CM.
4. Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

(iv)Hill Cipher

In general terms, the Hill system can be expressed as

$$\mathbf{C} = \mathrm{E}(\mathbf{K}, \mathbf{P}) = \mathbf{PK} \bmod 26$$
$$\mathbf{P} = \mathrm{D}(\mathbf{K}, \mathbf{C}) = \mathbf{CK}^{-1} \bmod 26 = \mathbf{PKK}^{-1} = \mathbf{P}$$

Consider this example. Suppose that the plaintext “hill cipher” is encrypted using a Hill cipher to yield the cipher text HCRZSSXNSP. Thus, we know that ; ; and so on. Using the first two plaintext–cipher text pairs, we have

$$\begin{pmatrix} 7 & 2 \\ 17 & 25 \end{pmatrix} = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \mathbf{K} \bmod 26$$

The inverse of $\mathbf{X}$ can be computed:

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}^{-1} = \begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix}$$

so

$$\mathbf{K} = \begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix} \begin{pmatrix} 7 & 2 \\ 17 & 25 \end{pmatrix} = \begin{pmatrix} 549 & 600 \\ 398 & 577 \end{pmatrix} \bmod 26 = \begin{pmatrix} 3 & 2 \\ 8 & 5 \end{pmatrix}$$

This result is verified by testing the remaining plaintext–ciphertext pairs.

(v)Polyalphabetic Ciphers

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

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

A general equation of the encryption process is

$$C_i = (p_i + k_{i \bmod m}) \bmod 26$$

Compare this with Equation (2.1) for the Caesar cipher. In essence, each plaintext character is encrypted with a different Caesar cipher, depending on the corresponding key character. Similarly, decryption is a generalization of Equation

$$p_i = (C_i - k_{i \bmod m}) \bmod 26$$

```
key:            deceptivedeceptivedeceptive
plaintext:      wearediscoveredsaveyourself
ciphertext:     ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

(v)One-Time Pad

The term "one-time pad" refers to any method of encryption where each byte of the plaintext is enciphered using one byte of the key stream, and each key byte is used one time then never used again. The key stream for a one-time pad must be a true-random stream, meaning that every key byte can take any of the values 0 to 255 with equal likelihood, and independently of the values of all other key bytes. By contrast, in a pseudo-random key stream the value of each byte after the first several is mathematically derived from the values of a few preceding bytes.

The practical difficulty of using a one-time pad is that the key bytes cannot be reused. This means that even for a two-way exchange of messages, each party must have a sufficient supply of key

material on hand so that they cannot run out before more key stream can be furnished. For N-way exchanges, the amount of key required increases quadratically.

1.2.2. Transposition techniques

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. The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. For example, to encipher the message "meet me after the toga party" with a rail fence of depth 2, we write the following:

```
m e m a t r h t g p r y
  e t e f e t e o a a t
```

This sort of thing would be trivial to cryptanalyze. 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 the columns then becomes the key to the algorithm. For example,

Key:

4 3 1 2 5 6 7

Plaintext:

a t t a c k p
o s t p o n e
d u n t i l t
w o a m x y z

Cipher text: TTNAAPTMTSUOAODWCOIXKNLYPETZ

The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed. Thus, if the foregoing message is re-encrypted using the same algorithm,

Key:

4 3 1 2 5 6 7

Input:

t t n a a p t
m t s u o a o
d w c o i x k

n l y p e t z

Output: NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

1.2.3. Rotor Machines

The example just given suggests that multiple stages of encryption can produce an algorithm that is significantly more difficult to cryptanalyze. This is as true of substitution ciphers as it is of transposition ciphers. Before the introduction of DES, the most important application of the principle of multiple stages of encryption was a class of systems known as rotor machines

1.2.4 LFSR Sequences

A linear feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state. The only linear function of single bits is xor, thus it is a shift register whose input bit is driven by the exclusive-or (xor) of some bits of the overall shift register value. The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle.

Applications of LFSRs include generating pseudo-random numbers, pseudo-noise sequences, fast digital counters, and whitening sequences. Both hardware and software implementations of LFSRs are common. A register of L delay (storage) elements each capable of storing one element from Fq , and a clock signal. Clocking, the register of delay elements is shifted one step and the new value of the last delay element is calculated as a linear function of the content of the register

Fibonacci LFSRs

The bit positions that affect the next state are called the taps. In the diagram the taps are [16,14,13,11]. The rightmost bit of the LFSR is called the output bit. The taps are XOR'd sequentially with the output bit and then fed back into the leftmost bit. The sequence of bits in the rightmost position is called the output stream.

- The bits in the LFSR state which influence the input are called *taps* (white in the diagram).

- A maximum-length LFSR produces an m-sequence (i.e. it cycles through all possible $2^n - 1$ states within the shift register except the state where all bits are zero), unless it contains all zeros, in which case it will never change.
- As an alternative to the XOR based feedback in an LFSR, one can also use XNOR.[1] This function is not linear, but it results in an equivalent polynomial counter whose state of this counter is the complement of the state of an LFSR. A state with all ones is illegal when using an XNOR feedback, in the same way as a state with all zeroes is illegal when using XOR. This state is considered illegal because the counter would remain "locked-up" in this state.
- The sequence of numbers generated by an LFSR or its XNOR counterpart can be considered a binary numeral system just as valid as Gray code or the natural binary code.

Properties of LFSR sequences

A sequence s=. . . , s0, s1, . . .is called periodic if there is a positive integer T such that si = si+T, for all i≥0. The period is the least such positive integer T for which si =si+T, for all i≥0.

The LFSR state runs through different values. The initial state will appear again after visiting a number of states. If deg C(D) =L, the period of a sequence is the same as the number of different states visited, before returning to the initial state. C(D) irreducible: the state corresponds to an element in FqL, say β.The sequence of different states that we are entering is then β, αβ, α2β, . . . , αT−1β, αTβ=β,where T is the order or α. If α is a primitive element (its order is qL −1), then obviously we will go through all qL−1 different states and the sequence will have period qL−1. Such sequences are called m-sequences and they appear if and only if the polynomial π(x) is a primitive polynomial.

Applications

LFSRs can be implemented in hardware, and this makes them useful in applications that require very fast generation of a pseudo-random sequence, such as direct-sequence spread spectrum radio. LFSRs have also been used for generating an approximation of white noise in various programmable sound generators.

(i) Uses as counters

The repeating sequence of states of an LFSR allows it to be used as a clock divider, or as a counter when a non-binary sequence is acceptable as is often the case where computer index or framing locations need to be machine-readable.[4] LFSR counters have simpler feedback logic than natural binary counters or Gray code counters, and therefore can operate at higher clock rates. However it is

necessary to ensure that the LFSR never enters an all-zeros state, for example by presetting it at start-up to any other state in the sequence. The table of primitive polynomials shows how LFSRs can be arranged in Fibonacci or Galois form to give maximal periods. One can obtain any other period by adding to an LFSR that has a longer period some logic that shortens the sequence by skipping some states.

(ii) Uses in cryptography

LFSRs have long been used as pseudo-random number generators for use in stream ciphers (especially in military cryptography), due to the ease of construction from simple electromechanical or electronic circuits, long periods, and very uniformly distributed output streams. However, an LFSR is a linear system, leading to fairly easy cryptanalysis. For example, given a stretch of known plaintext and corresponding ciphertext, an attacker can intercept and recover a stretch of LFSR output stream used in the system described, and from that stretch of the output stream can construct an LFSR of minimal size that simulates the intended receiver by using the Berlekamp-Massey algorithm. This LFSR can then be fed the intercepted stretch of output stream to recover the remaining plaintext.

Three general methods are employed to reduce this problem in LFSR-based stream ciphers:

- Non-linear combination of several bits from the LFSR state;
- Non-linear combination of the output bits of two or more LFSRs (see also: shrinking generator); or
- Irregular clocking of the LFSR, as in the alternating step generator.

Important LFSR-based stream ciphers include A5/1 and A5/2, used in GSM cell phones, E0, used in Bluetooth, and the shrinking generator. The A5/2 cipher has been broken and both A5/1 and E0 have serious weaknesses.

The linear feedback shift register has a strong relationship to linear congruential generators.

(iii) Uses in circuit testing

LFSRs are used in circuit testing, for test pattern generation(for exhaustive testing, pseudo random testing or pseudo exhaustive testing) and signature analysis.

(iv) Test Pattern Generation

Complete LFSRs are commonly used as a pattern generator for exhaustive testing, since they cover all possible inputs for an n input circuit. Maximum length LFSRs and weighted LFSRs are widely used as a pseudo-random test pattern generators for pseudo-random test applications.

1.3 Basic Number Theory

1.3.1 Divisibility

Number theory is concerned with the properties of the integers. One of the most important is divisibility.

Definition: Let a and b be integers with a ^ 0. We say that a divides b, if there is an integer k such that b = ak. This is denoted by a|b. Another way to express this is that b is a multiple of a.

Examples: 3[15, —15|60, 7 f 18 (does not divide).

The following properties of divisibility are useful.

Proposition: Let a ,b ,c represent integers.

1. For every a ^ 0, a|0 and a |a. Also, 1|6 for every b.

2. I f a|b and b|c, then a|c.

3. I f a|b and a|c, then a |(sb+tc) for all integers s and t;

Proof. Since 0 = a . 0, we may take k = 0 in the definition to obtain c|0. Since a = a . 1, we take k = 1 to prove a|a. Since b = b . 1, we have 1|b. This proves (1). In (2), there exist k and l such that b = ak and c = bl. Therefore, c = (kl)a, so a|c. For (3), write $b = ak_1$ and $c = ak_2$. Then $sb + tc = a(sk_1 + tk_2)$, so a|sb + tc.

1.3.2 Prime Numbers

A number $p > 1$ that is divisible only by 1 and itself is called a prime number. The first few primes are 2, 3, 5, 7, 11, 13, 17, ... An integer $n > 1$ that is not prime is called composite , which means that n must expressible as a product ab of integers with $1 < a, b < n$. A fact, known already to Euclid, is that there are infinitely many prime numbers.

Prime Number Theorem: Let π(x) be the member of primes less than x. Then

$$\pi(x) \approx \frac{x}{\ln x},$$

in the sense that the ratio π(x)=x/In x —> **1** as x —►∞.

Theorem: *Every positive integer is a product of primes. This factorization into primes is unique, up to reordering the factors.*

Proof. There is a small technicality that must be dealt with before we begin. When dealing with products, it is convenient to make the convention that an empty product equals 1. This is similar to the

convention that $x^\circ = 1$. Therefore, the positive integer 1 is a product of primes, namely the empty product. Also, each prime is regarded as a one factor product of primes.

Lemma. I f p is a prime and p divides a product of integers ab, then either p|a or p|b. More generally, if a prime p divides a product a,b,... z, then p must divide one of the factors a ,b ,. .. ,*z.* For example, when $p = 2$, this says that if a product of two integers is even then one of the two integers must be even. The proof of the lemma will be given at the end of this section, after we discuss the Euclidean algorithm. Continuing with the proof of the theorem, suppose th a t an integer *n* can be written as a product of primes in two different ways; *n = pa1Y 21 ---Ps' = < 7 ^* where p i , . . . , ps and *qi*, . , . , *qt* are primes, and the exponents a,- and *bj* are nonzero. If a prime occurs in both factorizations, divide both sides by it to obtain a shorter relation. Continuing in this way, vve may assume that none of the primes *p i*, . . . ,*p £* occur among the *qj's.*

1.3.3 Greatest Common Divisor

The greatest common divisor of *a a*nd *b* is the largest positive integer dividing both *a* and *b* and is denoted by either *gcd(a, b)* or by *(a, b).* In this book, we shall use the first notation.

Examples. gcd(6, 4) = 2, gcd(5, 7) = 1, gcd(24, GO) = 12.

Example. Compute gcd(482, 1180).

Solution: Divide 482 into 1180. The quotient is 2 and the remainder is 216. Now divide the remainder 216 into 482. The quotient is 2 and the remainder is 50. Divide the remainder 50 into the previous remainder 216. The quotient is 4 and the remainder is 16. Continue this process of dividing the most recent remainder into the previous one. The last nonzero remainder is the gcd, which is 2 in this case:

1180 = 2 . 482 + 216

482 = 2 .216 + 50

216 = 4 . 50 + 16

50 = 3. 16 + 2

16 = 8 .2 + 0.

Notice how the numbers are shifted:

remainder —» divisor —> dividend —> ignore.

Here is another example:

12345 = 1.1111+ 1234

11111 = 9.1234 + 5

1234 = 246 . 5 + 4

5 = 1 . 4 + 1

4 = 4. 1 + 0.

Therefore, gcd(12345,11111) = 1.

Using these examples as guidelines, we can now give a more formal description of the Euc idean algorith m . Suppose that a is greater than b. If not, switch a and 6. The first step is to divide a by b, hence represent a in the form a = q1b + r 1. If r! = 0, then b divides a and the greatest common divisor is b. If r 1 != 0, then continue by representing b in the form b = qar1 + r2 . Continue in this way until the remainder th a t is zero, giving the following sequence of steps:

$a = q_1 b + r\ 1$

$b = q_2 r_1 + r\ 2$

$r_1 = q_3 r_2 + r\ 3$

……

$r_{k-2} = q_k r_{k-1} + r_k$

$r_{k-1} = q_{k+1} r_k$.

The conclusion is that

gcd(a, b) = r k.

There are two important aspects to this algorithm:

1. It does not require factorization of the numbers.
2. It is fast.

1.3.4 Solving ax + by = d

Given integers a and b, there are integers x and y such that ax + by = gcd(a, b). Suppose we start by dividing a into b, so $b = q_1 a + r_1$, and then proceed as in the Euclidean algorithm. Let the successive quotients be q1, qs ,... qn, we have q1 = 2, q2 = 2, q3 = 4, q4 = 3, qs = 8. Form the following sequences:

$x0 = 0, x1 = 1, xj = - q_{j-1} x_{j-1} + x_{j-2}$,

$yo = 1, y1 = o, yj = - q_{j-1} y_{j-1} + y_{j-2}$.

Then

$ax_n + by_n = gcd(a,b)$.

In the first example, we have the following calculation:

x0 = 0, x 1 = 1

x2 = -2x1 + xo = -2

x3 = - 2 x 2 + x1 = 5

x4 = -4x3 + x2 = -22

x5 = -3x4 + x3 = 71.

Similarly, we calculate y5 = —29. An easy calcu*l*ation shows that

482 . 71 + 1180 . (-2 9) = 2 = gcd(482,1180).

1.3.5 Congruences

One of the most basic and useful notions in number theory is modular arithmetic, or congruences.

Definition . *Let a ,b ,n be integers with n ≠O . We say that* a ≡ 6 (mod n)

(read: a is congruent *to b mod n) if a -b is a multiple (positive or negative)of n .*

Another formulation is that a ≡ *b* (mod *n)* if *a* and *b* differ by a multiple of *n.* This can be rewritten as *a* = *b* + nk for some integer *k* (positive or negative).

Examples.

32 = 7 (mod 5), -1 2 = 37 (mod 7), 1 7 = 1 7 (mod 13).

Proposition. *Let* a, 6, c, *n be integers with n £* 0.

1. a ≡ 0 (mod *n) if and only if n|a.*

2 . a ≡ *a* (mod *n).*

3. a ≡ *b* (mod n) *if and only i f b* ≡ *a* (mod *n).*

4. If a ≡ *b and b* ≡ *c* (mod n), *then a* ≡ c (mod n).

Proposition . *Let a ,b ,c ,d ,n be integers with n* 0, *and suppose a s b* (mod n) *and c = d* (mod *n). Then a + c* ≡ *b + d, a - c* ≡ *b - d, ac* ≡ *bd* (mod n).

Proof. Write *a = b + n k* and *c = d + nl,* for integers *k* and *i.* Then a + c = *b + d + n (k + l), so a + c = b + d* (mod n). The proof that a - c ≡ *b - d* is similar. For multiplication, we have *ac = bd + n(dk + bl + nkl),* so *ac = bd.*

Example . Here is an example of how we can do algebra m od *n.* Consider the following problem:

Solve **2** + 7 = 3 (mod 17),

Solution: r = 3 - 7 = - 4 = 1 3 (mod 17).

1.3.6 Division

Division is much trickier mod *n* than it is w ith rational numbers. The general rule is th at you can divide by *a* (mod n) when gcd(a, n) = 1.

Proposition . *Let a ,b ,c ,n be integers with, n≠0 and with* gcd(a,n) = 1. *If ab* ≡ ac (mod n), *then b* ≡*c* (mod n). *In other words, i f* a *and n are relatively prime, we can divide both sides of the congruence by a.*

Proof. Since gcd(a, n) = 1, there exist integers *x, y* such th at *ax* + *n y* = 1. Multiply by b ^ c to obtain *(ab — ac)x* + *n(b — c)y* = *b* — c. Since *ab — ac* is a multiple of n, by assumption, and *n(b — c)y* is also a multiple of n, we find th at 6 — c is a multiple of *n.* This means that *b* = c
(mod n).

Example. Solve: 2x + 7 = 3 (mod 17).

Solution: 2x = 3 — 7 = —4, so *x* = — 2 = 15 (mod 17). The division By 2 is allowed since gcd(2,17) = 1.

1.3.7 The Chinese Remainder Theorem

In many situations, it is useful to break a congruence mod *n* into a system of congruences mod factors of *n.* Consider the following example. Suppose we know that a number *x* satisfies *x* = 25 (mod 42). This means that we can write *x* = 25 + 42*k* for some integer *k.* Rewriting 42 as 7 . 6 , we obtain *x* — 25 + 7(6*k),* which implies that *x* ≡ 25 ≡ 4 (mod 7). Similarly, since *x* = 25 + 6(7*k),* we have *x* ≡ 25 ≡ 1 (mod 6). Therefore,

$$x \equiv 25 \pmod{42} \Rightarrow \begin{cases} x \equiv 4 \pmod{7} \\ x \equiv 1 \pmod{6}. \end{cases}$$

The Chinese remainder theorem shows that this process can be reversed; namely, a system of congruences can be replaced by a single congruence under certain conditions.

Chinese Remainder Theorem. *Suppose* gcd(m, n) = 1. *Given integers a and b, there exists exactly one solution x* (mod *m n) to the simultaneous congruences*

$x \equiv a \pmod{m}$, $x \equiv b \pmod{n}$.

Proof*.* There exist integers s, *t* such that *m s*+ *n t* = 1. Then *m s* ≡ 1 (mod *n)* and *nt* ≡ 1 (mod m). Let *x* = *bms* + *ant.* Then *x* ≡ *a n t* ≡ *a* (mod m), and *x* ≡ *bms* ≡ *b* (mod n), so a solution *x* exists. Suppose X1 is another solution. Then *x* ≡ *X1* (mod *m)* and *x* ≡ *x1* (mod n), so *x* —*x1* is a multiple of both *m* and *n.*

Lemma. *Let m ,n be integers with* gcd(m, n) = 1. *I f an integer c is a multiple of both m and n, then c is a multiple of mn.*

Proof. Let c = *m k* = *n l.* W rite *m s* + *n t* = 1 with integers s, *t.* Multiply by c to obtain *c = cms* + *cnt = mnls* + *mnkt = mn(ls + kt).*

Example: Solve x = 3 (mod 7), x = 5 (mod 15).

Solution: x = 80 (mod 105) (note: 105 = 7 . 15). Since 80 s 3 (mod 7) and 80 = 5 (mod 15), 80 is a solution. The theorem guarantees that such a solution exists, and says that it is uniquely determined mod the product *m n,* which is 105 in the present example.

Example: Solve *x* = 7 (mod 12345), *x* = 3 (mod 11111).

Solution: First, we know from our calculations that the inverse of 11111 (mod 12345) is *i* = 2471. Therefore, *k* = 2471(7 - 3) = 9884 (mod 12345). This yields *x* = 3 + 11111 - 9884 = 109821127 (mod (11111 . 12345)).

Chinese Remainder Theorem (General Form). *Let be integers with* gcd(m_i , m_j) = 1 *whenever i ^ j . Given integers* a_1, . . . , a_k, *there exists exactly one solution x* (mod m_1, ... m_k) *to the simultaneous congruences*

$x \equiv a_1 \pmod{m_1}$, $x \equiv a_2 \pmod{m_2}$, $x \equiv a_k \pmod{m_k}$.

For example, the theorem guarantees there is a solution to the simultaneous congruences

$x \equiv I \pmod{11}$, $x \equiv -1 \pmod{13}$, $x \equiv 1 \pmod{17}$.

In fact, $x \equiv 1871 \pmod{11 \bullet 13 \bullet 17}$ is the answer.

1.3.8 Mo*dul*ar Exponentiation

Suppose we want to compute 2123,1 (mod 789). If we first compute 2123,1, then reduce mod 789, we'll be working with very large numbers, even though the final answer has only 3 digits. We should therefore perform each multiplication and then calculate the remainder. Calculating the consecutive powers of 2 would require that we perform the modular multiplication 1233 times. This is method is too slow to be practical, especially when the exponent becomes very large. A more efficient way is the following (all congruences will be mod 789). We start with 22 = 4 (mod 789) and repeatedly square both sides to obtain the following congruences:

$2^4 \equiv 4^2 \equiv 16$

$2^8 \equiv 16^2 \equiv 256$

$2^{16} \equiv 256^2 \equiv 49$

$2^{32} \equiv 34$

$2^{64} \equiv 367$

$2^{128} \equiv 559$

$2^{256} \equiv 37$

$2^{512} \equiv 580$

$2^{1024} \equiv 286.$

Since 1234 = 1024 + 128 + 64 + 16 + 2 (this just means that 1234 equals 10011010010 in binary), we have 2123'1 = 286 . 559 . 367 . 49 . 4 = 481 (mod 789). Note that we never needed to work with a number larger than 7882.

1.3.9 Fermat's Little Theorem and Euler's Theorem

Two of the most basic results in number theory are Fermat's and Euler's theorems. Originally admired for their theoretical value, they have more recently proved to have important cryptographic applications and will be used repeatedly throughout this book.

Fermat 's Little Theorem . *If p is a prime and p does not divide a, then*

$a^{p-1} \equiv 1 \pmod{p}$.

Proof.

Let S= {*1*,2,3..... .. p-1}.

Consider the map Ψ : S —> *S* defined by $\Psi(x) = ax \pmod{p}$. For example, when $p — 7$ and a = 2, the map Ψ takes a number *x,* multiplies it by 2, then reduces the result mod 7. We need to check that if x € S, then $\Psi[x)$ is actually in S; that is, $\Psi(x)$!= 0. Suppose $\Psi(x) = 0$. Then $ax \equiv 0 \pmod{p}$. Since gcd(a,p) = 1, we can divide this congruence by *a* to obtain $x \equiv 0 \pmod{p}$, so x € *S.* This contradiction means that $\Psi(x)$ cannot be 0, hence $\Psi(x)$ € *S.* Now suppose there are x, y € *S* with $\Psi(x) = \Psi\{y)$. This means $ax \equiv ay \pmod{p}$. Since gcd(a,p) = 1, we can divide this congruence by *a* to obtain $x \equiv y \pmod{p}$.

1.3.10 Legendre and Jacobi Symbols

Suppose we want to determine whether or not $x^2 \equiv a \pmod{p)}$ has a solution, where p is prime. If p is small, we could square all of the numbers mod p and see if *a* is on the list. When p is large, this is impractical. If $p \equiv 3 \pmod{4}$, we can use the technique of the previous section and com pute $s \equiv a^{(p+1)/4} \pmod{p}$. If *a* has a square root, then *s* is one of them, so we simply have to square a and see if we get *a.* If not, then *a* has no square root mod p. The following proposition gives a method for deciding whether a is a square mod *p* that works for arbitrary odd p.

Proposition . *Let p be an odd prime and let a be an integer with a* ≠0 (mod p). *Then* $a^{(p-1)/2} \equiv \pm 1 \pmod p$. *The congruence* $x^2 \equiv a \pmod p$ *has a solution if and only* = 1 (mod p).

Proof. Let $y = a^{(p-1)/2} \pmod p$. Then $y^2 = a^{p-1} \equiv 1 \pmod p$, by F ermat's theorem.

Therefore $y \equiv \pm 1 \pmod p$.

If $a \equiv x^2$, then $a^{(p-1/2)} = x^{p-1} \equiv 1 \pmod p$. The hard part is showing the converse. Let *g* be a primitive root mod p. Then $a = g^j$ for some j , If $a^{(p-1/2)} \equiv 1 \pmod p$, then

$g^{j(p-1)/2} = a^{(p-1)/2} = 1 \pmod p$.

Proposition. *Let p be an odd prime.*

1. *If* $a \equiv b \not\equiv 0 \pmod p$, *then*

$$\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right).$$

2. *If* $a \not\equiv 0 \pmod p$, *then*

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod p.$$

3. *If* $ab \not\equiv 0 \pmod p$, *then*

$$\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right).$$

4.

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}.$$

1.3.11 Finite Fields

The basic examples of Gelds are the real numbers, the complex numbers, the rational numbers, and the integers mod a prime. The set of all integers is not a field since we sometimes cannot divide and obtain an answer in the set (for example, 4 /3 is not an integer 0.

Example. Here is a field with four elements. Consider the set

G F(4) = { 0 , l, ω, ω^2 }, with the following laws:

1. 0 + *x* = *x* for all *x.*

2. *x + x* = 0 for all *x.*

3. *I . x = x lor* all x.

4. $\omega + 1 = \omega^2$.

5. Addition and multiplication are commutative and associative, and the distributive law $x(y + z) = xy + xz$ holds for all x, y, z.

Since

$$\omega^3 = \omega \cdot \omega^2 = \omega \cdot (1 + \omega) = \omega + \omega^2 = \omega + (1 + \omega) = 1.$$

In general, a field is a set containing elements 0 and 1 (with 1 ^ 0) and satisfying the following:

1. It has a multiplication and addition satisfying (1), (3), (5) in the preceding list.

2. Every element has an additive inverse (for each *x,* this means there exists an element — *x* such that *x* + (—*x)* = 0).

3. Every nonzero element has a multiplicative inverse.

A field is closed under subtraction. To compute *x* - y, simply compute x+(-y)