BCS3752 Network Security Lab

Tools/Software used: C/C++/Java

1. Write program for Mono alphabetic cipher.
2. Implementation of Play Fair cipher.
3. Implementation of Vigenere cipher (Polyalphabetic substitution).
4. Implementation of Hill cipher.
5. Implementation of Gauss cipher.
6. Implementation of Rail Fence cipher.
7. Implementation of S-DES algorithm for data encryption.
8. Implement RSA asymmetric (public key and private key)-Encryption. Encryption key (e, n) & (d,n)
9. Generate digital signature using Hash code.
10. Generate digital signature using MAC code.

BCS3702 Network Security and Cryptography

Course Objective:

1. Have a fundamental understanding of the objectives of cryptography and network security
2. Getting familiar with the cryptographic techniques that provide information and network security
3. To know the different types of algorithms of exchanging information in a secret way.
4. To know the possible threats which can breach the secure communication

Learning Outcome:

At the end of the course, the student should be able to:

1. Understanding cryptography and network security concepts and applications
2. Apply security principals to system design and Real time Scenarios.
3. Identify and investigate network security threats
4. Analysis of network traffic and security threats

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction to Cryptography and Symmetric Ciphers** Security Attacks: Security Services and mechanism; Classical encryption techniques: Substitution ciphers and Transposition ciphers, Steganography, Cryptanalysis; Modern Block Ciphers: Stream and Block Cipher, Block Cipher Principles, Block Cipher Modes of Operations; Shannon's theory of Confusion and Diffusion; Fiestal structure; Data encryption standard(DES); Strength of DES; Idea of differential cryptanalysis; Triple DES; Symmetric Key Distribution; Finite Fields: Introduction to groups, rings and fields, Modular Arithmetic, Euclidean Algorithm, Finite Fields of the form GF(p).	30 Hours	1
II	**Basics of Number Theory and Public key Cryptography** Introduction to Number Theory: Prime and Relative Prime Numbers, Fermat's and Euler's theorem, Testing for Primality, Chinese Remainder theorem, Discrete Logarithms; Public Key Cryptography: Principles of Public-Key Cryptography, RSA Algorithm, Security of RSA; Key Management: Deffie-Hellman Key Exchange.	30 Hours	1
III	**Hash Functions and Digital Signatures** Message Authentication; Hash Functions; Secure Hash Functions; Security of Hash functions and MACs; Digital Signatures; Digital Signature Standards (DSS); Proof of digital signature algorithm; Advanced Encryption Standard (AES) encryption and decryption.	30 Hours	1
IV	**Network and System Security** Authentication Applications: Kerberos, X.509 Certificates; Electronic Mail Security: Pretty Good Privacy, S/MIME; IP Security: IP Security Architecture, Authentication Header, Encapsulating security payloads, Combining Security Associations; Web Security: Secure Socket Layer and Transport Layer Security, Secure Electronic transaction; Intruder; Viruses; Firewalls.	30 Hours	1

Text/Reference Books:

1. William Stallings, “Cryptography and Network Security: Principals and Practice”, Pearson Education.

2. Behrouz A. Frouzan: “Cryptography and Network Security”, Tata McGraw-Hill
3. Bruce Schiener, “Applied Cryptography”. John Wiley & Sons
4. Bernard Menezes, “Network Security and Cryptography”, Cengage Learning.
5. Atul Kahate, “Cryptography and Network Security”, Tata McGraw-Hill

UNIT - I

INTRODUCTION

Computer data often travels from one computer to another, leaving the safety of its protected physical surroundings. Once the data is out of hand, people with bad intention could modify or forge your data, either for amusement or for their own benefit.

Cryptography can reformat and transform our data, making it safer on its trip between computers. The technology is based on the essentials of secret codes, augmented by modern mathematics that protects our data in powerful ways.

- **Computer Security** - generic name for the collection of tools designed to protect data and to thwart hackers
- **Network Security** - measures to protect data during their transmission
- **Internet Security** - measures to protect data during their transmission over a collection of interconnected networks

Security Attacks, Services and Mechanisms

To assess the security needs of an organization effectively, the manager responsible for security needs some systematic way of defining the requirements for security and characterization of approaches to satisfy those requirements. One approach is to consider three aspects of information security:

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.

Basic Concepts

Cryptography The art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form

Plaintext The original intelligible message

Cipher text The transformed message

Cipher An algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods

Key Some critical information used by the cipher, known only to the sender& receiver

Encipher (encode) The process of converting plaintext to cipher text using a cipher and a key

Decipher (decode) the process of converting cipher text back into plaintext using a cipher and a key

Cryptanalysis The study of principles and methods of transforming an unintelligible message back into an intelligible message *without* knowledge of the key. Also called **code breaking**

Cryptology Both cryptography and cryptanalysis

Code An algorithm for transforming an intelligible message into an unintelligible one using a code-book

Cryptography

Cryptographic systems are generally classified along 3 independent dimensions:

Type of operations used for transforming plain text to cipher text

All the encryption algorithms are based on two general principles: **substitution**, in which each element in the plaintext is mapped into another element, and **transposition**, in which elements in the plaintext are rearranged.

The number of keys used

If the sender and receiver uses same key then it is said to be **symmetric key (or)**
single key (or) conventional encryption.

If the sender and receiver use different keys then it is said to be **public key encryption**.

The way in which the plain text is processed

A **block cipher** processes the input and block of elements at a time, producing output block for each input block.

A **stream cipher** processes the input elements continuously, producing output element one at a time, as it goes along.

Cryptanalysis

The process of attempting to discover X or K or both is known as cryptanalysis. The strategy used by the cryptanalysis depends on the nature of the encryption scheme and the information available to the cryptanalyst.

There are various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst.

Cipher text only – A copy of cipher text alone is known to the cryptanalyst.

Known plaintext – The cryptanalyst has a copy of the cipher text and the corresponding plaintext.

Chosen plaintext – The cryptanalysts gains temporary access to the encryption machine. They cannot open it to find the key, however; they can encrypt a large number of suitably chosen plaintexts and try to use the resulting cipher texts to deduce the key.

Chosen cipher text – The cryptanalyst obtains temporary access to the decryption machine, uses it to decrypt several string of symbols, and tries to use the results to deduce the key.

STEGANOGRAPHY

A plaintext message may be hidden in any one of the two ways. The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text.

A simple form of steganography, but one that is time consuming to construct is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message.

e.g., (i) the sequence of first letters of each word of the overall message spells out the real (Hidden) message.
(ii) Subset of the words of the overall message is used to convey the hidden
message.

Various other techniques have been used historically, some of them are

Character marking – selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held to an angle to bright light.

Invisible ink – a number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.

Pin punctures – small pin punctures on selected letters are ordinarily not visible unless the paper is held in front of the light. Typewritten correction ribbon – used between the lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Drawbacks of steganography

Requires a lot of overhead to hide a relatively few bits of information.

Once the system is discovered, it becomes virtually worthless.

SECURITY SERVICES

The classification of security services are as follows:

Confidentiality: Ensures that the information in a computer system a n d transmitted information are accessible only for reading by authorized parties.

E.g. Printing, displaying and other forms of disclosure.

Authentication: Ensures that the origin of a message or electronic document is correctly identified, with an assurance that the identity is not false.

Integrity: Ensures that only authorized parties are able to modify computer system assets and transmitted information. Modification includes writing, changing status, deleting, creating and delaying or replaying of transmitted messages.

Non repudiation: Requires that neither the sender nor the receiver of a message be able to deny the transmission.

Access control: Requires that access to information resources may be controlled by or the target system.

Availability: Requires that computer system assets be available to authorized parties when needed.

SECURITY MECHANISMS

One of the most specific security mechanisms in use is cryptographic techniques. Encryption or encryption-like transformations of information are the most common means of providing security. Some of the mechanisms are

1 Encipherment

2 Digital Signature

3 Access Control

SECURITY ATTACKS

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

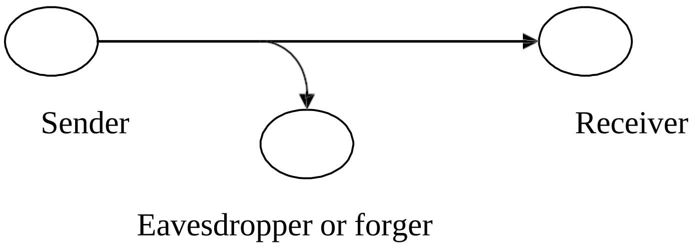
Interruption

An asset of the system is destroyed or becomes unavailable or unusable. This is an attack on availability e.g., destruction of piece of hardware, cutting of a communication line or

Disabling of file management system.

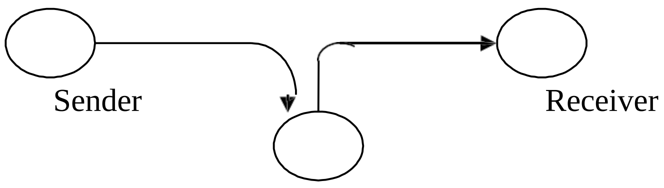
Interception

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.e.g., wire tapping to capture data in the network, illicit copying of files



Modification

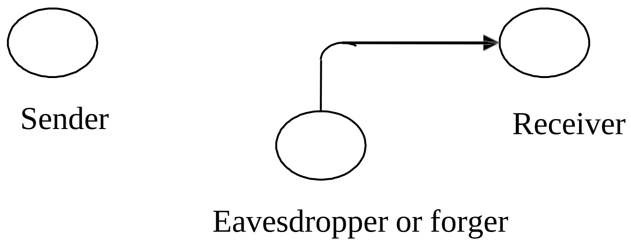
An unauthorized party not only gains access to but tampers with an asset. This is an attack on integrity. e.g., changing values in data file, altering a program, modifying the contents of messages being transmitted in a network.



Eavesdropper or forger

Fabrication

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



Cryptographic Attacks

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: If we had encryption protection in place, an opponent might still be able to observe the pattern of the message. The opponent could determine the location and identity of communication hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of communication that was taking place.

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.

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 pretends to be a different entity.

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

Modification of messages – Some portion of message is altered or the messages are delayed or recorded, to produce an unauthorized effect.

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.

Symmetric and public key algorithms

Encryption/Decryption methods fall into two categories.

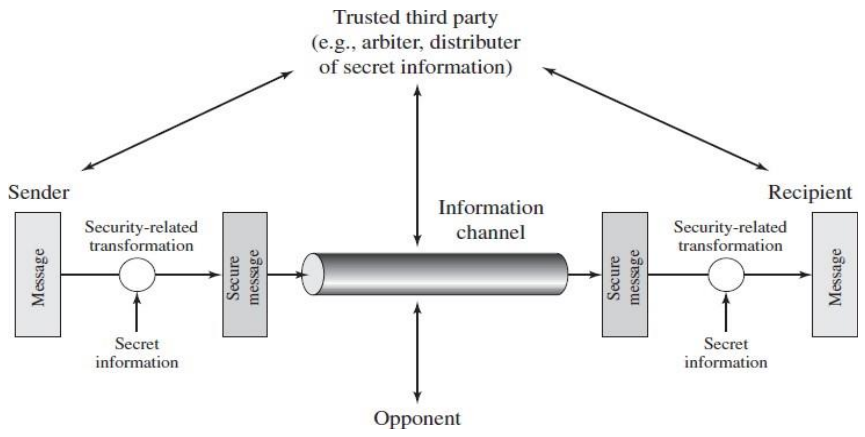
Symmetric key

Public key

In symmetric key algorithms, the encryption and decryption keys are known both to sender and receiver. The encryption key is shared and the decryption key is easily calculated from it. In many cases, the encryption and decryption keys are the same.

In public key cryptography, encryption key is made public, but it is computationally infeasible to find the decryption key without the information known to the receiver.

A MODEL FOR NETWORK SECURITY



A message is to be transferred from one party to another across some sort of internet. The two parties, who are the principals in this transaction, must cooperate for the exchange to take place. A logical information channel is established by defining a route through the internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.

Using this model requires us to:

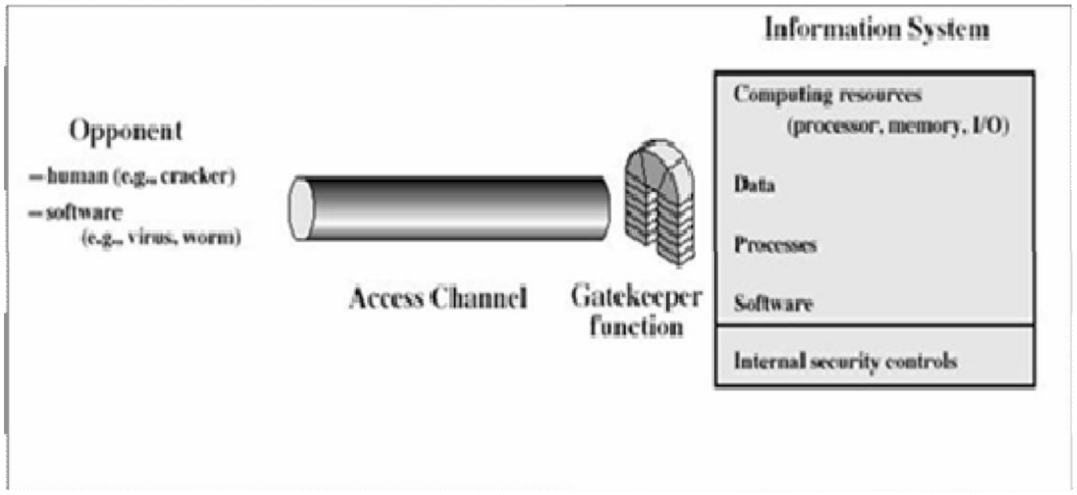
– design a suitable algorithm for the security transformation

– 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

MODEL FOR NETWORK ACCESS SECURITY



Using this model requires us to:

– select appropriate gatekeeper functions to identify users

– implement security controls to ensure only authorized users access designated information or resources

• **Trusted computer systems can be used to implement this model**

CONVENTIONAL ENCRYPTION

• Referred conventional / private-key / single-key

• Sender and recipient share a common key

All classical encryption algorithms are private-key was only type prior to invention of public-key in 1970‟**plaintext** - the original message
Some basic terminologies used:

• **cipher text** - the coded message

• **Cipher** - algorithm for transforming plaintext to cipher text

• **Key** - info used in cipher known only to sender/receiver

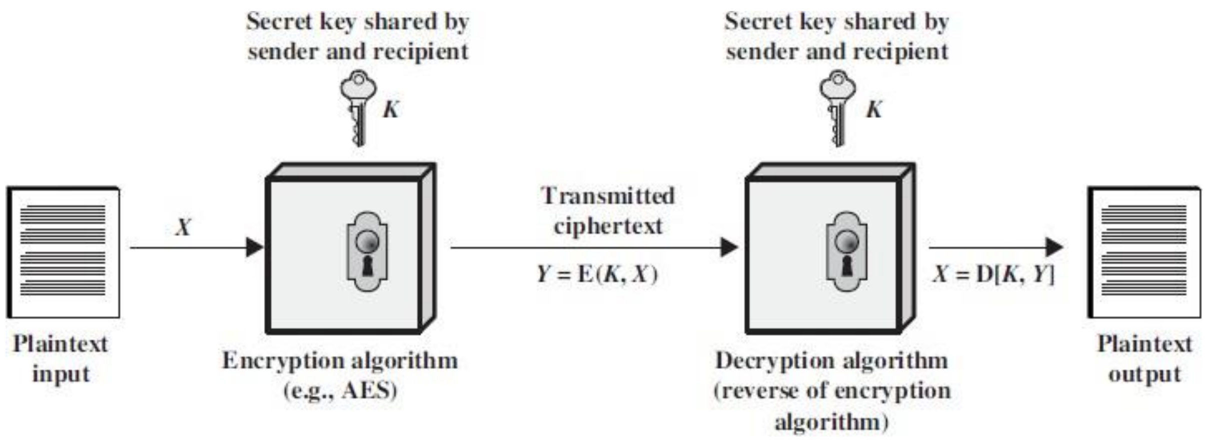
• **encipher (encrypt)** - converting plaintext to cipher text

• **decipher (decrypt)** - recovering cipher text from plaintext

• **Cryptography** - study of encryption principles/methods

• **Cryptanalysis (code breaking)** - the study of principles/ methods of deciphering cipher text *without* knowing key

• **Cryptology** - the field of both cryptography and cryptanalysis


Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E(K, X)
X = D[K, Y]
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Here the original message, referred to as plaintext, is converted into apparently random nonsense, referred to as cipher text. The encryption process consists of an algorithm and a key. The key is a value independent of the plaintext. Changing the key changes the output of the algorithm. Once the cipher text is produced, it may be transmitted. Upon reception, the cipher text can be transformed back to the original plaintext by using a decryption algorithm and the same key that was used for encryption. The security depends on several factors. First, the encryption algorithm must be powerful enough that it is impractical to decrypt a message on the basis of cipher text alone. Beyond that, the security depends on the secrecy of the key, not the secrecy of the algorithm.

- **Two requirements for secure use of symmetric encryption:**

– A strong encryption algorithm

– A secret key known only to sender / receiver

$Y = \mathrm{E}K(X)$

$X = \mathrm{D}K(Y)$

- **assume encryption algorithm is known**
- **implies a secure channel to distribute key**

A source produces a message in plaintext, X = [X1, X2… XM] where M are the number of letters in the message. A key of the form K = [K1, K2… KJ] is generated. If the key is generated at the source, then it must be provided to the destination by means of some secure channel.

With the message X and the encryption key K as input, the encryption algorithm forms the cipher text Y = [Y1, Y2, YN]. This can be expressed as

Y = EK(X)

The intended receiver, in possession of the k e y , is able to invert the transformation:

X = DK(Y)

An opponent, observing Y but not having access to K or X, may attempt to recover X or K or both. It is assumed that the opponent knows the encryption and decryption algorithms.

If the opponent is interested in only this particular message, then the focus of effort is to recover X by generating a plaintext estimate. Often if the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover K by generating an estimate.

CLASSICAL ENCRYPTION TECHNIQUES

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

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.

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.

e.g., 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

Playfair cipher

The best known multiple letter encryption cipher is the playfair, which treats digrams in the plaintext as single units and translates these units into cipher text digrams. The playfair algorithm is based on the use of 5x5 matrix of letters constructed using a keyword. Let the keyword be „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 alphabetical order.

The letter „i‟ and „j‟ count as one letter. Plaintext is encrypted two letters at a time
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.

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

Plaintext = meet me at the school house

Splitting two letters as a unit => me et me at th es ch o x 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.

1.15.1.3 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.

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

	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
	z	Z	A	B	C	D	E	F	G	H	I	J	…			Y

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.

e.g., 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 PT = w e a r e d i s c o v e r e d s a v e y o u r s e l f CT = ZICVTWQNGRZGVTWAVZHCQYGLMGJ

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.

<u>Strength of Vigenere cipher</u>

o There are multiple cipher text letters for each plaintext letter.

o Letter frequency information is obscured.

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 0‟s and 1‟s 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:

$C_i = P_i \oplus K_i$ C_i - ith binary digit of cipher text P_i - ith binary digit of plaintext K_i - ith binary digit of key

Exclusive OR operation

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:

$P_i = C_i \oplus K_i$

e.g., plaintext = 0 0 1 0 1 0 0 1

Key = 1 0 1 0 1 1 0 0

------------------ ciphertext = 1 0 0 0 0 1 0 1

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.

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.

Rail fence

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.

Plaintext = meet at the school house

To encipher this message with a rail fence of depth 2, we write the message as follows:

```
m e  a  t e  c  o l o s
e  t   t   h  s   h  o  h  u  e
```

The encrypted message is

MEATECOLOSETTHSHOHUE

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.

e.g., plaintext = meet at the school house

Key =	4	3	1	2	5	6	7
PT =	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.

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 L_0 and 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 L_{i-1} and R_{i-1}, derived from the previous round, as well as the subkey K_i, derived from the overall key K. in general, the subkeys 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 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

Subkey 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 & ease of analysis - are more recent concerns for practical use and testing.

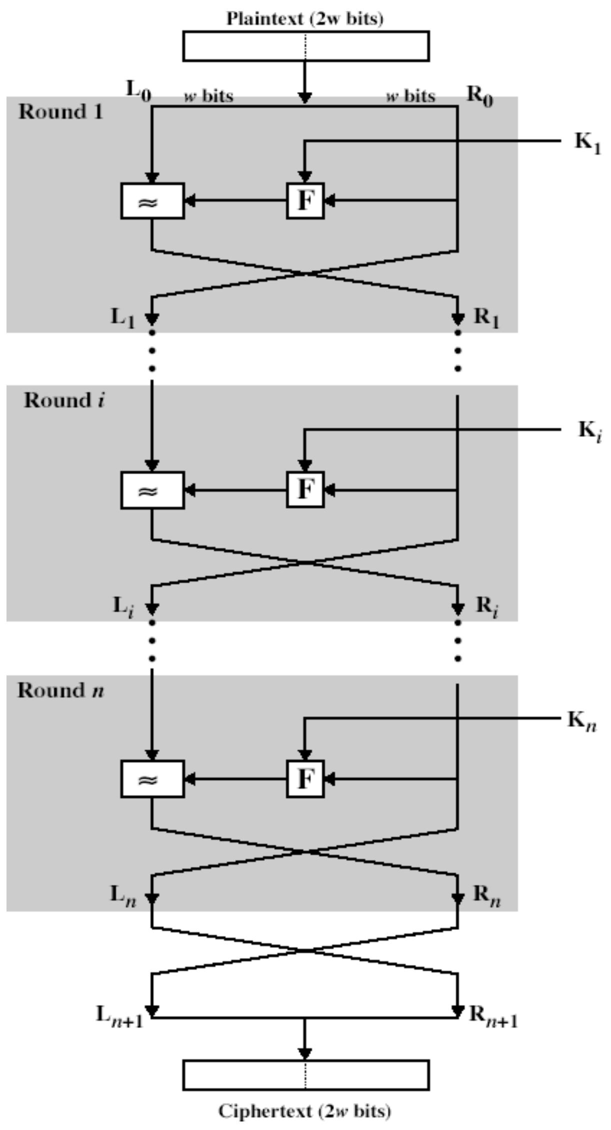


Fig: Classical Feistel Network

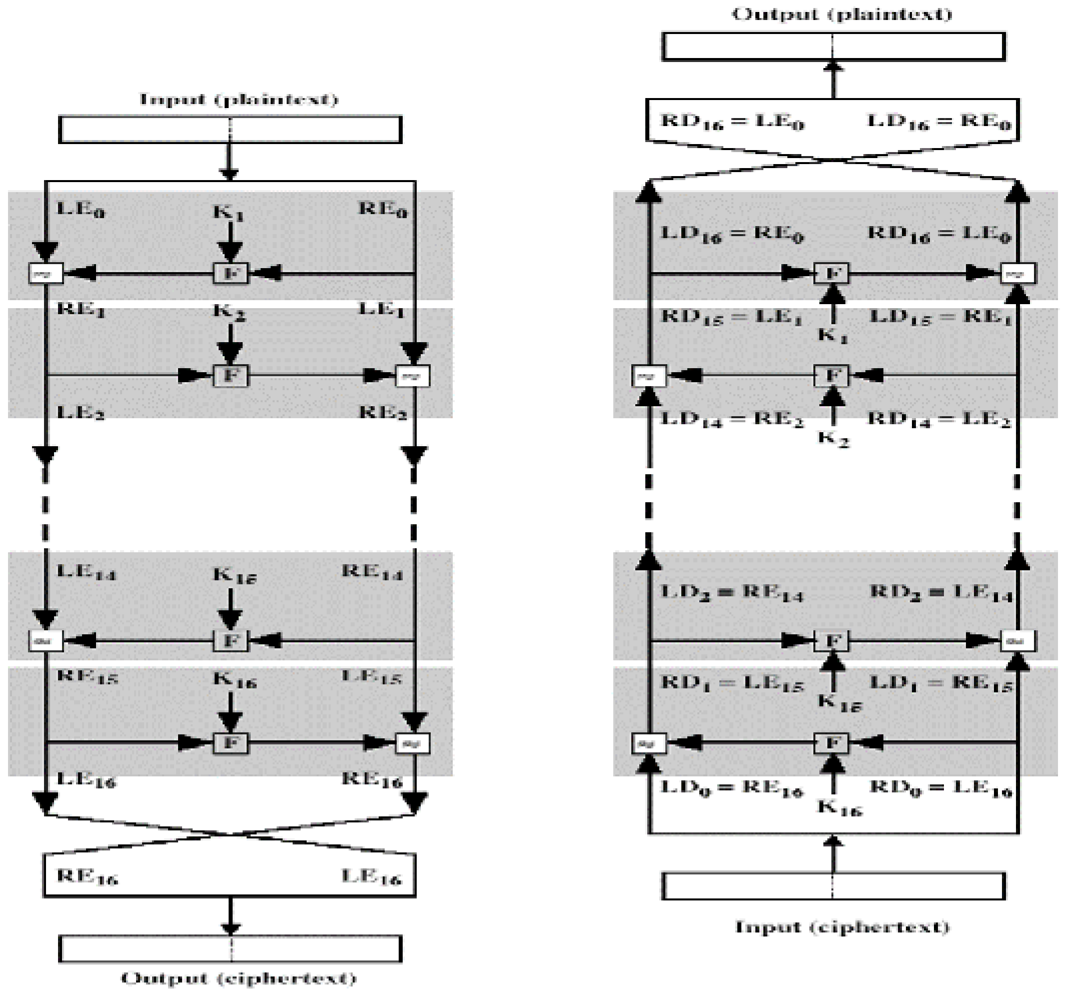


Fig: Feistel encryption and decryption

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 subkey k_i in reverse order. i.e., k_n in the first round, k_{n-1} in second round and so on. For clarity, we use the notation LE_i and 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., $RE_i \| LE_i$ (or) equivalently $RD_{16-i} \| LD_{16-i}$

After the last iteration of the encryption process, the two halves of the output are swapped, so that the cipher text is $RE_{16} \| LE_{16}$. 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 $RE_{16} \| LE_{16}$, 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. First consider the encryption process,

$LE_{16} = RE_{15}$

$RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16})$ On the decryption side,

$LD_1 = RD_0 = LE_{16} = RE_{15}$

$RD_1 = LD_0 \oplus F(RD_0, K_{16})$

$= RE_{16} \oplus F(RE_{15}, K_{16})$

$= [LE_{15} \oplus F(RE_{15}, K_{16})] \oplus F(RE_{15}, K_{16})$

$= LE_{15}$

Therefore, $LD_1 = RE_{15}$

$RD_1 = LE_{15}$ In general, for the ith iteration of the encryption algorithm, $LE_i = RE_{i-1}$

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

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

Cryptography and Network Security

Third Edition

by William Stallings

Lecture slides by Lawrie Brown

Chapter 1 – Introduction

The art of war teaches us to rely not on the likelihood of the enemy's not coming, but on our own readiness to receive him; not on the chance of his not attacking, but rather on the fact that we have made our position unassailable.

—*The Art of War,* Sun Tzu

Background

- Information Security requirements have changed in recent times
- traditionally provided by physical and administrative mechanisms
- computer use requires automated tools to protect files and other stored information
- use of networks and communications links requires measures to protect data during transmission

Definitions

- **Computer Security** - generic name for the collection of tools designed to protect data and to thwart hackers
- **Network Security** - measures to protect data during their transmission
- **Internet Security** - measures to protect data during their transmission over a collection of interconnected networks

Aim of Course

- our focus is on **Internet Security**
- consists of measures to deter, prevent, detect, and correct security violations that involve the transmission of information

Services, Mechanisms, Attacks

- need systematic way to define requirements
- consider three aspects of information security:
  - **security attack**
  - **security mechanism**
  - **security service**
- consider in reverse order

Security Service

- is something that enhances the security of the data processing systems and the information transfers of an organization
- intended to counter security attacks
- make use of one or more security mechanisms to provide the service
- replicate functions normally associated with physical documents
  - eg have signatures, dates; need protection from disclosure, tampering, or destruction; be notarized or witnessed; be recorded or licensed

Security Mechanism

- a mechanism that is designed to detect, prevent, or recover from a security attack
- no single mechanism that will support all functions required
- however one particular element underlies many of the security mechanisms in use: **cryptographic techniques**
- hence our focus on this area

Security Attack

- any action that compromises the security of information owned by an organization
- information security is about how to prevent attacks, or failing that, to detect attacks on information-based systems
- have a wide range of attacks
- can focus of generic types of attacks
- note: often *threat* & *attack* mean same

OSI Security Architecture

- ITU-T X.800 Security Architecture for OSI
- defines a systematic way of defining and providing security requirements
- for us it provides a useful, if abstract, overview of concepts we will study

Security Services

- X.800 defines it as: a service provided by a protocol layer of communicating open systems, which ensures adequate security of the systems or of data transfers
- RFC 2828 defines it as: a processing or communication service provided by a system to give a specific kind of protection to system resources
- X.800 defines it in 5 major categories

Security Services (X.800)

- **Authentication** - assurance that the communicating entity is the one claimed
- **Access Control** - prevention of the unauthorized use of a resource
- **Data Confidentiality** –protection of data from unauthorized disclosure
- **Data Integrity** - assurance that data received is as sent by an authorized entity
- **Non-Repudiation** - protection against denial by one of the parties in a communication

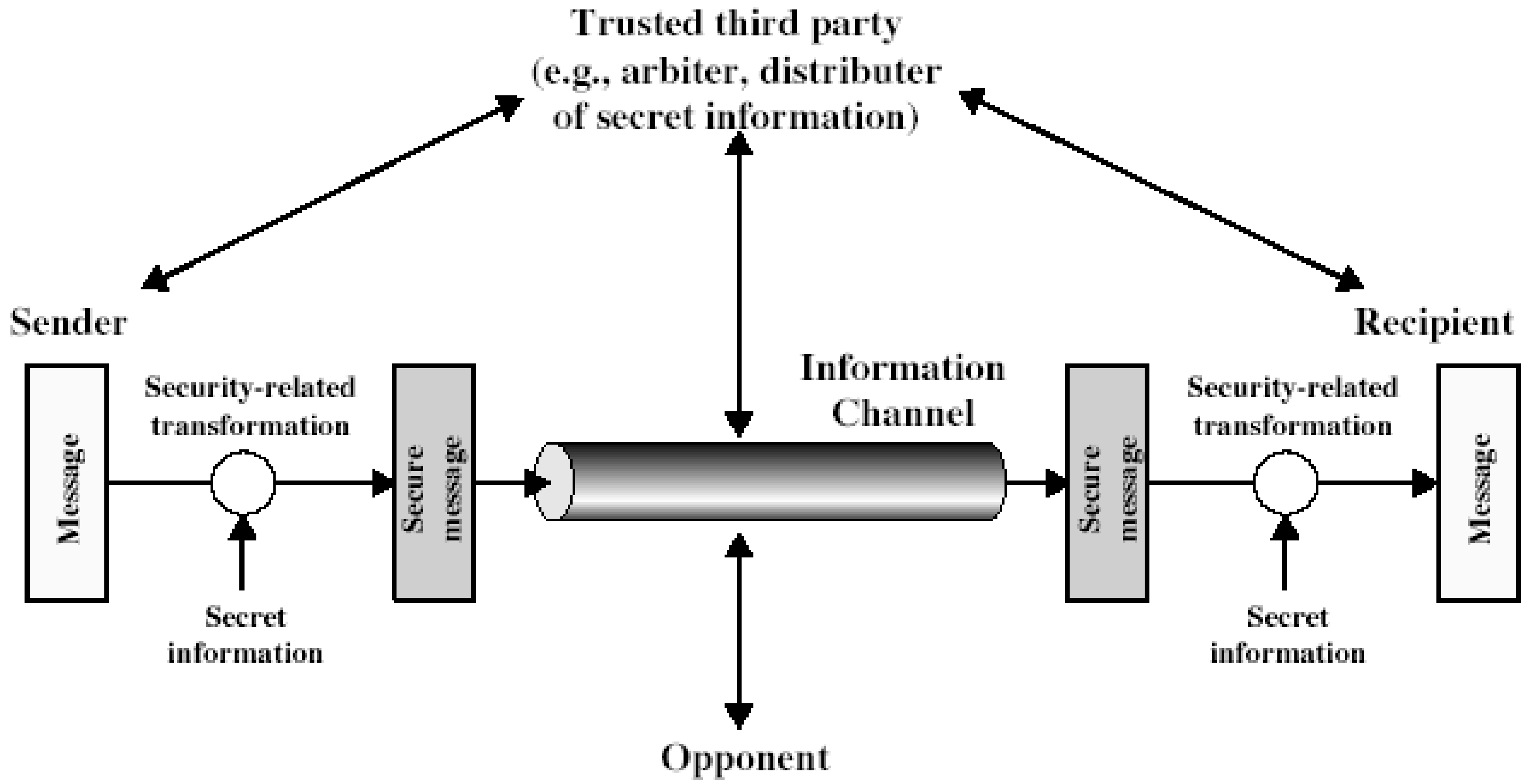
Security Mechanisms (X.800)

- 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

Classify Security Attacks as

- **passive attacks** - eavesdropping on, or monitoring of, transmissions to:
  - obtain message contents, or
  - monitor traffic flows
- **active attacks** – modification of data stream to:
  - masquerade of one entity as some other
  - replay previous messages
  - modify messages in transit
  - denial of service

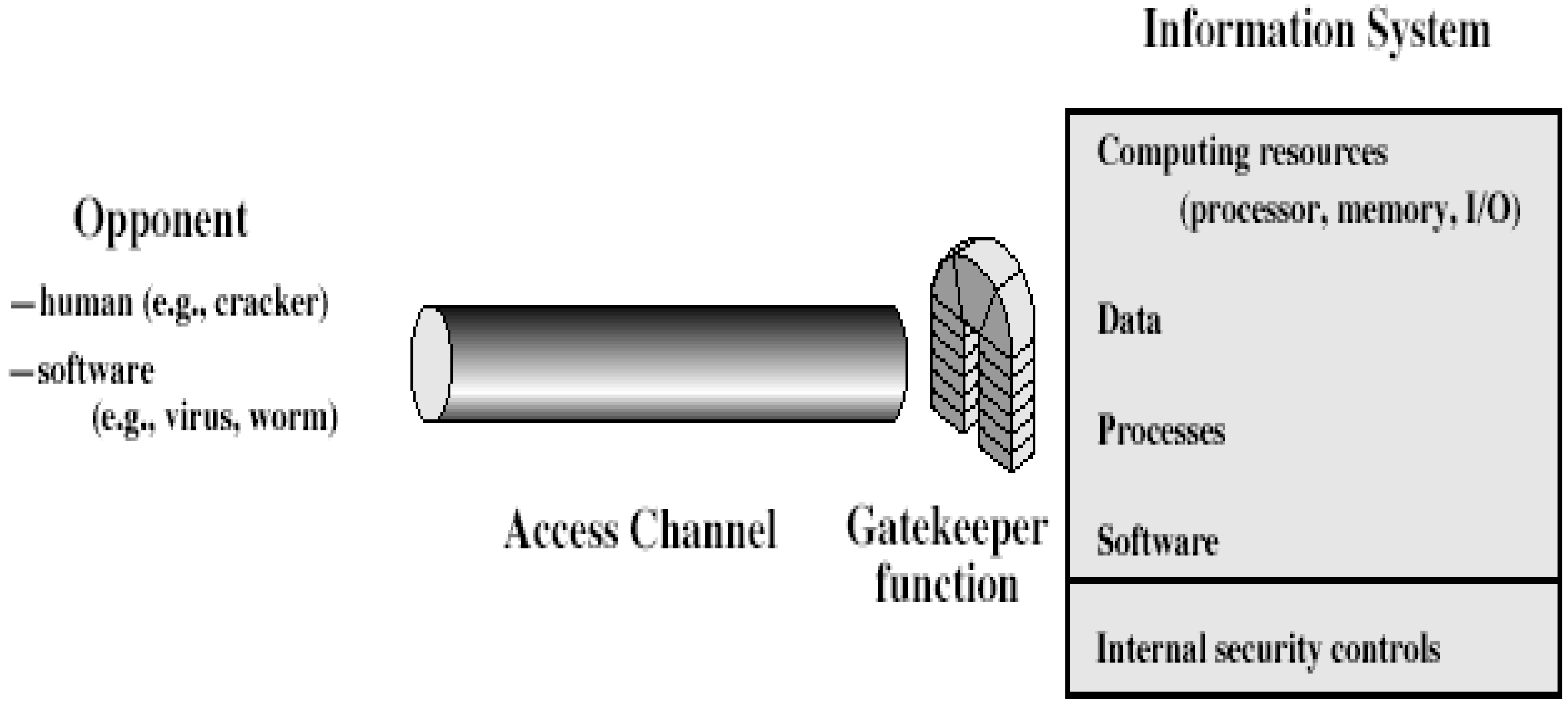
Model for Network Security



Model for Network Security

- using this model requires us to:
  - 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

Model for Network Access Security



Model for Network Access Security

- 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

Summary

- have considered:
  - computer, network, internet security def’s
  - security services, mechanisms, attacks
  - X.800 standard
  - models for network (access) security

Chapter 3 – Block Ciphers and the Data Encryption Standard

Last Chapter

- have considered:
  - terminology
  - classical cipher techniques
  - substitution ciphers
    - cryptanalysis using letter frequencies
  - transposition ciphers

Modern Block Ciphers

- will now look at modern block ciphers
- one of the most widely used types of cryptography algorithms
- provide strong secrecy and/or authentication services
- in particular will introduce DES (Data Encryption Standard)

Block vs Stream Ciphers

- block ciphers process messages into blocks, each of which is then en/decrypted
- like a substitution on very big characters
  - 64-bits or more
- stream ciphers process messages a bit or byte at a time when en/decrypting
- many current ciphers are block ciphers
- hence are focus of course

Block Cipher Principles

- block ciphers look like an extremely large substitution
- would need table of 2^{64} entries for a 64-bit block
- arbitrary reversible substitution cipher for a large block size is not practical
  - 64-bit general substitution block cipher, key size 2^{64}!
- most symmetric block ciphers are based on a **Feistel Cipher Structure**
- needed since must be able to **decrypt** ciphertext to recover messages efficiently

C. Shannon and Substitution-Permutation Ciphers

- in 1949 Shannon introduced idea of substitution-permutation (S-P) networks
  - 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) (transposition)
- provide *confusion* and *diffusion* of message

Diffusion and Confusion

- Introduced by Claude Shannon to thwart cryptanalysis based on statistical analysis
  - Assume the attacker has some knowledge of the statistical characteristics of the plaintext
- cipher needs to completely obscure statistical properties of original message
- a one-time pad does this

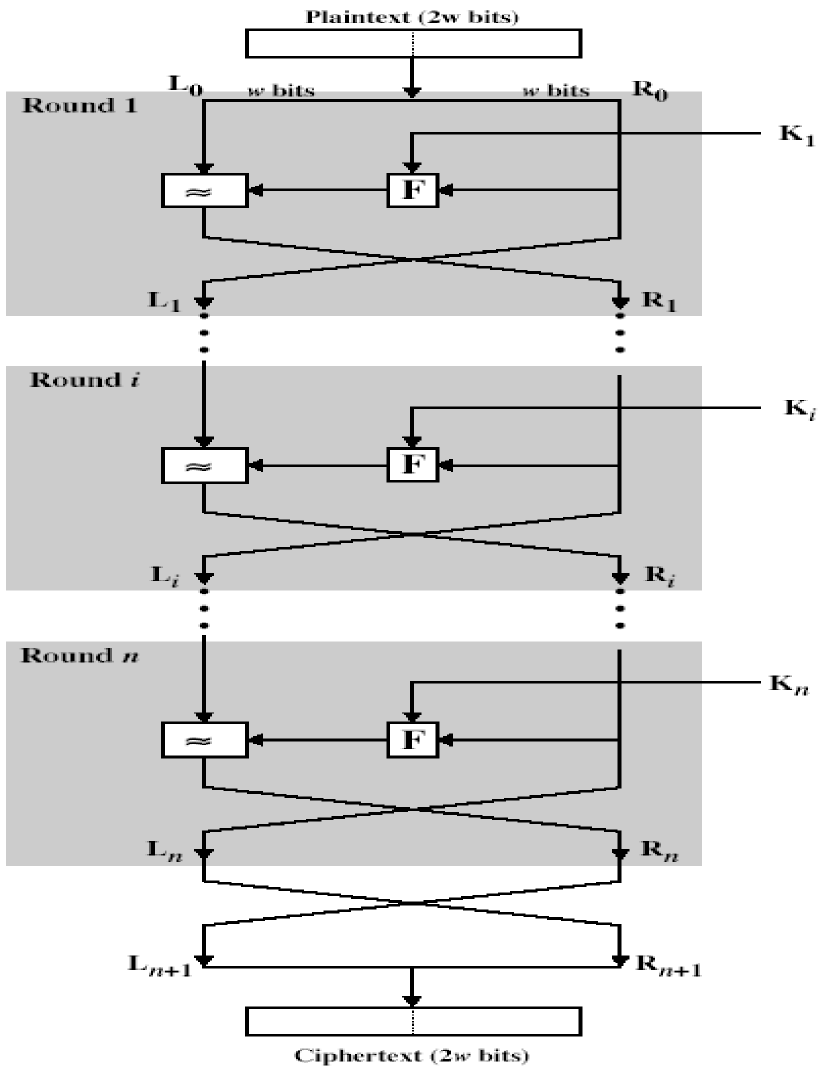
Diffusion and Confusion

- more practically Shannon suggested combining elements to obtain:
- **diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext
- **confusion** – makes relationship between ciphertext and key as complex as possible

Feistel Cipher Structure

- Horst Feistel devised the **feistel cipher**
  - implements Shannon’s substitution-permutation network concept
- partitions input block into two halves
  - process through multiple rounds which
  - perform a substitution on left data half
  - based on round function of right half & subkey
  - then have permutation swapping halves

Feistel Cipher Structure



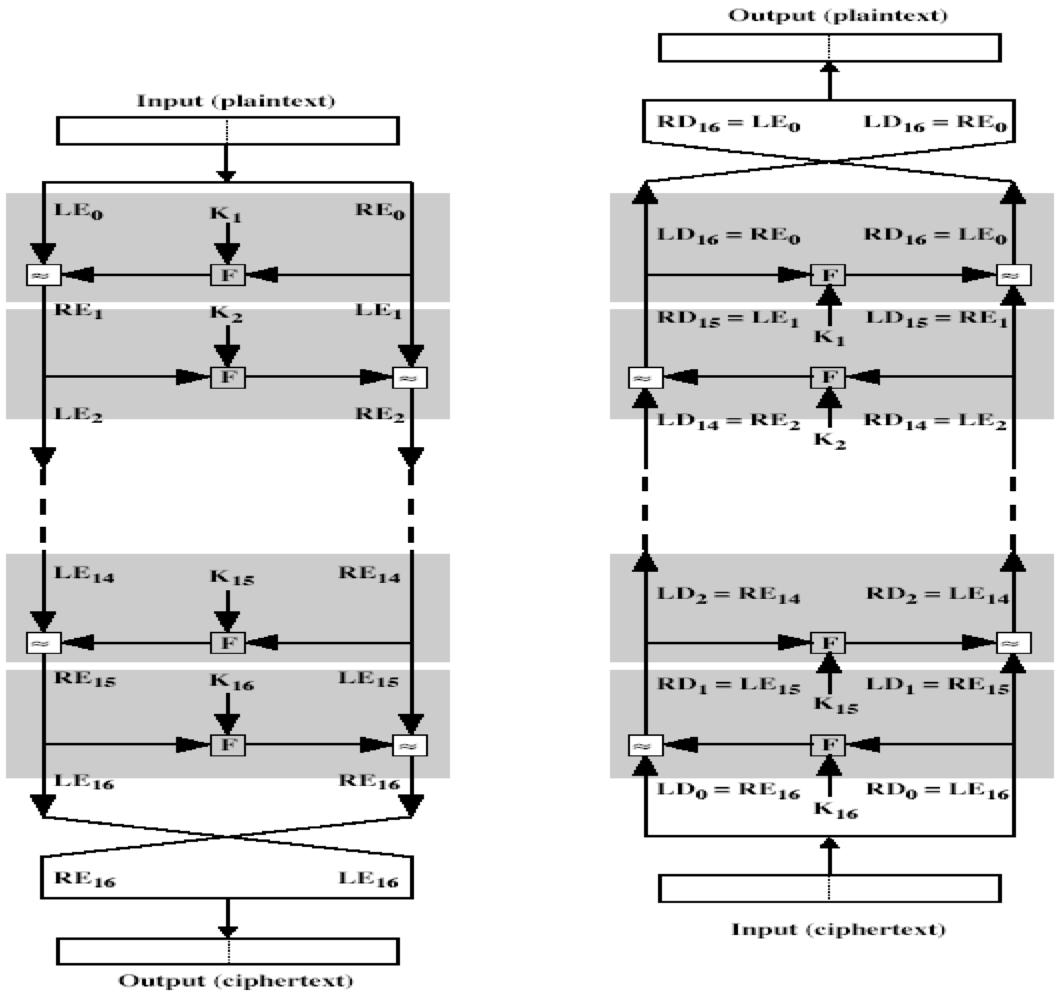
Feistel Cipher

- n sequential rounds
- A substitution on the left half L_i
  - 1. Apply a round function F to the right half R_i and
  - 2. Take XOR of the output of (1) and L_i
- The round function is parameterized by the subkey K_i
  - K_i are derived from the overall key K

Feistel Cipher Design Principles

- **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
- **subkey 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 & ease of analysis**
  - are more recent concerns for practical use and testing

Feistel Cipher Decryption



Data Encryption Standard (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
  - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use

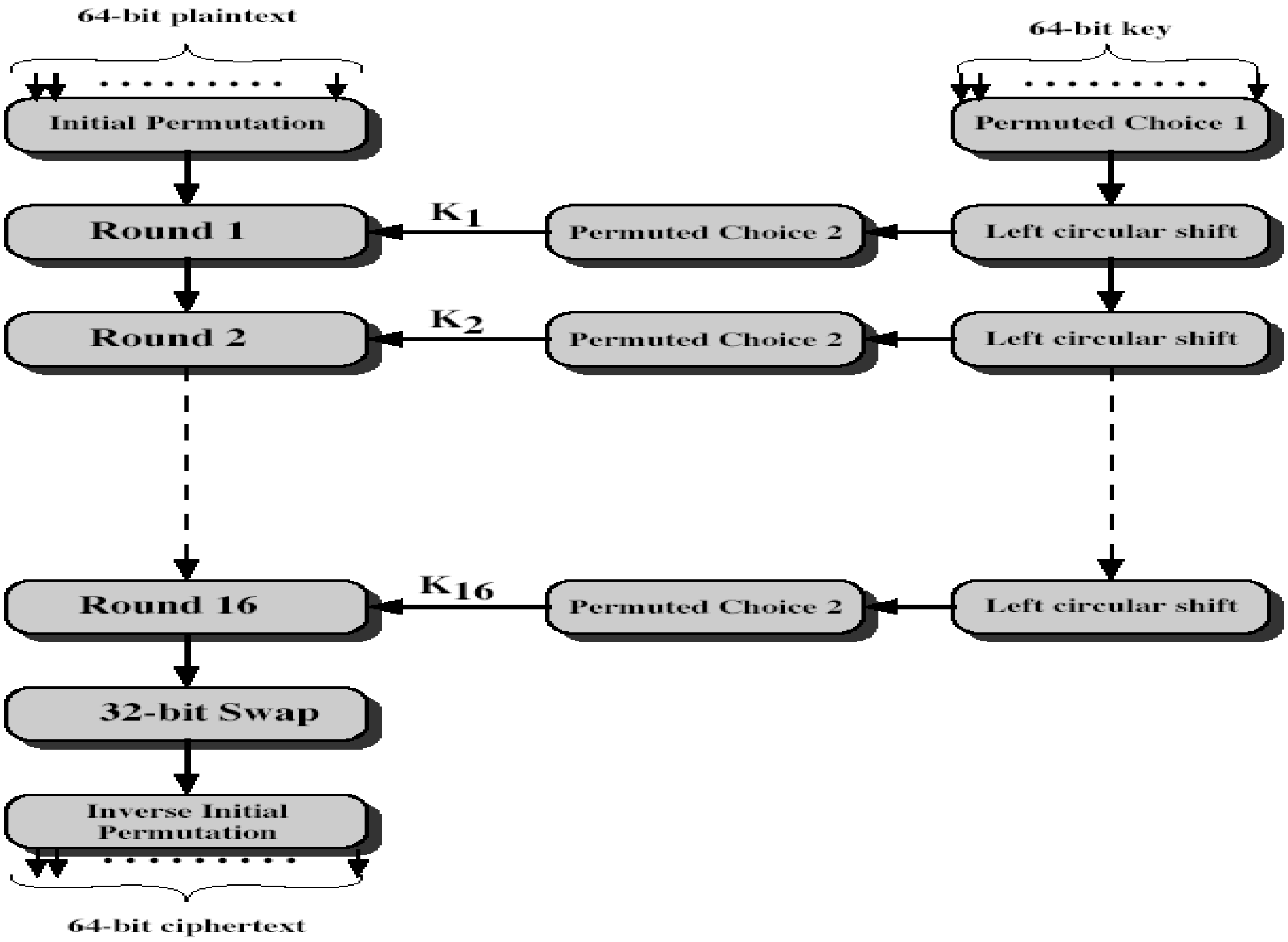
DES History

- IBM developed Lucifer cipher
  - by team led by Feistel
  - used 64-bit data blocks with 128-bit key
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES

DES Design Controversy

- although DES standard is public
- was considerable controversy over design
  - in choice of 56-bit key (vs Lucifer 128-bit)
- subsequent events and public analysis show in fact design was appropriate
- DES has become widely used, especially in financial applications

DES Encryption



Initial Permutation IP

- first step of the data computation
- IP reorders the input data bits
- quite regular in structure
  - see text Table 3.2
- example:

```
IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)
```

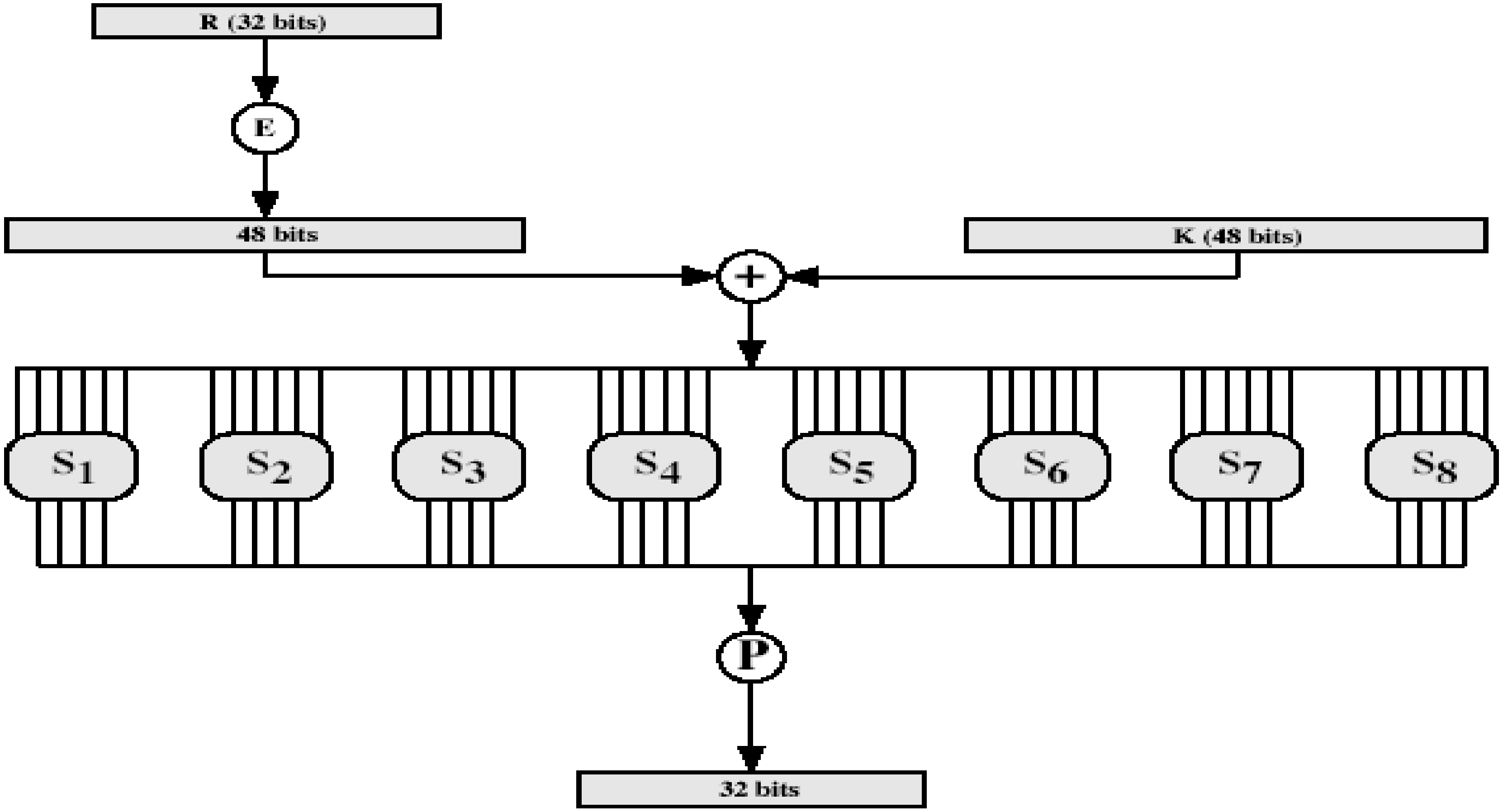
DES Round Structure

- uses two 32-bit L & R halves
- as for any Feistel cipher can describe as:

  $L_i = R_{i-1}$

  $R_i = L_{i-1}$ xor $F(R_{i-1}, K_i)$
- takes 32-bit R half and 48-bit subkey and:
  - expands R to 48-bits using Expansion Permutation E (Table 3.2 c.)
  - adds to subkey
  - passes through 8 S-boxes to get 32-bit result
  - finally permutes this using 32-bit Permutation Function P (Table 3.2 d)

The round function F(R,K)



Substitution Boxes S

- 8 S-boxes (Table 3.3)
- Each S-Box mapps 6 to 4 bits
  - outer bits 1 & 6 (**row** bits) select the row
  - inner bits 2-5 (**col** bits) select the column
  - For example, in S1, for input 011001,
    - the row is 01 (row 1)
    - the column is 1100 (column 12).
    - The value in row 1, column 12 is 9
    - The output is 1001.
- result is 8 X 4 bits, or 32 bits

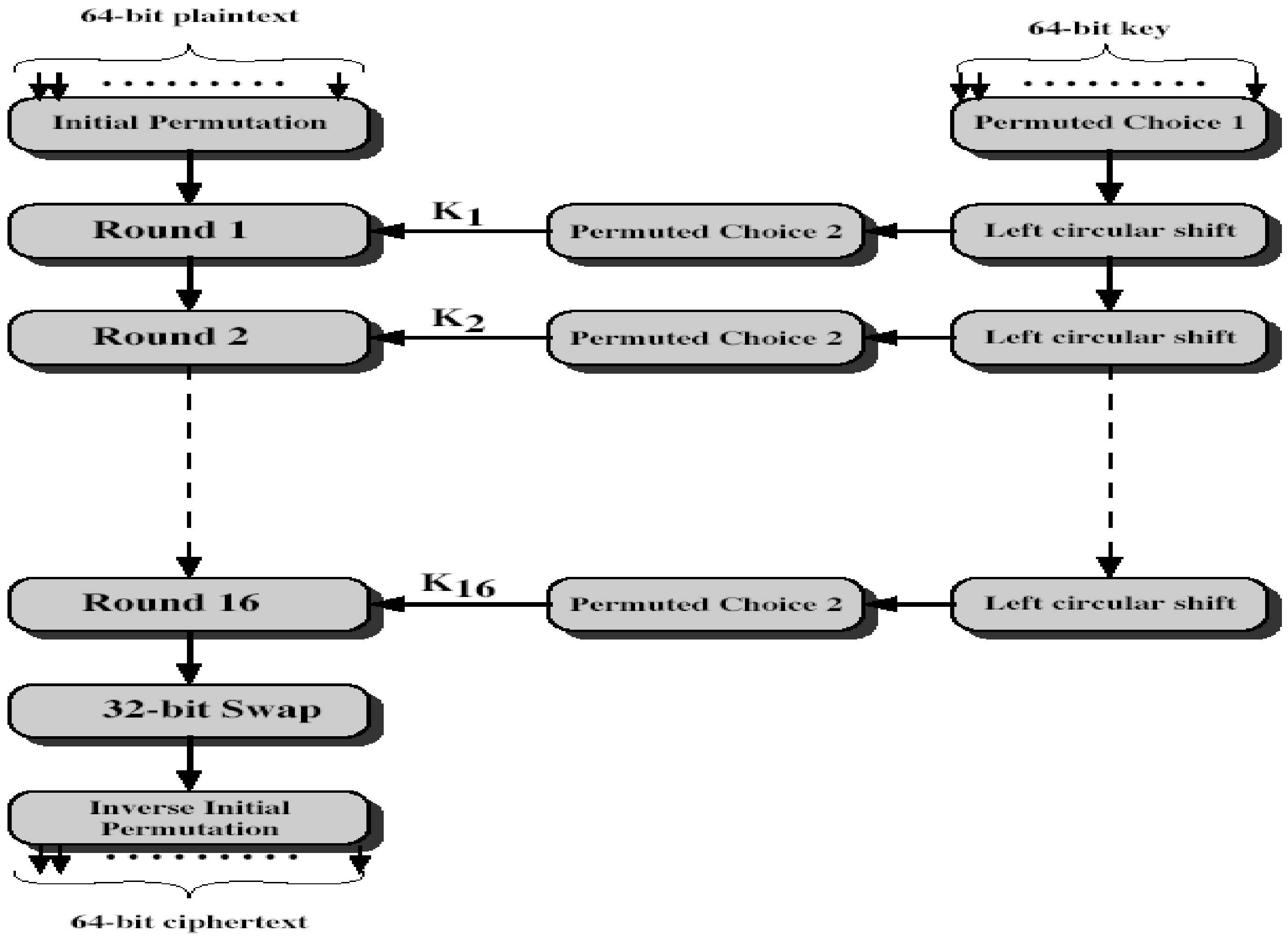
DES Key Schedule

- forms subkeys used in each round
- 1. initial permutation of the key PC1 (Table 3.4b)
- 2. divide the 56-bits in two 28-bit halves
- 3. at each round
  - 3.1. Left shift each half (28bits) separately either 1 or 2 places based on the left shift schedule (Table 3.4d)
    - Shifted values will be input for next round
  - 3.2. Combine two halfs to 56 bits, permuting them by PC2 (Table 3.4c) for use in function f
    - PC2 takes 56-bit input, outputs 48 bits

DES Decryption

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again
- using subkeys in reverse order (SK16 … SK1)
- note that IP undoes final FP step of encryption
- 1st round with SK16 undoes 16th encrypt round
- ….
- 16th round with SK1 undoes 1st encrypt round
- then final FP undoes initial encryption IP
- thus recovering original data value

DES Decryption (reverse encryption)



Avalanche Effect

- key desirable property of encryption alg
- DES exhibits strong avalanche
- where a change of **one** input or key bit results in changing approx **half** output bits

Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- brute force search looks hard
- recent advances have shown is possible
  – in 1997 on Internet in a few months
  – in 1998 on dedicated hardware (EFF) in a few days
  – in 1999 above combined in 22hrs!
- still must be able to recognize plaintext
- now considering alternatives to DES

Strength of DES – Timing Attacks

- attacks actual implementation of cipher
- use knowledge of consequences of implementation to derive knowledge of some/all subkey bits
- specifically use fact that calculations can take varying times depending on the value of the inputs to it

Strength of DES – Analytic Attacks

- now have several analytic attacks on DES
- these utilise some deep structure of the cipher
  - by gathering information about encryptions
  - can eventually recover some/all of the sub-key bits
  - if necessary then exhaustively search for the rest
- generally these are statistical attacks
- include
  - differential cryptanalysis
  - linear cryptanalysis
  - related key attacks

Differential Cryptanalysis

- one of the most significant recent (public) advances in cryptanalysis
- known in 70's with DES design
- Murphy, Biham & Shamir published 1990
- powerful method to analyse block ciphers
- used to analyse most current block ciphers with varying degrees of success
- DES reasonably resistant to it

Differential Cryptanalysis

- a statistical attack against Feistel ciphers
- uses cipher structure not previously used
- design of S-P networks has output of function *f* influenced by both input & key
- hence cannot trace values back through cipher without knowing values of the key
- Differential Cryptanalysis compares two related pairs of encryptions

Differential Cryptanalysis Compares Pairs of Encryptions

- Differential cryptanalysis is complex
- with a known difference in the input
- searching for a known difference in output

$$\Delta m_{i+1} = m_{i+1} \oplus m'_{i+1}$$

$$= \left[m_{i-1} \oplus \mathrm{f}\left(m_i, K_i\right)\right] \oplus \left[m'_{i-1} \oplus \mathrm{f}\left(m'_i, K_i\right)\right]$$

$$= \Delta m_{i-1} \oplus \left[\mathrm{f}\left(m_i, K_i\right) \oplus \mathrm{f}\left(m'_i, K_i\right)\right]$$

Differential Cryptanalysis

- have some input difference giving some output difference with probability p
- if find instances of some higher probability input / output difference pairs occurring
- can infer subkey that was used in round
- then must iterate process over many rounds

Differential Cryptanalysis

- perform attack by repeatedly encrypting plaintext pairs with known input XOR until obtain desired output XOR
- when found
  - if intermediate rounds match required XOR have a **right pair**
  - if not then have a **wrong pair**
- can then deduce keys values for the rounds
  - right pairs suggest same key bits
  - wrong pairs give random values
- larger numbers of rounds makes it more difficult
- Attack on full DES requires an effort on the order of 2^{47}, requiring 2^{47} chosen plaintexts to be encrypted

Linear Cryptanalysis

- another recent development
- also a statistical method
- based on finding linear approximations to model the transformation of DES
- can attack DES with 2^{47} known plaintexts, still in practise infeasible

Criteria for S-Boxes

- No output of any S-Box is too close to a linear function of the input bits
- Each row of an S-Box includes all 16 possible output bit combinations
- If two inputs to an S-box differ in one bit, the output bits differ in at least two bits
- If two inputs differ is the two middle bits, outputs must differ at least two bits
- Defend against differential analysis and provide good confusion properties

Block Cipher Design Principles

- basic principles still like Feistel in 1970’s
- number of rounds
  - more is better, makes exhaustive search best attack
  - 16 rounds: brute force 2^{55}
  - differential analysis: $2^{55.1}$

Block Cipher Design Principles

- function F:
  - provides “confusion”, is nonlinear, avalanche
  - Strict Avalanche Criterion (SAC)
    - Any output bit i should change with p=1/2 when any single input bit j is inverted, for all i,j
    - Applies to both S-Boxes and the overall F function
- key schedule
  - No general rule has been discovered
  - complex subkey creation, key avalanche

Modes of Operation

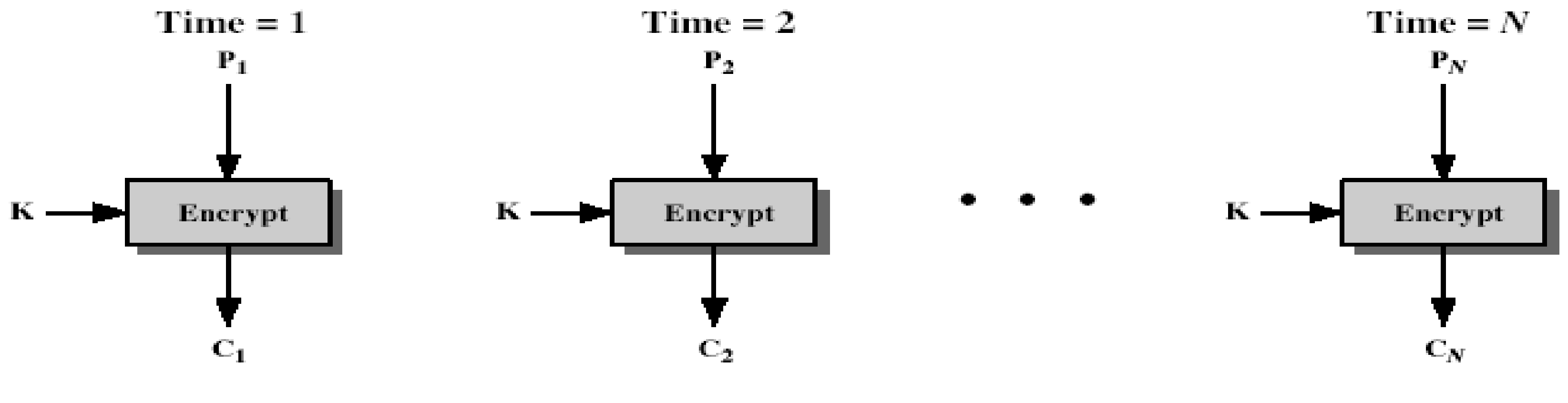
- block ciphers encrypt fixed size blocks
- eg. DES encrypts 64-bit blocks, with 56-bit key
- need way to use in practise, given usually have arbitrary amount of information to encrypt
- four were defined for DES in ANSI standard **ANSI X3.106-1983 Modes of Use**
  - DES is the basic building block
- have **block** and **stream** modes

Electronic Codebook Book (ECB)

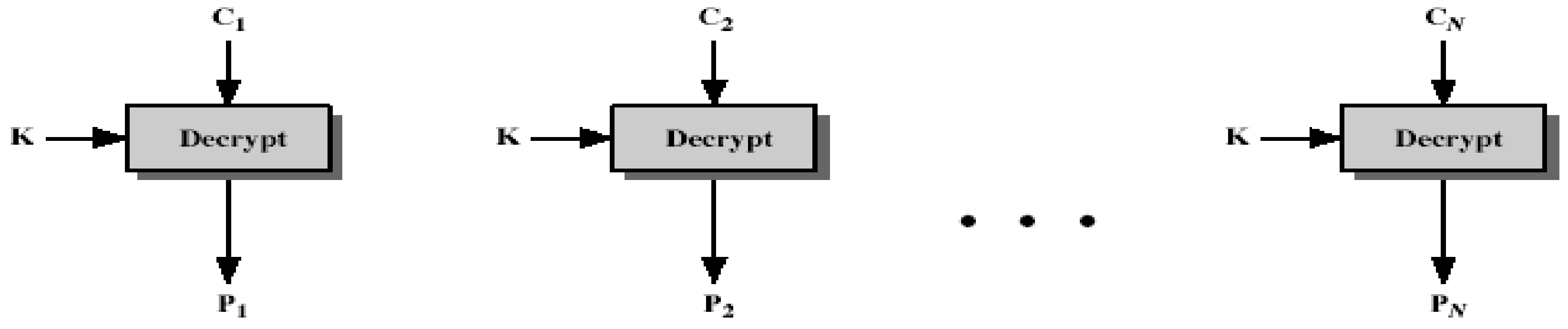
- message is broken into independent blocks which are encrypted
- each block is a value which is substituted, like a codebook, hence name
  - Each DES is a very complex 64-bit to 64-bit substitution
- each block is encoded independently of the other blocks

  $C_i = DES_{K1} (P_i)$
- uses: secure transmission of single values
  - Repeated input blocks have same output
  - Not secure for long transmission

Electronic Codebook Book (ECB)



(a) Encryption



(b) Decryption

Advantages and Limitations of ECB

- repetitions in message may show 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 due to encrypted message blocks being independent
- main use is sending a few blocks of data

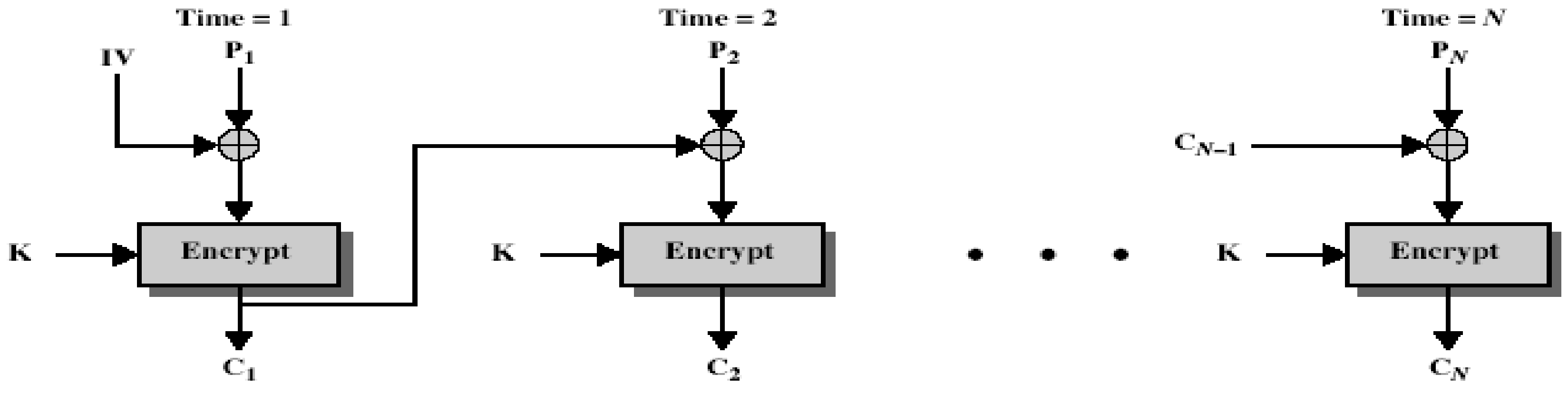
Cipher Block Chaining (CBC)

- message is broken into blocks
- but these are linked together in the encryption operation
- each previous cipher blocks is chained with current plaintext block, hence name
- use Initial Vector (IV) to start process

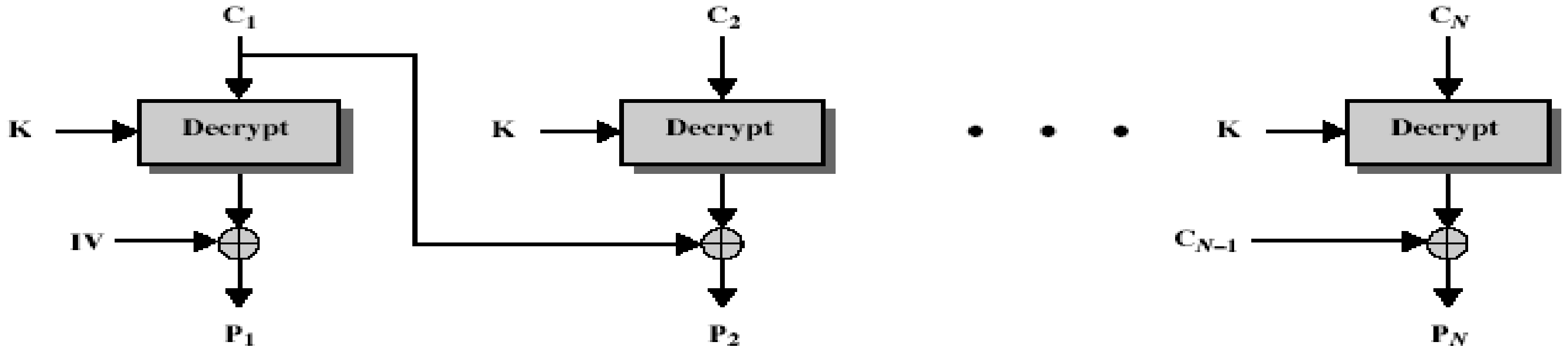
  $C_i = DES_{K1}(P_i \text{ XOR } C_{i-1})$

  $C_{-1} = IV$
- uses: bulk data encryption, authentication

Cipher Block Chaining (CBC)



(a) Encryption



(b) Decryption

Advantages and Limitations of CBC

- each ciphertext block depends on **all** message blocks
- thus a change in the message affects all ciphertext blocks after the change as well as the original block
- need **Initial Value** (IV) known to sender & 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

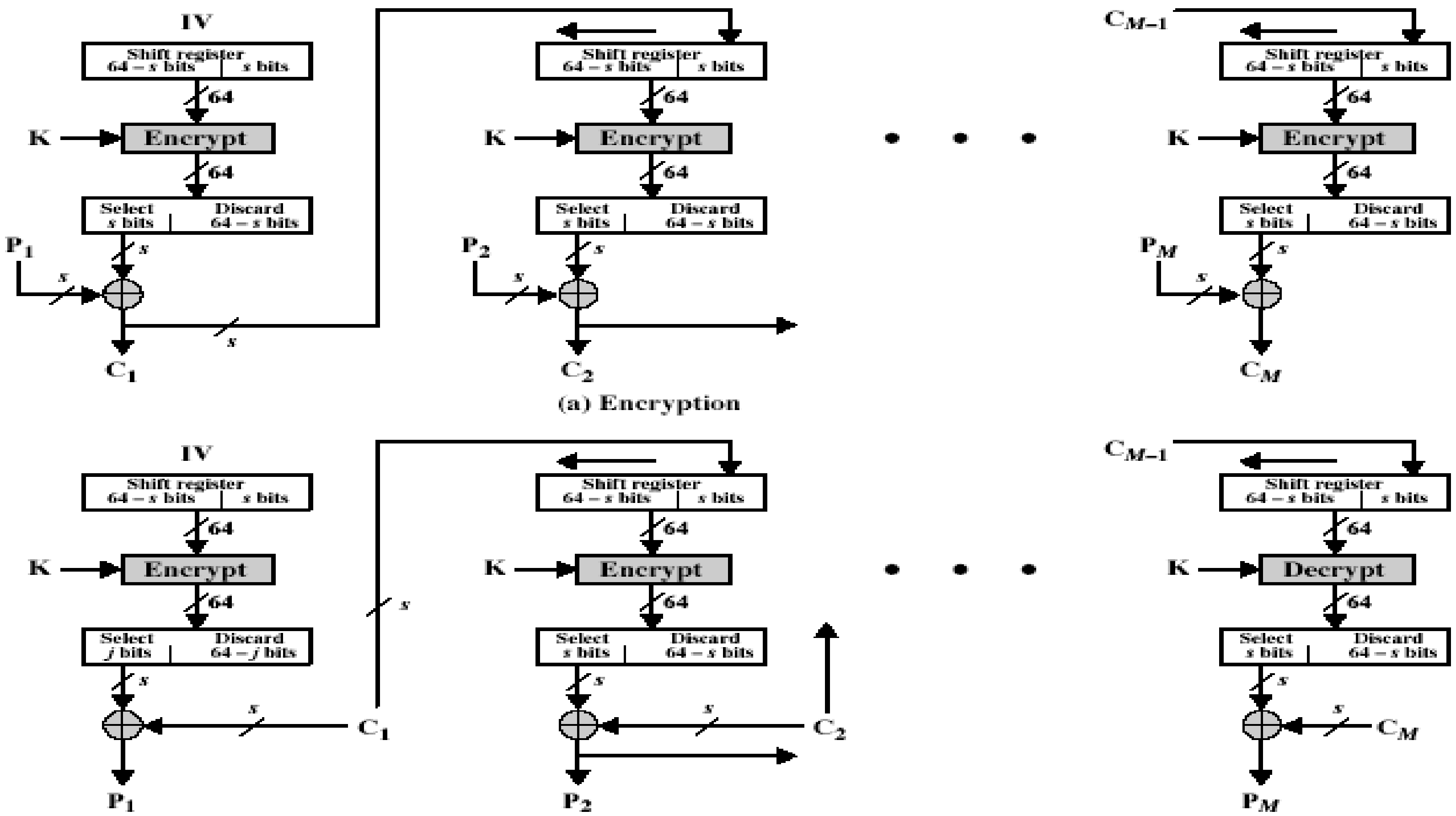
Cipher FeedBack (CFB)

- message is treated as a stream of bits
- added to the output of the block cipher
- result is feed back for next stage (hence name)
- standard allows any number of bit (1,8 or 64 or whatever) to be feed back
  - denoted CFB-1, CFB-8, CFB-64 etc
- is most efficient to use all 64 bits (CFB-64)

  $C_i = P_i \text{ XOR } DES_{K1}(C_{i-1})$

  $C_{-1} = IV$
- uses: stream data encryption, authentication

Cipher FeedBack (CFB)



(a) Encryption

(b) Decryption

Advantages and Limitations of CFB

- appropriate when data arrives in bits/bytes
- most common stream mode
- note that the block cipher is used in **encryption** mode at **both** ends
- errors propagate for several blocks after the error
  - Must use over a reliable network channel

Output FeedBack (OFB)

- message is treated as a stream of bits
- output of cipher is added to message
- output is then feed back (hence name)
- feedback is independent of message
- can be computed in advance

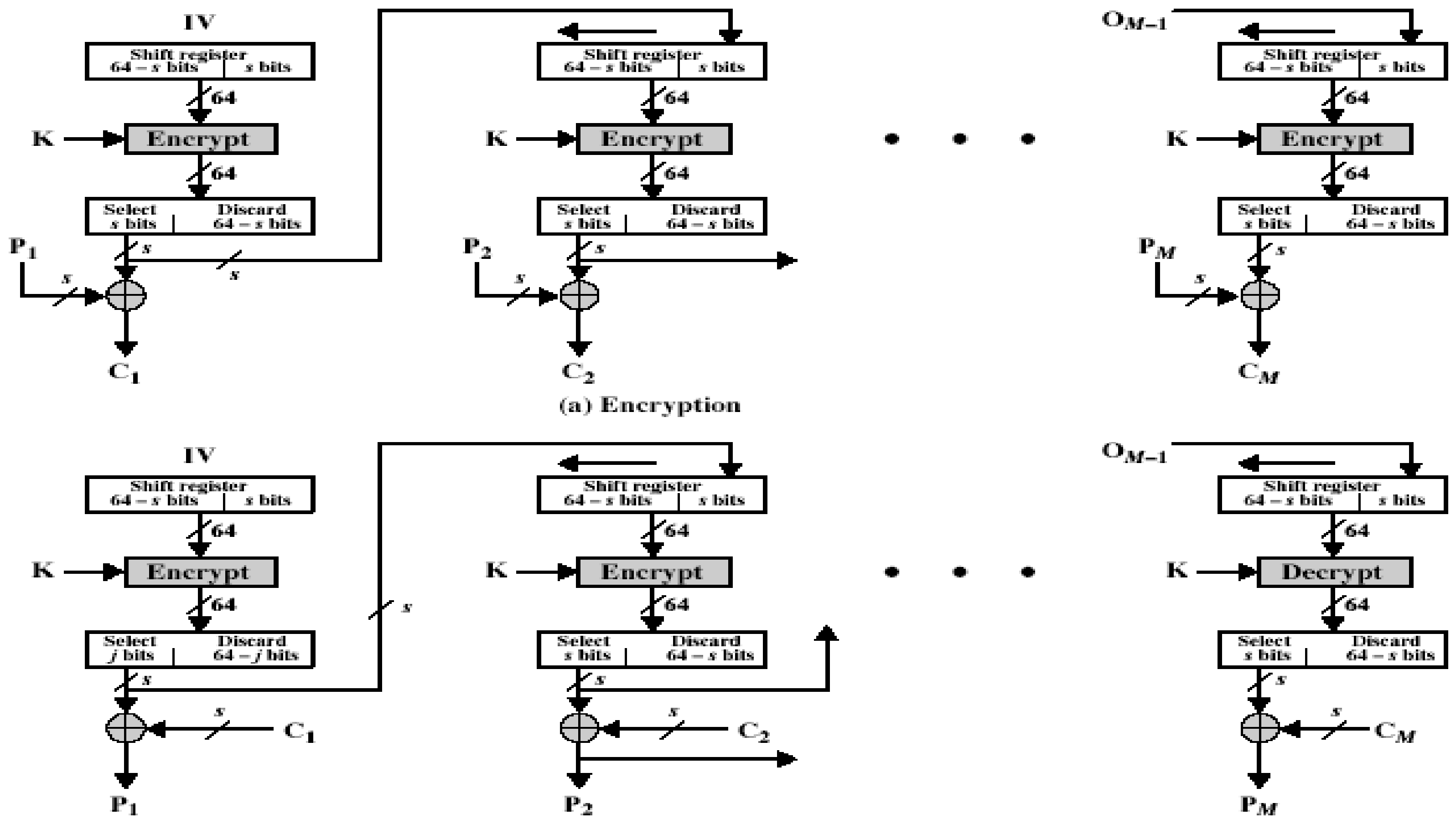
  $C_i = P_i \text{ XOR } O_i$

  $O_i = DES_{K1}(O_{i-1})$

  $O_{-1} = IV$

- uses: stream encryption over noisy channels

Output FeedBack (OFB)



(a) Encryption

(b) Decryption

Advantages and Limitations of OFB

- used when error feedback a problem or where need to encryptions before message is available
- superficially similar to CFB
- but feedback is from the output of cipher and is independent of message
  - Errors do not propagate
- sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs
- Because the "random" bits are independent of the message, they must **never** be used more than once
  - otherwise the 2 ciphertexts can be combined, cancelling these bits)

Counter (CTR)

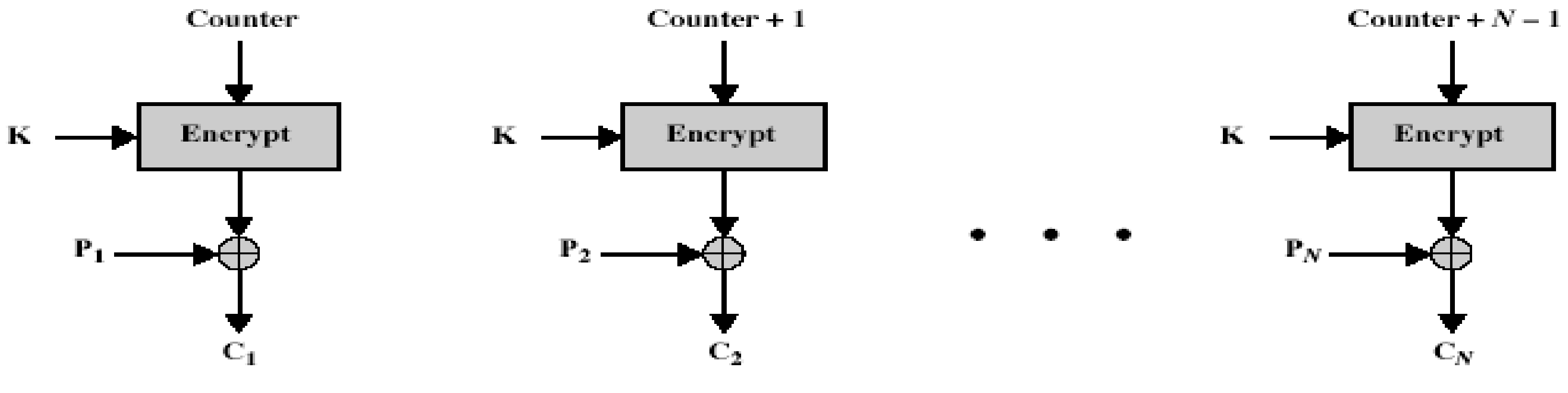
- a “new” mode, though proposed early on
- encrypts counter value rather than any feedback value
- must have a different key & counter value for every plaintext block (never reused)

  $C_i = P_i \text{ XOR } O_i$

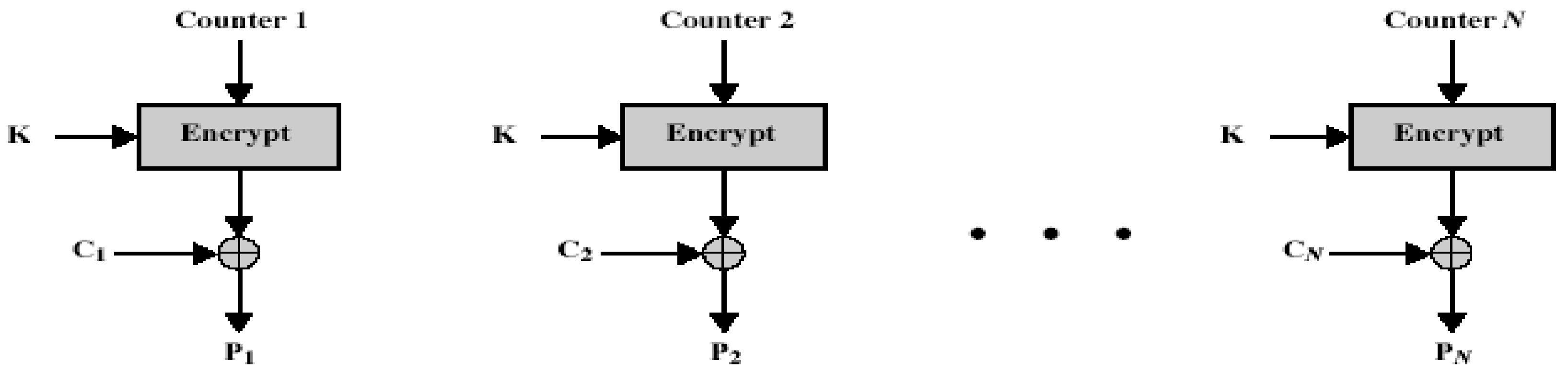
  $O_i = DES_{K1}(i)$

- uses: high-speed network encryptions

Counter (CTR)



(a) Encryption



(b) Decryption

Advantages and Limitations of CTR

- efficiency
  - can do parallel encryptions
  - in advance of need
  - good for bursty high speed links
- random access to encrypted data blocks
  - Do not have to decode from the beginning
- provable security (good as other modes)
- but must ensure never reuse key/counter values, otherwise could break (cf OFB)

Summary

- have considered:
  - block cipher design principles
  - DES
    - details
    - strength
  - Differential & Linear Cryptanalysis
  - Modes of Operation
    - ECB, CBC, CFB, OFB, CTR

Chapter 4 – Finite Fields

Introduction

- will now introduce finite fields
- of increasing importance in cryptography
  - AES, Elliptic Curve, IDEA, Public Key
- concern operations on “numbers”
  - what constitutes a “number”
  - the type of operations and the properties
- start with concepts of groups, rings, fields from abstract algebra

Group

- a set of elements or “numbers”
  - A generalization of usual arithmetic
- obeys:
  - closure: `a.b also in G`
  - associative law: `(a.b).c = a.(b.c)`
  - has identity `e`: `e.a = a.e = a`
  - has inverses a^{-1}: $a.a^{-1} = e$
- if commutative `a.b = b.a`
  - then forms an **abelian group**
- Examples in P.105

Cyclic Group

- define **exponentiation** as repeated application of operator
  - example: $a^3 = a.a.a$
- and let identity be: $e=a^0$
- a group is cyclic if every element is a power of some fixed element
  - ie $b = a^k$ for some a and every b in group
- a is said to be a generator of the group
- Example: positive numbers with addition

Ring

- a set of “numbers” with two operations (addition and multiplication) which are:
- an abelian group with addition operation
- multiplication:
  - has closure
  - is associative
  - distributive over addition: `a(b+c) = ab + ac`
- In essence, a ring is a set in which we can do addition, subtraction [a – b = a + (–b)], and multiplication without leaving the set.
- With respect to addition and multiplication, the set of all *n*-square matrices over the real numbers form a ring.

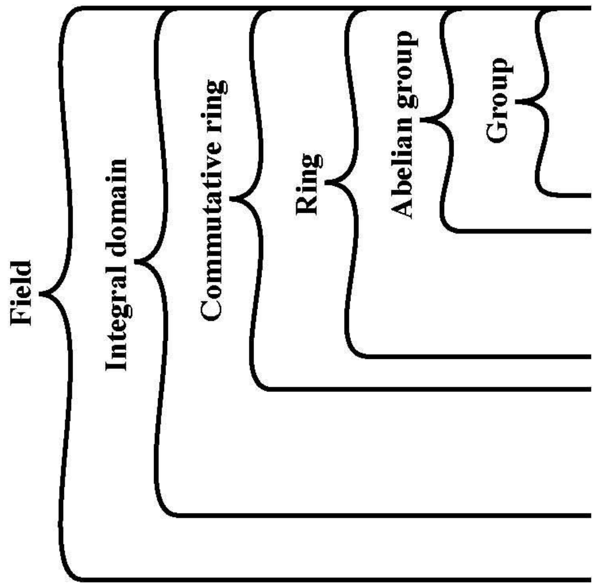
Ring

- if multiplication operation is commutative, it forms a **commutative ring**
- if multiplication operation has an identity element and no zero divisors (ab=0 means either a=0 or b=0), it forms an **integral domain**
- The set of Integers with usual + and x is an integral domain

Field

- a set of numbers with two operations:
  - Addition and multiplication
  - F is an integral domain
  - F has multiplicative reverse
    - For each a in F other than 0, there is an element b such that ab=ba=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 with the following rule: $a/b = a\,(b^{-1})$
- Examples of fields: rational numbers, real numbers, complex numbers. Integers are NOT a field.

Definitions



(A1) Closure under addition:	If a and b belong to S, then $a + b$ is also in S
(A2) Associativity of addition:	$a + (b + c) = (a + b) + c$ for all a, b, c in S
(A3) Additive identity:	There is an element 0 in R such that $a + 0 = 0 + a = a$ for all a in S
(A4) Additive inverse:	For each a in S there is an element $-a$ in S such that $a + (-a) = (-a) + a = 0$
(A5) Commutativity of addition:	$a + b = b + a$ for all a, b in S
(M1) Closure under multiplication:	If a and b belong to S, then ab is also in S
(M2) Associativity of multiplication:	$a(bc) = (ab)c$ for all a, b, c in S
(M3) Distributive laws:	$a(b + c) = ab + ac$ for all a, b, c in S $(a + b)c = ac + bc$ for all a, b, c in S
(M4) Commutativity of multiplication:	$ab = ba$ for all a, b in S
(M5) Multiplicative identity:	There is an element 1 in S such that $a1 = 1a = a$ for all a in S
(M6) No zero divisors:	If a, b in S and $ab = 0$, then either $a = 0$ or $b = 0$
(M7) Multiplicative inverse:	If a belongs to S and a 0, there is an element a^{-1} in S such that $aa^{-1} = a^{-1}a = 1$

Figure 4.1 Group, Ring, and Field

Modular Arithmetic

- define **modulo operator** `a mod n` to be remainder when a is divided by n
  - e.g. 1 = 7 mod 3 ; 4 = 9 mod 5
- use the term **congruence** for: $a \equiv b \pmod{n}$
  - when divided by *n*, a & b have same remainder
  - eg. $100 \equiv 34 \pmod{11}$
- b is called the **residue** of a mod n
  - since with integers can always write: $a = qn + b$
- usually have $0 <= b <= n-1$

  $-12 \bmod 7 = -5 \bmod 7 = 2 \bmod 7 = 9 \bmod 7$

Modulo 7 Example

```
...
-21 -20 -19 -18 -17 -16 -15
-14 -13 -12 -11 -10  -9  -8
 -7  -6  -5  -4  -3  -2  -1
  0   1   2   3   4   5   6
  7   8   9  10  11  12  13
 14  15  16  17  18  19  20
 21  22  23  24  25  26  27
 28  29  30  31  32  33  34
...
```

all numbers in a column are equivalent (have same remainder) and are called a **residue class**

Divisors

- say a non-zero number `b` **divides** `a` if for some `m` have `a=mb` (`a,b,m` all integers)
  - 0 ≡ `a mod b`
- that is `b` divides into `a` with no remainder
- denote this `b|a`
- and say that `b` is a **divisor** of `a`
- eg. all of 1,2,3,4,6,8,12,24 divide 24

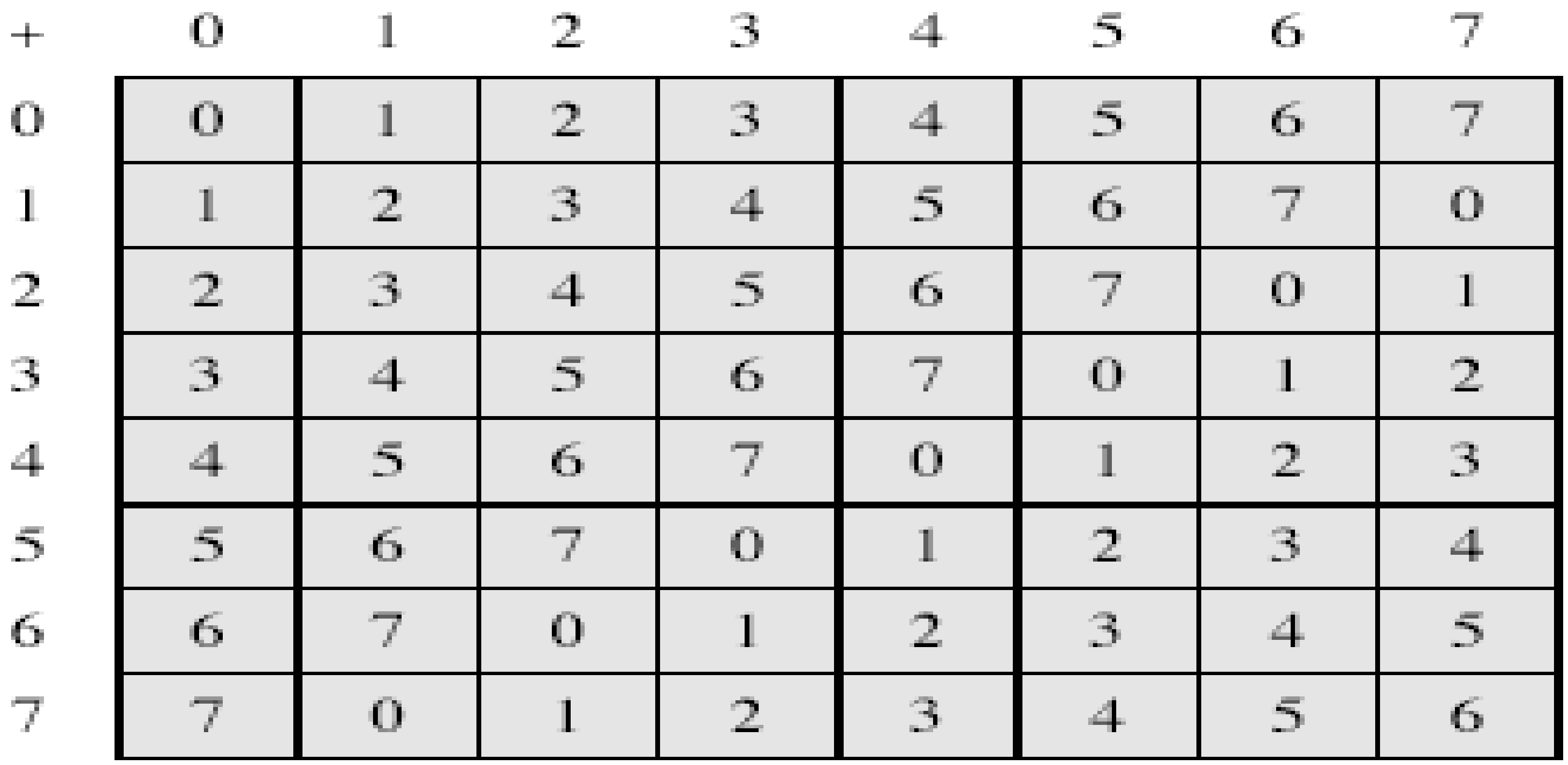
Modular Arithmetic Operations

- has a finite number of values, and loops back from either end
- modular arithmetic
  - Can perform addition & multiplication
  - Do modulo to reduce the answer to the finite set
- can do reduction at any point, ie
  - `a+b mod n = a mod n + b mod n`

Modular Arithmetic

- can do modular arithmetic with any group of integers: $Z_n = \{0, 1, \ldots, n-1\}$
- form a commutative ring for addition
- with an additive identity (Table 4.2)
- some additional properties
  - if $(a+b) \equiv (a+c) \bmod n$ then $b \equiv c \bmod n$
  - but $(ab) \equiv (ac) \bmod n$ then $b \equiv c \bmod n$ only if a is relatively prime to n

Modulo 8 Example



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

Greatest Common Divisor (GCD)

- a common problem in number theory
- GCD (a,b) of a and b is the largest number that divides both a and b
  - eg GCD(60,24) = 12
- often want **no common factors** (except 1) and hence numbers are **relatively prime**
  - eg GCD(8,15) = 1
  - hence 8 & 15 are relatively prime

Euclid's GCD Algorithm

- an efficient way to find the GCD(a,b)
- uses theorem that:
  - `GCD(a,b) = GCD(b, a mod b)`
- **Euclid's Algorithm** to compute GCD(a,b):

```
– A=a, B=b
– while B>0
    • R = A mod B
    • A = B, B = R
– return A
```

Example GCD(1970,1066)

```
1970 = 1 x 1066 + 904   gcd(1066, 904)
1066 = 1 x 904 + 162    gcd(904, 162)
904 = 5 x 162 + 94            gcd(162, 94)
162 = 1 x 94 + 68             gcd(94, 68)
94 = 1 x 68 + 26        gcd(68, 26)
68 = 2 x 26 + 16        gcd(26, 16)
26 = 1 x 16 + 10        gcd(16, 10)
16 = 1 x 10 + 6         gcd(10, 6)
10 = 1 x 6 + 4                gcd(6, 4)
6 = 1 x 4 + 2                 gcd(4, 2)
4 = 2 x 2 + 0                 gcd(2, 0)
```

- Compute successive instances of GCD(a,b) = GCD(b,a mod b).
- Note this MUST always terminate since will eventually get a mod b = 0 (ie no remainder left).

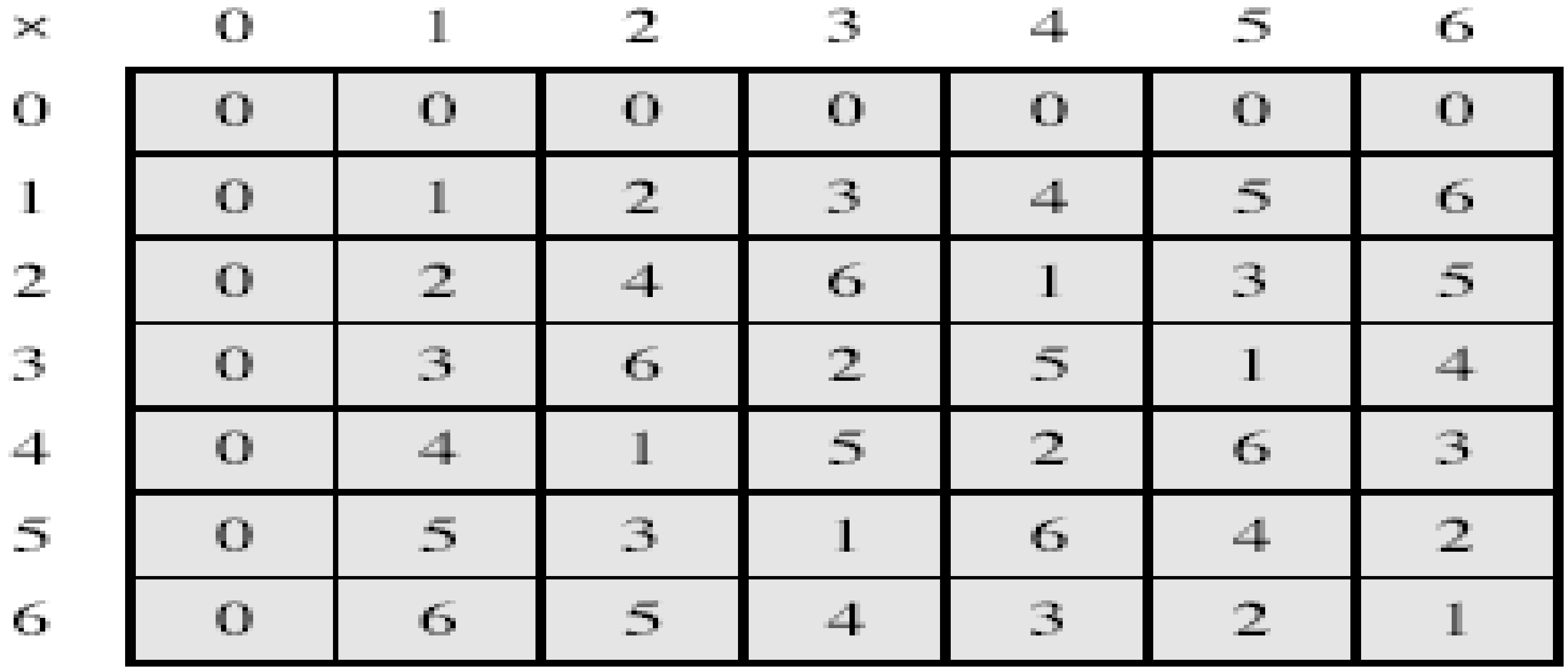
Galois Fields

- finite fields play a key role in many cryptography algorithms
- can show number of elements in any finite field **must** be a power of a prime number p^n
- known as Galois fields
- denoted $GF(p^n)$
- in particular often use the fields:
  - $GF(p)$
  - $GF(2^n)$

Galois Fields GF(p)

- GF(p) is the set of integers {0,1, … , p-1} with arithmetic operations modulo prime p
- these form a finite field
  - since have multiplicative inverses
- hence arithmetic is “well-behaved” and can do addition, subtraction, multiplication, and division without leaving the field GF(p)
  - Division depends on the existence of multiplicative inverses. Why p has to be prime?

Example GF(7)



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

(b) Multiplication modulo 7

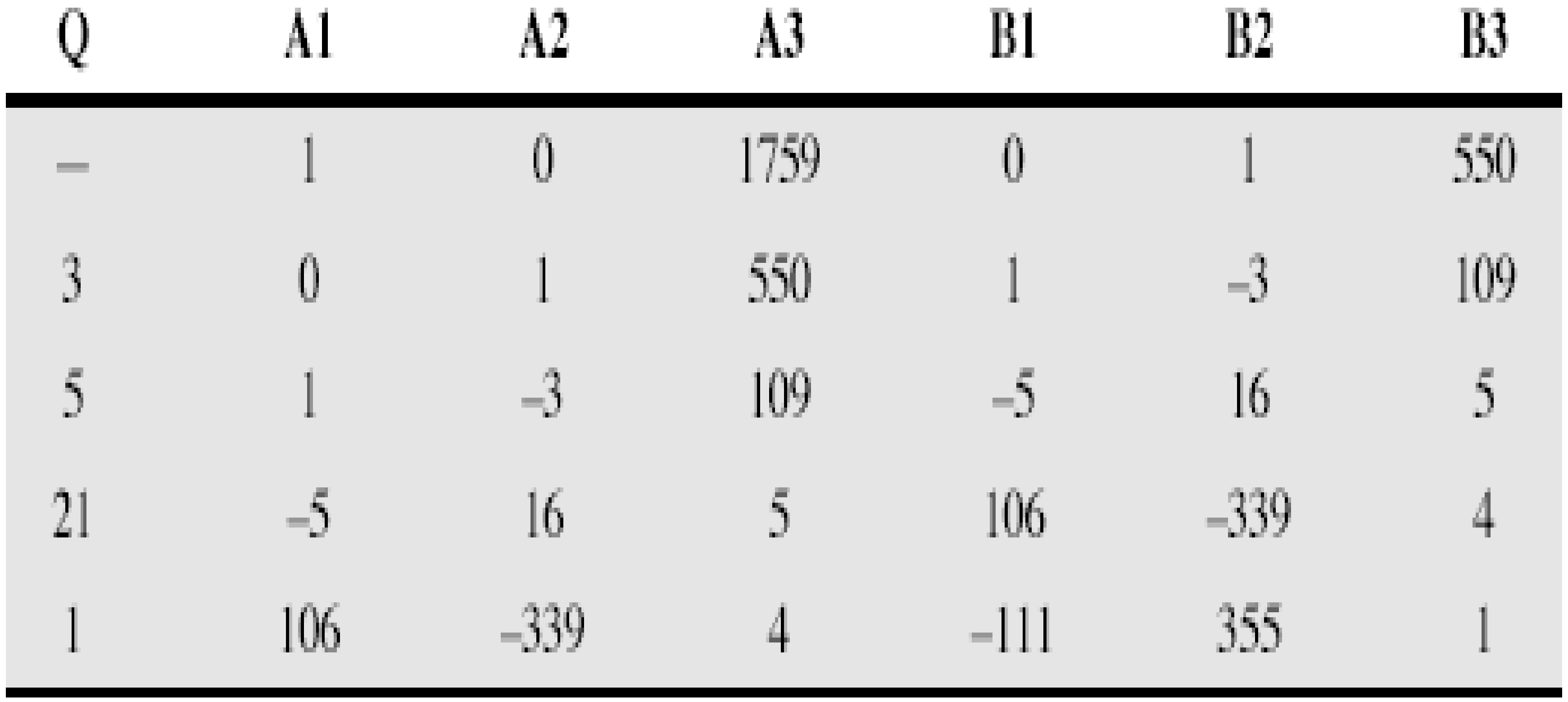
Example: 3/2=5
GP(6) does not exist

Finding Inverses

- Finding inverses for large P is a problem
- can extend Euclid’s algorithm:

```
EXTENDED EUCLID($m$, $b$)
1. (A1, A2, A3)=(1, 0, $m$);
   (B1, B2, B3)=(0, 1, $b$)
2. if B3 = 0
   return A3 = gcd($m$, $b$); no inverse
3. if B3 = 1
   return B3 = gcd($m$, $b$); B2 = $b^{-1}$ mod $m$
4. Q = A3 div B3
5. (T1, T2, T3)=(A1 – Q B1, A2 – Q B2, A3 – Q B3)
6. (A1, A2, A3)=(B1, B2, B3)
7. (B1, B2, B3)=(T1, T2, T3)
8. goto 2
```

Inverse of 550 in GF(1759)



Q	A1	A2	A3	B1	B2	B3
—	1	0	1759	0	1	550
3	0	1	550	1	–3	109
5	1	–3	109	–5	16	5
21	–5	16	5	106	–339	4
1	106	–339	4	–111	355	1

Prove correctness

Polynomial Arithmetic

- can compute using polynomials
- se

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$$

  –
  – poly arithmetic with coefficients mod p
  – poly arithmetic with coefficients mod p and polynomials mod another polynomial M(x)
- Motivation: use polynomials to model Shift and XOR operations

Ordinary Polynomial Arithmetic

- add or subtract corresponding coefficients
- multiply all terms by each other
- eg
  - let $f(x) = x^3 + x^2 + 2$ and $g(x) = x^2 - x + 1$

    $f(x) + g(x) = x^3 + 2x^2 - x + 3$

    $f(x) - g(x) = x^3 + x + 1$

    $f(x) \text{ x } g(x) = x^5 + 3x^2 - 2x + 2$

Polynomial Arithmetic with Modulo Coefficients

- when computing value of each coefficient, modulo some value
- could be modulo any prime
- but we are most interested in mod 2
  - ie all coefficients are 0 or 1
  - eg. let $f(x) = x^3 + x^2$ and $g(x) = x^2 + x + 1$

  $f(x) + g(x) = x^3 + x + 1$

  $f(x) \text{ x } g(x) = x^5 + x^2$

Modular Polynomial Arithmetic

- Given any polynomials f,g, can write in the form:
  - $f(x) = q(x)\, g(x) + r(x)$
  - can interpret $r(x)$ as being a remainder
  - $r(x) = f(x) \bmod g(x)$
- if have no remainder say $g(x)$ divides $f(x)$
- if $g(x)$ has no divisors other than itself & 1 say it is **irreducible** (or **prime**) polynomial
- Modular polynomial arithmetic modulo an irreducible polynomial forms a field
  - Check the definition of a field

Polynomial GCD

- can find greatest common divisor for polys
- GCD: the one with the greatest degree
  - $c(x)$ = GCD($a(x)$, $b(x)$) if $c(x)$ is the poly of greatest degree which divides both $a(x)$, $b(x)$
  - can adapt Euclid’s Algorithm to find it:
  - EUCLID[$a(x)$, $b(x)$]

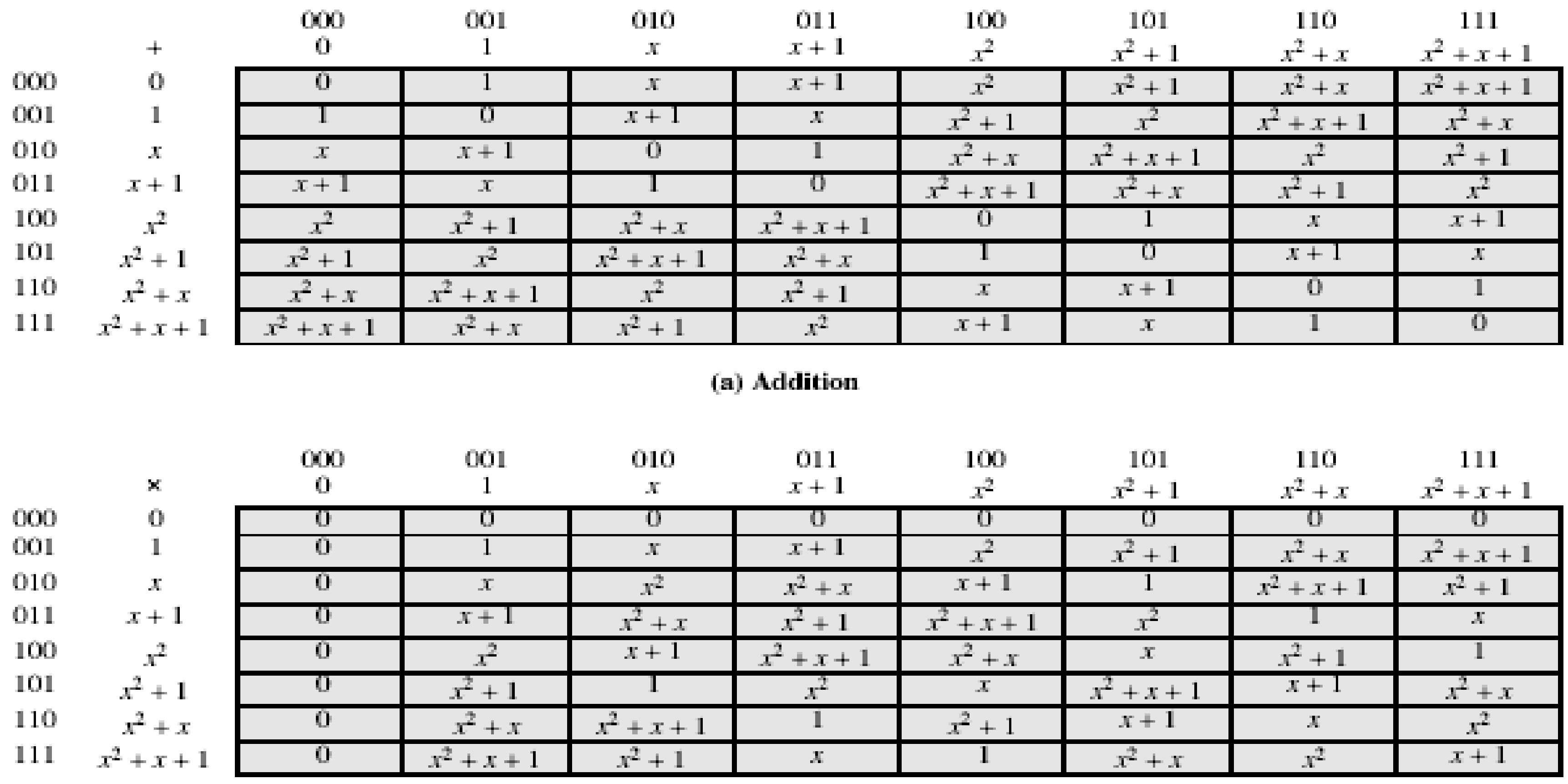
```
1. A(x) = a(x); B(x) = b(x)
2. 2. if B(x) = 0 return A(x) = gcd[a(x), b(x)]
3. R(x) = A(x) mod B(x)
4. A(x) ¨ B(x)
5. B(x) ¨ R(x)
6. goto 2
```

Modular Polynomial Arithmetic

- can compute in field GF(2^n)
  - polynomials with coefficients modulo 2
  - whose degree is less than n
  - Coefficients always modulo 2 in an operation
  - hence must modulo an irreducible polynomial of degree n (for multiplication only)
- form a finite field
- can always find an inverse
  - can extend Euclid’s Inverse algorithm to find

Example GF(2^3)

Table 4.6 Polynomial Arithmetic Modulo ($x^3 + x + 1$)



		000	001	010	011	100	101	110	111
	+	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
000	0	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
001	1	1	0	$x+1$	x	x^2+1	x^2	x^2+x+1	x^2+x
010	x	x	$x+1$	0	1	x^2+x	x^2+x+1	x^2	x^2+1
011	$x+1$	$x+1$	x	1	0	x^2+x+1	x^2+x	x^2+1	x^2
100	x^2	x^2	x^2+1	x^2+x	x^2+x+1	0	1	x	$x+1$
101	x^2+1	x^2+1	x^2	x^2+x+1	x^2+x	1	0	$x+1$	x
110	x^2+x	x^2+x	x^2+x+1	x^2	x^2+1	x	$x+1$	0	1
111	x^2+x+1	x^2+x+1	x^2+x	x^2+1	x^2	$x+1$	x	1	0

(a) Addition

		000	001	010	011	100	101	110	111
	×	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
000	0	0	0	0	0	0	0	0	0
001	1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
010	x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
011	$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
100	x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
101	x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
110	x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
111	x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

(b) Multiplication

Computational Considerations

- since coefficients are 0 or 1, can represent any such polynomial as a bit string
- addition becomes XOR of these bit strings
- multiplication is shift & XOR
  - Example in P.133
- modulo reduction done by repeatedly substituting highest power with remainder of irreducible poly (also shift & XOR)

Summary

- have considered:
  - concept of groups, rings, fields
  - modular arithmetic with integers
  - Euclid’s algorithm for GCD
  - finite fields GF(p)
  - polynomial arithmetic in general and in GF(2^n)

Chapter 5
Advanced Encryption Standard

Origins

- clear a replacement for DES was needed
  - have theoretical attacks that can break it
  - have demonstrated exhaustive key search attacks
- can use Triple-DES – pretty safe
  - but slow, small blocks
- issued call for ciphers in `97
- 15 candidates accepted in Jun 98
- 5 were short-listed in Aug-99
- AES selected in Oct-2000
- issued as FIPS PUB 197 standard in Nov-2001

AES Requirements

- private key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- stronger & faster than Triple-DES
- active life of 20-30 years (+ archival use)
- provide full specification & design details
- both C & Java implementations
- NIST have released all submissions & unclassified analyses

AES Evaluation Criteria

- initial criteria (15 to 5):
  - security – effort to practically cryptanalyse
  - cost – computational, high-speed applications
  - algorithm & implementation characteristics
    - Flexibility, simplicity, maintainability
- final criteria
  - general security
  - software & hardware implementation ease
  - implementation attacks
  - flexibility (in changing en/decrypt, keying, #rounds, other factors)

AES Shortlist

- after testing and evaluation, shortlist in Aug-99:
  - MARS (IBM) - complex, fast, high security margin
  - RC6 (USA) - v. simple, v. fast, low security margin
  - Rijndael (Belgium) - clean, fast, good security margin
  - Serpent (Euro) - slow, clean, v. high security margin
  - Twofish (USA) - complex, v. fast, high security margin
- then subject to further analysis & comment
- All were thought to be good – came down to best balance of attributes to meet criteria.
- Note mix of commercial (MARS, RC6, Twofish) verses academic (Rijndael, Serpent) proposals

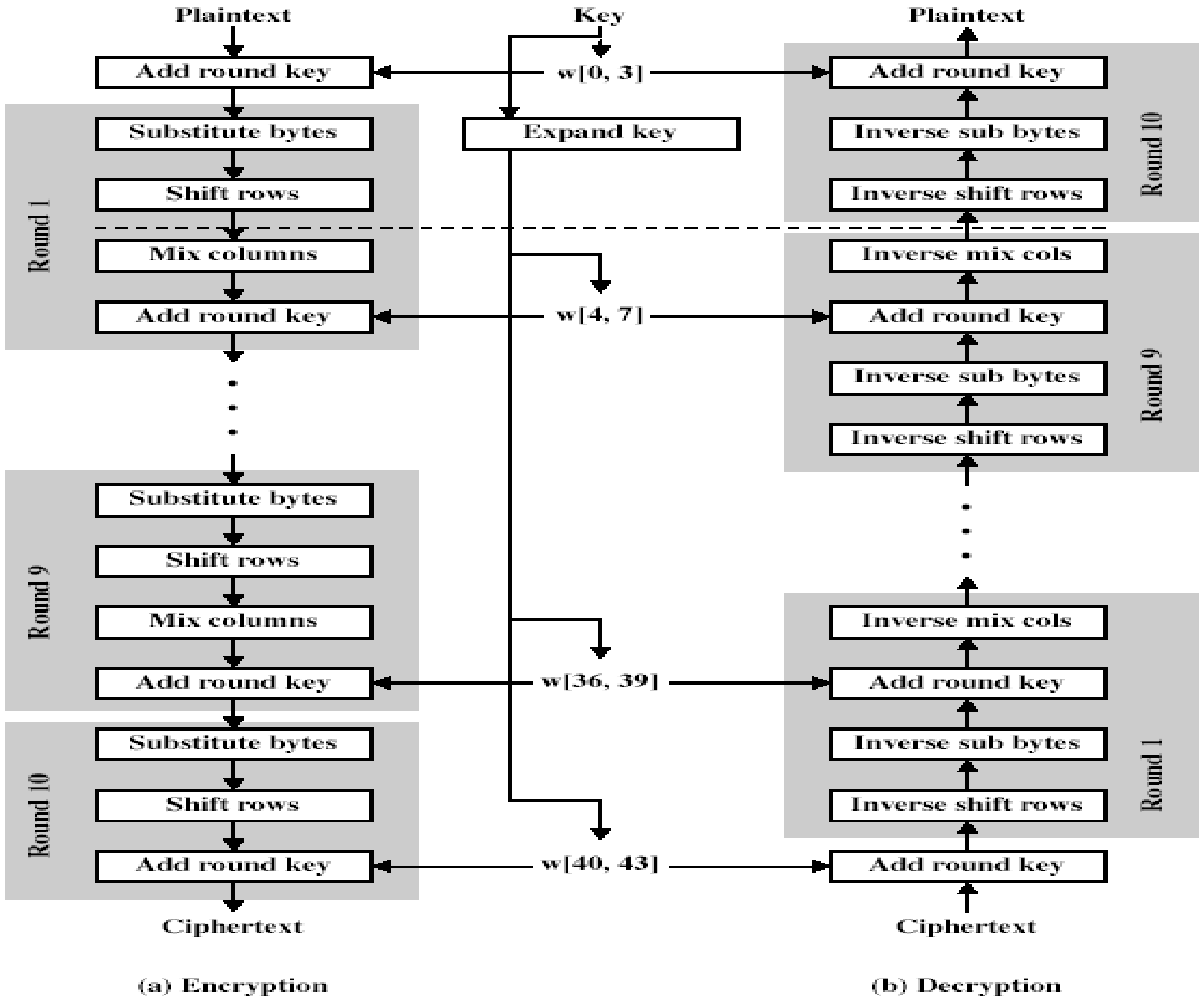
The AES Cipher

- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **feistel** cipher
  - treats data in 4 groups of 4 bytes
  - operates an entire block in every round
  - rather than feistel (operate on halves at a time)
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

AES

- processes data as 4 groups of 4 bytes (state)
- has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multiply of groups)
  - add round key (XOR state with key material)
- initial XOR key material & incomplete last round
- all operations can be combined into XOR and table lookups - hence very fast & efficient

Rijndael


Plaintext
Key
Plaintext
Add round key
w[0, 3]
Add round key
Substitute bytes
Expand key
Inverse sub bytes
Shift rows
Inverse shift rows
Round 1
Round 10
Mix columns
Inverse mix cols
Add round key
w[4, 7]
Add round key
Inverse sub bytes
Inverse shift rows
Round 9
Substitute bytes
Shift rows
Mix columns
Round 9
Add round key
w[36, 39]
Inverse mix cols
Add round key
Inverse sub bytes
Inverse shift rows
Round 1
Substitute bytes
Shift rows
Round 10
Add round key
w[40, 43]
Add round key
Ciphertext
Ciphertext
(a) Encryption
(b) Decryption

Byte Substitution

- a simple substitution of each byte
- uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- each byte of state is replaced by byte in row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by row 9 col 5 byte
  - which is the value {2A}
- S-box is constructed using a defined transformation of the values in GF(2^8)
- designed to be resistant to all known attacks

Shift Rows

- a circular byte shift in each row
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- decrypt does shifts to right
- since state is processed by columns, this step permutes bytes between the columns

Mix Columns

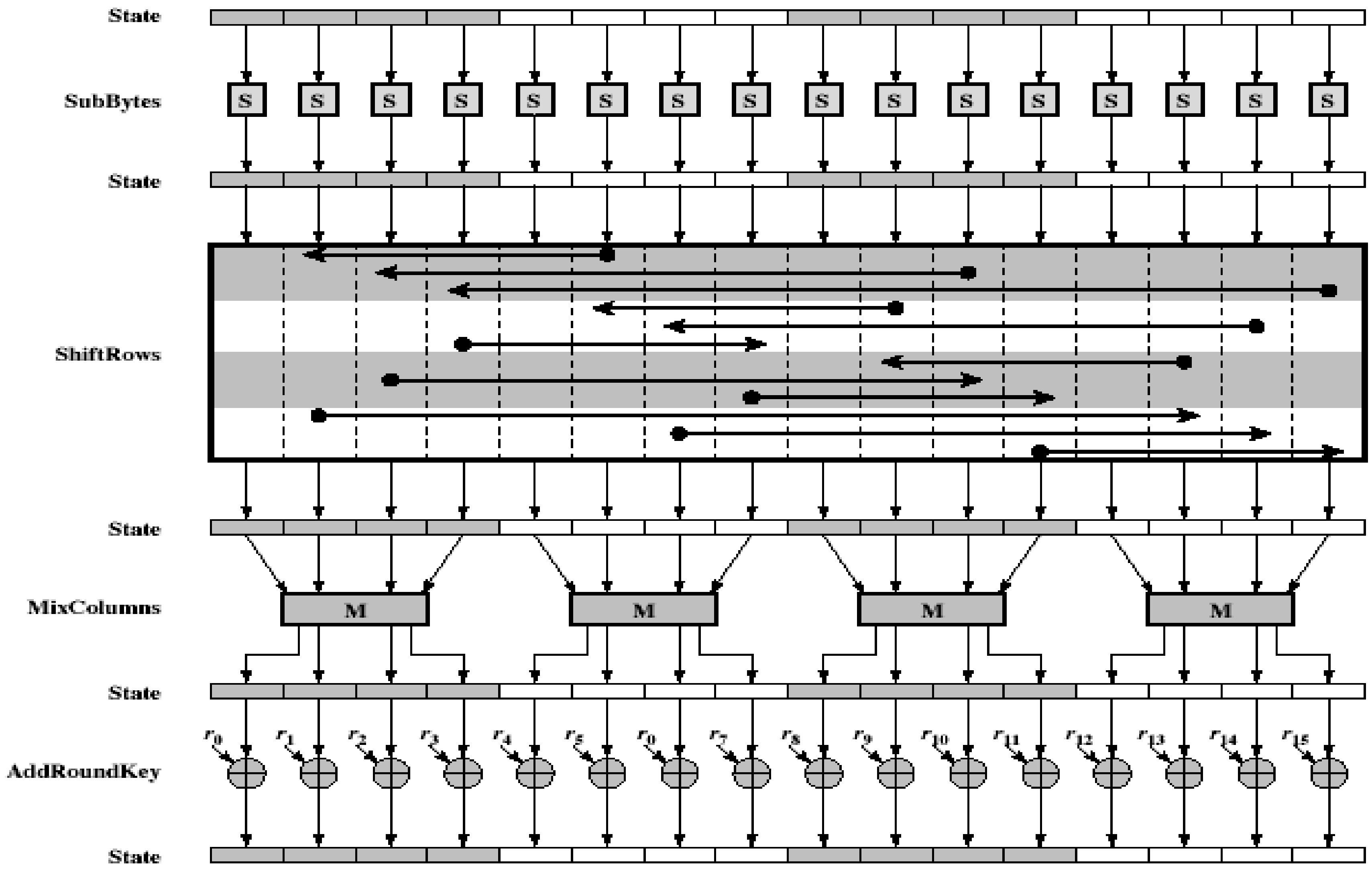
- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in GF(2^8) using prime poly m(x) =$x^8+x^4+x^3+x+1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Add Round Key

- XOR state with 128-bits of the round key
- again processed by column (though effectively a series of byte operations)
- inverse for decryption is identical since XOR is own inverse, just with correct round key
- designed to be as simple as possible

AES Round


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

AES Key Expansion

- takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
- start by copying key into first 4 words
- then loop creating words that depend on values in previous & 4 places back
  - in 3 of 4 cases just XOR these together
  - every 4th has S-box + rotate + XOR constant of previous before XOR together
- designed to resist known attacks

AES Decryption

- AES decryption is not identical to encryption since steps done in reverse
- but can define an equivalent inverse cipher with steps as for encryption
  - but using inverses of each step
  - with a different key schedule
- works since result is unchanged when
  - swap byte substitution & shift rows
  - swap mix columns & add (tweaked) round key

Implementation Aspects

- can efficiently implement on 8-bit CPU
  - byte substitution works on bytes using a table of 256 entries
  - shift rows is simple byte shifting
  - add round key works on byte XORs
  - mix columns requires matrix multiply in GF(2^8) which works on byte values, can be simplified to use a table lookup

Implementation Aspects

- can efficiently implement on 32-bit CPU
  - redefine steps to use 32-bit words
  - can pre-compute 4 tables of 256-words
  - then each column in each round can be computed using 4 table lookups + 4 XORs
  - at a cost of 16Kb to store tables
- designers believe this very efficient implementation was a key factor in its selection as the AES cipher

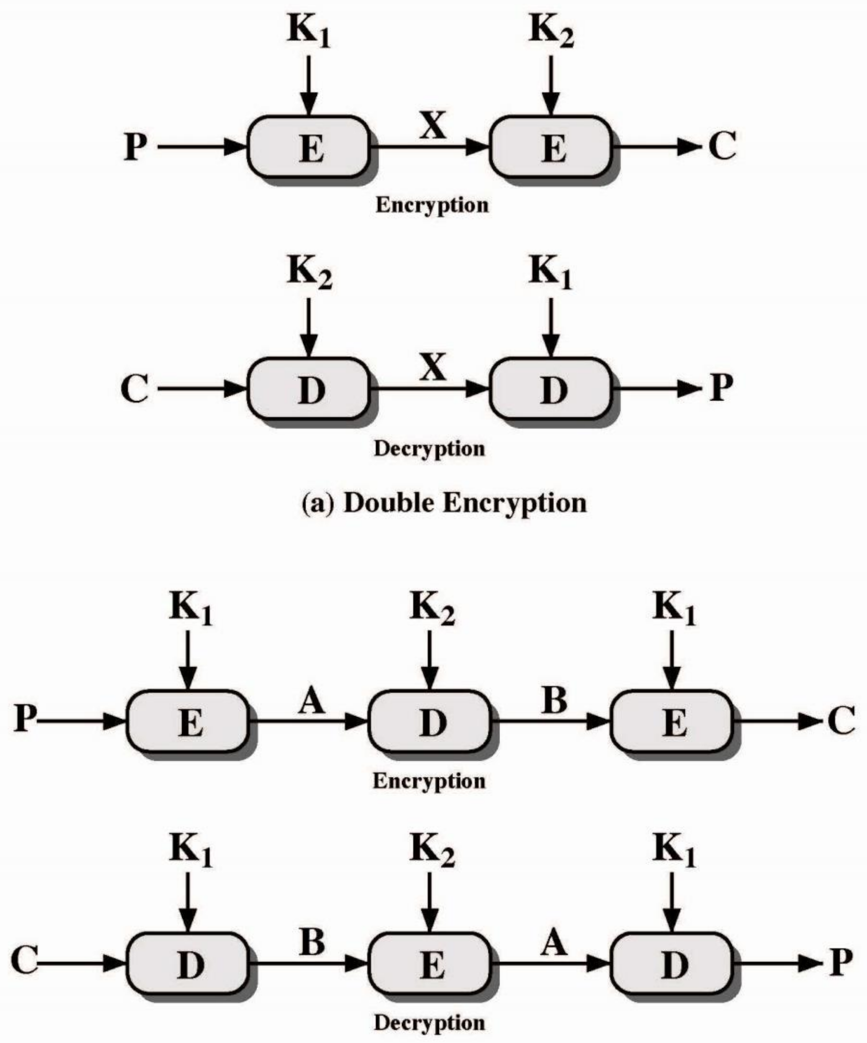
Summary

- have considered:
  - – the AES selection process
  - – the details of Rijndael – the AES cipher
  - – looked at the steps in each round
  - – the key expansion
  - – implementation aspects

Chapter 6 – Contemporary Symmetric Ciphers

Triple DES

- A replacement for DES was needed
  - theoretical attacks that can break it
  - demonstrated exhaustive key search attacks
- AES is a new cipher alternative
- Before AES alternative
  - use multiple encryptions with DES
- Triple-DES is the chosen form



(a) Double Encryption

(b) Triple Encryption

Figure 6.1 Multiple Encryption

Why Triple-DES?

- why not Double-DES?
  - NOT same as some other single-DES use, but have
- meet-in-the-middle attack
  - works whenever use a cipher twice
  - since $X = E_{K1}[P] = D_{K2}[C]$
  - attack by encrypting P with all keys and store
  - then decrypt C with keys and match X value
  - can show takes $O(2^{56})$ steps

Triple-DES with Two-Keys

- hence must use 3 encryptions
  - would seem to need 3 distinct keys
  - Key of 56 X 3 = 168 bits seems too long
- but can use 2 keys with E-D-E sequence
  - $C = E_{K1}[D_{K2}[E_{K1}[P]]]$
  - No cryptographic significance to the use of D in the second step
- standardized in ANSI X9.17 & ISO8732
- no current known practical attacks
  - some are now adopting Triple-DES with three keys for greater security

Triple-DES with Three-Keys

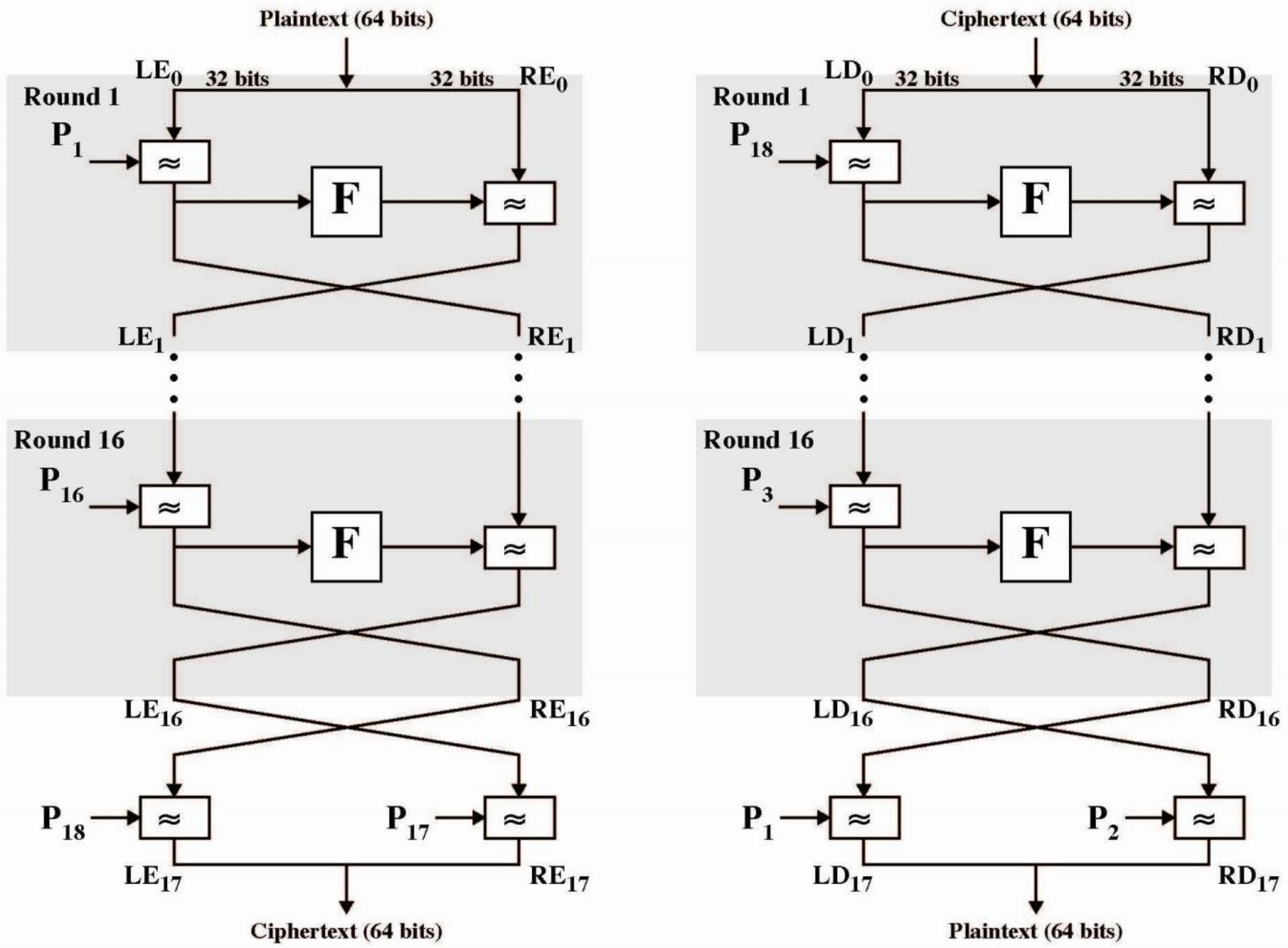
- although are no practical attacks on two-key Triple-DES have some indications
- can use Triple-DES with Three-Keys to avoid even these
  - $C = E_{K3}[D_{K2}[E_{K1}[P]]]$
- has been adopted by some Internet applications

Blowfish

- a symmetric block cipher designed by Bruce Schneier in 1993/94
- characteristics
  - fast implementation on 32-bit CPUs, 18 clock cycles per byte
  - compact in use of memory, less than 5KB
  - simple structure for analysis/implementation
  - variable security by varying key size
    - Allows tuning for speed/security tradeoff

Blowfish Key Schedule

- uses a 32 to 448 bit key
- used to generate
  - 18 32-bit subkeys stored in P-array: P1 to P18
  - S-boxes stored in $S_{i,j}$,
    - `i=1..4`
    - `j=0..255`



(a) Encryption

(b) Decryption

Figure 6.3 Blowfish Encryption and Decryption

Blowfish Encryption

- uses two primitives: addition & XOR
- data is divided into two 32-bit halves L_0 & R_0

```
for $i$ = 1 to 16 do
    $R_i = L_{i-1}$ XOR $P_i$;
    $L_i = F[R_i]$ XOR $R_{i-1}$;
$L_{17} = R_{16}$ XOR $P_{18}$;
$R_{17} = L_{16}$ XOR $i_{17}$;
```

- where

$F[a,b,c,d] = ((S_{1,a} + S_{2,b})$ XOR $S_{3,c}) + S_{4,a}$

Break 32-bit R_i into (a,b,c,d)

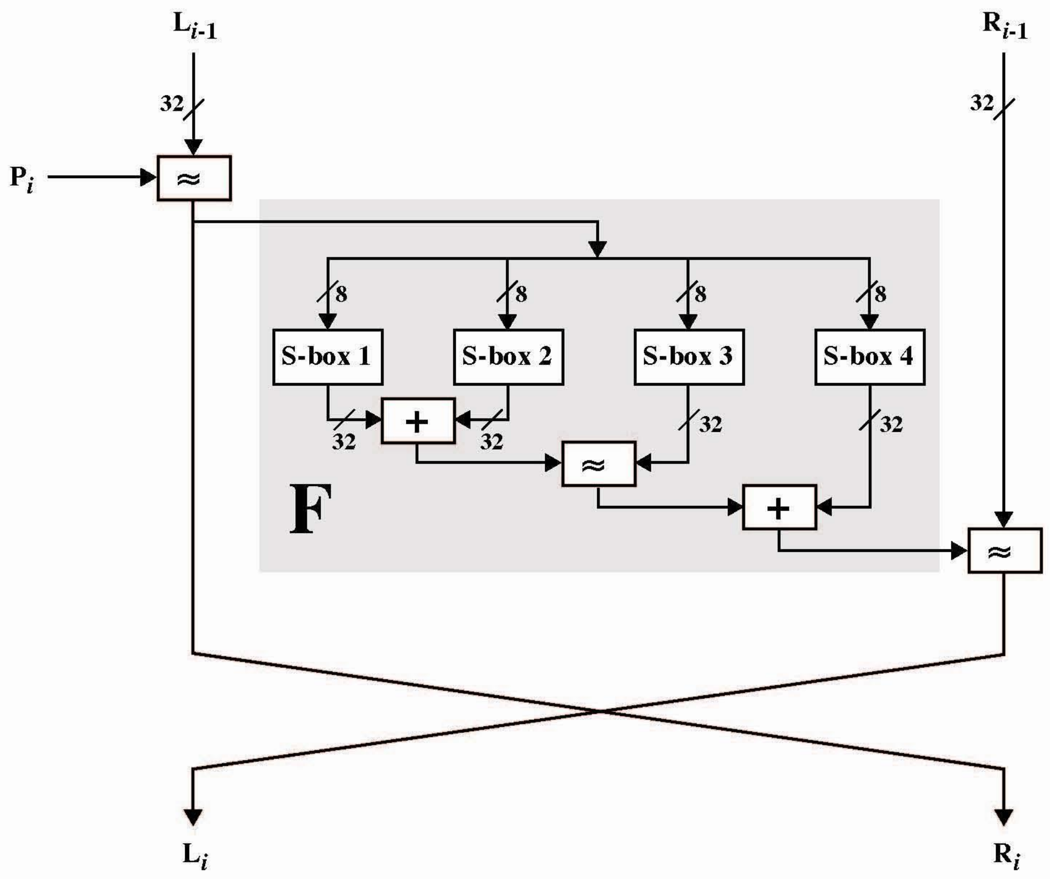


Figure 6.4 Detail of Single Blowfish Round

Discussion

- provided key is large enough, brute-force key search is not practical, especially given the high key schedule cost
- key dependent S-boxes and subkeys make analysis very difficult
  - Very few cryptoanalysis results on blowfish
- changing both halves in each round increases security
  - Some study shows improved avalanche effects

RC5

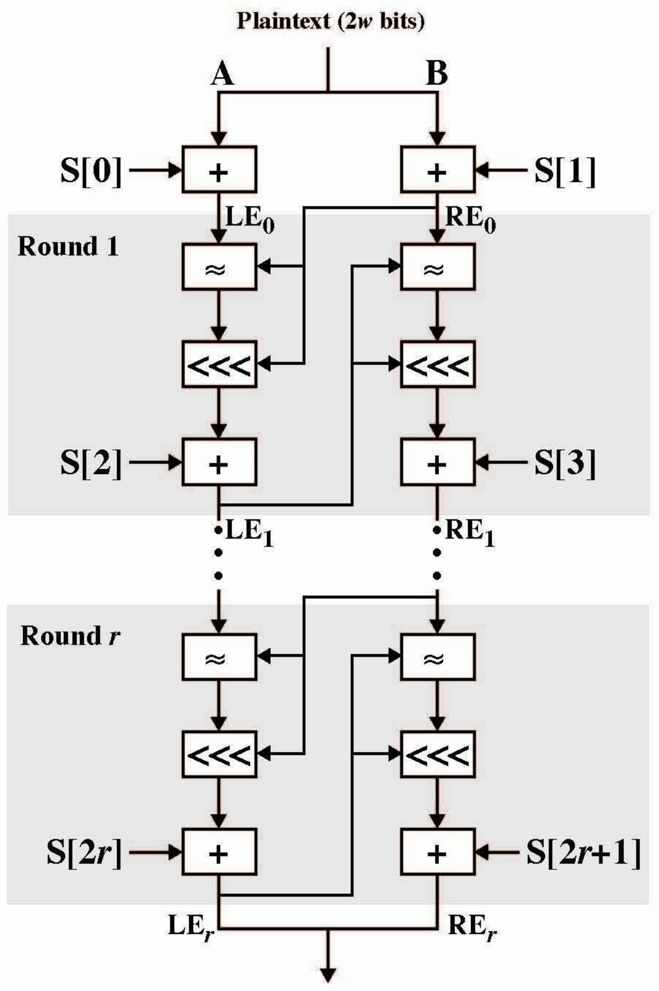
- can vary key size / input data size / #rounds
- very clean and simple design
- easy implementation on various CPUs
- yet still regarded as secure
  - Vary parameters to achieve tradeoffs

RC5 Ciphers

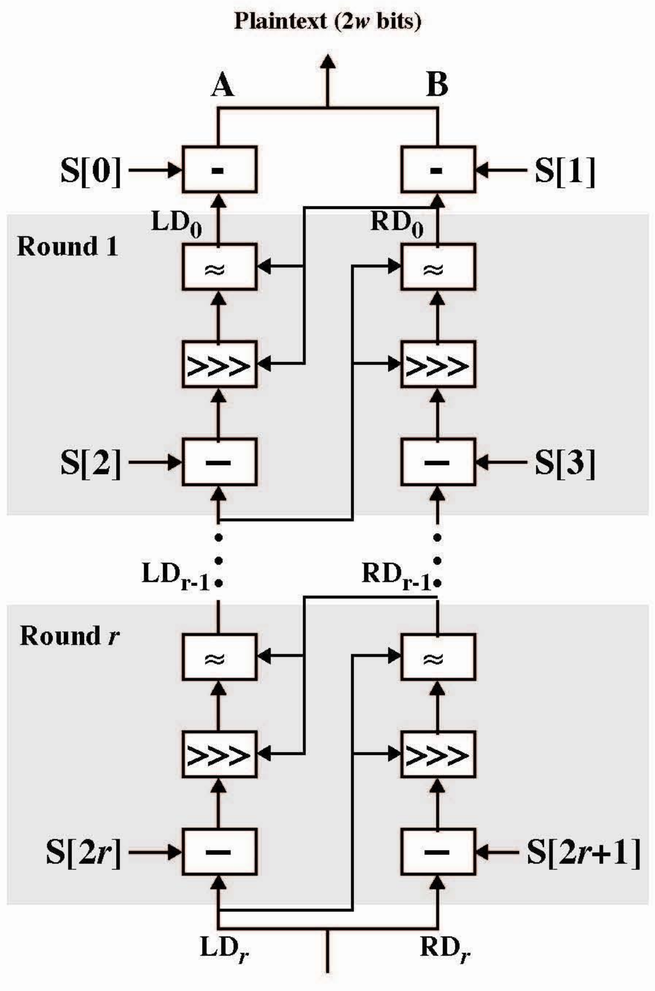
- RC5 is a family of ciphers RC5-w/r/b
  - w = word size in bits (16/32/64) data=2w
  - r = number of rounds (0..255)
  - b = number of bytes in key (0..255)
- nominal version is RC5-32/12/16
  - ie 32-bit words so encrypts 64-bit data blocks
  - using 12 rounds
  - with 16 bytes (128-bit) secret key

RC5 Key Expansion

- RC5 uses 2r+2 subkey words (w-bits)
  - Two subkeys for each round
  - 2 subkeys for additional operations
- subkeys are stored in array `S[i]`, i=0..t-1
- Key expansion: fill in pseudo-random bits to the original key K
- Certain amount of *one-wayness*
  - Difficult to determine K from S



(a) Encryption



(b) Decryption

Figure 6.6 RC5 Encryption and Decryption

RC5 Encryption

- split input into two halves A & B

```
$L_0$ = $A$ + S[0];
$R_0$ = $B$ + S[1];
for $i$ = 1 to $r$ do
    $L_i$ = (($L_{i-1}$ XOR $R_{i-1}$) <<< $R_{i-1}$) + S[2 x $i$];
    $R_i$ = (($R_{i-1}$ XOR $L_i$) <<< $L_i$) + S[2 x $i$ + 1];
```

- each round is like 2 DES rounds
- note rotation is main source of non-linearity
- need reasonable number of rounds (eg 12-16)
- Striking features: simplicity, data-dependent rotations

RC5 Modes

- RFC2040 defines 4 modes used by RC5
  - RC5 Block Cipher, is ECB mode
  - RC5-CBC, input length is a multiples of 2w
  - RC5-CBC-PAD, any length CBC with padding
    - Output can be longer than input
  - RC5-CTS, CBC with padding
    - Output has same length than input

Block Cipher Characteristics

- features seen in modern block ciphers are:
  - variable key length / block size / no rounds
  - mixed operators
    - data/key dependent rotation
    - key dependent S-boxes
  - more complex key scheduling
    - Lengthy key generation, simple encryption rounds
  - operation of full data in each round

Stream Ciphers

- process the message bit by bit (as a stream)
- typically have a (pseudo) random **key stream**
- combined (XOR) with plaintext bit by bit
- randomness of **key stream** completely destroys any statistically properties in the message
  - `C`$_i$ `= M`$_i$ `XOR StreamKey`$_i$
- what could be simpler!!!!
- but must never reuse key stream
  - otherwise can remove effect and recover messages

Block/Stream Ciphers

- Stream ciphers
  - For applications that require encryt/decryt of a stream of data
  - Examples: data communication channel, brower/web link
- Block ciphers
  - For applications dealing with blocks of data
  - Examples: file transfer, e-mail, database
- Either type can be used in virtually any application

Stream Cipher Properties

- some design considerations are:
  - long period with no repetitions
  - statistically random
  - Highly nonlinear correlation

RC4

- variable key size, byte-oriented stream cipher
- widely used (web SSL/TLS between browser and server, wireless WEP)
- key forms random permutation of a 8-bit string
- uses that permutation to scramble input info processed a byte at a time

RC4 Security

- claimed secure against known attacks
  - have some analyses, none practical
- result is very non-linear
- since RC4 is a stream cipher, must **never reuse a key**

Summary

- have considered:
  - some other modern symmetric block ciphers
  - Triple-DES
  - Blowfish
  - RC5
  - briefly introduced stream ciphers

Cryptography and Network Security

Third Edition

by William Stallings

Lecture slides by Lawrie Brown

Chapter 10 – Key Management; Other Public Key Cryptosystems

No Singhalese, whether man or woman, would venture out of the house without a bunch of keys in his hand, for without such a talisman he would fear that some devil might take advantage of his weak state to slip into his body.

—*The Golden Bough*, Sir James George Frazer

Key Management

- public-key encryption helps address key distribution problems
- distribution of public keys
- use of public-key encryption to distribute secret keys

Distribution of Public Keys

- can be considered as using one of:
  - Public announcement
  - Publicly available directory
  - Public-key authority
  - Public-key certificates

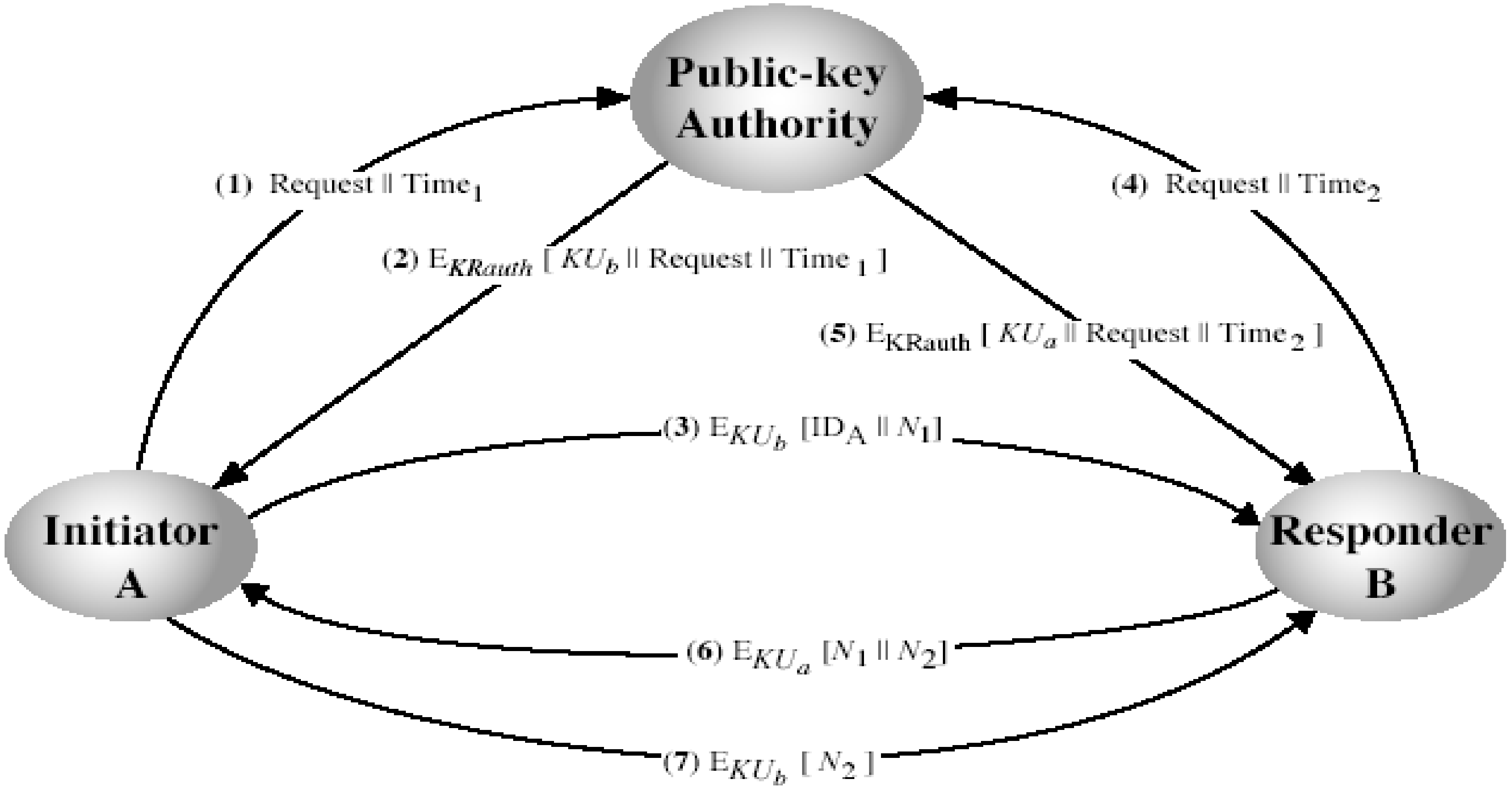
Public Announcement

- users distribute public keys to recipients or broadcast to community at large
  - eg. append PGP keys to email messages or post to news groups or email list
- major weakness is forgery
  - anyone can create a key claiming to be someone else and broadcast it
  - until forgery is discovered can masquerade as claimed user for authentication

Publicly Available Directory

- can obtain greater security by registering keys with a public directory
- directory must be trusted with properties:
  - contains {name, public-key} entries
  - participants register securely with directory
  - participants can replace key at any time
  - directory is periodically published
  - directory can be accessed electronically
- still vulnerable to tampering or forgery

Public-Key Authority


Public-key Authority
Initiator A
Responder B
(1) Request || Time$_1$
(2) E_{KRauth} [KU_b || Request || Time$_1$]
(3) E_{KU_b} [ID_A || N_1]
(4) Request || Time$_2$
(5) E_{KRauth} [KU_a || Request || Time$_2$]
(6) E_{KU_a} [N_1 || N_2]
(7) E_{KU_b} [N_2]

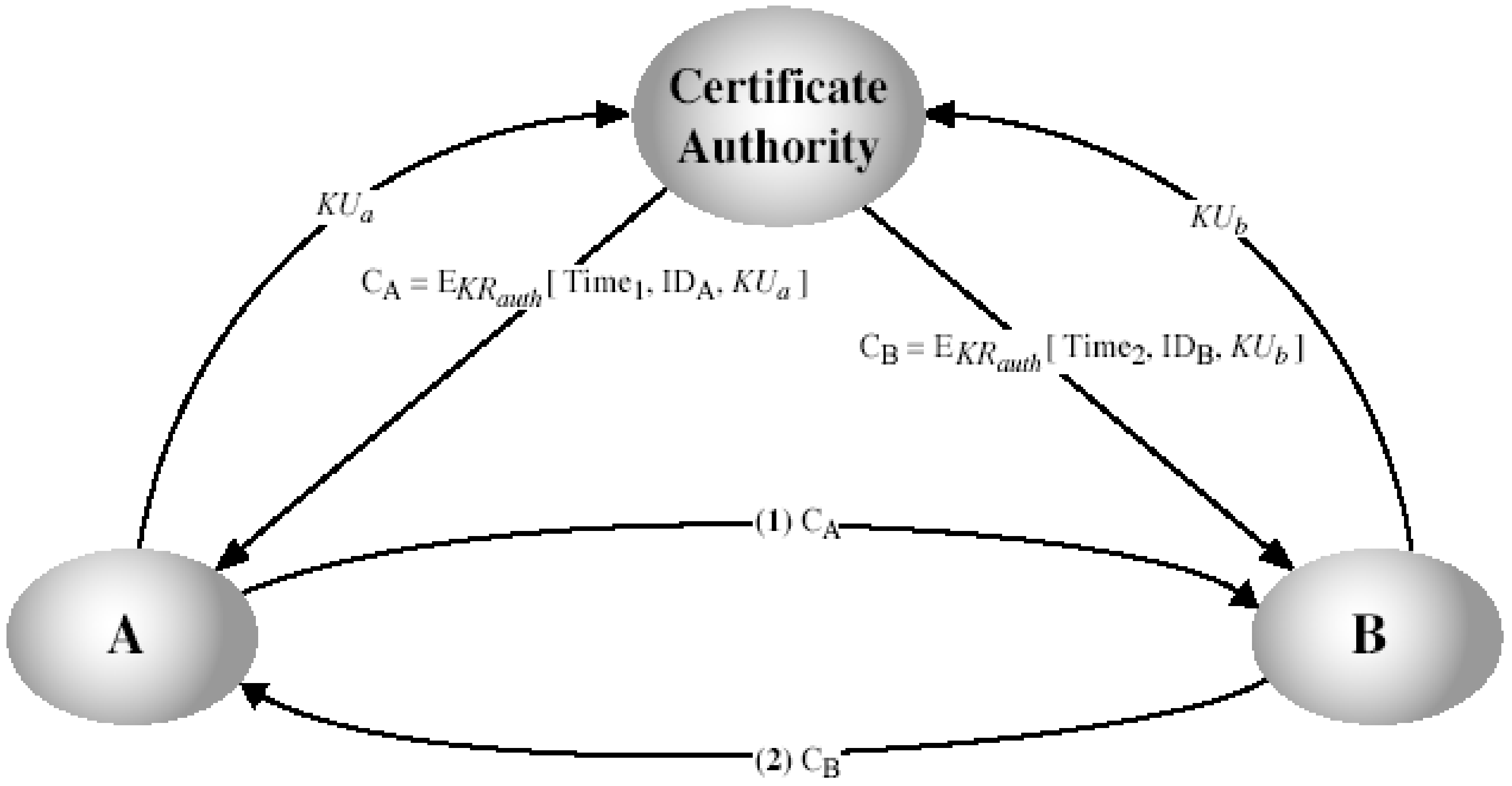
Public-Key Authority

- improve security by tightening control over distribution of keys from directory
- requires users to know public key for the directory
- then users interact with directory to obtain any desired public key securely
  - does require real-time access to directory when keys are needed

Public-Key Certificates

- The public-key authority could be a bottleneck in the system.
  - must appeal to the authority for the key of every other user
- certificates allow key exchange without real-time access to public-key authority
- a certificate binds **identity** to **public key**
- with all contents **signed** by a trusted Public-Key or Certificate Authority (CA)
  - Certifies the identity
  - Only the CA can make the certificates

Public-Key Certificates


Certificate Authority
KU_a
KU_b
$C_A = E_{KR_{auth}}[\text{Time}_1, ID_A, KU_a]$
$C_B = E_{KR_{auth}}[\text{Time}_2, ID_B, KU_b]$
(1) C_A
A
B
(2) C_B

Public-Key Distribution of Secret Keys

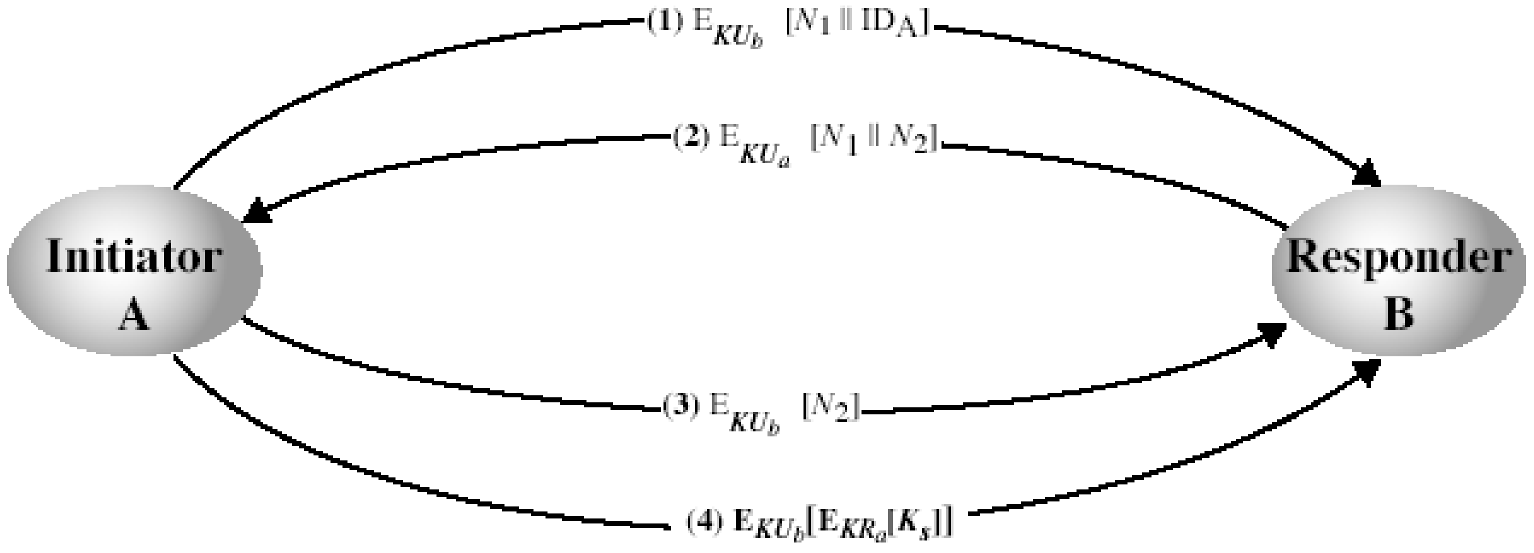
- public-key algorithms are slow
- so usually want to use private-key encryption to protect message contents
- hence need a session key
- have several alternatives for negotiating a suitable session using public-key

Simple Secret Key Distribution

- proposed by Merkle in 1979
  - A generates a new temporary public key pair
  - A sends B the public key and their identity
  - B generates a session key K sends it to A encrypted using the supplied public key
  - A decrypts the session key and both use
- problem is that an opponent can intercept and impersonate both halves of protocol
  - The scenario

Public-Key Distribution of Secret Keys

- First securely exchanged public-keys using a previous method


(1) E_{KU_b} [$N_1 \| ID_A$]
(2) E_{KU_a} [$N_1 \| N_2$]
Initiator A
Responder B
(3) E_{KU_b} [N_2]
(4) $E_{KU_b}[E_{KR_a}[K_s]]$

Diffie-Hellman Key Exchange

- first public-key type scheme proposed
  - For key distribution only
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
  - note: now know that James Ellis (UK CESG) secretly proposed the concept in 1970
- is a practical method for public exchange of a secret key
- used in a number of commercial products

Diffie-Hellman Key Exchange

- a public-key distribution scheme
  - cannot be used to exchange an arbitrary message
  - rather it can establish a common key
  - known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial) - easy
- security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

Diffie-Hellman Setup

- all users agree on global parameters:
  - large prime integer or polynomial q
  - α a primitive root mod q
- each user (eg. A) generates their key
  - chooses a secret key (number): $x_A < q$
  - compute their **public key**: $y_A = \alpha^{x_A} \bmod q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

- shared session key for users A & B is K:

  $K = y_A^{x_B} \bmod q$ (which **B** can compute)

  $K = y_B^{x_A} \bmod q$ (which **A** can compute)

  (example)
- K is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an x, must solve discrete log

Diffie-Hellman Example

- users Alice & Bob who wish to swap keys:
- agree on prime $q=353$ and $\alpha=3$
- select random secret keys:
  - A chooses $x_A=97$, B chooses $x_B=233$
- compute public keys:
  - $y_A=3^{97} \bmod 353 = 40$ (Alice)
  - $y_B=3^{233} \bmod 353 = 248$ (Bob)
- compute shared session key as:

$K_{AB}= y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)

$K_{AB}= y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)

Elliptic Curve Cryptography

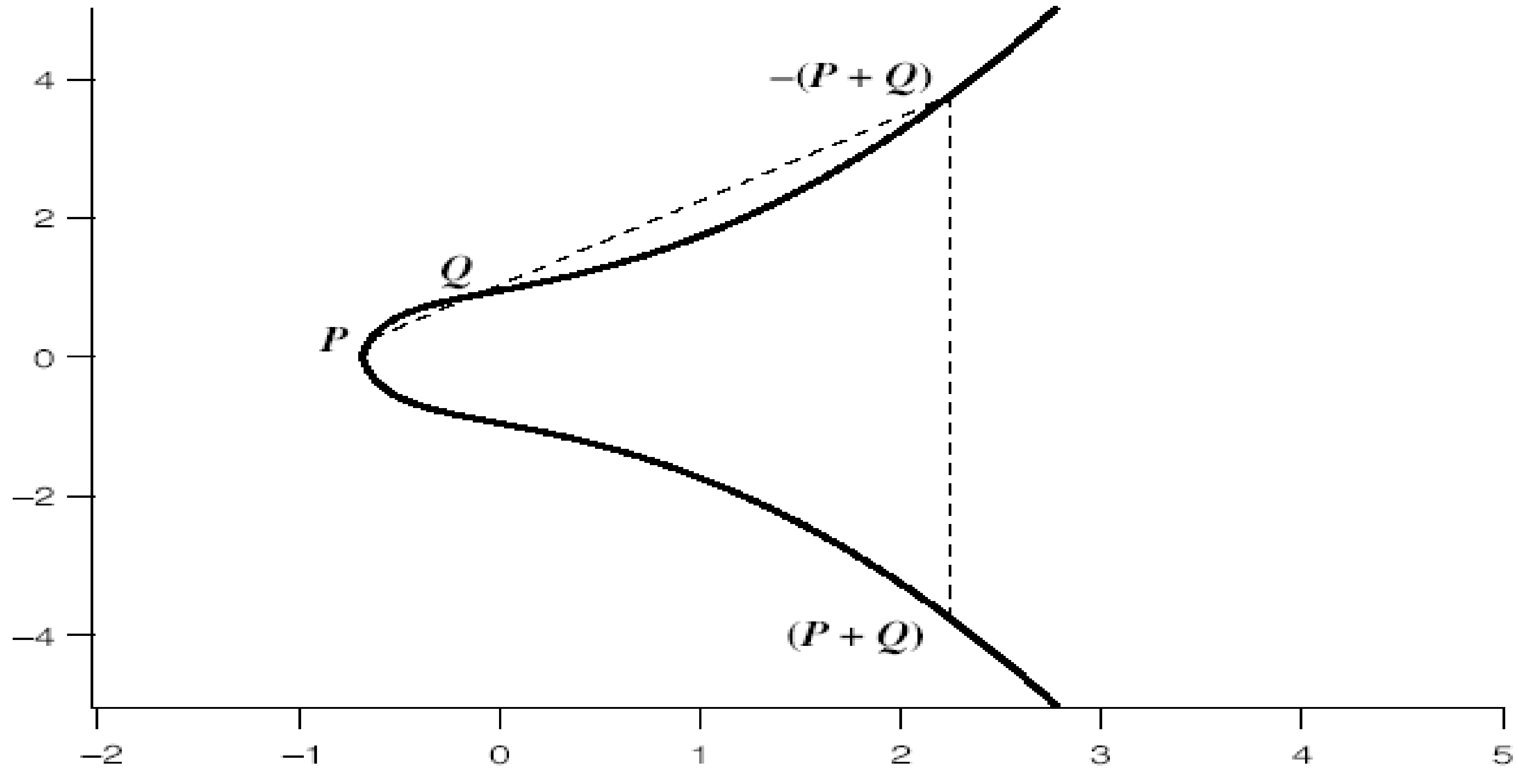
- majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials
- imposes a significant load in storing and processing keys and messages
- an alternative is to use elliptic curves
- offers same security with smaller bit sizes

Real Elliptic Curves

- an elliptic curve is defined by an equation in two variables x & y, with coefficients
- consider a cubic elliptic curve of form
  - $y^2 = x^3 + ax + b$
  - where x,y,a,b are all real numbers
  - also define zero point O
- have addition operation for elliptic curve
  - Q+R is reflection of intersection R
  - Closed form for additions
    - (10.3) and (10.4) P.300-301

Real Elliptic Addition

Rule 1-5 in P.300



(b) $y^2 = x^3 + x + 1$

Finite Elliptic Curves

- Elliptic curve cryptography uses curves whose variables & coefficients are finite integers
- have two families commonly used:
  - prime curves $E_p(a,b)$ defined over Z_p
    - $y^2 \bmod p = (x^3+ax+b) \bmod p$
    - use integers modulo a prime for both variables and coeff
    - best in software
  - Closed form of additions: P.303
  - Example: P=(3,10), Q=(9,7), in $E_{23}(1,1)$
    - P+Q = (17,20)
    - 2P = (7,12)

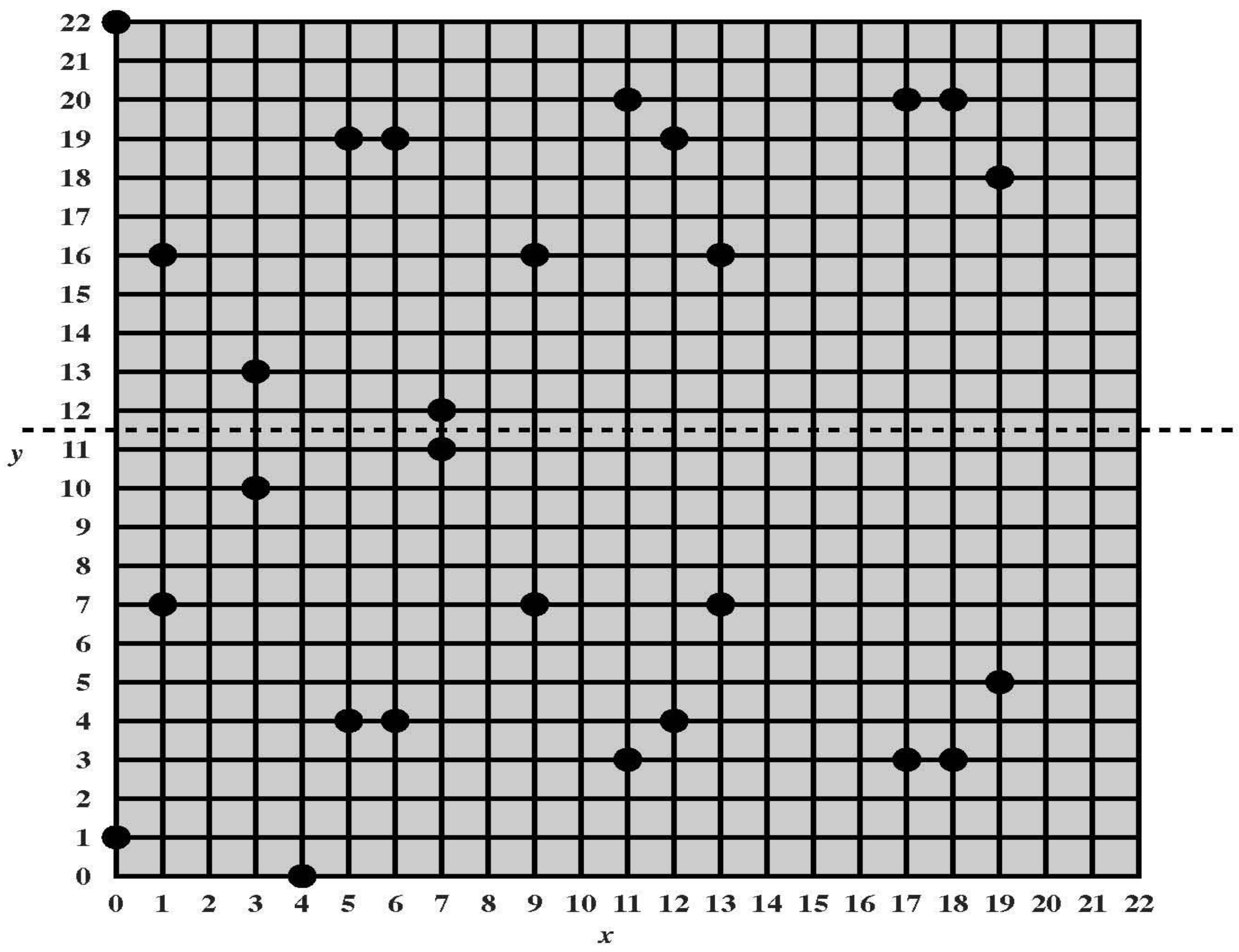


Figure 10.10 The Elliptic Curve $E_{23}(1,1)$

Finite Elliptic Curves

- have two families commonly used:
  - binary curves $E_{2^m}(a,b)$ defined over GF(2^m)
    - use polynomials with binary coefficients
    - best in hardware
  - Take a slightly different form of the equation
  - Different close forms for addition (P.304)

Elliptic Curve Cryptography

- ECC addition is analog of multiply
- ECC repeated addition is analog of exponentiation
- need “hard” problem equiv to discrete log
  - `Q=kP`, where Q,P are points in an elliptic curve
  - is “easy” to compute Q given k,P
  - but “hard” to find k given Q,P
  - known as the elliptic curve logarithm problem
- Certicom example: $E_{23}(9,17)$ (P.305)
  - k could be so large as to make brute-force fail

ECC Key Exchange

- can do key exchange similar to D-H
- users select a suitable curve $E_p(a,b)$
  - Either a prime curve, or a binary curve
- select base point $G=(x_1,y_1)$ with large order n s.t. $nG=O$
- A & B select private keys $n_A<n,\ n_B<n$
- compute public keys: $P_A=n_A \times G,\ P_B=n_B \times G$
- compute shared key: $K=n_A \times P_B,\ K=n_B \times P_A$
  - same since $K=n_A \times n_B \times G$
- Example: P.305

ECC Encryption/Decryption

- select suitable curve & point G as in D-H
- encode any message M as a point on the elliptic curve $P_m=(x,y)$
- each user chooses private key $n_A<n$
- and computes public key $P_A=n_A\times G$
- to encrypt pick random k: $C_m=\{kG,\ P_m+k\ P_b\}$,
- decrypt C_m compute:

$$P_m+kP_b-n_B(kG) = P_m+k(n_BG)-n_B(kG) = P_m$$

- Example: P.307

Table 10.2 Computational Effort for Cryptanalysis of Elliptic Curve Cryptography Compared to RSA

Key Size	MIPS-Years
150	3.8×10^{10}
205	7.1×10^{18}
234	1.6×10^{28}

(a) Elliptic Curve Logarithms using the Pollard rho Method

Key Size	MIPS-Years
512	3×10^{4}
768	2×10^{8}
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

(b) Integer Factorization using the General Number Field Sieve

ECC Security

- relies on elliptic curve logarithm problem
- fastest method is “Pollard rho method”
- compared to factoring, can use much smaller key sizes than with RSA etc
- for equivalent key lengths computations are roughly equivalent
- hence for similar security ECC offers significant computational advantages

Summary

- have considered:
  - distribution of public keys
  - public-key distribution of secret keys
  - Diffie-Hellman key exchange
  - Elliptic Curve cryptography

Chapter 8 – Introduction to Number Theory

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
```

Prime Factorisation

- to **factor** a number `n` is to write it as a product of other numbers: `n=a × b × 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. $91=7\times13$; $3600=2^4\times3^2\times5^2$
  - It is unique $a = \prod_{p\in P} p^{a_p}$

Relatively Prime Numbers & GCD

- two numbers `a, b` are **relatively prime** if have **no common divisors** apart from 1
  - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
- conversely can determine the greatest common divisor by comparing their prime factorizations and using least powers
  - eg. $300=2^1\times3^1\times5^2$ $18=2^1\times3^2$ hence $GCD(18,300)=2^1\times3^1\times5^0=6$

Fermat's Little Theorem

- $a^{p-1} \bmod p = 1$

  where p is prime and a is a positive integer not divisible by p

Euler Totient Function `ø(n)`

- when doing arithmetic modulo n
- **complete set of residues** is: `0..n-1`
- **reduced set of residues** includes those numbers which are relatively prime to n
  - eg for n=10,
  - complete set of residues is {0,1,2,3,4,5,6,7,8,9}
  - reduced set of residues is {1,3,7,9}
- **Euler Totient Function ø(n):**
  - number of elements in reduced set of residues of n
  - **ø(10) = 4**

Euler Totient Function $\emptyset(n)$

- to compute ø(n) need to count number of elements to be excluded
- in general need prime factorization, but
  - for p (p prime) $\emptyset(p) = p-1$
  - for p.q (p,q prime) $\emptyset(p.q) = (p-1)(q-1)$
- eg.
  - $\emptyset(37) = 36$
  - $\emptyset(21) = (3-1)\times(7-1) = 2\times6 = 12$

Euler's Theorem

- a generalisation of Fermat's Theorem
- $a^{\varnothing(n)} \bmod n = 1$
  - where $\gcd(a,n)=1$
- eg.
  - $a=3; n=10; \varnothing(10)=4;$
  - hence $3^4 = 81 = 1 \bmod 10$
  - $a=2; n=11; \varnothing(11)=10;$
  - hence $2^{10} = 1024 = 1 \bmod 11$

Primality Testing

- A number of cryptographic algorithms need to find large prime numbers
- traditionally **sieve** using **trial division**
  - ie. divide by all numbers (primes) in turn less than the square root of the number
  - only works for small numbers
- statistical primality tests
  - for which all primes numbers satisfy property
  - but some composite numbers, called pseudo-primes, also satisfy the property, with a low probability
- Prime is in P:
  - Deterministic polynomial algorithm found in 2002

Miller Rabin Algorithm

- a test based on Fermat's Theorem
- algorithm is:

```
TEST ($n$) is:
1. Find biggest $k$, $k > 0$, so that $(n-1)=2^k q$
2. Select a random integer $a$, $1<a<n-1$
3. if $a^q \bmod n = 1$ then return ("maybe prime");
4. for $j = 0$ to $k - 1$ do
   5. if ($a^{2^j q} \bmod n = n-1$)
      then return(" maybe prime ")
6. return ("composite")
```

- Proof and examples

Probabilistic Considerations

- if Miller-Rabin returns “composite” the number is definitely not prime
- otherwise is a prime or a pseudo-prime
- chance it detects a pseudo-prime is < ¼
- hence if repeat test with different random a then chance n is prime after t tests is:
  - Pr(n prime after t tests) = $1-4^{-t}$
  - eg. for t=10 this probability is > 0.99999

Prime Distribution

- there are infinite prime numbers
  - Euclid’s proof
- prime number theorem states that
  - primes near `n` occur roughly every (`ln n`) integers
- since can immediately ignore evens and multiples of 5, in practice only need test `0.4 ln(n)` numbers before locate a prime around `n`
  - note this is only the “average” sometimes primes are close together, at other times are quite far apart

Chinese Remainder Theorem

- Used to speed up modulo computations
- Used to modulo a product of numbers
  - eg. mod $M = m_1m_2..m_k$, where $\gcd(m_i,m_j)=1$
- Chinese Remainder theorem lets us work in each moduli m_i separately
- since computational cost is proportional to size, this is faster than working in the full modulus M

Chinese Remainder Theorem

- to compute (A mod M) can firstly compute all (a_i mod m_i) separately and then combine results to get answer using:

$$A \equiv \left(\sum_{i=1}^{k} a_i c_i \right) \bmod M$$

$$c_i = M_i \times \left(M_i^{-1} \bmod m_i \right) \quad \text{for } 1 \le i \le k$$

Exponentiation mod p

- $A^x = b \pmod p$
- from Euler's theorem have $a^{ø(n)} \bmod n=1$
- consider $a^m \bmod n=1$, $GCD(a,n)=1$
  - must exist for m= ø(n) but may be smaller
  - once powers reach m, cycle will repeat
- if smallest is m= ø(n) then `a` is called a **primitive root**

Discrete Logarithms or Indices

- the inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo p
- Given a, b, p, find x where $a^x = b \bmod p$
- written as $x=\log_a b \bmod p$ or $x=\mathrm{ind}_{a,p}(b)$
- Logirthm may not always exist
  - $x = \log_3 4 \bmod 13$ (x st $3^x = 4 \bmod 13$) has no answer
  - $x = \log_2 3 \bmod 13 = 4$ by trying successive powers
- whilst exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem
  - Oneway-ness: desirable in modern cryptography

Summary

- have considered:
  - prime numbers
  - Fermat’s and Euler’s Theorems
  - Primality Testing
  - Chinese Remainder Theorem
  - Discrete Logarithms

Cryptography and Network Security

Third Edition

by William Stallings

Lecture slides by Lawrie Brown

Chapter 9 – Public Key Cryptography and RSA

Every Egyptian received two names, which were known respectively as the true name and the good name, or the great name and the little name; and while the good or little name was made public, the true or great name appears to have been carefully concealed.

—*The Golden Bough,* Sir James George Frazer

Private-Key Cryptography

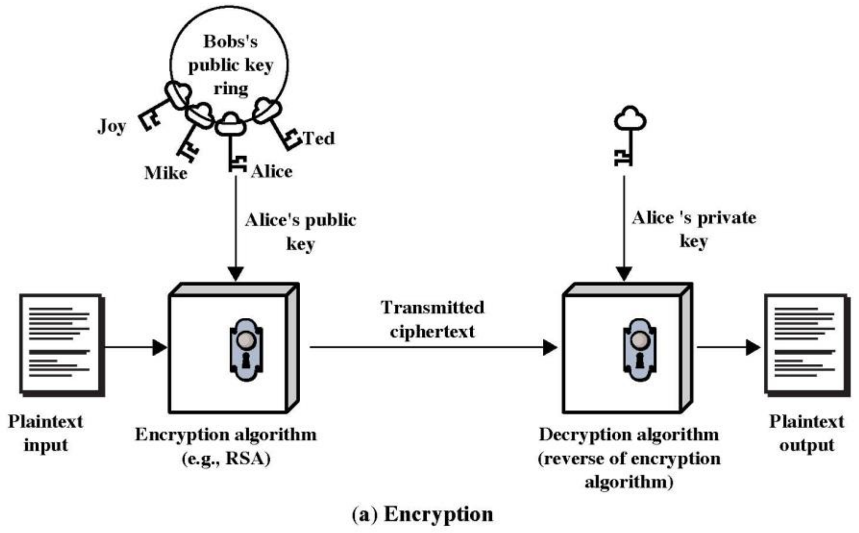
- traditional **private/secret/single key** cryptography uses **one** key
- shared by both sender and receiver
- if this key is disclosed, communications are compromised
- also is **symmetric**, parties are equal
- hence does not protect sender from receiver forging a message & claiming is sent by sender

Public-Key Cryptography

- probably most significant advance in the 3000 year history of cryptography
- uses **two** keys – a public & a private key
  - Anyone knowing the public key can encrypt messages or verify signatures
  - **But cannot** decrypt messages or create signatures
- **asymmetric** since parties are **not** equal
- complements **rather than** replaces private key crypto

Public-Key Cryptography

- **public-key/two-key/asymmetric** 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**
- is **asymmetric** because
  - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures



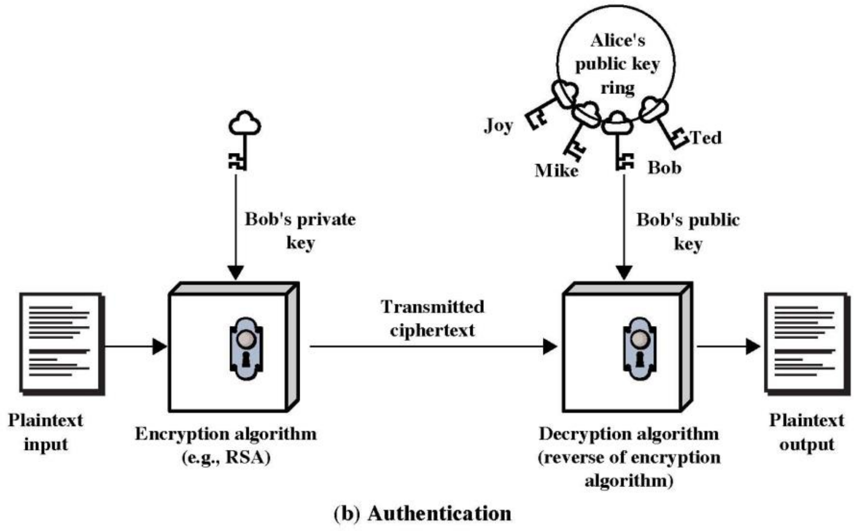


Figure 9.1 Public-Key Cryptography

Why Public-Key Cryptography?

- developed to address two key issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
    - No need for secure key delivery
    - No one else needs to know your private key
  - **digital signatures** – how to verify a message comes intact from the claimed sender

Public-Key Characteristics

- Public-Key algorithms rely on two keys with the characteristics that it is:
  - computationally infeasible to find decryption key knowing only algorithm & encryption key
  - computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - Oneway-ness is desirable: exp/log, mul/fac
  - either of the two related keys can be used for encryption, with the other used for decryption (in some schemes)

Public-Key Cryptosystems:Secrecy

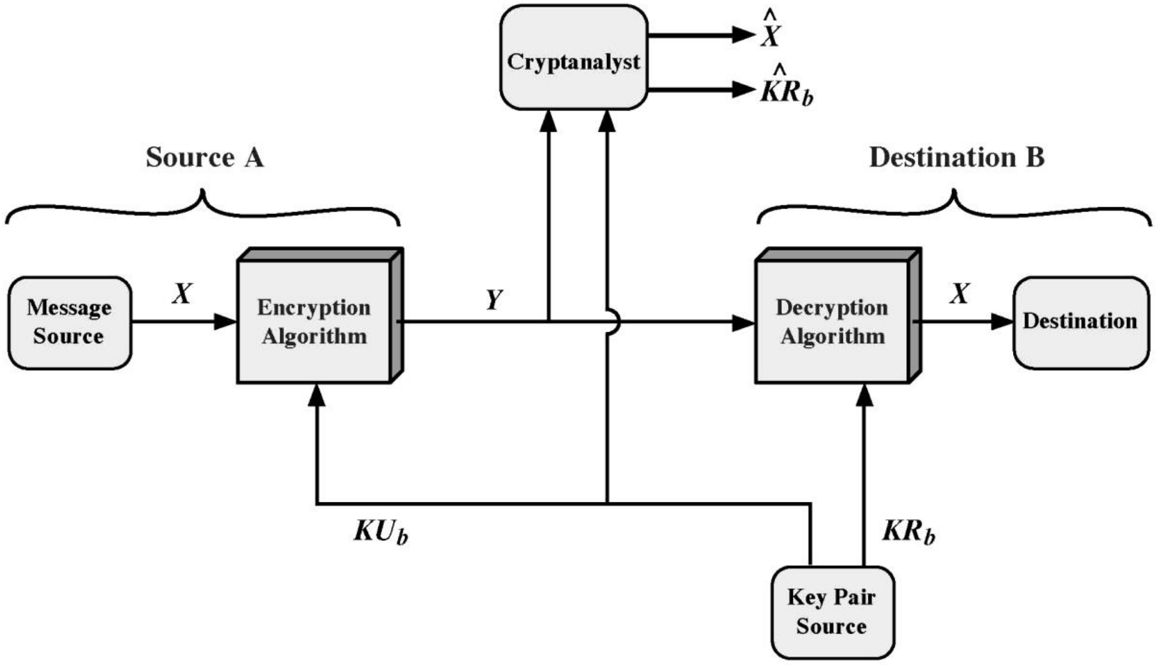


Figure 9.2 Public-Key Cryptosystem: Secrecy

Public-Key Cryptosystems: Authentication

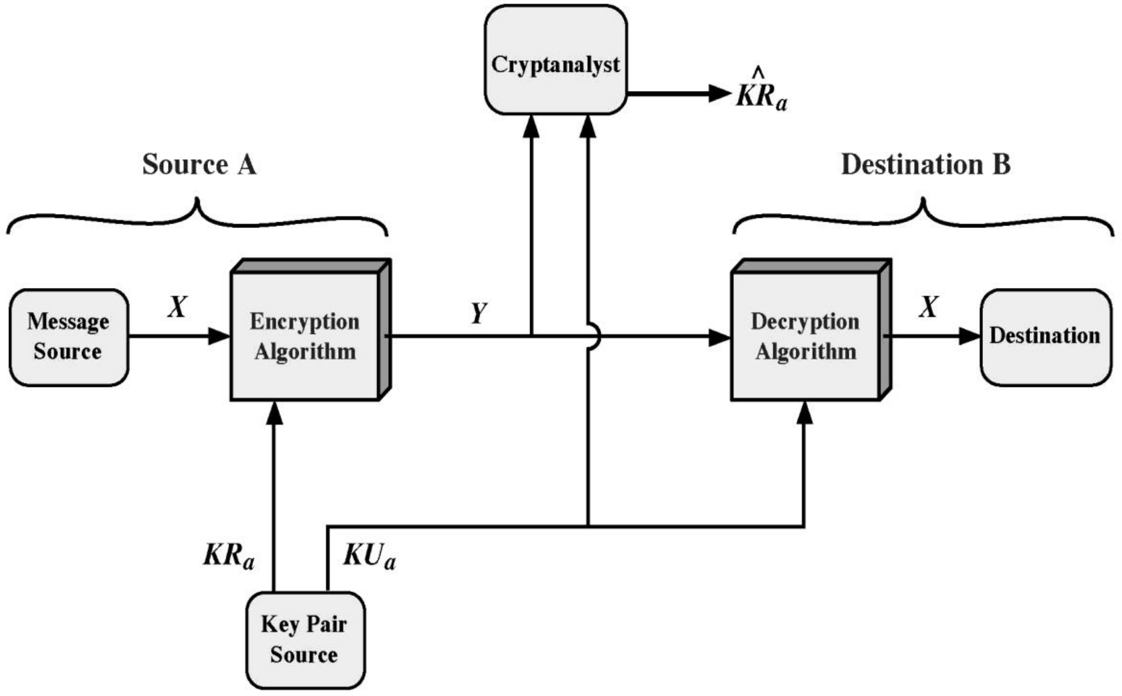


Figure 9.3 Public-Key Cryptosystem: Authentication

Public-Key Cryptosystems: Secrecy and Authentication

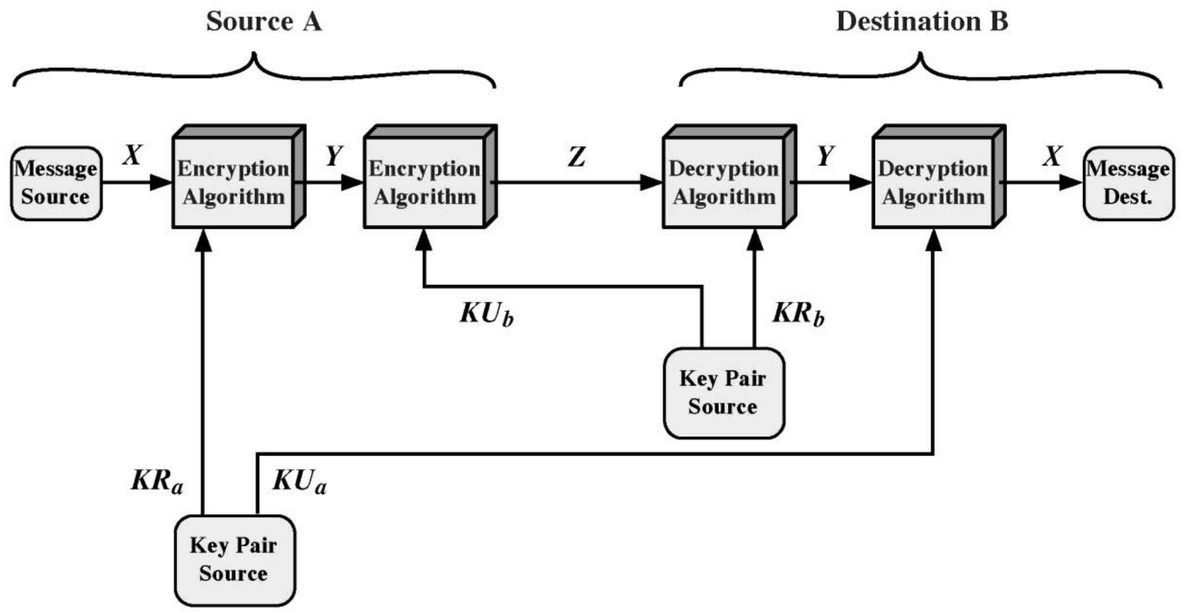


Figure 9.4 Public-Key Cryptosystem: Secrecy and Authentication

Public-Key Applications

- can classify uses into 3 categories:
  - **encryption/decryption** (provide secrecy)
  - **digital signatures** (provide authentication)
  - **key exchange** (of session keys)
- some algorithms are suitable for all uses, others are specific to one

Security of Public Key Schemes

- like private key schemes brute force **exhaustive search** attack is always theoretically possible
- but keys used are too large (>512bits)
- security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems
- requires the use of **very large numbers**
- hence is **slow** compared to private key schemes

RSA

- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- based on exponentiation of integers in a finite (Galois) field
  – Defined over integers modulo a prime
  – exponentiation takes $O((\log n)^3)$ operations (easy)
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
  – factorization takes $O(e^{\log n \log \log n})$ operations (hard)

RSA Key Setup

- each user generates a public/private key pair by:

1. selecting two large primes at random - p, q (secret)
2. computing their system modulus $N=p.q$ (public)
    - note $\varnothing(N)=(p-1)(q-1)$ (secret)
3. selecting at random the encryption key e (public)
    - where $1<e<\varnothing(N)$, $\gcd(e,\varnothing(N))=1$
4. solve following equation to find decryption key d (secret)
    - $e.d=1 \bmod \varnothing(N)$ and $0\leq d\leq N$
    - Use the extended Euclid's algorithm to find the multiplicative inverse of e (mod $\varnothing(N)$)

- publish their public encryption key: KU={e,N}
- keep secret private decryption key: KR={d,p,q}

Block size of RSA

- Each block is represented as an integer number
- Each block has a value M less than N
- The block size is <= $\log_2(N)$ bits
- If the block size is k bits then

  $2^k <= N <= 2^{K+1}$

RSA Use

- to encrypt a message M the sender:
  - obtains **public key** of recipient `KU={e,N}`
  - computes: $C=M^e \bmod N$, where $0 \le M < N$
- to decrypt the ciphertext C the owner:
  - uses their private key `KR={d,p,q}`
  - computes: $M=C^d \bmod N$
- note that the message M must be smaller than the modulus N (block if needed)

Why RSA Works

- because of Euler's Theorem:
- $a^{\varnothing(n)} \bmod N = 1$
  - where $\gcd(a,N)=1$
- in RSA have:
  - $N=p.q$
  - $\varnothing(N)=(p-1)(q-1)$
  - carefully chosen e & d to be inverses $\bmod \varnothing(N)$
  - hence $e.d=1+k.\varnothing(N)$ for some k
- Two cases:
  - 1. gcd(M, N) = 1
  - 2. gcd(M, N) > 1, see equation (8.6) in P.243

RSA Example

1. Select primes: $p=17$ & $q=11$
2. Compute $n = pq = 17 \times 11 = 187$
3. Compute $\varnothing(n)=(p-1)(q-1)=16 \times 10=160$
4. Select e : $\gcd(e,160)=1$; choose $e=7$
5. Determine d: $de=1 \bmod 160$ and $d < 160$
   Value is $d=23$ since $23 \times 7=161= 10 \times 160+1$
6. Publish public key $KU=\{7,187\}$
7. Keep secret private key $KR=\{23,17,11\}$

RSA Example cont

- sample RSA encryption/decryption is:
- given message `M = 88` (`88<187`)
- encryption:

  $C = 88^7 \bmod 187 = 11$

- decryption:

  $M = 11^{23} \bmod 187 = 88$

Exponentiation

- can use the Square and Multiply Algorithm
- a fast, efficient algorithm for exponentiation
- concept is based on repeatedly squaring base
- and multiplying in the ones that are needed to compute the result
- look at binary representation of exponent

Exponentiation

```
c ← 0; d ← 1
for i  ← k downto 0
        do  c ← 2 × c
            d ← (d × d) mod n
            if   $b_i = 1$
                 then   c ←  c + 1
                        d ← (d × a) mod n
return d
```

RSA Key Generation

- users of RSA must:
  - determine two primes at random - `p, q`
  - select either `e` or `d` and compute the other
- primes `p,q` must not be easily derived from modulus `N=p.q`
  - means must be sufficiently large
  - typically guess and use probabilistic test
- exponents `e, d` are inverses, so use Inverse algorithm to compute the other

RSA Security

- three approaches to attacking RSA:
  - brute force key search (infeasible given size of numbers)
  - mathematical attacks (based on difficulty of computing ø(N), by factoring modulus N)
  - timing attacks (on running of decryption)

Factoring Problem

- mathematical approach takes 3 forms:
  - factor `N=p.q`, hence find `ø(N)` and then d
  - determine `ø(N)` directly and find d
  - find d directly
- currently believe all equivalent to factoring
  - have seen slow improvements over the years
    - as of Aug-99 best is 130 decimal digits (512) bit with GNFS
  - biggest improvement comes from improved algorithm
    - cf “Quadratic Sieve” to “Generalized Number Field Sieve”
  - barring dramatic breakthrough 1024+ bit RSA secure
    - ensure p, q of similar size and matching other constraints

Timing Attacks

- developed in mid-1990’s
- exploit timing variations in operations
  - infer bits of d based on time taken
- countermeasures
  - use constant exponentiation time
  - add random delays
  - blind values used in calculations
    - $C' = (Mr)^e$, $M' = (C')^d$, $M=M'r^{-1}$

Summary

- have considered:
  - principles of public-key cryptography
  - RSA algorithm, implementation, security

Cryptography and Network Security

Third Edition

by William Stallings

Lecture slides by Lawrie Brown

Basic Terminology

- **plaintext** - the original message
- **ciphertext** - the coded message
- **cipher** - algorithm for transforming plaintext/ciphertext
- **key** - info used in cipher known only to sender/receiver
- **encipher (encrypt) -** converting plaintext to ciphertext
- **decipher (decrypt)** - recovering ciphertext from plaintext
- **cryptography** - study of encryption principles/methods
- **cryptanalysis (codebreaking)** - the study of principles/ methods of deciphering ciphertext *without* knowing key
- **cryptology** - the field of both cryptography and cryptanalysis

Two kinds of Ciphers

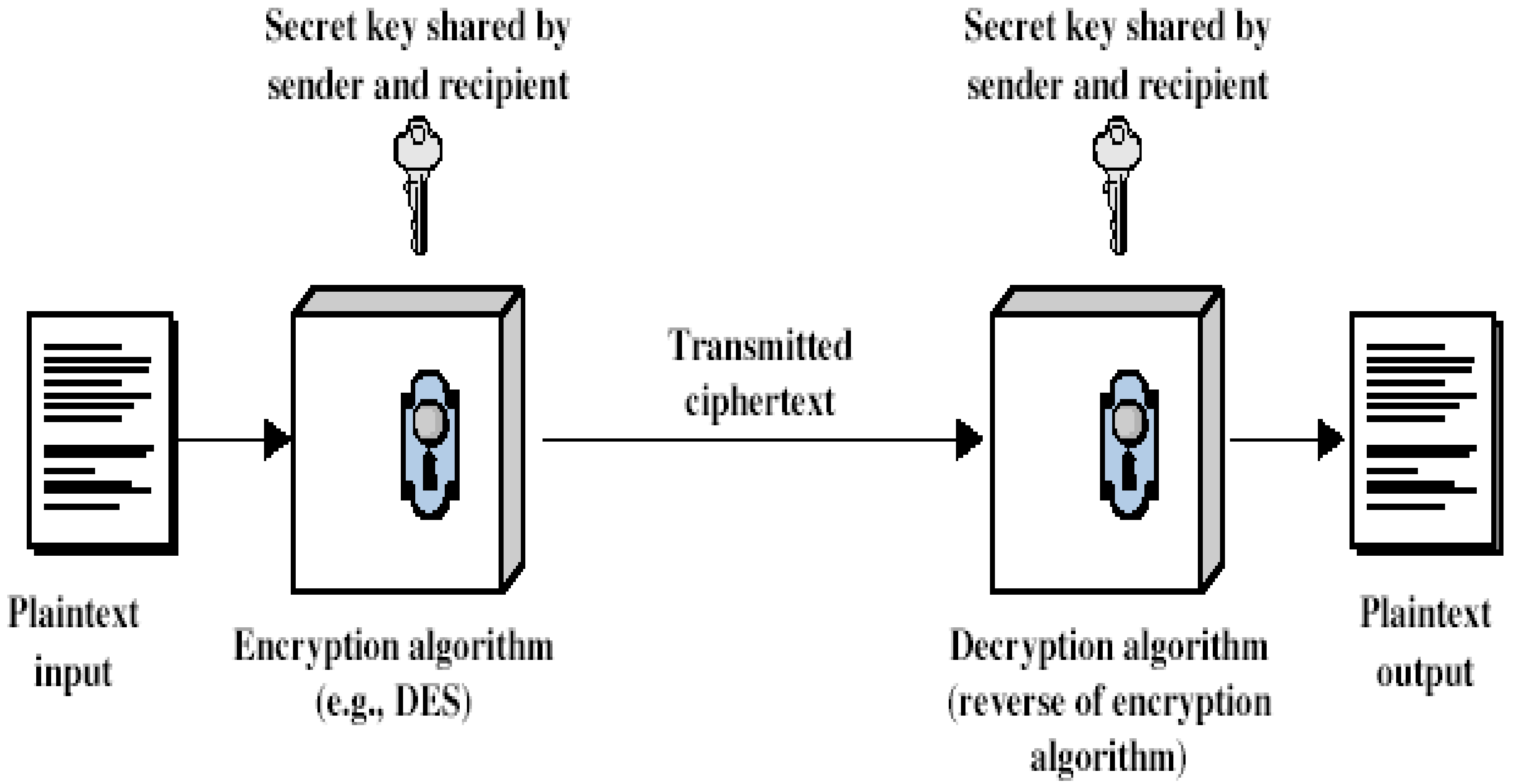
State-of-the-art: two kinds of most popular encryption algorithms

- Symmetric ciphers
  - Sender and receiver share a common key
- Public key ciphers
  - Sender and receiver have asymmetric information of the key(s)

Symmetric Encryption

- or conventional / private-key / single-key
- was only type prior to invention of public-key in 1970's
- remains very widely used
- sender and recipient share a common key
  - Both parties have full information of the key
- all classical encryption algorithms are common key (private-key)
  - Characteristic of conventional algorithms

Symmetric Cipher Model


Secret key shared by sender and recipient
Secret key shared by sender and recipient
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Requirements

- two requirements for secure use of symmetric encryption:
  - a strong encryption algorithm (keeping key secret is sufficient for security)
  - a secret key known only to sender / receiver

    $Y = \mathrm{E}_K(X)$

    $X = \mathrm{D}_K(Y)$
- assume encryption algorithm is known
- implies a secure channel to distribute key

Cryptography

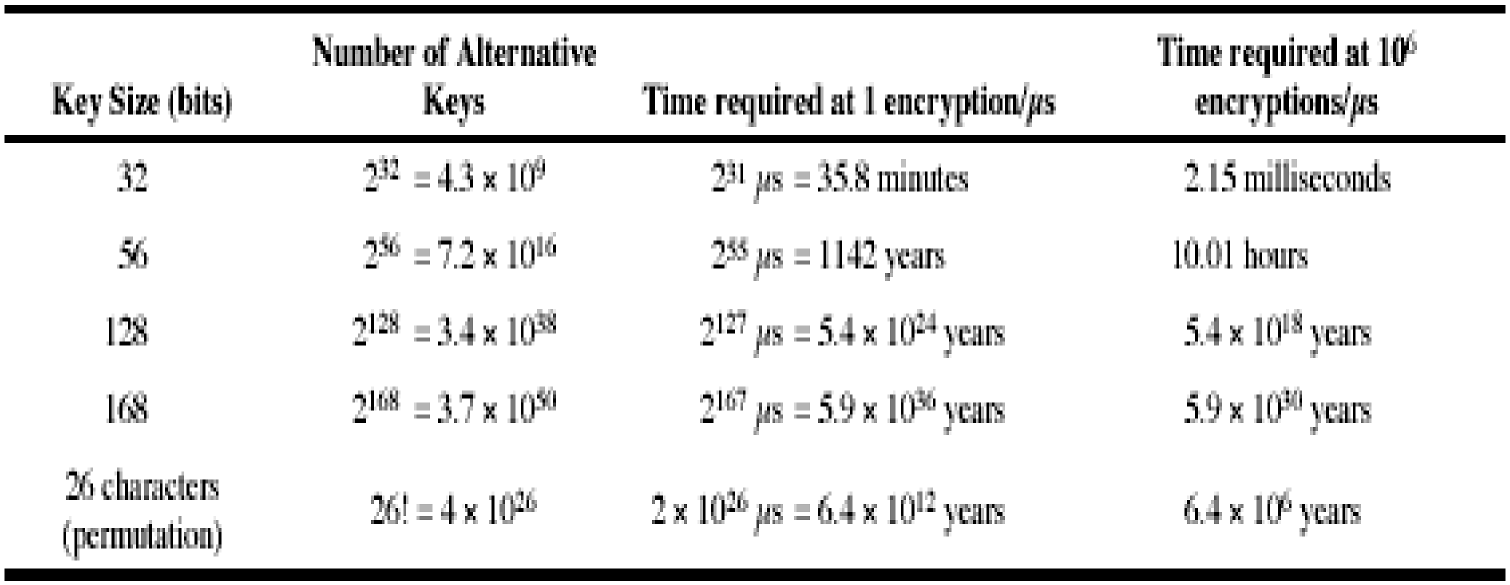
- can characterize by:
  - type of encryption operations used
    - substitution / transposition / product systems
  - number of keys used
    - single-key or private / two-key or public
  - way in which plaintext is processed
    - Block: process one block of elements a time
    - Stream: continuous input, output one element a time

Types of Cryptanalytic Attacks

- **ciphertext only**
  - know a) algorithm b) ciphertext
- **known plaintext**
  - know some given plaintext/ciphertext pairs
- **chosen plaintext**
  - select plaintext and obtain ciphertext
- **chosen ciphertext**
  - select ciphertext and obtain plaintext
- **chosen text**
  - select either plaintext or ciphertext to en/decrypt to attack cipher

Brute Force Search

- always possible to simply try every key
- most basic attack, proportional to key size
- assume either know / recognise plaintext



Key Size (bits)	Number of Alternative Keys	Time required at 1 encryption/μs	Time required at 10^6 encryptions/μs
32	$2^{32} = 4.3 \times 10^9$	2^{31} μs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} μs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} μs = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} μs = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} μs = 6.4×10^{12} years	6.4×10^6 years

More Definitions

- **unconditional security**
  - no matter how much computer power is available, the cipher cannot be broken since the ciphertext provides insufficient information to uniquely determine the corresponding plaintext (non-exist in real applications)
- **computational security**
  - given limited computing resources (eg time needed for calculations is greater than age of universe), the cipher cannot be broken

Classical Ciphers

- Examine a sampling of what might be called classical encryption techniques.
- Illustrate the basic approaches to symmetric encryption and the types of cryptanalytic attacks that must be anticipated.
- The two basic building blocks of all encryption techniques: substitution and transposition.

Classical Substitution Ciphers

- where letters of plaintext are replaced by other letters or by numbers or symbols
- or if plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns

Caesar Cipher

- earliest known substitution cipher
- by Julius Caesar
- first attested use in military affairs
- replaces each letter by a letter *three* places down the alphabet
- example:

```
meet me after the toga party
PHHW PH DIWHU WKH WRJD SDUWB
```

Caesar Cipher

- can define transformation as:

```
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
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
```

- mathematically give each letter a number

```
a b c d e f g h i j k  l  m
0 1 2 3 4 5 6 7 8 9 10 11 12
n  o  p  q  r  s  t  u  v  w  x  y  Z
13 14 15 16 17 18 19 20 21 22 23 24 25
```

- then have Caesar cipher as:

  C = E(p) = (p + k) mod (26)

  p = D(C) = (C – k) mod (26)

  – **modulo arithmetic:** 1 = 27 mod 26, 3 = 29 mod 26

Cryptanalysis of Caesar Cipher

- only have 26 possible keys
  - Could shift K = 0, 1, 2, …, 25 slots
- could simply try each in turn
- a **brute force search**
- given ciphertext, just try all shifts of letters
- do need to recognize when have plaintext
- Test:break ciphertext

  GCUA VQ DTGCM

Monoalphabetic Cipher

- rather than just shifting the alphabet
- could shuffle the letters arbitrarily
- each plaintext letter maps to a different random ciphertext letter
- hence key is 26 letters long

```
Plain:  abcdefghijklmnopqrstuvwxyz
Cipher: DKVQFIBJWPESCXHTMYAUOLRGZN
Plaintext:  ifwewishtoreplaceletters
Ciphertext: WIRFRWAJUHYFTSDVFSFUUFYA
```

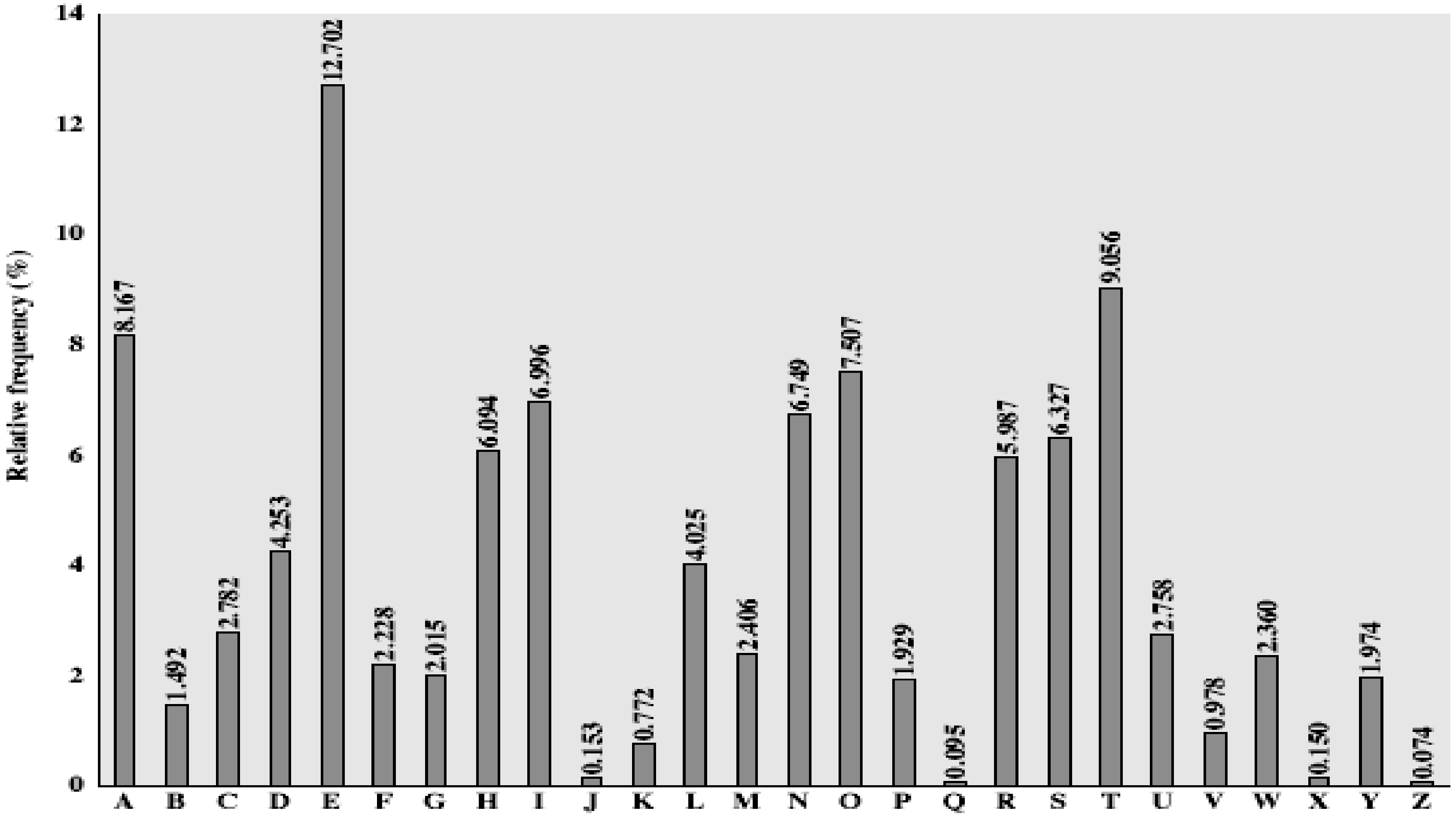
Monoalphabetic Cipher Security

- now have a total of 26! = 4 x 10^26 keys
- with so many keys, might think is secure
  - The simplicity and strength of the monoalphabetic substitution cipher dominated for the first millenium AD.
- but would be **!!!WRONG!!!**
  - First broken by Arabic scientists in 9th century

Frequency Analysis

- letters are not equally commonly used
- in English **e** is by far the most common letter
- then T,R,N,I,O,A,S
- other letters are fairly rare
- cf. Z,J,K,Q,X
- have tables of single, double & triple letter frequencies

English Letter Frequencies



Use in Cryptanalysis

- key concept - monoalphabetic substitution ciphers do not change relative letter frequencies
- discovered by Arabian scientists in 9th century
- calculate letter frequencies for ciphertext
- compare counts/plots against known values
- for monoalphabetic must identify each letter
  - tables of common double/triple letters help

Example Cryptanalysis

- given ciphertext:

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
```

- count relative letter frequencies (see text)
- guess P & Z are e and t
- guess ZW is th and hence ZWP is the
- proceeding with trial and error finally get:

```
it was disclosed yesterday that several informal but
direct contacts have been made with political
representatives of the viet cong in moscow
```

Playfair Cipher

- not even the large number of keys in a monoalphabetic cipher provides security
- one approach to improving security was to encrypt multiple letters
- the **Playfair Cipher** is an example
- invented by Charles Wheatstone in 1854, but named after his friend Baron Playfair

Playfair Key Matrix

- a 5X5 matrix of letters based on a keyword
- fill in letters of keyword (sans duplicates)
- fill rest of matrix with other letters
- eg. using the keyword MONARCHY

```
MONAR
CHYBD
EFGIK
LPQST
UVWXZ
```

Encrypting and Decrypting

- plaintext encrypted two letters at a time:
  1. if a pair is a repeated letter, insert a filler like 'X', eg. "balloon" encrypts as "ba lx lo on"
  2. if both letters fall in the same row, replace each with letter to right (wrapping back to start from end), eg. “ar" encrypts as "RM"
  3. if both letters fall in the same column, replace each with the letter below it (again wrapping to top from bottom), eg. “mu" encrypts to "CM"
  4. otherwise each letter is replaced by the one in its row in the column of the other letter of the pair, eg. “hs" encrypts to "BP", and “ea" to "IM" or "JM" (as desired)

Security of the Playfair Cipher

- security much improved over monoalphabetic
- since have 26 x 26 = 676 digrams
- would need a 676-entry frequency table to analyse (verses 26 for a monoalphabetic)
- and correspondingly more ciphertext
- was widely used for many years (eg. US & British military in WW1)
- it **can** be broken, given a few hundred letters
- since still has much of plaintext structure

Polyalphabetic Ciphers

- another approach to improving security is to use multiple cipher alphabets
- called **polyalphabetic substitution ciphers**
- makes cryptanalysis harder with more alphabets to guess and flatter frequency distribution
- use a key to select which alphabet is used for each letter of the message
- use each alphabet in turn
- repeat from start after end of key is reached

Example

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

- write the plaintext out
- write the keyword repeated above it
  - eg using keyword *deceptive*
- use each key letter as a caesar cipher key
- encrypt the corresponding plaintext letter

Vigenère Cipher

- simplest polyalphabetic substitution cipher is the **Vigenère Cipher**
- effectively multiple caesar ciphers
- key is d-letter long K = k1 k2 ... kd
- i^{th} letter specifies i^{th} alphabet to use
- use each alphabet in turn
- repeat from start after d letters in message
- decryption simply works in reverse

Security of Vigenère Ciphers

- have multiple ciphertext letters for each plaintext letter
- hence letter frequencies are obscured
- but not totally lost
- start with letter frequencies
  - see if look monoalphabetic or not
- if not, then need to determine number of alphabets, since then can attach each

Kasiski Method

repetitions in ciphertext give clues to period

- so find same plaintext an exact period apart
- which results in the same ciphertext
- eg repeated “VTW” in previous example

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

- suggests size of 3 or 9
- find a number of duplicated sequences, collect all their distances apart, look for common factors
- then attack each monoalphabetic cipher individually using same techniques as before

Autokey Cipher

- Use the plain text itself as part of the key
- eg. given key *deceptive*

```
 key:        deceptivewearediscoveredsav
 plaintext:  wearediscoveredsaveyourself
 ciphertext:ZICVTWQNGKZEIIGASXSTSLVVWLA
```

- but still have frequency characteristics to attack

One-Time Pad

- if a truly random key as long as the message is used, the cipher will be secure
- called a One-Time pad
- is unbreakable since ciphertext bears no statistical relationship to the plaintext
  - No repetition of patterns
- since for **any plaintext** & **any ciphertext** there exists a key mapping one to other
- can only use the key **once** though
- have problem of safe distribution of key

Transposition Ciphers

- now consider classical **transposition** or **permutation** ciphers
- these hide the message by rearranging the letter order
- without altering the actual letters used
- can recognise these since have the same frequency distribution as the original text

Rail Fence cipher

- write message letters out diagonally over a number of rows
- then read off cipher row by row
- eg. write message out as:

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

- giving ciphertext

```
MEMATRHTGPRYETEFETEOAAT
```

Row Transposition Ciphers

- a more complex scheme
- write letters of message out in rows over a specified number of columns
- then reorder the columns according to some key before reading off the rows

```
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
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Product Ciphers

- ciphers using substitutions or transpositions are not secure because of language characteristics
- hence consider using several ciphers in succession to make harder, but:
  - two substitutions make a more complex substitution
  - two transpositions make more complex transposition
  - but a substitution followed by a transposition makes a new much harder cipher
- this is bridge from classical to modern ciphers

Rotor Machines

- Multiple-stage substitution algorithms
- before modern ciphers, rotor machines were most common product cipher
- were widely used in WW2
  - German Enigma, Allied Hagelin, Japanese Purple
- implemented a very complex, varying substitution cipher
- used a series of cylinders, each giving one substitution, which rotated and changed after each letter was encrypted

Steganography

- an alternative to encryption
- hides existence of message
  - using only a subset of letters/words in a longer message marked in some way
  - using invisible ink
  - hiding graphic image or sound file
- has drawbacks
  - high overhead to hide relatively few info bits

Summary

- have considered:
  - classical cipher techniques and terminology
  - monoalphabetic substitution ciphers
  - cryptanalysis using letter frequencies
  - Playfair ciphers
  - polyalphabetic ciphers
  - transposition ciphers
  - product ciphers and rotor machines
  - stenography