Advance Security 1
Lecture 1

Introduction

Assessment Methods

- Written examination – 60%
- Continuous assessment – 40%
  - Theory test 1 (week 6)- 10%
  - Theory test 2 (week 12)- 10% (All Lectures included)
  - Assignment 1 Cryptographic Tools Research - 5%
  - Lab sheets – 15% (Lab1=1%, Lab2=6%, Lab3=2%, and Lab 4=6%)

Module Contents

- Introduction to Advanced Security
- Number theory, Discrete logarithms and Elliptic Curves
- Steganography
- **Symmetric Encryption:** Block Ciphers and Advanced Encryption Standard, Confidentiality Using Conventional Encryption.
- **Asymmetric Encryption:** Public-Key Cryptography and RSA,
- **Mutual Trust:** Key management and Authentication Protocols
- **Cryptographic Hash Functions:** Message Authentication and Hash Functions, Hash and Mac Algorithms, Digital Signatures.

Text Book

Cryptography and Network Security : Principles and Practices, 6th Ed, Williams Stallings (2014) Pearson.

Book Chapters

- Chapter 1: Overview
- Chapter 2: Classical Encryption Techniques
- Chapter 3: Block ciphers and the data encryption standard
- Chapter 4: Basic Concepts in Number Theory and Finite Fields
- Chapter 5 Advanced Encryption Standard
- Chapter 6 Block Cipher Operation
- Chapter 7 Pseudorandom Number Generation and Stream Ciphers
- Chapter 8 More Number Theory
- Chapter 9 Public-Key Cryptography and RSA
- Chapter 11 Cryptographic Hash Functions
- Chapter 14 Key Management and Distribution

References

- Network Security Essentials: Applications and Standards, 4th Ed, William Stallings (2010), Prentice Hall.
- Introduction to Cryptography with Java Applets, David Bishop (2003), Jones and Batlett Computer Science.
- Cryptography Engineering, Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno (2010), John Wiley and Sons
- Any Book on Cryptography
- Web

Assessment guidelines

- This is not a distant education module, therefore attendance to lectures and labs is necessary and make sure you sign the attendance sheet.
- Anyone who is not attending the assumption is that you know what we are doing.
- If you miss an assessment submission date marks will be deducted as follows:
  - Each day 20%
  - No submission will be accepted after I finish marking and give feedback in class.

Assessment guidelines

- The success of the module is a team effort:
- Depends on honest participation, increase knowledge, share and dare
- Submission guidelines
  - naming files (Full-Name_Student-Number_Assignment-Name), use Brightspace, no email submission
- Report guidelines
  - Cover page, introduction, body, discussion, conclusion and references
- Lab demonstration guidelines – must be done in the lab

Postgraduate Studies

- MSc in Computing (Security & Forensics)
  - Full time - DT228A Part time - DT228B
  - Digital forensics, also known as computer and network forensics is the application of science to the identification, collection, examination, and analysis of data while preserving the integrity of the information and maintaining a strict chain of custody for the data. Every organisation and user needs to have the capability to perform digital forensics. Without such a capability, organisations and users will have difficulty determining what events have occurred within their systems, networks and mobile devices, such as the exposure of protected, sensitive data.

Security and Forensics Course at other Universities in Ireland

- Institute of Technology Blanchardstown
  - Master of Science in Computing (Information Security & Digital Forensics)
- Cork Institute of Technology
  - Master of Science in Networking & Security
- Dublin City University
  - M.Sc. in Security and Forensic Computing
- Letterkeny IT
  - Master of Science in Computing in Systems & Software Security

Security and Forensics Course at other Universities in Ireland

- University College Dublin
  - MSc Digital Investigation and Forensic Computing
  - Forensic Computing and Cybercrime Investigation (FCCI). Qualifications include Graduate Certificate, Graduate Diploma, Master of Science and Continuous Professional Development programme (CPD).

- University of Limerick
  - MEng Information and Network Security

Final Year Projects in Security

- Use of Cryptography in applications running in mobile devices.
- How Cryptography can be used to provide secure transmission of information in Cloud and the Internet of things
- Efficient deployment of Cryptography in systems
- Security protocols or algorithms
- Security Games development

Final Year Projects in Security

- Intelligent rules creation in Firewall, IDS and IPS.
- E-learning project to demonstrate vulnerabilities of applications, protocols or Operating Systems.
- Extend or modify the functionalities of Security Tools such as Kali Linux, Snort, Wireshark, nmap, netcat etc
- Test the performance of security tools

Cryptographic algorithms and protocols can be grouped into four main areas:

Symmetric encryption

- Used to conceal the contents of blocks or streams of data of any size, including messages, files, encryption keys, and passwords

Asymmetric encryption

- Used to conceal small blocks of data, such as encryption keys and hash function values, which are used in digital signatures

Data integrity algorithms

- Used to protect blocks of data, such as messages, from alteration

Authentication protocols

- Schemes based on the use of cryptographic algorithms designed to authenticate the identity of entities

The field of network and Internet security consists of:

measures to deter, prevent, detect, and correct security violations that involve the transmission of information

Computer Security

- The NIST *Computer Security Handbook* defines the term computer security as:

  "the protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and confidentiality of information system resources" (includes hardware, software, firmware, information/ data, and telecommunications)

Computer Security Objectives

Confidentiality

- Data confidentiality
  - Assures that private or confidential information is not made available or disclosed to unauthorized individuals
- Privacy
  - Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed

Integrity

- Data integrity
  - Assures that information and programs are changed only in a specified and authorized manner
- System integrity
  - Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system

Availability

- Assures that systems work promptly and service is not denied to authorized users

CIA Triad

Confidentiality
Integrity
Availability
Data
and
services

Possible additional concepts:

Authenticity

- Verifying that users are who they say they are and that each input arriving at the system came from a trusted source

Accountability

- The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity

Breach of Security Levels of Impact

Computer Security Challenges

- Security is not simple
- Potential attacks on the security features need to be considered
- It is necessary to decide where to use the various security mechanisms
- Requires constant monitoring
- Security mechanisms typically involve more than a particular algorithm or protocol
- Security is essentially a battle of wits between a perpetrator and the designer
- Little benefit from security investment is perceived until a security failure occurs
- Strong security is often viewed as an impediment to efficient and user-friendly operation

OSI Security Architecture

- Security attack
  - Any action that compromises the security of information owned by an organization
- Security mechanism
  - A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack
- Security service
  - A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization
  - Intended to counter security attacks, and they make use of one or more security mechanisms to provide the service

Table 1.1
Threats and Attacks (RFC 4949)

Threat

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

Attack

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

Security Attacks

- A means of classifying security attacks, used both in X.800 and RFC 4949, is in terms of *passive attacks* and *active attacks*

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

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

Figure 1.1 Security Attacks

Passive Attacks

- Are in the nature of eavesdropping on, or monitoring of, transmissions
- Goal of the opponent is to obtain information that is being transmitted

- Two types of passive attacks are:
  - The release of message contents
  - Traffic analysis

Active Attacks

- Involve some modification of the data stream or the creation of a false stream
- Difficult to prevent because of the wide variety of potential physical, software, and network vulnerabilities
- Goal is to detect attacks and to recover from any disruption or delays caused by them

Masquerade
- Takes place when one entity pretends to be a different entity
- Usually includes one of the other forms of active attack

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

Modification of messages
- Some portion of a legitimate message is altered, or messages are delayed or reordered to produce an unauthorized effect

Denial of service
- Prevents or inhibits the normal use or management of communications facilities

Security Services

- Defined by X.800 as:
  - A service provided by a protocol layer of communicating open systems and that ensures adequate security of the systems or of data transfers
- Defined by RFC 4949 as:
  - A processing or communication service provided by a system to give a specific kind of protection to system resources

X.800 Service Categories

- Authentication
- Access control
- Data confidentiality
- Data integrity
- Nonrepudiation

Authentication

- Concerned with assuring that a communication is authentic
  - In the case of a single message, assures the recipient that the message is from the source that it claims to be from
  - In the case of ongoing interaction, assures the two entities are authentic and that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties

Two specific authentication services are defined in X.800:

- Peer entity authentication
- Data origin authentication

Access Control

- The ability to limit and control the access to host systems and applications via communications links
- To achieve this, each entity trying to gain access must first be indentified, or authenticated, so that access rights can be tailored to the individual

Data Confidentiality

- The protection of transmitted data from passive attacks
  - Broadest service protects all user data transmitted between two users over a period of time
  - Narrower forms of service includes the protection of a single message or even specific fields within a message
- The protection of traffic flow from analysis
  - This requires that an attacker not be able to observe the source and destination, frequency, length, or other characteristics of the traffic on a communications facility

Data Integrity

Can apply to a stream of messages, a single message, or selected fields within a message

Connection-oriented integrity service, one that deals with a stream of messages, assures that messages are received as sent with no duplication, insertion, modification, reordering, or replays

A connectionless integrity service, one that deals with individual messages without regard to any larger context, generally provides protection against message modification only

Nonrepudiation

- Prevents either sender or receiver from denying a transmitted message
- When a message is sent, the receiver can prove that the alleged sender in fact sent the message
- When a message is received, the sender can prove that the alleged receiver in fact received the message

Table 1.2

Security Services (X.800)

(This table is found on page 18 in textbook)

AUTHENTICATION

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

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

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

ACCESS CONTROL

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

DATA CONFIDENTIALITY

The protection of data from unauthorized disclosure.

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

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

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

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

DATA INTEGRITY

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

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

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

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

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

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

NONREPUDIATION

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

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

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

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

Table 1.3

Security Mechanisms (X.800)

(This table is found on pages 20-21 in textbook)

SPECIFIC SECURITY MECHANISMS

May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

Encipherment
The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.

Digital Signature
Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).

Access Control
A variety of mechanisms that enforce access rights to resources.

Data Integrity
A variety of mechanisms used to assure the integrity of a data unit or stream of data units.

Authentication Exchange
A mechanism intended to ensure the identity of an entity by means of information exchange.

Traffic Padding
The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.

Routing Control
Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.

Notarization
The use of a trusted third party to assure certain properties of a data exchange.

PERVASIVE SECURITY MECHANISMS

Mechanisms that are not specific to any particular OSI security service or protocol layer.

Trusted Functionality
That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).

Security Label
The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.

Event Detection
Detection of security-relevant events.

Security Audit Trail
Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.

Security Recovery
Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

Model for Network Security

Figure 1.2 Model for Network Security

Network Access Security Model

Figure 1.3 Network Access Security Model

Unwanted Access

- Placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application programs as well as utility programs such as editors and compilers
- Programs can present two kinds of threats:
  - Information access threats
    - Intercept or modify data on behalf of users who should not have access to that data
  - Service threats
    - Exploit service flaws in computers to inhibit use by legitimate users

Summary

- Computer security concepts
  - Definition
  - Examples
  - Challenges
- The OSI security architecture
- Security attacks
  - Passive attacks
  - Active attacks

- Security services
  - Authentication
  - Access control
  - Data confidentiality
  - Data integrity
  - Nonrepudiation
  - Availability service
- Security mechanisms

1

2

3

4

I haven't
got any
reason

Chapter 2

Classical Encryption Techniques

Symmetric Encryption

- Also referred to as conventional encryption or single-key encryption
- Was the only type of encryption in use prior to the development of public-key encryption in the 1970s
- Remains by far the most widely used of the two types of encryption

Basic Terminology

- Plaintext
  - The original message
- Ciphertext
  - The coded message
- Enciphering or encryption
  - Process of converting from plaintext to ciphertext
- Deciphering or decryption
  - Restoring the plaintext from the ciphertext
- Cryptography
  - Study of encryption
- Cryptographic system or cipher
  - Schemes used for encryption
- Cryptanalysis
  - Techniques used for deciphering a message without any knowledge of the enciphering details
- Cryptology
  - Areas of cryptography and cryptanalysis together

Simplified Model of Symmetric Encryption

Figure 2.1 Simplified Model of Symmetric Encryption

Model of Symmetric Cryptosystem

Figure 2.2 Model of Symmetric Cryptosystem

Cryptographic Systems

- Characterized along three independent dimensions:

The type of operations used for transforming plaintext to ciphertext	The number of keys used	The way in which the plaintext is processed
Substitution	Symmetric, single-key, secret-key, conventional encryption	Block cipher
Transposition	Asymmetric, two-key, or public-key encryption	Stream cipher

Cryptanalysis and Brute-Force Attack

Cryptanalysis

- Attack relies on the nature of the algorithm plus some knowledge of the general characteristics of the plaintext
- Attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used

Brute-force attack

- Attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained
- On average, half of all possible keys must be tried to achieve success

Type of Attack	Known to Cryptanalyst
Ciphertext Only	• Encryption algorithm • Ciphertext
Known Plaintext	• Encryption algorithm • Ciphertext • One or more plaintext-ciphertext pairs formed with the secret key
Chosen Plaintext	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	• Encryption algorithm • Ciphertext • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Table 2.1

Types of Attacks on Encrypted Messages

Encryption Scheme Security

- Unconditionally secure
  - No matter how much time an opponent has, it is impossible for him or her to decrypt the ciphertext simply because the required information is not there
- Computationally secure
  - The cost of breaking the cipher exceeds the value of the encrypted information
  - The time required to break the cipher exceeds the useful lifetime of the information

Brute-Force Attack

Involves trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained

On average, half of all possible keys must be tried to achieve success

To supplement the brute-force approach, some degree of knowledge about the expected plaintext is needed, and some means of automatically distinguishing plaintext from garble is also needed

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 ciphertext bit patterns

Caesar Cipher

- Simplest and earliest known use of a substitution cipher
- Used by Julius Caesar
- Involves replacing each letter of the alphabet with the letter standing three places further down the alphabet
- Alphabet is wrapped around so that the letter following Z is A

plain: meet me after the toga party

cipher: PHHW PH DIWHU WKH WRJD SDUWB

Caesar Cipher Algorithm

- 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 n o p q r s t u v w x y z
  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

- Algorithm can be expressed as:

$$c = E(3, p) = (p + 3) \bmod (26)$$

- A shift may be of any amount, so that the general Caesar algorithm is:

$$C = E(k, p) = (p + k) \bmod 26$$

- Where k takes on a value in the range 1 to 25; the decryption algorithm is simply:

$$p = D(k, C) = (C - k) \bmod 26$$

Brute-Force Cryptanalysis of Caesar Cipher

KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

Figure 2.3 Brute-Force Cryptanalysis of Caesar Cipher

(This chart can be found on page 35 in the textbook)

Sample of Compressed Text

Figure 2.4 Sample of Compressed Text

Monoalphabetic Cipher

- Permutation
  - Of a finite set of elements *S* is an ordered sequence of all the elements of *S* , with each element appearing exactly once
- If the "cipher" line can be any permutation of the 26 alphabetic characters, then there are 26! or greater than 4×10^{26} possible keys
  - This is 10 orders of magnitude greater than the key space for DES
  - Approach is referred to as a *monoalphabetic substitution cipher* because a single cipher alphabet is used per message

Figure 2.5 Relative Frequency of Letters in English Text

Monoalphabetic Ciphers

- Cipher Text
  UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
  VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
  EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

- Frequency of letter in cipher text and compare it with English Frequency

P	13.33	H	5.83	F	3.33	B	1.67	C	0.00
Z	11.67	D	5.00	W	3.33	G	1.67	K	0.00
S	8.33	E	5.00	Q	2.50	Y	1.67	L	0.00
U	8.33	V	4.17	T	2.50	I	0.83	N	0.00
O	7.50	X	4.17	A	1.67	J	0.83	R	0.00
M	6.67								

- Putting value in cipher text

```
UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
 t a        e  e te  a that e e a       a
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
   e t   ta t ha e ee  a e  th    t  a
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
 e  e e tat  e   the   t
```

Monoalphabetic Ciphers

- Easy to break because they reflect the frequency data of the original alphabet
- Countermeasure is to provide multiple substitutes (homophones) for a single letter
- Digram
  - Two-letter combination
  - Most common is *th*

- Trigram
  - Three-letter combination
  - Most frequent is *the*

Playfair Cipher

- Best-known multiple-letter encryption cipher
- Treats digrams in the plaintext as single units and translates these units into ciphertext digrams
- Based on the use of a 5 x 5 matrix of letters constructed using a keyword
- Invented by British scientist Sir Charles Wheatstone in 1854
- Used as the standard field system by the British Army in World War I and the U.S. Army and other Allied forces during World War II

Playfair Key Matrix

- Fill in letters of keyword (minus duplicates) from left to right and from top to bottom, then fill in the remainder of the matrix with the remaining letters in alphabetic order
- Using the keyword MONARCHY:

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

Hill Cipher

- Developed by the mathematician Lester Hill in 1929
- Strength is that it completely hides single-letter frequencies
  - The use of a larger matrix hides more frequency information
  - A 3 x 3 Hill cipher hides not only single-letter but also two-letter frequency information
- Strong against a ciphertext-only attack but easily broken with a known plaintext attack

Hill Cipher Example

For example, consider the plaintext "paymoremoney" and use the encryption key

$$\mathbf{K} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

The first three letters of the plaintext are represented by the vector (15 0 24). Then(15 0 24)$\mathbf{K}$ = (303 303 531) mod 26 = (17 17 11) = RRL. Continuing in this fashion, the ciphertext for the entire plaintext is RRLMWBKASPDH.

$$\mathbf{K}^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

This is demonstrated as

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}\begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

It is easily seen that if the matrix $\mathbf{K}^{-1}$ is applied to the ciphertext, then the plaintext is recovered.

In general terms, the Hill system can be expressed as

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

Polyalphabetic Ciphers

- Polyalphabetic substitution cipher
  - Improves on the simple monoalphabetic technique by using different monoalphabetic substitutions as one proceeds through the plaintext message

All these techniques have the following features in common:

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

Vigenère Cipher

- Best known and one of the simplest polyalphabetic substitution ciphers
- In this scheme the set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers with shifts of 0 through 25
- Each cipher is denoted by a key letter which is the ciphertext letter that substitutes for the plaintext letter a

Example of Vigenère Cipher

- To encrypt a message, a key is needed that is as long as the message
- Usually, the key is a repeating keyword
- For example, if the keyword is *deceptive*, the message “we are discovered save yourself” is encrypted as:

key: deceptivedeceptivedeceptive

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Example of Vigenère Cipher

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

Expressed numerically, we have the following result.

key	3	4	2	4	15	19	8	21	4	3	4	2	4	15
plaintext	22	4	0	17	4	3	8	18	2	14	21	4	17	4
ciphertext	25	8	2	21	19	22	16	13	6	17	25	6	21	19

key	19	8	21	4	3	4	2	4	15	19	8	21	4
plaintext	3	18	0	21	4	24	14	20	17	18	4	11	5
ciphertext	22	0	21	25	7	2	16	24	6	11	12	6	9

Vigenère Autokey System

- A keyword is concatenated with the plaintext itself to provide a running key
- Example:

  key: deceptivewearediscoveredsav

  plaintext: wearediscoveredsaveyourself

  ciphertext: ZICVTWQNGKZEIIGASXSTSLVVWLA
- Even this scheme is vulnerable to cryptanalysis
  - Because the key and the plaintext share the same frequency distribution of letters, a statistical technique can be applied

Vernam Cipher

Figure 2.7 Vernam Cipher

One-Time Pad

- Improvement to Vernam cipher proposed by an Army Signal Corp officer, Joseph Mauborgne
- Use a random key that is as long as the message so that the key need not be repeated
- Key is used to encrypt and decrypt a single message and then is discarded
- Each new message requires a new key of the same length as the new message
- Scheme is unbreakable
  - Produces random output that bears no statistical relationship to the plaintext
  - Because the ciphertext contains no information whatsoever about the plaintext, there is simply no way to break the code

Difficulties

- The one-time pad offers complete security but, in practice, has two fundamental difficulties:
  - There is the practical problem of making large quantities of random keys
    - Any heavily used system might require millions of random characters on a regular basis
  - Mammoth key distribution problem
    - For every message to be sent, a key of equal length is needed by both sender and receiver
- Because of these difficulties, the one-time pad is of limited utility
  - Useful primarily for low-bandwidth channels requiring very high security

Rail Fence Cipher

- Simplest transposition cipher
- Plaintext is written down as a sequence of diagonals and then read off as a sequence of rows
- To encipher the message "meet me after the toga party" with a rail fence of depth 2, we would write:

  m e m a t r h t g p r y

  e t e f e t e o a a t

  Encrypted message is:

  MEMATRHTGPRYETEFETEOAAT

Row Transposition Cipher

- Is a more complex transposition
- Write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns
  - The order of the columns then becomes the key to the algorithm

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

Row Transposition Cipher…

- The transposition cipher can be made significantly more secure by performing more than one stage of transposition.
  - The order of the columns then becomes the key to the algorithm

Key: 4 3 1 2 5 6 7

Plaintext:

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

Ciphertext:

NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

Rotor Machines

Figure 2.8 Three-Rotor Machine With Wiring Represented by Numbered Contacts

Steganography

3rd March

Dear George,

Greetings to all at Oxford. Many thanks for your letter and for the Summer examination package. All Entry Forms and Fees Forms should be ready for final despatch to the Syndicate by Friday 20th or at the very latest, I'm told, by the 21st. Admin has improved here, though there's room for improvement still; just give us all two or three more years and we'll really show you! Please don't let these wretched 16+ proposals destroy your basic O and A pattern. Certainly this sort of change, if implemented immediately, would bring chaos.

Sincerely yours,

Figure 2.9 A Puzzle for Inspector Morse
(from *The Silent World of Nicholas Quinn*, by Colin Dexter)

Other Steganography Techniques

- Character marking
  - Selected letters of printed or typewritten text are over-written in pencil
  - The marks are ordinarily not visible unless the paper is held at 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 up in front of a light
- Typewriter correction ribbon
  - Used between lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light

Summary

- Symmetric Cipher Model
  - Cryptography
  - Cryptanalysis and Brute-Force Attack
- Transposition techniques
- Rotor machines

- Substitution techniques
  - Caesar cipher
  - Monoalphabetic ciphers
  - Playfair cipher
  - Hill cipher
  - Polyalphabetic ciphers
  - One-time pad
- Steganography

A can proof
B get Msg

$$C = E(K, X)$$

$$X = D(K, C)$$

$C = E(K, P)$

~~$C = E(3, P) = E(3, 0) \bmod 26$~~

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

$= (\overset{A}{0} + 3) \bmod 26 = 3$

$= D$

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

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

$(3 - 3) \bmod 26 = 0 \bmod 26 = 0 \Rightarrow A$

① Repeating plain text

balloon

balx

② Same Row go Right

A → R

R → M

③ M → C U → M Same Column Replaced By letter beneath

④ Pair Replace by its own Row + colum

H → B S → P E A

I M

Brute force

1. Encryption + Decryption Algo known.
2. 25 keys to try.
3. language of plaintext known

⇒ we assume Algorithm known

⇒ Impractical if key is large
DES algo 168 bit key 2^{168}

⇒ Plaintext unknown Output can't recognizable

Monoalphabetic Cipher

$S = \{a, b, c\}$

Sin permutation of S

abc, acb, bac, bca, cab, cba

3 element Means 3! permutation

n element Means n! permutation

26 element $26! = 4 \times 10^{26}$ keys

$$\text{Freq} = \frac{\text{number of occurrence}}{\text{Total Element}} \times 100$$

$$J = \frac{1}{120} \times 100 = 0.83$$

$$Y = \frac{2}{120} \times 100 = 1.67$$

Hill Cipher Example

$$C = E(K, P) = PK \bmod 26$$

$$P = D(K, P) = CK^{-1} \bmod 26$$

Plaintext

15	0	24
p	a	y

Key

$$\begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

$$= 15 \times 17 + 0 \times 21 + 24 \times 2 = 303$$

$$= 15 \times 17 + 0 \times 18 + 24 \times 2 = 303$$

$$= 15 \times 5 + 0 \times 21 + 24 \times 19 = 531$$

$$C = (303 \quad 303 \quad 531) \bmod 26 = \begin{matrix} 17 & 17 & 11 \\ R & R & L \end{matrix}$$

$$P = CK^{-1} \bmod 26$$

C

17	17	11
R	R	L

K^{-1}

$$\begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}$$

$$= 17 \times 4 + 17 \times 15 + 11 \times 24 = 587$$

$$= 17 \times 9 + 17 \times 17 + 11 \times 0 = 442$$

$$= 17 \times 15 + 17 \times 6 + 11 \times 17 = 544$$

$$P = (587 \quad 442 \quad 544) \bmod 26 = \begin{matrix} 15 & 0 & 24 \\ p & a & y \end{matrix}$$

Vigenere Cipher

Key = d e c

P = w e a

↓ ↓ ↓

Z I C

a = 0 b = 1 c = 2 d = 3

e = 4

① w x y z

1 2 3

② e f g h i

1 2 3 4

③ a b c

1 2

Chapter 3

Block Ciphers and the Data Encryption Standard

Stream Cipher

Encrypts a digital data stream one bit or one byte at a time

Examples:

- Autokeyed Vigenère cipher
- Vernam cipher

In the ideal case a one-time pad version of the Vernam cipher would be used, in which the keystream is as long as the plaintext bit stream

If the cryptographic keystream is random, then this cipher is unbreakable by any means other than acquiring the keystream

- Keystream must be provided to both users in advance via some independent and secure channel
- This introduces insurmountable logistical problems if the intended data traffic is very large

For practical reasons the bit-stream generator must be implemented as an algorithmic procedure so that the cryptographic bit stream can be produced by both users

It must be computationally impractical to predict future portions of the bit stream based on previous portions of the bit stream

The two users need only share the generating key and each can produce the keystream

Block Cipher

A block of plaintext is treated as a whole and used to produce a ciphertext block of equal length

Typically a block size of 64 or 128 bits is used

As with a stream cipher, the two users share a symmetric encryption key

The majority of network-based symmetric cryptographic applications make use of block ciphers

Stream Cipher and Block Cipher

(a) Stream Cipher Using Algorithmic Bit Stream Generator

(b) Block Cipher

Figure 3.1 Stream Cipher and Block Cipher

Figure 3.2 General *n*-bit-*n*-bit Block Substitution (shown with *n* = 4)

Table 3.1

Encryption and Decryption Tables for Substitution Cipher of Figure 3.2

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Feistel Cipher

- Proposed the use of a cipher that alternates substitutions and permutations

Substitutions
- Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements

Permutation
- No elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed

- Is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates confusion and diffusion functions

- Is the structure used by many significant symmetric block ciphers currently in use

Diffusion and Confusion

- Terms introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system
  - Shannon's concern was to thwart cryptanalysis based on statistical analysis

Diffusion

- The statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext
- This is achieved by having each plaintext digit affect the value of many ciphertext digits

Confusion

- Seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible
- Even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key

Feistel Cipher Structure

Figure 3.3 Feistel Encryption and Decryption (16 rounds)

Feistel Cipher Design Features

- Block size
  - Larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm
- Key size
  - Larger key size means greater security but may decrease encryption/decryption speeds
- Number of rounds
  - The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security
- Subkey generation algorithm
  - Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis
- Round function F
  - Greater complexity generally means greater resistance to cryptanalysis
- Fast software encryption/decryption
  - In many cases, encrypting is embedded in applications or utility functions in such a way as to preclude a hardware implementation; accordingly, the speed of execution of the algorithm becomes a concern
- Ease of analysis
  - If the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength

Feistel Example

Figure 3.4 Feistel Example

Data Encryption Standard (DES)

- Issued in 1977 by the National Bureau of Standards (now NIST) as Federal Information Processing Standard 46
- Was the most widely used encryption scheme until the introduction of the Advanced Encryption Standard (AES) in 2001
- Algorithm itself is referred to as the Data Encryption Algorithm (DEA)
  - Data are encrypted in 64-bit blocks using a 56-bit key
  - The algorithm transforms 64-bit input in a series of steps into a 64-bit output
  - The same steps, with the same key, are used to reverse the encryption

Figure 3.5 General Depiction of DES Encryption Algorithm

DES Encryption Algorithm

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

Table 3.2 DES Example

(Table can be found on page 75 in textbook)

Round	*Ki*	*Li*	*Ri*
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP–1		da02ce3a	89ecac3b

Note: DES subkeys are shown as eight 6-bit values in hex format

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33

Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP−1	da02ce3a89ecac3b 057cde97d7683f2a	32

Table 3.3 Avalanche Effect in DES: Change in Plaintext

Round		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		δ
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP−1	da02ce3a89ecac3b ee92b50606b62b0b	30

Table 3.4 Avalanche Effect in DES: Change in Key

Table 3.5

Average Time Required for Exhaustive Key Search

Table 3.5 Average Time Required for Exhaustive Key Search

Key Size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 Decryptions/s	Time Required at 10^{13} Decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns $= 1.125$ years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns $= 5.3 \times 10^{21}$ years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns $= 5.8 \times 10^{33}$ years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns $= 9.8 \times 10^{40}$ years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns $= 1.8 \times 10^{60}$ years	1.8×10^{56} years
26 characters (permutation)	Monoalphabetic	$2! = 4 \times 10^{26}$	2×10^{26} ns $= 6.3 \times 10^{9}$ years	6.3×10^{6} years

Strength of DES

- Timing attacks
  - One in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts
  - Exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs
  - So far it appears unlikely that this technique will ever be successful against DES or more powerful symmetric ciphers such as triple DES and AES

Block Cipher Design Principles: Number of Rounds

The greater the number of rounds, the more difficult it is to perform cryptanalysis

In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack

If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than a brute-force key search

Block Cipher Design Principles: Design of Function F

- The heart of a Feistel block cipher is the function F
- The more nonlinear F, the more difficult any type of cryptanalysis will be
- The SAC and BIC criteria appear to strengthen the effectiveness of the confusion function

The algorithm should have good avalanche properties

Strict avalanche criterion (SAC)

States that any output bit j of an S-box should change with probability 1/2 when any single input bit i is inverted for all i , j

Bit independence criterion (BIC)

States that output bits j and k should change independently when any single input bit i is inverted for all i , j , and k

Block Cipher Design Principles: Key Schedule Algorithm

- With any Feistel block cipher, the key is used to generate one subkey for each round
- In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key
- It is suggested that, at a minimum, the key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion

Summary

- Traditional Block Cipher Structure
  - Stream ciphers
  - Block ciphers
  - Feistel cipher
- The Data Encryption Standard (DES)
  - Encryption
  - Decryption
  - Avalanche effect

- The strength of DES
  - Use of 56-bit keys
  - Nature of the DES algorithm
  - Timing attacks
- Block cipher design principles
  - DES design criteria
  - Number of rounds
  - Design of function F
  - Key schedule algorithm

Plaintext: 02468aceeca86420

Number	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
P	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0

Row	Binary	Hexa
1	0 1 0 1 1 0 1 0	5 A
2	0 0 0 0 0 0 0 0	0 0
3	0 1 0 1 1 0 1 0	5 A
4	0 0 0 0 0 0 0 0	0 0
5	0 0 1 1 1 1 0 0	3 C
6	1 1 1 1 0 0 0 0	F 0
7	0 0 1 1 1 1 0 0	3 C
8	0 0 0 0 1 1 1 1	0 F

58	50
60	52
62	54
64	56
57	49
59	51
61	53
63	55

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0

IP

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

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Motivation of Feistel Cipher

⇒ In block Cipher, n bit of plain text produce n bit of cipher text.

⇒ 2^n combination of Encryption Reversible

$n = 2 \Rightarrow 4$ combination

Reversible Mapping

P	C
00	11
01	10
10	00
11	01

Irreversible Mapping

P	C
00	11
01	10
10	01
11	01

Chapter 4

Basic Concepts in Number Theory and Finite Fields

Divisibility

- We say that a nonzero b **divides** a if $a = mb$ for some m, where a, b, and m are integers
- b divides a if there is no remainder on division
- The notation $b \mid a$ is commonly used to mean b divides a
- If $b \mid a$ we say that b is a **divisor** of a

The positive divisors of 24 are 1, 2, 3, 4, 6, 8, 12, and 24
$13 \mid 182$; $-5 \mid 30$; $17 \mid 289$; $-3 \mid 33$; $17 \mid 0$

Properties of Divisibility

- If $a \mid 1$, then $a = \pm 1$
- If $a \mid b$ and $b \mid a$, then $a = \pm b$
- Any $b \neq 0$ divides 0
- If $a \mid b$ and $b \mid c$, then $a \mid c$

$$\boxed{11 \mid 66 \text{ and } 66 \mid 198 = 11 \mid 198}$$

- If $b \mid g$ and $b \mid h$, then $b \mid (mg + nh)$ for arbitrary integers m and n

Properties of Divisibility

- To see this last point, note that:
  - If $b \mid g$, then g is of the form $g = b * g_1$ for some integer g_1
  - If $b \mid h$, then h is of the form $h = b * h_1$ for some integer h_1
- So:
  - $mg + nh = mbg_1 + nbh_1 = b * (mg_1 + nh_1)$

  and therefore b divides $mg + nh$

$b = 7$; $g = 14$; $h = 63$; $m = 3$; $n = 2$
$7 \mid 14$ and $7 \mid 63$.
To show $7 \mid (3 * 14 + 2 * 63)$,
we have $(3 * 14 + 2 * 63) = 7(3 * 2 + 2 * 9)$,
and it is obvious that $7 \mid (7(3 * 2 + 2 * 9))$.

Division Algorithm

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

$$a = qn + r \qquad 0 \le r < n;\ q = [a/n]$$

Figure 4.1 The Relationship $a = qn + r;\ 0 \le r < n$

Euclidean Algorithm

- One of the basic techniques of number theory
- Procedure for determining the greatest common divisor of two positive integers
- Two integers are **relatively prime** if their only common positive integer factor is 1

Greatest Common Divisor (GCD)

- The greatest common divisor of *a* and *b* is the largest integer that divides both *a* and *b*
- We can use the notation gcd(*a*,*b*) to mean the **greatest common divisor** of *a* and *b*
- We also define gcd(0,0) = 0
- Positive integer *c* is said to be the gcd of *a* and *b* if:
  - *c* is a divisor of *a* and *b*
  - Any divisor of *a* and *b* is a divisor of *c*
- An equivalent definition is:

$$\gcd(a,b) = \max[k, \text{ such that } k \mid a \text{ and } k \mid b]$$

GCD

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

$$\boxed{\gcd(60, 24) = \gcd(60, -24) = 12}$$

- Also, because all nonzero integers divide 0, we have $\gcd(a,0) = |a|$
- We stated that two integers a and b are relatively prime if their only common positive integer factor is 1; this is equivalent to saying that a and b are relatively prime if $\gcd(a,b) = 1$

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

GCD

- Euclid algorithm for easily finding the greatest common divisor of two integers.
- Suppose we have integers *a*, *b* such that *d* = gcd(*a*, *b*).
  - Now dividing *a* by *b* and applying the division algorithm

$$a = q_1 b + r_1 \qquad 0 \le r_1 < b$$

$$b = q_2 r_1 + r_2 \qquad 0 \le r_2 < r_1$$

$$\left.\begin{array}{ll}
a = q_1 b + r_1 & 0 < r_1 < b \\
b = q_2 r_1 + r_2 & 0 < r_2 < r_1 \\
r_1 = q_3 r_2 + r_3 & 0 < r_3 < r_2 \\
\vdots & \vdots \\
r_{n-2} = q_n r_{n-1} + r_n & 0 < r_n < r_{n-1} \\
r_{n-1} = q_{n+1} r_n + 0 & \\
d = \gcd(a, b) = r_n &
\end{array}\right\}$$

Euclidean Algorithm Example

To find $d = \gcd(a,b) = \gcd(1160718174, 316258250)$		
$a = q_1 b + r_1$	$1160718174 = 3 \times 316258250 + 211943424$	$d = \gcd(316258250, 211943424)$
$b = q_2 r_1 + r_2$	$316258250 = 1 \times 211943424 + 104314826$	$d = \gcd(211943424, 104314826)$
$r_1 = q_3 r_2 + r_3$	$211943424 = 2 \times 104314826 + 3313772$	$d = \gcd(104314826, 3313772)$
$r_2 = q_4 r_3 + r_4$	$104314826 = 31 \times 3313772 + 1587894$	$d = \gcd(3313772, 1587894)$
$r_3 = q_5 r_4 + r_5$	$3313772 = 2 \times 1587894 + 137984$	$d = \gcd(1587894, 137984)$
$r_4 = q_6 r_5 + r_6$	$1587894 = 11 \times 137984 + 70070$	$d = \gcd(137984, 70070)$
$r_5 = q_7 r_6 + r_7$	$137984 = 1 \times 70070 + 67914$	$d = \gcd(70070, 67914)$
$r_6 = q_8 r_7 + r_8$	$70070 = 1 \times 67914 + 2156$	$d = \gcd(67914, 2156)$
$r_7 = q_9 r_8 + r_9$	$67914 = 31 \times 2516 + 1078$	$d = \gcd(2156, 1078)$
$r_8 = q_{10} r_9 + r_{10}$	$2156 = 2 \times 1078 + 0$	$d = \gcd(1078, 0) = 1078$
Therefore, $d = \gcd(1160718174, 316258250) = 1078$		

Table 4.1
Euclidean Algorithm Example

Dividend	Divisor	Quotient	Remainder
$a = 1160718174$	$b = 316258250$	$q_1 = 3$	$r_1 = 211943424$
$b = 316258250$	$r_1 = 211943424$	$q_2 = 1$	$r_2 = 104314826$
$r_1 = 211943424$	$r_2 = 104314826$	$q_3 = 2$	$r_3 = 3313772$
$r_2 = 104314826$	$r_3 = 3313772$	$q_4 = 31$	$r_4 = 1587894$
$r_3 = 3313772$	$r_4 = 1587894$	$q_5 = 2$	$r_5 = 137984$
$r_4 = 1587894$	$r_5 = 137984$	$q_6 = 11$	$r_6 = 70070$
$r_5 = 137984$	$r_6 = 70070$	$q_7 = 1$	$r_7 = 67914$
$r_6 = 70070$	$r_7 = 67914$	$q_8 = 1$	$r_8 = 2156$
$r_7 = 67914$	$r_8 = 2156$	$q_9 = 31$	$r_9 = 1078$
$r_8 = 2156$	$r_9 = 1078$	$q_{10} = 2$	$r_{10} = 0$

(This table can be found on page 91 in the textbook)

Modular Arithmetic

- The modulus
  - If *a* is an integer and *n* is a positive integer, we define *a* mod *n* to be the remainder when *a* is divided by *n*; the integer *n* is called the **modulus**
  - thus, for any integer *a:*

$$a = qn + r \qquad 0 \le r < n; \; q = [a/n]$$

$$a = [a/n] * n + (a \bmod n)$$

11 mod 7 = 4; - 11 mod 7 = 3

Modular Arithmetic

- Congruent modulo *n*
  - Two integers *a* and *b* are said to be **congruent modulo *n*** if $(a \bmod n) = (b \bmod n)$
  - This is written as $a = b \pmod{n}$
  - Note that if $a = 0 \pmod{n}$, then $n \mid a$

$73 = 4 \pmod{23}$; $21 = -9 \pmod{10}$

Properties of Congruences

- Congruences have the following properties:

  1. $a = b \pmod{n}$ if $n \mid (a - b)$

  2. $a = b \pmod{n}$ implies $b = a \pmod{n}$

  3. $a = b \pmod{n}$ and $b = c \pmod{n}$ imply $a = c \pmod{n}$

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

23 = 8 (mod 5) because 23 - 8 = 15 = 5 * 3
- 11 = 5 (mod 8) because - 11 - 5 = - 16 = 8 * (- 2)
81 = 0 (mod 27) because 81 - 0 = 81 = 27 * 3

Modular Arithmetic

- Modular arithmetic exhibits the following properties:

  1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
  2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
  3. $[(a \bmod n) * (b \bmod n)] \bmod n = (a * b) \bmod n$

Remaining Properties:

- Examples of the three remaining properties:

11 mod 8 = 3; 15 mod 8 = 7

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

(11 + 15) mod 8 = 26 mod 8 = 2

[(11 mod 8) - (15 mod 8)] mod 8 = - 4 mod 8 = 4

(11 - 15) mod 8 = - 4 mod 8 = 4

[(11 mod 8) * (15 mod 8)] mod 8 = 21 mod 8 = 5

(11 * 15) mod 8 = 165 mod 8 = 5

Table 4.2(a) Arithmetic Modulo 8

+	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

Table 4.2(b)
Multiplication Modulo 8

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

Table 4.2(c)

Additive and Multiplicative Inverses Modulo 8

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

Table 4.3

Properties of Modular Arithmetic for Integers in Z_n

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

Extended Euclidean Algorithm Example

$y \backslash x$	−3	−2	−1	0	1	2	3
−3	−216	−174	−132	−90	−48	−6	36
−2	−186	−144	−102	−60	−18	24	66
−1	−156	−114	−72	−30	12	54	96
0	−126	−84	−42	0	42	84	126
1	−96	−54	−12	30	72	114	156
2	−66	−24	18	60	102	144	186
3	−36	6	48	90	132	174	216

Table 4.4

Extended Euclidean Algorithm Example

i	r_i	q_i	x_i	Y_i
–1	1759		1	0
0	550		0	1
1	109	3	1	–3
2	5	5	–5	16
3	4	21	106	–339
4	1	1	–111	355
5	0	4		

Result: $d = 1$; $x = -111$; $y = 355$

Summary

- Divisibility and the division algorithm
- The Euclidean algorithm
- Modular arithmetic
- Groups, rings, and fields

- Finite fields of the form GF(p)
- Polynomial arithmetic
- Finite fields of the form GF(2^n)

The 10 integers in Z_{10}, that is the integers 0 through 9, can be reconstructed from their two residues modulo 2 and 5 (the relatively prime factors of 10). Say the known residues of a decimal digit x are $r_2 = 0$ and $r_5 = 3$; that is, $x \bmod 2 = 0$ and $x \bmod 5 = 3$. Therefore, x is an even integer in Z_{10} whose remainder, on division by 5, is 3. The unique solution is $x = 8$.

Divisibility

$a = 4 \quad b = 2 \quad m = 2$

$$\begin{array}{r} 2 \\ 2 \overline{)\,4} \\ 4 \\ \hline 0 \end{array} \qquad \begin{array}{r} m \\ b \overline{)\,a} \\ a \\ \hline 0 \end{array}$$

$b \mid a$

b is divisor of a

$13 \mid 182$

$$\begin{array}{r} 14 \\ 13 \overline{)\,182} \\ 182 \\ \hline 0 \end{array}$$

$17 \mid 289$

$$\begin{array}{r} 17 \\ 17 \overline{)\,289} \\ 289 \\ \hline 0 \end{array}$$

$17 \mid 0$

$$\begin{array}{r} 0 \\ 17 \overline{)\,0} \\ 0 \\ \hline 0 \end{array}$$

Division Algo

$$\begin{array}{r} 4 \\ 15 \overline{)\,70} \\ 60 \\ \hline 10 \end{array} \qquad \begin{array}{r} q \\ n \overline{)\,a} \\ qn \\ \hline r \end{array}$$

$0 \le r < n$

$a = 11$

$n = 7$

$$\begin{array}{r} 1 \\ 7 \overline{)\,11} \\ 7 \\ \hline 4 \end{array}$$

$qn + r$

$= 1 \times 7 + 4$

Euclidean Algo

(A)

$d = \gcd(a, b)$

$a = 100 \qquad b = 50$

$d = \gcd(100, 50)$

$a = q_1 b + r_1 \Rightarrow 100 = \underset{q_1}{2} \times \underset{b}{50} + \underset{r_1}{0}$

$d = 50$

(B)

$a = 1071 \qquad b = 462$

$\gcd(1071, 462)$

(1) $a = q_1 b + r_1$

$1071 = \underset{q_1}{2} \times \underset{b}{462} + \underset{r_1}{147}$

Divide b by r_1

(2) $b = q_2 r_1 + r_2$

$462 = 3 \times 147 + 21$

Divide r_1 by r_2

(3) $r_1 = q_3 r_2 + r_3$

$147 = 7 \times 21 + 0$

$d = 21$

Modular Arithmetic

$a \bmod n = r$

$a = 11 \qquad n = 7$

$$\begin{array}{r} 1 \\ 7 \overline{)11} \\ 7 \\ \hline 4 \end{array}$$

$11 \bmod 7 = 4$

$a = -11 \qquad n = 7$

$$\begin{array}{r} 2 \\ 7 \overline{)-11} \\ 14 \\ \hline 3 \end{array}$$

$-11 \bmod 7 = 3$

Congruent Modulo

(1)

$a = 73 \qquad b = 4 \qquad n = 23$

$a \bmod n$

$73 \bmod 23$

$= 4$

$b \bmod n$

$4 \bmod 23$

$= 4$

$a \bmod n = b \bmod n$

$a \equiv b \bmod n$

$73 \equiv 4 \bmod 23$

(2)

$7 \equiv 4 \bmod 3$

$$\begin{array}{r} 2 \\ 3 \overline{)7} \\ 6 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ 3 \overline{)4} \\ 3 \\ \hline 1 \end{array}$$

Both 7 and 4 have remainder of 1 when divided by 3

Euclidean Algo Revisted

$a \geq b \geq 0$

$$\gcd(a,b) = \gcd(b, a \bmod b)$$

$$\gcd(55,22) = \gcd(22, 55 \bmod 22)$$
$$= \gcd(22, 11)$$
$$= \gcd(11, 0) = 11$$

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

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

Extended Euclidean Algo

$\gcd(42,30)$ $\quad a = 42 \quad b = 30$

$$ax + by = d = \gcd(a,b)$$

$$42x + 30y$$

$x = 0$	$y = 0$
$y = -3 \Rightarrow -90$	$x = -3 \Rightarrow -126$
$y = -2 \Rightarrow -60$	$x = -2 \Rightarrow -84$
$y = -1 \Rightarrow -30$	$x = -1 \Rightarrow -42$
$y = 0 \Rightarrow 0$	$x = 0 \Rightarrow 0$
$y = 1 \Rightarrow 30$	$x = 1 \Rightarrow 42$
$y = 2 \Rightarrow 60$	$x = 2 \Rightarrow 84$
$y = 3 \Rightarrow 90$	$x = 3 \Rightarrow 126$

Extended Euclidean Algo Example

i	r	x	y_0	q_i
-1	1759	1		-
0	550	0	1	-
1	109	1	-3	3
2	5	-5	16	5

(1)

$r_i = ax_i + by_i$

$r_{-1} = a$ $\quad a = 1759 \quad x_{-1} = 1 \quad y_{-1} = 0$

$r_0 = b$ $\quad b = 550 \quad x_0 = 0 \quad y_0 = 1$

(2)

$r_1 = ax_1 + by_1$

$x_i = x_{(i-2)} - q_i x_{i-1} \Rightarrow x_{-1} - q_1 x_0$

$x_1 = x_{-1} - q_1 x_0$

$1 - 3(0) = 1$

$y_1 = y_{-1} - q_1 y_0$

$0 - 3(1) = -3$

$r_1 = 1759 + 550(-3) \Rightarrow 109$

$q_1 = |a/b|$

$q_1 = \frac{1759}{550} = 3$

$$\begin{array}{r|l} & 3 \\ 550 & 1759 \\ & 1650 \\ \hline & 109 \end{array}$$

(3)

$x_2 = x_0 - q_2 x_1$

$0 - q_2(1) = -q_2 \Rightarrow -5$

$y_2 = y_0 - q_2 y' \Rightarrow 1 - 5(-3) = 16$

$r_2 = ax_2 + by_2$

$= 1759(-5) + 550(16) = 5$

$\Rightarrow r_2 = b \bmod r_1 \quad , r_3 = r_1 \bmod r_2$

$q_2 = \left|\frac{b}{r_1}\right|$

$$\begin{array}{r|l} & 5 \\ 109 & 550 \\ & 545 \\ \hline & 5 \end{array}$$

PRIME Number

$91 = 7 * 13$

$3600 = 2^4 \times 3^2 \times 5^2$

$11011 = 7 \times 11^2 \times 13$

Fermat Theorem

① $p = 5 \quad a = 3$

$a^p = a \bmod p$

$3^5 = 3 \bmod 5$

$243 = 3 \bmod 5$

② $p = 5 \quad a = 10$

$a^p = 10^5 = 100000 = 10 \bmod 5$

Euler Totient Function $\varphi(n)$

$\varphi(1) = 1$

$\varphi(P) = P - 1$

$\varphi(19) = 19 - 1 = 18$

$\varphi(29) = 28$

$n = 10$

1, 2, 3, 4, 5, 6, 7, 8, 9

Factors

1, 10
1
1, 2, 5, 10 — Relative Prime

2, 10
1, 2
1, 2, 5, 10 — Not Relative Prime

3, 10
1, 3
1, 2, 5, 10 — Relative Prime

4, 10
1, 2, 4
1, 2, 5, 10 — Not Relative Prime

5, 10
1, 5
1, 2, 5, 10 — Not Relative Prime

10 = 1, 3, 7, 9

$\varphi(10) = 4$

$\varphi(15) = \varphi(3) \times \varphi(5)$

$\varphi(p) = p-1$

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

$= 2 \times 4 = 8$

$\varphi(21) = \varphi(3) \times \varphi(7)$

$(3-1) \times (7-1)$

$= 12$

Euler Theorem

$a^{\varphi(n)} = 1 \bmod n$

① $a = 3$ $n = 10$

$\varphi(n) = \varphi(10) = 4$

$a^{\varphi(n)} = 3^4 = 81 = 1 \bmod 10$

$= 1 \bmod n$

② $a = 2$ $n = 11$

$\varphi(11) = 10$

$a^{\varphi(n)} = 2^{10} = 1024 = 1 \bmod 11$

$= 1 \bmod n$

Miller Rabin Algo

(1) $n = 29$ Find K and q

$n-1 = 2^K q$ $K > 0$

$28 = 2^2 \times 7$ $K = 2$ $q = 7$

(2) Select a $a > 1$

$a = 10$ $a < n-1$

(3) $a^q \bmod n = 10^7 \bmod 29 = 17$

Stop 1 or $n-1$

(4) Loop $j = 0$

$a^{2^j q} \bmod n \Rightarrow a^q \bmod n$

$j = 1$

$a^{2 \times 7} \bmod n = 10^{14} \bmod 29 = 28$

May be Prime

Try again $a = 2$

(3) $a^q \bmod n = 2^7 \bmod 29 = 12$

Stop 1 or $n-1$

(4) $j = 0$ $a^q \bmod n = 12$

$j = 1$ $a^{2q} \bmod n$

$= 2^{2 \times 7} \bmod 29 = 28$

Go through 1 to 28 and get same Result

means n being Prime

Chapter 8

More Number Theory

Prime Numbers

- Prime numbers only have divisors of 1 and itself
  - They cannot be written as a product of other numbers
- Prime numbers are central to number theory
- Any integer a > 1 can be factored in a unique way as

$$a = {}_{p1}{}^{a1} * p_2{}^{a2} * \ldots * p_t{}^{at}$$

where $p_1 < p_2 < \ldots < p_t$ are prime numbers and where each a_i is a positive integer

- This is known as the fundamental theorem of arithmetic

Table 8.1
Primes Under 2000

2	101	211	307	401	503	601	701	809	907	1009	1103	1201	1301	1409	1511	1601	1709	1801	1901
3	103	223	311	409	509	607	709	811	911	1013	1109	1213	1303	1423	1523	1607	1721	1811	1907
5	107	227	313	419	521	613	719	821	919	1019	1117	1217	1307	1427	1531	1609	1723	1823	1913
7	109	229	317	421	523	617	727	823	929	1021	1123	1223	1319	1429	1543	1613	1733	1831	1931
11	113	233	331	431	541	619	733	827	937	1031	1129	1229	1321	1433	1549	1619	1741	1847	1933
13	127	239	337	433	547	631	739	829	941	1033	1151	1231	1327	1439	1553	1621	1747	1861	1949
17	131	241	347	439	557	641	743	839	947	1039	1153	1237	1361	1447	1559	1627	1753	1867	1951
19	137	251	349	443	563	643	751	853	953	1049	1163	1249	1367	1451	1567	1637	1759	1871	1973
23	139	257	353	449	569	647	757	857	967	1051	1171	1259	1373	1453	1571	1657	1777	1873	1979
29	149	263	359	457	571	653	761	859	971	1061	1181	1277	1381	1459	1579	1663	1783	1877	1987
31	151	269	367	461	577	659	769	863	977	1063	1187	1279	1399	1471	1583	1667	1787	1879	1993
37	157	271	373	463	587	661	773	877	983	1069	1193	1283		1481	1597	1669	1789	1889	1997
41	163	277	379	467	593	673	787	881	991	1087		1289		1483		1693			1999
43	167	281	383	479	599	677	797	883	997	1091		1291		1487		1697			
47	173	283	389	487		683		887		1093		1297		1489		1699			
53	179	293	397	491		691				1097				1493					
59	181			499										1499					
61	191																		
67	193																		
71	197																		
73	199																		
79																			
83																			
89																			
97																			

Fermat's Theorem

- States the following:
  - If *p* is prime and *a* is a positive integer not divisible by *p* then

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

- Sometimes referred to as Fermat's Little Theorem
- An alternate form is:
  - If *p* is prime and *a* is a positive integer then

$$a^{p} = a \pmod{p}$$

- Plays an important role in public-key cryptography

Table 8.2

Some Values of Euler's Totient Function ø(*n*)

n	$\phi(n)$
1	1
2	1
3	2
4	2
5	4
6	2
7	6
8	4
9	6
10	4

n	$\phi(n)$
11	10
12	4
13	12
14	6
15	8
16	8
17	16
18	6
19	18
20	8

n	$\phi(n)$
21	12
22	10
23	22
24	8
25	20
26	12
27	18
28	12
29	28
30	8

Euler's Theorem

- States that for every *a* and *n* that are relatively prime:

$$a^{\varnothing(n)} = 1 (\text{mod } n)$$

- An alternative form is:

$$a^{\varnothing(n)+1} = a (\text{mod } n)$$

- Plays an important role in public-key cryptography

Miller-Rabin Algorithm

- Typically used to test a large number for primality
- Algorithm is:

```
TEST (n)
1. Find integers k, q, with k > 0, q odd, so that
   (n − 1 = 2^k q);
2. Select a random integer a, 1 < a < n − 1;
3. if a^q mod n = 1 then return("inconclusive");
4. for j = 0 to k − 1 do
5. if a^(2^j q) mod n = n − 1 then return("inconclusive");
6. return("composite");
```

Deterministic Primality Algorithm

- Prior to 2002 there was no known method of efficiently proving the primality of very large numbers
- All of the algorithms in use produced a probabilistic result
- In 2002 Agrawal, Kayal, and Saxena developed an algorithm that efficiently determines whether a given large number is prime
  - Known as the AKS algorithm
  - Does not appear to be as efficient as the Miller-Rabin algorithm

Chinese Remainder Theorem (CRT)

- Believed to have been discovered by the Chinese mathematician Sun-Tsu in around 100 A.D.
- One of the most useful results of number theory
- Says it is possible to reconstruct integers in a certain range from their residues modulo a set of pairwise relatively prime moduli
- Can be stated in several ways

Provides a way to manipulate (potentially very large) numbers mod M in terms of tuples of smaller numbers

- This can be useful when M is 150 digits or more
- However, it is necessary to know beforehand the factorization of M

Table 8.3

Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Table 8.4 Tables of Discrete Logarithms, Modulo 19

(a) Discrete logarithms to the base 2, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(b) Discrete logarithms to the base 3, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c) Discrete logarithms to the base 10, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d) Discrete logarithms to the base 13, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e) Discrete logarithms to the base 14, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9

(f) Discrete logarithms to the base 15, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\log_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	14	9

Summary

- Prime numbers
- Fermat's Theorem
- Euler's totient function
- Euler's Theorem
- Testing for primality
  - Miller-Rabin algorithm
  - A deterministic primality algorithm
  - Distribution of primes

- The Chinese Remainder Theorem
- Discrete logarithms
  - Powers of an integer, modulo *n*
  - Logarithms for modular arithmetic
  - Calculation of discrete logarithms

Chapter 5

Advanced Encryption Standard

Finite Field Arithmetic

- In the Advanced Encryption Standard (AES) all operations are performed on 8-bit bytes
- The arithmetic operations of addition, multiplication, and division are performed over the finite field.
- 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:
  - $a / b = a (b^{-1})$
- An example of a finite field (one with a finite number of elements) is the set Z_p consisting of all the integers {0, 1, ... , p - 1}, where p is a prime number and in which arithmetic is carried out modulo p

AES Encryption Process

No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Figure 5.1 AES Encryption Process

AES Data Structures

(a) Input, state array, and output

(b) Key and expanded key

Figure 5.2 AES Data Structures

Table 5.1
AES Parameters

Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

AES Encryption and Decryption

Figure 5.3 AES Encryption and Decryption

Detailed Structure

- Processes the entire data block as a single matrix during each round using substitutions and permutation
- The key that is provided as input is expanded into an array of forty-four 32-bit words, *w[i]*

Four different stages are used:

- Substitute bytes – uses an S-box to perform a byte-by-byte substitution of the block
- ShiftRows – a simple permutation
- MixColumns – a substitution that makes use of arithmetic operation over finite field
- AddRoundKey – a simple bitwise XOR of the current block with a portion of the expanded key

- The cipher begins and ends with an AddRoundKey stage
- Can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on
- Each stage is easily reversible
- The decryption algorithm makes use of the expanded key in reverse order, however the decryption algorithm is not identical to the encryption algorithm
- State is the same for both encryption and decryption
- Final round of both encryption and decryption consists of only three stages

Figure 5.4 AES Encryption Round

AES Byte Level Operations

Figure 5.5 AES Byte-Level Operations

Table 5.2

(a) S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(Table can be found on page 139 in textbook)

Table 5.2

(b) Inverse S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

(Table can be found on page 139 in textbook)

SubByte Transformation Example

Here is an example of the SubBytes transformation:

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

S-Box Rationale

- The S-box is designed to be resistant to known cryptanalytic attacks
- The Rijndael developers sought a design that has a low correlation between input bits and output bits and the property that the output is not a linear mathematical function of the input
- The nonlinearity is due to the use of the multiplicative inverse

Shift Row Transformation

(a) Shift row transformation

Figure 5.7 AES Row and Column Operations

(Figure can be found on page 144 in textbook)

Shift Row Example

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

MixColumn Transformation

(b) Mix column transformation

Figure 5.7 AES Row and Column Operations

(Figure can be found on page 144 in textbook)

Mix Columns Rationale

$$s'_{0,j} = (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$

$$s'_{1,j} = s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j} \oplus s_{3,j}$$

$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j})$$

$$s'_{3,j} = (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j})$$

The following is an example of MixColumns:

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

$\rightarrow$

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$$\begin{array}{llllll}
(\{02\} \cdot \{87\}) & \oplus (\{03\} \cdot \{6E\}) & \oplus \{46\} & \oplus \{A6\} & = \{47\} \\
\{87\} & \oplus (\{02\} \cdot \{6E\}) & \oplus (\{03\} \cdot \{46\}) & \oplus \{A6\} & = \{37\} \\
\{87\} & \oplus \{6E\} & \oplus (\{02\} \cdot \{46\}) & \oplus (\{03\} \cdot \{A6\}) & = \{94\} \\
(\{03\} \cdot \{87\}) & \oplus \{6E\} & \oplus \{46\} & \oplus (\{02\} \cdot \{A6\}) & = \{ED\}
\end{array}$$

$$\begin{array}{lcl}
\{02\} \cdot \{87\} & = & 0001\ 0101 \\
\{03\} \cdot \{6E\} & = & 1011\ 0010 \\
\{46\} & = & 0100\ 0110 \\
\{A6\} & = & \underline{1010\ 0110} \\
 & & 0100\ 0111 \ = \{47\}
\end{array}$$

Multiple by 2

```
0x00,0x02,0x04,0x06,0x08,0x0a,0x0c,0x0e,0x10,0x12,0x14,0x16,0x18,0x1a,0x1c,0x1e,
0x20,0x22,0x24,0x26,0x28,0x2a,0x2c,0x2e,0x30,0x32,0x34,0x36,0x38,0x3a,0x3c,0x3e,
0x40,0x42,0x44,0x46,0x48,0x4a,0x4c,0x4e,0x50,0x52,0x54,0x56,0x58,0x5a,0x5c,0x5e,
0x60,0x62,0x64,0x66,0x68,0x6a,0x6c,0x6e,0x70,0x72,0x74,0x76,0x78,0x7a,0x7c,0x7e,
0x80,0x82,0x84,0x86,0x88,0x8a,0x8c,0x8e,0x90,0x92,0x94,0x96,0x98,0x9a,0x9c,0x9e,
0xa0,0xa2,0xa4,0xa6,0xa8,0xaa,0xac,0xae,0xb0,0xb2,0xb4,0xb6,0xb8,0xba,0xbc,0xbe,
0xc0,0xc2,0xc4,0xc6,0xc8,0xca,0xcc,0xce,0xd0,0xd2,0xd4,0xd6,0xd8,0xda,0xdc,0xde,
0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xee,0xf0,0xf2,0xf4,0xf6,0xf8,0xfa,0xfc,0xfe,
0x1b,0x19,0x1f,0x1d,0x13,0x11,0x17,0x15,0x0b,0x09,0x0f,0x0d,0x03,0x01,0x07,0x05,
0x3b,0x39,0x3f,0x3d,0x33,0x31,0x37,0x35,0x2b,0x29,0x2f,0x2d,0x23,0x21,0x27,0x25,
0x5b,0x59,0x5f,0x5d,0x53,0x51,0x57,0x55,0x4b,0x49,0x4f,0x4d,0x43,0x41,0x47,0x45,
0x7b,0x79,0x7f,0x7d,0x73,0x71,0x77,0x75,0x6b,0x69,0x6f,0x6d,0x63,0x61,0x67,0x65,
0x9b,0x99,0x9f,0x9d,0x93,0x91,0x97,0x95,0x8b,0x89,0x8f,0x8d,0x83,0x81,0x87,0x85,
0xbb,0xb9,0xbf,0xbd,0xb3,0xb1,0xb7,0xb5,0xab,0xa9,0xaf,0xad,0xa3,0xa1,0xa7,0xa5,
0xdb,0xd9,0xdf,0xdd,0xd3,0xd1,0xd7,0xd5,0xcb,0xc9,0xcf,0xcd,0xc3,0xc1,0xc7,0xc5,
0xfb,0xf9,0xff,0xfd,0xf3,0xf1,0xf7,0xf5,0xeb,0xe9,0xef,0xed,0xe3,0xe1,0xe7,0xe5
```

Multiple by 3

```
0x00,0x03,0x06,0x05,0x0c,0x0f,0x0a,0x09,0x18,0x1b,0x1e,0x1d,0x14,0x17,0x12,0x11,
0x30,0x33,0x36,0x35,0x3c,0x3f,0x3a,0x39,0x28,0x2b,0x2e,0x2d,0x24,0x27,0x22,0x21,
0x60,0x63,0x66,0x65,0x6c,0x6f,0x6a,0x69,0x78,0x7b,0x7e,0x7d,0x74,0x77,0x72,0x71,
0x50,0x53,0x56,0x55,0x5c,0x5f,0x5a,0x59,0x48,0x4b,0x4e,0x4d,0x44,0x47,0x42,0x41,
0xc0,0xc3,0xc6,0xc5,0xcc,0xcf,0xca,0xc9,0xd8,0xdb,0xde,0xdd,0xd4,0xd7,0xd2,0xd1,
0xf0,0xf3,0xf6,0xf5,0xfc,0xff,0xfa,0xf9,0xe8,0xeb,0xee,0xed,0xe4,0xe7,0xe2,0xe1,
0xa0,0xa3,0xa6,0xa5,0xac,0xaf,0xaa,0xa9,0xb8,0xbb,0xbe,0xbd,0xb4,0xb7,0xb2,0xb1,
0x90,0x93,0x96,0x95,0x9c,0x9f,0x9a,0x99,0x88,0x8b,0x8e,0x8d,0x84,0x87,0x82,0x81,
0x9b,0x98,0x9d,0x9e,0x97,0x94,0x91,0x92,0x83,0x80,0x85,0x86,0x8f,0x8c,0x89,0x8a,
0xab,0xa8,0xad,0xae,0xa7,0xa4,0xa1,0xa2,0xb3,0xb0,0xb5,0xb6,0xbf,0xbc,0xb9,0xba,
0xfb,0xf8,0xfd,0xfe,0xf7,0xf4,0xf1,0xf2,0xe3,0xe0,0xe5,0xe6,0xef,0xec,0xe9,0xea,
0xcb,0xc8,0xcd,0xce,0xc7,0xc4,0xc1,0xc2,0xd3,0xd0,0xd5,0xd6,0xdf,0xdc,0xd9,0xda,
0x5b,0x58,0x5d,0x5e,0x57,0x54,0x51,0x52,0x43,0x40,0x45,0x46,0x4f,0x4c,0x49,0x4a,
0x6b,0x68,0x6d,0x6e,0x67,0x64,0x61,0x62,0x73,0x70,0x75,0x76,0x7f,0x7c,0x79,0x7a,
0x3b,0x38,0x3d,0x3e,0x37,0x34,0x31,0x32,0x23,0x20,0x25,0x26,0x2f,0x2c,0x29,0x2a,
0x0b,0x08,0x0d,0x0e,0x07,0x04,0x01,0x02,0x13,0x10,0x15,0x16,0x1f,0x1c,0x19,0x1a
```

AddRoundKey Transformation

- The 128 bits of State are bitwise XORed with the 128 bits of the round key
- Operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key
  - Can also be viewed as a byte-level operation

Rationale:

Is as simple as possible and affects every bit of State

The complexity of the round key expansion plus the complexity of the other stages of AES ensure security

AddRoundKey Transformation Example

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

$=$

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

Inputs for Single AES Round

Figure 5.8 Inputs for Single AES Round

AES Key Expansion

- Takes as input a four-word (16 byte) key and produces a linear array of 44 words (176) bytes
  - This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher
- Key is copied into the first four words of the expanded key
  - The remainder of the expanded key is filled in four words at a time
- Each added word *w*[i] depends on the immediately preceding word, *w[i – 1]*, and the word four positions back, w[i – 4]
  - In three out of four cases a simple XOR is used
  - For a word whose position in the *w* array is a multiple of 4, a more complex function is used

AES Key Expansion

Figure 5.9 AES Key Expansion

```
KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)   w[i] = (key[4*i], key[4*i+1],
                                      key[4*i+2],
                                      key[4*i+3]);
     for (i = 4; i < 44; i++)
     {
      temp = w[i - 1];
      if (i mod 4 = 0)   temp = SubWord (RotWord (temp))
                                ⊕ Rcon[i/4];
      w[i] = w[i-4] ⊕ temp
     }
}
```

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

For example, suppose that the round key for round 8 is

EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F

Then the first 4 bytes (first column) of the round key for round 9 are calculated as follows:

i (decimal)	temp	After RotWord	After SubWord	Rcon (9)	After XOR with Rcon	w[i−4]	w[i] = temp ⊕ w[i−4]
36	7F8D292F	8D292F7F	5DA515D2	1B000000	46A515D2	EAD27321	AC7766F3

Key Expansion Rationale

- The Rijndael developers designed the expansion key algorithm to be resistant to known cryptanalytic attacks
- Inclusion of a round-dependent round constant eliminates the symmetry between the ways in which round keys are generated in different rounds

The specific criteria that were used are:

- Knowledge of a part of the cipher key or round key does not enable calculation of many other round-key bits
- An invertible transformation
- Speed on a wide range of processors
- Usage of round constants to eliminate symmetries
- Simplicity of description

Table 5.3

AES Example Key Expansion

Key Words	Auxiliary Function
w0 = 0f 15 71 c9 w1 = 47 d9 e8 59 w2 = 0c b7 ad d6 w3 = af 7f 67 98	RotWord(w3)= 7f 67 98 af = x1 SubWord(x1)= d2 85 46 79 = y1 Rcon(1)= 01 00 00 00 y1 ⊕ Rcon(1)= d3 85 46 79 = z1
w4 = w0 ⊕ z1 = dc 90 37 b0 w5 = w4 ⊕ w1 = 9b 49 df e9 w6 = w5 ⊕ w2 = 97 fe 72 3f w7 = w6 ⊕ w3 = 38 81 15 a7	RotWord(w7)= 81 15 a7 38 = x2 SubWord(x4)= 0c 59 5c 07 = y2 Rcon(2)= 02 00 00 00 y2 ⊕ Rcon(2)= 0e 59 5c 07 = z2
w8 = w4 ⊕ z2 = d2 c9 6b b7 w9 = w8 ⊕ w5 = 49 80 b4 5e w10 = w9 ⊕ w6 = de 7e c6 61 w11 = w10 ⊕ w7 = e6 ff d3 c6	RotWord(w11)= ff d3 c6 e6 = x3 SubWord(x2)= 16 66 b4 8e = y3 Rcon(3)= 04 00 00 00 y3 ⊕ Rcon(3)= 12 66 b4 8e = z3
w12 = w8 ⊕ z3 = c0 af df 39 w13 = w12 ⊕ w9 = 89 2f 6b 67 w14 = w13 ⊕ w10 = 57 51 ad 06 w15 = w14 ⊕ w11 = b1 ae 7e c0	RotWord(w15)= ae 7e c0 b1 = x4 SubWord(x3)= e4 f3 ba c8 = y4 Rcon(4)= 08 00 00 00 y4 ⊕ Rcon(4)= ec f3 ba c8 = 4
w16 = w12 ⊕ z4 = 2c 5c 65 f1 w17 = w16 ⊕ w13 = a5 73 0e 96 w18 = w17 ⊕ w14 = f2 22 a3 90 w19 = w18 ⊕ w15 = 43 8c dd 50	RotWord(w19)= 8c dd 50 43 = x5 SubWord(x4)= 64 c1 53 1a = y5 Rcon(5)= 10 00 00 00 y5 ⊕ Rcon(5)= 74 c1 53 1a = z5
w20 = w16 ⊕ z5 = 58 9d 36 eb w21 = w20 ⊕ w17 = fd ee 38 7d w22 = w21 ⊕ w18 = 0f cc 9b ed w23 = w22 ⊕ w19 = 4c 40 46 bd	RotWord(w23)= 40 46 bd 4c = x6 SubWord(x5)= 09 5a 7a 29 = y6 Rcon(6)= 20 00 00 00 y6 ⊕ Rcon(6)= 29 5a 7a 29 = z6
w24 = w20 ⊕ z6 = 71 c7 4c c2 w25 = w24 ⊕ w21 = 8c 29 74 bf w26 = w25 ⊕ w22 = 83 e5 ef 52 w27 = w26 ⊕ w23 = cf a5 a9 ef	RotWord(w27)= a5 a9 ef cf = x7 SubWord(x6)= 06 d3 df 8a = y7 Rcon(7)= 40 00 00 00 y7 ⊕ Rcon(7)= 46 d3 df 8a = z7
w28 = w24 ⊕ z7 = 37 14 93 48 w29 = w28 ⊕ w25 = bb 3d e7 f7 w30 = w29 ⊕ w26 = 38 d8 08 a5 w31 = w30 ⊕ w27 = f7 7d a1 4a	RotWord(w31)= 7d a1 4a f7 = x8 SubWord(x7)= ff 32 d6 68 = y8 Rcon(8)= 80 00 00 00 y8 ⊕ Rcon(8)= 7f 32 d6 68 = z8
w32 = w28 ⊕ z8 = 48 26 45 20 w33 = w32 ⊕ w29 = f3 1b a2 d7 w34 = w33 ⊕ w30 = cb c3 aa 72 w35 = w34 ⊕ w32 = 3c be 0b 38	RotWord(w35)= be 0b 38 3c = x9 SubWord(x8)= ae 2b 07 eb = y9 Rcon(9)= 1B 00 00 00 y9 ⊕ Rcon(9)= b5 2b 07 eb = z9
w36 = w32 ⊕ z9 = fd 0d 42 cb w37 = w36 ⊕ w33 = 0e 16 e0 1c w38 = w37 ⊕ w34 = c5 d5 4a 6e w39 = w38 ⊕ w35 = f9 6b 41 56	RotWord(w39)= 6b 41 56 f9 = x10 SubWord(x9)= 7f 83 b1 99 = y10 Rcon(10)= 36 00 00 00 y10 ⊕ Rcon(10)= 49 83 b1 99 = z10
w40 = w36 ⊕ z10 = b4 8e f3 52 w41 = w40 ⊕ w37 = ba 98 13 4e w42 = w41 ⊕ w38 = 7f 4d 59 20 w43 = w42 ⊕ w39 = 86 26 18 76	

(Table is located on page 151 in textbook)

Table 5.4

AES Example

Start of round	After SubBytes	After ShiftRows	After MixColumns	Round Key
01 89 fe 76 23 ab dc 54 45 cd ba 32 67 ef 98 10				0f 47 0c af 15 d9 b7 7f 71 e8 ad 67 c9 59 d6 98
0e ce f2 d9 36 72 6b 2b 34 25 17 55 ae b6 4e 88	ab 8b 89 35 05 40 7f f1 18 3f f0 fc e4 4e 2f c4	ab 8b 89 35 40 7f f1 05 f0 fc 18 3f c4 e4 4e 2f	b9 94 57 75 e4 8e 16 51 47 20 9a 3f c5 d6 f5 3b	dc 9b 97 38 90 49 fe 81 37 df 72 15 b0 e9 3f a7
65 0f c0 4d 74 c7 e8 d0 70 ff e8 2a 75 3f ca 9c	4d 76 ba e3 92 c6 9b 70 51 16 9b e5 9d 75 74 de	4d 76 ba e3 c6 9b 70 92 9b e5 51 16 de 9d 75 74	8e 22 db 12 b2 f2 dc 92 df 80 f7 c1 2d c5 1e 52	d2 49 de e6 c9 80 7e ff 6b b4 c6 d3 b7 5e 61 c6
5c 6b 05 f4 7b 72 a2 6d b4 34 31 12 9a 9b 7f 94	4a 7f 6b bf 21 40 3a 3c 8d 18 c7 c9 b8 14 d2 22	4a 7f 6b bf 40 3a 3c 21 c7 c9 8d 18 22 b8 14 d2	b1 c1 0b cc ba f3 8b 07 f9 1f 6a c3 1d 19 24 5c	c0 89 57 b1 af 2f 51 ae df 6b ad 7e 39 67 06 c0
71 48 5c 7d 15 dc da a9 26 74 c7 bd 24 7e 22 9c	a3 52 4a ff 59 86 57 d3 f7 92 c6 7a 36 f3 93 de	a3 52 4a ff 86 57 d3 59 c6 7a f7 92 de 36 f3 93	d4 11 fe 0f 3b 44 06 73 cb ab 62 37 19 b7 07 ec	2c a5 f2 43 5c 73 22 8c 65 0e a3 dd f1 96 90 50
f8 b4 0c 4c 67 37 24 ff ae a5 c1 ea e8 21 97 bc	41 8d fe 29 85 9a 36 16 e4 06 78 87 9b fd 88 65	41 8d fe 29 9a 36 16 85 78 87 e4 06 65 9b fd 88	2a 47 c4 48 83 e8 18 ba 84 18 27 23 eb 10 0a f3	58 fd 0f 4c 9d ee cc 40 36 38 9b 46 eb 7d ed bd
72 ba cb 04 1e 06 d4 fa b2 20 bc 65 00 6d e7 4e	40 f4 1f f2 72 6f 48 2d 37 b7 65 4d 63 3c 94 2f	40 f4 1f f2 6f 48 2d 72 65 4d 37 b7 2f 63 3c 94	7b 05 42 4a 1e d0 20 40 94 83 18 52 94 c4 43 fb	71 8c 83 cf c7 29 e5 a5 4c 74 ef a9 c2 bf 52 ef
0a 89 c1 85 d9 f9 c5 e5 d8 f7 f7 fb 56 7b 11 14	67 a7 78 97 35 99 a6 d9 61 68 68 0f b1 21 82 fa	67 a7 78 97 99 a6 d9 35 68 0f 61 68 fa b1 21 82	ec 1a c0 80 0c 50 53 c7 3b d7 00 ef b7 22 72 e0	37 bb 38 f7 14 3d d8 7d 93 e7 08 a1 48 f7 a5 4a
db a1 f8 77 18 6d 8b ba a8 30 08 4e ff d5 d7 aa	b9 32 41 f5 ad 3c 3d f4 c2 04 30 2f 16 03 0e ac	b9 32 41 f5 3c 3d f4 ad 30 2f c2 04 ac 16 03 0e	b1 1a 44 17 3d 2f ec b6 0a 6b 2f 42 9f 68 f3 b1	48 f3 cb 3c 26 1b c3 be 45 a2 aa 0b 20 d7 72 38
f9 e9 8f 2b 1b 34 2f 08 4f c9 85 49 bf bf 81 89	99 1e 73 f1 af 18 15 30 84 dd 97 3b 08 08 0c a7	99 1e 73 f1 18 15 30 af 97 3b 84 dd a7 08 08 0c	31 30 3a c2 ac 71 8c c4 46 65 48 eb 6a 1c 31 62	fd 0e c5 f9 0d 16 d5 6b 42 e0 4a 41 cb 1c 6e 56
cc 3e ff 3b a1 67 59 af 04 85 02 aa a1 00 5f 34	4b b2 16 e2 32 85 cb 79 f2 97 77 ac 32 63 cf 18	4b b2 16 e2 85 cb 79 32 77 ac f2 97 18 32 63 cf	4b 86 8a 36 b1 cb 27 5a fb f2 f2 af cc 5a 5b cf	b4 ba 7f 86 8e 98 4d 26 f3 13 59 18 52 4e 20 76
ff 08 69 64 0b 53 34 14 84 bf ab 8f 4a 7c 43 b9				

(Table is located on page 153 in textbook)

Table 5.5

Avalanche Effect in AES: Change in Plaintext

Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

(Table is located on page 154 in textbook)

Table 5.6

Avalanche Effect in AES: Change in Key

Round		Number of Bits that Differ
	0123456789abcdeffedcba9876543210 0123456789abcdeffedcba9876543210	0
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c5a9ad090ec7ff3fc1e8e8ca4cd02a9c	22
2	5c7bb49a6b72349b05a2317ff46d1294 90905fa9563356d15f3760f3b8259985	58
3	7115262448dc747e5cdac7227da9bd9c 18aeb7aa794b3b66629448d575c7cebf	67
4	f867aee8b437a5210c24c1974cffeabc f81015f993c978a876ae017cb49e7eec	63
5	721eb200ba06206dcbd4bce704fa654e 5955c91b4e769f3cb4a94768e98d5267	81
6	0ad9d85689f9f77bc1c5f71185e5fb14 dc60a24d137662181e45b8d3726b2920	70
7	db18a8ffa16d30d5f88b08d777ba4eaa fe8343b8f88bef66cab7e977d005a03c	74
8	f91b4fbfe934c9bf8f2f85812b084989 da7dad581d1725c5b72fa0f9d9d1366a	67
9	cca104a13e678500ff59025f3bafaa34 0ccb4c66bbfd912f4b511d72996345e0	59
10	ff0b844a0853bf7c6934ab4364148fb9 fc8923ee501a7d207ab670686839996b	53

(Table is located on page 155 in textbook)

Equivalent Inverse Cipher

- AES decryption cipher is not identical to the encryption cipher
  - The sequence of transformations differs although the form of the key schedules is the same
  - Has the disadvantage that two separate software or firmware modules are needed for applications that require both encryption and decryption

Two separate changes are needed to bring the decryption structure in line with the encryption structure

The first two stages of the decryption round need to be interchanged

The second two stages of the decryption round need to be interchanged

Interchanging InvShiftRows and InvSubBytes

- InvShiftRows *affects the sequence* of bytes in State but *does not alter byte contents* and *does not depend on byte contents* to perform its transformation

- InvSubBytes *affects the contents* of bytes in State but *does not alter byte sequence* and *does not depend on byte sequence* to perform its transformation

Thus, these two operations commute and can be interchanged

Interchanging AddRoundKey and InvMixColumns

The transformations AddRoundKey and InvMixColumns do not alter the sequence of bytes in State

If we view the key as a sequence of words, then both AddRoundKey and InvMixColumns operate on State one column at a time

These two operations are linear with respect to the column input

Equivalent Inverse Cipher

Figure 5.10 Equivalent Inverse Cipher

Implementation Aspects

- AES can be implemented very efficiently on an 8-bit processor
- AddRoundKey is a bytewise XOR operation
- ShiftRows is a simple byte-shifting operation
- SubBytes operates at the byte level and only requires a table of 256 bytes
- MixColumns requires matrix multiplication in the field $GF(2^8)$, which means that all operations are carried out on bytes

Summary

- Finite field arithmetic
- AES structure
  - General structure
  - Detailed structure
- AES key expansion
  - Key expansion algorithm
  - Rationale

- AES transformation functions
  - Substitute bytes
  - ShiftRows
  - MixColumns
  - AddRoundKey
- AES implementation
  - Equivalent inverse cipher
  - Implementation aspects

Mix Columns

$$\begin{bmatrix} ab & 8b & 89 & 35 \\ 40 & 7f & f1 & 05 \\ f0 & fc & 18 & 3f \\ c4 & e4 & 4e & 2f \end{bmatrix} \qquad \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

Select Column 1 and Multiply it with Row 1, 2, 3, 4

1. Multiply with Row 1

$$02.ab \oplus 03.40 \oplus f0 \oplus c4$$
$$4d \oplus c0 + f0 + c4 = B9$$

2. Mutiply with Row 2

$$ab \oplus 02.40 \oplus 03.f0 \oplus c4$$
$$ab \oplus 80 \oplus 0b \oplus c4 = E4$$

3. Mutiply with Row 3

$$ab \oplus 40 \oplus 02.f0 \oplus 03.c4$$
$$ab \oplus 40 \oplus fb \oplus 57 = 47$$

4. Mutiply with Row 4

$$03.ab \oplus 40 \oplus f0 \oplus 02.c4$$
$$e6 \oplus 40 \oplus f0 + 93 = c5$$

Chapter 11

Cryptographic Hash Functions

Not included in exam and quiz

Hash Functions

- A hash function H accepts a variable-length block of data *M* as input and produces a fixed-size hash value
  - $h = H(M)$
  - Principal object is data integrity
- Cryptographic hash function
  - An algorithm for which it is computationally infeasible to find either:

    (a) a data object that maps to a pre-specified hash result (the one-way property)

    (b) two data objects that map to the same hash result (the collision-free property)

Figure 11.1 Cryptographic Hash Function; $h = H(M)$

(a) Use of hash function to check data integrity

(b) Man-in-the-middle attack

Figure 11.2 Attack Against Hash Function

Figure 11.3 Simplified Examples of the Use of a Hash Function for Message Authentication

Message Authentication Code (MAC)

- Also known as a *keyed hash function*
- Typically used between two parties that share a secret key to authenticate information exchanged between those parties

Takes as input a secret key and a data block and produces a hash value (MAC) which is associated with the protected message

- If the integrity of the message needs to be checked, the MAC function can be applied to the message and the result compared with the associated MAC value
- An attacker who alters the message will be unable to alter the associated MAC value without knowledge of the secret key

Digital Signature

- Operation is similar to that of the MAC
- The hash value of a message is encrypted with a user's private key
- Anyone who knows the user's public key can verify the integrity of the message
- An attacker who wishes to alter the message would need to know the user's private key
- Implications of digital signatures go beyond just message authentication

Figure 11.4 Simplified Examples of Digital Signatures

Other Hash Function Uses

Commonly used to create a one-way password file

When a user enters a password, the hash of that password is compared to the stored hash value for verification

This approach to password protection is used by most operating systems

Can be used for intrusion and virus detection

Store H(F) for each file on a system and secure the hash values

One can later determine if a file has been modified by recomputing H(F)

An intruder would need to change F without changing H(F)

Can be used to construct a pseudorandom function (PRF) or a pseudorandom number generator (PRNG)

A common application for a hash-based PRF is for the generation of symmetric keys

Two Simple Hash Functions

- Consider two simple insecure hash functions that operate using the following general principles:
  - The input is viewed as a sequence of *n*-bit blocks
  - The input is processed one block at a time in an iterative fashion to produce an *n*-bit hash function
- Bit-by-bit exclusive-OR (XOR) of every block
  - $C_i = b_{i1}\ xor\ b_{i2}\ xor\ \ldots\ xor\ b_{im}$
  - Produces a simple parity for each bit position and is known as a longitudinal redundancy check
  - Reasonably effective for random data as a data integrity check
- Perform a one-bit circular shift on the hash value after each block is processed
  - Has the effect of randomizing the input more completely and overcoming any regularities that appear in the input

Two Simple Hash Functions

Figure 11.5 Two Simple Hash Functions

Requirements and Security

Preimage

- x is the preimage of h for a hash value $h = H(x)$
- Is a data block whose hash function, using the function H, is h
- Because H is a many-to-one mapping, for any given hash value h, there will in general be multiple preimages

Collision

- Occurs if we have $x \neq y$ and $H(x) = H(y)$
- Because we are using hash functions for data integrity, collisions are clearly undesirable

Table 11.1

Requirements for a Cryptographic Hash Function H

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness

(Table can be found on page 323 in textbook.)

Figure 11.6 Relationship Among Hash Function Properties

Table 11.2

Hash Function Resistance Properties Required for Various Data Integrity Applications

	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

* Resistance required if attacker is able to mount a chosen message attack

Attacks on Hash Functions

Brute-Force Attacks

- Does not depend on the specific algorithm, only depends on bit length
- In the case of a hash function, attack depends only on the bit length of the hash value
- Method is to pick values at random and try each one until a collision occurs

Cryptanalysis

- An attack based on weaknesses in a particular cryptographic algorithm
- Seek to exploit some property of the algorithm to perform some attack other than an exhaustive search

Birthday Attacks

- For a collision resistant attack, an adversary wishes to find two messages or data blocks that yield the same hash function
  - The effort required is explained by a mathematical result referred to as the *birthday paradox*

- How the birthday attack works:
  - The source (A) is prepared to sign a legitimate message *x* by appending the appropriate *m*-bit hash code and encrypting that hash code with A's private key
  - Opponent generates $2^{m/2}$ variations *x'* of *x*, all with essentially the same meaning, and stores the messages and their hash values
  - Opponent generates a fraudulent message *y* for which A's signature is desired
  - Two sets of messages are compared to find a pair with the same hash
  - The opponent offers the valid variation to A for signature which can then be attached to the fraudulent variation for transmission to the intended recipient
    - Because the two variations have the same hash code, they will produce the same signature and the opponent is assured of success even though the encryption key is not known

A Letter in 2^{37} Variation

Dear Anthony,

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --}

Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the}

Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --}

responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches}

in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs}

to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the

market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the

{latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum}

of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document}

as proof of identity. An order with his signature, which is {appended / attached}

{authorizes / allows} you to charge the cost to this company at the {above / head office}

address. We {fully / --} expect that our {level / volume} of orders will increase in

the {following / next} year and {trust / hope} that the new appointment will {be / prove}

{advantageous / an advantage} to both our companies.

Figure 11.7 A Letter in 2^{37} Variations

(Letter is located on page 326 in textbook)

IV = Initial value
CV_i = chaining variable
Y_i = ith input block
f = compression algorithm
L = number of input blocks
n = length of hash code
b = length of input block

Figure 11.8 General Structure of Secure Hash Code

Hash Functions Based on Cipher Block Chaining

- Can use block ciphers as hash functions
  - Using $H_0=0$ and zero-pad of final block
  - Compute: $H_i = E(M_i\ H_{i-1})$
  - Use final block as the hash value
  - Similar to CBC but without a key
- Resulting hash is too small (64-bit)
  - Both due to direct birthday attack
  - And "meet-in-the-middle" attack
- Other variants also susceptible to attack

Secure Hash Algorithm (SHA)

- SHA was originally designed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993
- Was revised in 1995 as SHA-1
- Based on the hash function MD4 and its design closely models MD4
- Produces 160-bit hash values
- In 2002 NIST produced a revised version of the standard that defined three new versions of SHA with hash value lengths of 256, 384, and 512
  - Collectively known as SHA-2

Table 11.3
Comparison of SHA Parameters

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Note: All sizes are measured in bits.

Figure 11.9 Message Digest Generation Using SHA-512

Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

Table 11.4
SHA-512 Constants

428a2f98d728ae22	7137449123ef65cd	b5c0fbcfec4d3b2f	e9b5dba58189dbbc
3956c25bf348b538	59f111f1b605d019	923f82a4af194f9b	ab1c5ed5da6d8118
d807aa98a3030242	12835b0145706fbe	243185be4ee4b28c	550c7dc3d5ffb4e2
72be5d74f27b896f	80deb1fe3b1696b1	9bdc06a725c71235	c19bf174cf692694
e49b69c19ef14ad2	efbe4786384f25e3	0fc19dc68b8cd5b5	240ca1cc77ac9c65
2de92c6f592b0275	4a7484aa6ea6e483	5cb0a9dcbd41fbd4	76f988da831153b5
983e5152ee66dfab	a831c66d2db43210	b00327c898fb213f	bf597fc7beef0ee4
c6e00bf33da88fc2	d5a79147930aa725	06ca6351e003826f	142929670a0e6e70
27b70a8546d22ffc	2e1b21385c26c926	4d2c6dfc5ac42aed	53380d139d95b3df
650a73548baf63de	766a0abb3c77b2a8	81c2c92e47edaee6	92722c851482353b
a2bfe8a14cf10364	a81a664bbc423001	c24b8b70d0f89791	c76c51a30654be30
d192e819d6ef5218	d69906245565a910	f40e35855771202a	106aa07032bbd1b8
19a4c116b8d2d0c8	1e376c085141ab53	2748774cdf8eeb99	34b0bcb5e19b48a8
391c0cb3c5c95a63	4ed8aa4ae3418acb	5b9cca4f7763e373	682e6ff3d6b2b8a3
748f82ee5defb2fc	78a5636f43172f60	84c87814a1f0ab72	8cc702081a6439ec
90befffa23631e28	a4506cebde82bde9	bef9a3f7b2c67915	c67178f2e372532b
ca273eceea26619c	d186b8c721c0c207	eada7dd6cde0eb1e	f57d4f7fee6ed178
06f067aa72176fba	0a637dc5a2c898a6	113f9804bef90dae	1b710b35131c471b
28db77f523047d84	32caab7b40c72493	3c9ebe0a15c9bebc	431d67c49c100d4c
4cc5d4becb3e42b6	597f299cfc657e2a	5fcb6fab3ad6faec	6c44198c4a475817

(Table can be found on page 333 in textbook)

Figure 11.11 Elementary SHA-512 Operation (single round)

Figure 11.12 Creation of 80-word Input Sequence for SHA-512 Processing of Single Block

SHA-512 Logic

The padded message consists blocks $M_1, M_2, \ldots M_N$. Each message block M_i consists of 16 64-bit words $M_{i,0}, M_{i,1} \ldots M_{i,15}$. All addition is performed modulo 2^{64}.

$H_{0,0}$ = 6A09E667F3BCC908 $\quad H_{0,4}$ = 510E527FADE682D1
$H_{0,1}$ = BB67AE8584CAA73B $\quad H_{0,5}$ = 9B05688C2B3E6C1F
$H_{0,2}$ = 3C6EF372FE94F82B $\quad H_{0,6}$ = 1F83D9ABFB41BD6B
$H_{0,3}$ = A54FF53A5F1D36F1 $\quad H_{0,7}$ = 5BE0CDI9137E2179

for $i = 1$ **to** N

1. Prepare the message schedule W:
   for $t = 0$ **to** 15
   $W_t = M_{i,t}$
   for t = 16 **to** 79
   $W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$
2. Initialize the working variables
   $a = H_{i-1,0} \quad e = H_{i-1,4}$
   $b = H_{i-1,1} \quad f = H_{i-1,5}$
   $c = H_{i-1,2} \quad g = H_{i-1,6}$
   $d = H_{i-1,3} \quad h = H_{i-1,7}$
3. Perform the main hash computation
   for $t = 0$ **to** 79
   $T_1 = h + \mathrm{Ch}(e, f, g) + \left(\sum_1^{512} e\right) + W_t + K_t$
   $T_2 = \left(\sum_0^{512} a\right) + \mathrm{Maj}(a, b, c)$
   $h = g$
   $g = f$
   $f = e$
   $e = d + T_1$
   $d = c$
   $c = b$
   $b = a$
   $a = T_1 + T_2$
4. Compute the inermediate hash value
   $H_{i,0} = a + H_{i-1,0} \quad H_{i,4} = e + H_{i-1,4}$
   $H_{i,1} = b + H_{i-1,1} \quad H_{i,5} = f + H_{i-1,5}$
   $H_{i,2} = c + H_{i-1,2} \quad H_{i,6} = g + H_{i-1,6}$
   $H_{i,3} = d + H_{i-1,3} \quad H_{i,7} = h + H_{i-1,7}$

return $\{H_{N,0} \| H_{N,1} \| H_{N,2} \| H_{N,3} \| H_{N,4} \| H_{N,5} \| H_{N,6} \| H_{N,7}\}$

Figure 11.13 SHA-512 Logic

(Figure can be found on page 337 in textbook)

SHA-3

SHA-1 has not yet been "broken"

- No one has demonstrated a technique for producing collisions in a practical amount of time
- Considered to be insecure and has been phased out for SHA-2

SHA-2 shares the same structure and mathematical operations as its predecessors so this is a cause for concern

- Because it will take years to find a suitable replacement for SHA-2 should it become vulnerable, NIST decided to begin the process of developing a new hash standard

NIST announced in 2007 a competition for the SHA-3 next generation NIST hash function

- Winning design was announced by NIST in October 2012
- SHA-3 is a cryptographic hash function that is intended to complement SHA-2 as the approved standard for a wide range of applications

The Sponge Construction

- Underlying structure of SHA-3 is a scheme referred to by its designers as a *sponge construction*
- Takes an input message and partitions it into fixed-size blocks
- Each block is processed in turn with the output of each iteration fed into the next iteration, finally producing an output block
- The sponge function is defined by three parameters:
  - f = the internal function used to process each input block
  - r = the size in bits of the input blocks, called the *bitrate*
  - pad = the padding algorithm

Figure 11.14 Sponge Function Input and Output

Table 11.5
SHA-3 Parameters

Message Digest Size	224	256	384	512
Message Size	no maximum	no maximum	no maximum	no maximum
Block Size (bitrate *r*)	1152	1088	832	576
Word Size	64	64	64	64
Number of Rounds	24	24	24	24
Capacity *c*	448	512	768	1024
Collision resistance	2^{112}	2^{128}	2^{192}	2^{256}
Second preimage resistance	2^{224}	2^{256}	2^{384}	2^{512}

Summary

- Applications of cryptographic hash functions
  - Message authentication
  - Digital signatures
  - Other applications
- Requirements and security
  - Security requirements for cryptographic hash functions
  - Brute-force attacks
  - Cryptanalysis

- Hash functions based on cipher block chaining
- Secure hash algorithm (SHA)
  - SHA-512 logic
  - SHA-512 round function
- SHA-3
  - The sponge construction
  - The SHA-3 Iteration Function *f*

Chapter 6

Block Cipher Operation

Double DES

(a) Double Encryption

Meet-in-the-Middle Attack

The use of double DES results in a mapping that is not equivalent to a single DES encryption

The meet-in-the-middle attack algorithm will attack this scheme and does not depend on any particular property of DES but will work against any block encryption cipher

Triple-DES with Two-Keys

- Obvious counter to the meet-in-the-middle attack is to use three stages of encryption with three different keys
  - This raises the cost of the meet-in-the-middle attack to 2^{112}, which is beyond what is practical
  - Has the drawback of requiring a key length of 56 x 3 = 168 bits, which may be somewhat unwieldy
  - As an alternative Tuchman proposed a triple encryption method that uses only two keys
- 3DES with two keys is a relatively popular alternative to DES and has been adopted for use in the key management standards ANSI X9.17 and ISO 8732

Multiple Encryption

(b) Triple Encryption

Figure 6.1 Multiple Encryption

(a) Two-key Triple Encryption with Candidate Pair of Keys

(b) Table of n known plaintext-ciphertext pairs, sorted on P

(c) Table of intermediate values and candidate keys

Figure 6.2 Known-Plaintext Attack on Triple DES

Triple DES with Three Keys

- Many researchers now feel that three-key 3DES is the preferred alternative

Three-key 3DES has an effective key length of 168 bits and is defined as:

- $C = E(K_3, D(K_2, E(K_1, P)))$

Backward compatibility with DES is provided by putting:

- $K_3 = K_2$ or $K_1 = K_2$

- A number of Internet-based applications have adopted three-key 3DES including PGP and S/MIME

Modes of Operation

- A technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application
- To apply a block cipher in a variety of applications, five *modes of operation* have been defined by NIST
  - The five modes are intended to cover a wide variety of applications of encryption for which a block cipher could be used
  - These modes are intended for use with any symmetric block cipher, including triple DES and AES

Table 6.1 Block Cipher Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next block of plaintext and the preceding block of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

Electronic Codebook Mode (ECB)

Figure 6.3 Electronic Codebook (ECB) Mode

Criteria and properties for evaluating and constructing block cipher modes of operation that are superior to ECB:

- Overhead
- Error recovery
- Error propagation
- Diffusion
- Security

Cipher Block Chaining (CBC)

Figure 6.4 Cipher Block Chaining (CBC) Mode

Cipher Feedback Mode

- For AES, DES, or any block cipher, encryption is performed on a block of *b* bits
  - In the case of DES $b = 64$
  - In the case of AES $b = 128$

There are three modes that make it possible to convert a block cipher into a stream cipher:

- Cipher feedback (CFB) mode
- Output feedback (OFB) mode
- Counter (CTR) mode

s-bit Cipher Feedback (CFB) Mode

Figure 6.5 *s*-bit Cipher Feedback (CFB) Mode

Output Feedback (OFB) Mode

Figure 6.6 Output Feedback (OFB) Mode

Counter (CTR) Mode

Figure 6.7 Counter (CTR) Mode

Advantages of CTR

- Hardware efficiency
- Software efficiency
- Preprocessing
- Random access
- Provable security
- Simplicity

Feedback Characteristics of Modes of Operation

Figure 6.8 Feedback Characteristic of Modes of Operation

XTS-AES Mode for Block-Oriented Storage Devices

- Approved as an additional block cipher mode of operation by NIST in 2010
- Mode is also an IEEE Standard, IEEE Std 1619-2007
  - Standard describes a method of encryption for data stored in sector-based devices where the threat model includes possible access to stored data by the adversary
  - Has received widespread industry support

Tweakable Block Ciphers

- XTS-AES mode is based on the concept of a *tweakable block cipher*
- General structure:
  - Has three inputs:

- Tweak need not be kept secret
  - Purpose is to provide variability

Tweakable Block Cipher

Figure 6.9 Tweakable Block Cipher

Summary

- Multiple encryption and triple DES
  - Double DES
  - Triple DES with two keys
  - Triple DES with three keys
- Electronic code book
- Cipher block chaining mode

- Cipher feedback mode
- Output feedback mode
- Counter mode
- XTS-AES mode for block-oriented storage devices
  - Storage encryption requirements
  - Operation on a single block
  - Operation on a sector

Double DES

$C = E(K_2, E(K1, P))$

$P = D(K1, D(K2, C))$

Key length $= 56 \times 2$
$= 112$ bits

$\Rightarrow$ Suppose

$E(K2, E(K1, P)) = E(K3, P)$

Meet in the Middle Attack

$X = E(K1, P) = D(K2, C)$

Given Known Pair (P, C)

1. Encrypt P for all values of K1 i.e 2^{56}
   Put in table and Sort by X

2. Decrypt C for all values of K2 i.e 2^{56}
   Check result against table for Match.

3. if match occurs, test two keys
   against new plain-cipher text pair.
   if produce correct cipher accept
   as Correct Keys.

$\Rightarrow$ The plaintext require 2^{56} effort

Same as DES 2^{55}

Multiple Encryption

$$C = E(K1, D(K2, E(K1, P)))$$

$$P = D(K1, E(K2, D(K1, C)))$$

Known PlainText Attack

1. obtain n (P, C) pairs and Sort by P.

2. Pick any value for 'a' & produce P_i

   2^{56} keys $K1 = i$

   $P_i = D(i, a)$ (Table 1 match)

   $B = D(i, C)$ (Table 2 key1, B)

3. calculate K2

   Pick any value of 'a'

   $B_j = D(j, a)$

   $K2 = j = 2^{56}$ keys

   if match i and j are keys

4. Test keys (i, j) on other plain-cipher

   if no pair find then select new value of 'a' & Repeat Algo

<u>ECB</u>

$Cj = E(K, Pj)\ j = 1, . . . , N$

$Pj = D(K, Cj)\ j = 1, . . . , N$

<u>CBC</u>

$C1 = E(K, [P1 \oplus IV])$

$Cj = E(K, [Pj \oplus Cj- 1])\ j = 2, ..., N$

$P1 = D(K, C1) \oplus IV$

$Pj = D(K, Cj) \oplus Cj- 1\ j = 2, ..., N$

Chapter 9

Public Key Cryptography and RSA

Misconceptions Concerning Public-Key Encryption

- Public-key encryption is more secure from cryptanalysis than symmetric encryption
- Public-key encryption is a general-purpose technique that has made symmetric encryption obsolete
- There is a feeling that key distribution is trivial when using public-key encryption, compared to the cumbersome handshaking involved with key distribution centers for symmetric encryption

Table 9.1

Terminology Related to Asymmetric Encryption

Asymmetric Keys
Two related keys, a public key and a private key that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.

Public Key Certificate
A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.

Public Key (Asymmetric) Cryptographic Algorithm
A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

Public Key Infrastructure (PKI)
A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

Source: *Glossary of Key Information Security Terms*, NIST IR 7298 [KISS06]

Principles of Public-Key Cryptosystems

- The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption:

Key distribution

- **How to have secure communications in general without having to trust a KDC with your key**

Digital signatures

- **How to verify that a message comes intact from the claimed sender**

- Whitfield Diffie and Martin Hellman from Stanford University achieved a breakthrough in 1976 by coming up with a method that addressed both problems and was radically different from all previous approaches to cryptography

Public-Key Cryptosystems

- A public-key encryption scheme has six ingredients:

Plaintext	Encryption algorithm	Public key	Private key	Ciphertext	Decryption algorithm
The readable message or data that is fed into the algorithm as input	Performs various transform-ations on the plaintext	Used for encryption or decryption	Used for encryption or decryption	The scrambled message produced as output	Accepts the ciphertext and the matching key and produces the original plaintext

Public-Key Cryptography

Figure 9.1 Public-Key Cryptography

Table 9.2
Conventional and Public-Key Encryption

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if the key is kept secret. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Public-Key Cryptosystem: Secrecy

Figure 9.2 Public-Key Cryptosystem: Secrecy

Public-Key Cryptosystem: Authentication

Figure 9.3 Public-Key Cryptosystem: Authentication

Public-Key Cryptosystem: Authentication and Secrecy

Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

Applications for Public-Key Cryptosystems

- Public-key cryptosystems can be classified into three categories:

Encryption/decryption	• The sender encrypts a message with the recipient's public key
Digital signature	• The sender "signs" a message with its private key
Key exchange	• Two sides cooperate to exchange a session key

- Some algorithms are suitable for all three applications, whereas others can be used only for one or two

Table 9.3

Applications for Public-Key Cryptosystems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Table 9.3 Applications for Public-Key Cryptosystems

Public-Key Requirements

- Conditions that these algorithms must fulfill:
  1. It is computationally easy for a party B to generate a pair (public-key PU_b, private key PR_b)
  2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, to generate the corresponding ciphertext
  3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message
  4. It is computationally infeasible for an adversary, knowing the public key, to determine the private key
  5. It is computationally infeasible for an adversary, knowing the public key and a ciphertext, to recover the original message
  6. The two keys can be applied in either order

Public-Key Requirements

- Need a trap-door one-way function
  - A one-way function is one that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy, whereas the calculation of the inverse is infeasible
    - $Y = f(X)$ easy
    - $X = f^{-1}(Y)$ infeasible
- A trap-door one-way function is a family of invertible functions f_k, such that
  - $Y = f_k(X)$ easy, if k and X are known
  - $X = f_k^{-1}(Y)$ easy, if k and Y are known
  - $X = f_k^{-1}(Y)$ infeasible, if Y known but k not known
- A practical public-key scheme depends on a suitable trap-door one-way function

Public-Key Cryptanalysis

- A public-key encryption scheme is vulnerable to a brute-force attack
  - Countermeasure: use large keys
  - Key size must be small enough for practical encryption and decryption
  - Key sizes that have been proposed result in encryption/decryption speeds that are too slow for general-purpose use
  - Public-key encryption is currently confined to key management and signature applications
- Another form of attack is to find some way to compute the private key given the public key
  - To date it has not been mathematically proven that this form of attack is infeasible for a particular public-key algorithm
- Finally, there is a probable-message attack
  - This attack can be thwarted by appending some random bits to simple messages

Rivest-Shamir-Adleman (RSA) Scheme

- Developed in 1977 at MIT by Ron Rivest, Adi Shamir & Len Adleman
- Most widely used general-purpose approach to public-key encryption
- Is a cipher in which the plaintext and ciphertext are integers between 0 and $n - 1$ for some n
  - A typical size for n is 1024 bits, or 309 decimal digits

RSA Algorithm

- RSA makes use of an expression with exponentials
- Plaintext is encrypted in blocks with each block having a binary value less than some number *n*
- Encryption and decryption are of the following form, for some plaintext block *M* and ciphertext block C

  $C = M^e \bmod n$

  $M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$

- Both sender and receiver must know the value of *n*
- The sender knows the value of *e*, and only the receiver knows the value of *d*
- This is a public-key encryption algorithm with a public key of *PU={e,n}* and a private key of *PR={d,n}*

Algorithm Requirements

- For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:

  1. It is possible to find values of *e*, *d*, *n* such that $M^{ed} \bmod n = M$ for all $M < n$

  2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$

  3. It is infeasible to determine *d* given *e* and *n*

RSA Key Setup

- each user generates a public/private key pair by:
- selecting two large primes at random - p, q
- computing their system modulus $n=p.q$
  - note $\varnothing(n)=(p-1)(q-1)$
- selecting at random the encryption key e
  - where $1<e<\varnothing(n)$, $\gcd(e,\varnothing(n))=1$
- solve following equation to find decryption key d
  - $e.d=1 \bmod \varnothing(n)$ and $0 \leq d \leq n$
- publish their public encryption key: PU={e,n}
- keep secret private decryption key: PR={d,n}

Key Generation by Alice	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ \ 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Private Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Figure 9.5 The RSA Algorithm

RSA Example - Key Setup

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 $PU=\{7,187\}$
7. Keep secret private key $PR=\{23,187\}$

RSA Example - En/Decryption

- sample RSA encryption/decryption is:
- given message $M = 88$
- encryption:

  $C = 88^7 \bmod 187 = 11$
- decryption:

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

Example of RSA Algorithm

Figure 9.6 Example of RSA Algorithm

(a) General approach

(b) Example

Figure 9.7 RSA Processing of Multiple Blocks

Exponentiation in Modular Arithmetic

- Both encryption and decryption in RSA involve raising an integer to an integer power, mod *n*
- Can make use of a property of modular arithmetic:

  $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

- With RSA you are dealing with potentially large exponents so efficiency of exponentiation is a consideration

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

Note: The integer b is expressed as a binary number $b_k b_{k-1} \ldots b_0$

Figure 9.8 Algorithm for Computing a^b mod n

Table 9.4

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
f	7	49	157	526	160	241	298	166	67	1

Table 9.4 Result of the Fast Modular Exponentiation Algorithm for $a^b \bmod n$, where $a = 7$, $b = 560 = 1000110000$, and $n = 561$

Efficient Operation Using the Public Key

- To speed up the operation of the RSA algorithm using the public key, a specific choice of *e* is usually made
- The most common choice is 65537 ($2^{16} + 1$)
  - Two other popular choices are *e*=3 and *e*=17
  - Each of these choices has only two 1 bits, so the number of multiplications required to perform exponentiation is minimized
  - With a very small public key, such as *e* = 3, RSA becomes vulnerable to a simple attack

Efficient Operation Using the Private Key

- Decryption uses exponentiation to power d
  - A small value of d is vulnerable to a brute-force attack and to other forms of cryptanalysis
- Can use the Chinese Remainder Theorem (CRT) to speed up computation
  - The quantities $d \bmod (p - 1)$ and $d \bmod (q - 1)$ can be precalculated
  - End result is that the calculation is approximately four times as fast as evaluating $M = C^d \bmod n$ directly

Key Generation

- Before the application of the public-key cryptosystem each participant must generate a pair of keys:
  - Determine two prime numbers *p* and *q*
  - Select either *e* or *d* and calculate the other

- Because the value of $n = pq$ will be known to any potential adversary, primes must be chosen from a sufficiently large set
  - The method used for finding large primes must be reasonably efficient

Procedure for Picking a Prime Number

1. Pick an odd integer *n* at random
2. Pick an integer $a < n$ at random
3. Perform the probabilistic primality test with *a* as a parameter. If *n* fails the test, reject the value *n* and go to step 1
4. If *n* has passed a sufficient number of tests, accept *n*; otherwise, go to step 2

The Security of RSA

Five possible approaches to attacking RSA are:

Brute force

- Involves trying all possible private keys

Mathematical attacks

- There are several approaches, all equivalent in effort to factoring the product of two primes

Timing attacks

- These depend on the running time of the decryption algorithm

Hardware fault-based attack

- This involves inducing hardware faults in the processor that is generating digital signatures

Chosen ciphertext attacks

- This type of attack exploits properties of the RSA algorithm

Factoring Problem

- We can identify three approaches to attacking RSA mathematically:
  1. Factor *n* into its two prime factors. This enables calculation of $ø(n) = (p - 1) \times (q - 1)$, which in turn enables determination of $d = e^{-1} \pmod{ø(n)}$
  2. Determine ø(n) directly without first determining *p* and *q*. Again this enables determination of $d = e^{-1} \pmod{ø(n)}$
  3. Determine *d* directly without first determining ø(n)

Number of Decimal Digits	Number of Bits	Date Achieved
100	332	April 1991
110	365	April 1992
120	398	June 1993
129	428	April 1994
130	431	April 1996
140	465	February 1999
155	512	August 1999
160	530	April 2003
174	576	December 2003
200	663	May 2005
193	640	November 2005
232	768	December 2009

Table 9.5 Progress in RSA Factorization

Table 9.5

MIPS-Years Needed to Factor

Figure 9.9 MIPS-years Needed to Factor

Timing Attacks

- Paul Kocher, a cryptographic consultant, demonstrated that a snooper can determine a private key by keeping track of how long a computer takes to decipher messages
- Are applicable not just to RSA but to other public-key cryptography systems
- Are alarming for two reasons:
  - It comes from a completely unexpected direction
  - It is a ciphertext-only attack

Countermeasures

Constant exponentiation time

- **Ensure that all exponentiations take the same amount of time before returning a result; this is a simple fix but does degrade performance**

Random delay

- **Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack**

Blinding

- **Multiply the ciphertext by a random number before performing exponentiation; this process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack**

Fault-Based Attack

- An attack on a processor that is generating RSA digital signatures
  - Induces faults in the signature computation by reducing the power to the processor
  - The faults cause the software to produce invalid signatures which can then be analyzed by the attacker to recover the private key
- The attack algorithm involves inducing single-bit errors and observing the results
- While worthy of consideration, this attack does not appear to be a serious threat to RSA
  - It requires that the attacker have physical access to the target machine and is able to directly control the input power to the processor

Chosen Ciphertext Attack (CCA)

- The adversary chooses a number of ciphertexts and is then given the corresponding plaintexts, decrypted with the target's private key
  - Thus the adversary could select a plaintext, encrypt it with the target's public key, and then be able to get the plaintext back by having it decrypted with the private key
  - The adversary exploits properties of RSA and selects blocks of data that, when processed using the target's private key, yield information needed for cryptanalysis
- To counter such attacks, RSA Security Inc. recommends modifying the plaintext using a procedure known as optimal asymmetric encryption padding (OAEP)

Optimal Asymmetric Encryption Padding (OAEP)

P = encoding parameters
M = message to be encoded
H = hash function
DB = data block
MGF = mask generating function
EM = encoded message

Figure 9.10 Encryption Using Optimal Asymmetric Encryption Padding (OAEP)

Summary

- Public-key cryptosystems
- Applications for public-key cryptosystems
- Requirements for public-key cryptography
- Public-key cryptanalysis

- The RSA algorithm
  - Description of the algorithm
  - Computational aspects
  - Security of RSA

Public Key Requirements

2. $C = E(PU_b, M)$

3. $M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$

6. $M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$

A mask generation function (MGF) is a cryptographic primitive similar to a cryptographic hash function except that while a hash function's output is a fixed size, a MGF supports output of a variable length.

Mask generation functions are completely deterministic: for any given input and desired output length the output is always the same.

Chapter 7

Pseudorandom Number Generation and Stream Ciphers

Random Numbers

- A number of network security algorithms and protocols based on cryptography make use of random binary numbers:
  - Key distribution and reciprocal authentication schemes
  - Session key generation
  - Generation of keys for the RSA public-key encryption algorithm
  - Generation of a bit stream for symmetric stream encryption

Randomness

- The generation of a sequence of allegedly random numbers being random in some well-defined statistical sense has been a concern

Two criteria are used to validate that a sequence of numbers is random:

Uniform distribution

- The frequency of occurrence of ones and zeros should be approximately equal

Independence

- No one subsequence in the sequence can be inferred from the others

Unpredictability

- The requirement is not just that the sequence of numbers be statistically random, but that the successive members of the sequence are unpredictable
- With "true" random sequences each number is statistically independent of other numbers in the sequence and therefore unpredictable
  - True random numbers have their limitations, such as inefficiency, so it is more common to implement algorithms that generate sequences of numbers that appear to be random
  - Care must be taken that an opponent not be able to predict future elements of the sequence on the basis of earlier elements

Pseudorandom Numbers

- Cryptographic applications typically make use of algorithmic techniques for random number generation
- These algorithms are deterministic and therefore produce sequences of numbers that are not statistically random
- If the algorithm is good, the resulting sequences will pass many tests of randomness and are referred to as *pseudorandom numbers*

True Random Number Generator (TRNG)

- Takes as input a source that is effectively random
- The source is referred to as an *entropy source* and is drawn from the physical environment of the computer
  - Includes things such as keystroke timing patterns, disk electrical activity, mouse movements, and instantaneous values of the system clock
  - The source, or combination of sources, serve as input to an algorithm that produces random binary output
- The TRNG may simply involve conversion of an analog source to a binary output
- The TRNG may involve additional processing to overcome any bias in the source

Pseudorandom Number Generator (PRNG)

- Takes as input a fixed value, called the *seed*, and produces a sequence of output bits using a deterministic algorithm
  - Quite often the seed is generated by a TRNG
- The output bit stream is determined solely by the input value or values, so an adversary who knows the algorithm and the seed can reproduce the entire bit stream
- Other than the number of bits produced there is no difference between a PRNG and a PRF

Two different forms of PRNG

Pseudorandom number generator

- An algorithm that is used to produce an open-ended sequence of bits
- Input to a symmetric stream cipher is a common application for an open-ended sequence of bits

Pseudorandom function (PRF)

- Used to produce a pseudorandom string of bits of some fixed length
- Examples are symmetric encryption keys and nonces

TRNG = true random number generator
PRNG = pseudorandom number generator
PRF = pseudorandom function

Figure 7.1 Random and Pseudorandom Number Generators

PRNG Requirements

- The basic requirement when a PRNG or PRF is used for a cryptographic application is that an adversary who does not know the seed is unable to determine the pseudorandom string
- The requirement for secrecy of the output of a PRNG or PRF leads to specific requirements in the areas of:
  - Randomness
  - Unpredictability
  - Characteristics of the seed

Randomness

- The generated bit stream needs to appear random even though it is deterministic
- There is no single test that can determine if a PRNG generates numbers that have the characteristic of randomness
  - If the PRNG exhibits randomness on the basis of multiple tests, then it can be assumed to satisfy the randomness requirement
- NIST SP 800-22 specifies that the tests should seek to establish three characteristics:
  - Uniformity
  - Scalability
  - Consistency

Randomness Tests

- SP 800-22 lists 15 separate tests of randomness

Frequency test

- The most basic test and must be included in any test suite
- Purpose is to determine whether the number of ones and zeros in a sequence is approximately the same as would be expected for a truly random sequence

Runs test

- Focus of this test is the total number of runs in the sequence, where a run is an uninterrupted sequence of identical bits bounded before and after with a bit of the opposite value
- Purpose is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence

Maurer's universal statistical test

- Focus is the number of bits between matching patterns
- Purpose is to detect whether or not the sequence can be significantly compressed without loss of information.

Three tests

Unpredictability

- A stream of pseudorandom numbers should exhibit two forms of unpredictability:
- Forward unpredictability
  - If the seed is unknown, the next output bit in the sequence should be unpredictable in spite of any knowledge of previous bits in the sequence
- Backward unpredictability
  - It should not be feasible to determine the seed from knowledge of any generated values. No correlation between a seed and any value generated from that seed should be evident; each element of the sequence should appear to be the outcome of an independent random event whose probability is 1/2
- The same set of tests for randomness also provides a test of unpredictability
  - A random sequence will have no correlation with a fixed value (the seed)

Seed Requirements

- The seed that serves as input to the PRNG must be secure and unpredictable
- The seed itself must be a random or pseudorandom number
- Typically the seed is generated by TRNG

Generation of Seed Input to PRNG

Figure 7.2 Generation of Seed Input to PRNG

Algorithm Design

- Algorithms fall into two categories:
  - Purpose-built algorithms
    - Algorithms designed specifically and solely for the purpose of generating pseudorandom bit streams
  - Algorithms based on existing cryptographic algorithms
    - Have the effect of randomizing input data

Three broad categories of cryptographic algorithms are commonly used to create PRNGs:

- Symmetric block ciphers
- Asymmetric ciphers
- Hash functions and message authentication codes

Linear Congruential Generator

- An algorithm first proposed by Lehmer that is parameterized with four numbers:

m	the modulus	$m > 0$
a	the multiplier	$0 < a < m$
c	the increment	$0 \leq c < m$
X_0	the starting value, or seed	$0 \leq X_0 < m$

- The sequence of random numbers $\{X_n\}$ is obtained via the following iterative equation:

$$X_{n+1} = (aX_n + c) \bmod m$$

- If m , a , c , and X_0 are integers, then this technique will produce a sequence of integers with each integer in the range $0 \leq X_n < m$

- The selection of values for a , c , and m is critical in developing a good random number generator

Blum Blum Shub (BBS) Generator

- Has perhaps the strongest public proof of its cryptographic strength of any purpose-built algorithm
- Referred to as a *cryptographically secure pseudorandom bit generator* (CSPRBG)
  - A CSPRBG is defined as one that passes the *next-bit-test* if there is not a polynomial-time algorithm that, on input of the first k bits of an output sequence, can predict the $(k + 1)$st bit with probability significantly greater than 1/2
- The security of BBS is based on the difficulty of factoring n

```
X0     = s^2 mod n
for i  = 1 to ∞
   Xi  = (Xi-1)^2 mod n
   Bi  = Xi mod 2
```

Figure 7.3 Blum Blum Shub Block Diagram

Table 7.1

Example Operation of BBS Generator

i	X_i	B_i
0	20749	
1	143135	1
2	177671	1
3	97048	0
4	89992	0
5	174051	1
6	80649	1
7	45663	1
8	69442	0
9	186894	0
10	177046	0

i	X_i	B_i
11	137922	0
12	123175	1
13	8630	0
14	114386	0
15	14863	1
16	133015	1
17	106065	1
18	45870	0
19	137171	1
20	48060	0

n = 192649 = 383 * 503, and the seed s = 101355.

PRNG Using Block Cipher Modes of Operation

- Two approaches that use a block cipher to build a PNRG have gained widespread acceptance:
  - CTR mode
    - Recommended in NIST SP 800-90, ANSI standard X.82, and RFC 4086
  - OFB mode
    - Recommended in X9.82 and RFC 4086

Figure 7.4 PRNG Mechanisms Based on Block Ciphers

Table 7.2

Key:	cfb0ef3108d49cc4562d5810b0a9af60
V:	4c89af496176b728ed1e2ea8ba27f5a4

Output Block	Fraction of One Bits	Fraction of Bits that Match with Preceding Block
1786f4c7ff6e291dbdfdd90ec3453176	0.57	—
5e17b22b14677a4d66890f87565eae64	0.51	0.52
fd18284ac82251dfb3aa62c326cd46cc	0.47	0.54
c8e545198a758ef5dd86b41946389bd5	0.50	0.44
fe7bae0e23019542962e2c52d215a2e3	0.47	0.48
14fdf5ec99469598ae0379472803accd	0.49	0.52
6aeca972e5a3ef17bd1a1b775fc8b929	0.57	0.48
f7e97badf359d128f00d9b4ae323db64	0.55	0.45

Example Results for PRNG Using OFB

Table 7.3

Key:	cfb0ef3108d49cc4562d5810b0a9af60
V:	4c89af496176b728ed1e2ea8ba27f5a4

Output Block	Fraction of One Bits	Fraction of Bits that Match with Preceding Block
1786f4c7ff6e291dbdfdd90ec3453176	0.57	—
60809669a3e092a01b463472fdcae420	0.41	0.41
d4e6e170b46b0573eedf88ee39bff33d	0.59	0.45
5f8fcfc5deca18ea246785d7fadc76f8	0.59	0.52
90e63ed27bb07868c753545bdd57ee28	0.53	0.52
0125856fdf4a17f747c7833695c52235	0.50	0.47
f4be2d179b0f2548fd748c8fc7c81990	0.51	0.48
1151fc48f90eebac658a3911515c3c66	0.47	0.45

Example Results for PRNG Using CTR

ANSI X9.17 PRNG

- One of the strongest PRNGs is specified in ANSI X9.17
  - A number of applications employ this technique including financial security applications and PGP

The algorithm makes use of triple DES for encryption. Ingredients are:

Input

- Two pseudorandom inputs drive the generator. One is a 64-bit representation of the current date and time. The other is a 64-bit seed value; this is initialized to some arbitrary value and is updated during the generation process.

Keys

- The generator makes use of three triple DES encryption modules. All three make use of the same pair of 56-bit keys, which must be kept secret and are used only for pseudorandom number generation.

Output

- The output consists of a 64-bit pseudorandom number and a 64-bit seed value.

Figure 7.5 ANSI X9.17 Pseudorandom Number Generator

NIST CTR_DRBG

- Counter mode-deterministic random bit generator
- PRNG defined in NIST SP 800-90 based on the CTR mode of operation
- Is widely implemented and is part of the hardware random number generator implemented on all recent Intel processor chips
- DRBG assumes that an entropy source is available to provide random bits
  - Entropy is an information theoretic concept that measures unpredictability or randomness
- The encryption algorithm used in the DRBG may be 3DES with three keys or AES with a key size of 128, 192, or 256 bits

Table 7.4

	3DES	AES-128	AES-192	AES-256
outlen	64	128	128	128
keylen	168	128	192	256
seedlen	232	256	320	384
reseed_interval	$\leq 2^{32}$	$\leq 2^{48}$	$\leq 2^{48}$	$\leq 2^{48}$

CTR_DRBG Parameters

Stream Ciphers

Figure 7.7 Stream Cipher Diagram

Stream Cipher Design Considerations

The encryption sequence should have a large period

- A pseudorandom number generator uses a function that produces a deterministic stream of bits that eventually repeats; the longer the period of repeat the more difficult it will be to do cryptanalysis

The keystream should approximate the properties of a true random number stream as close as possible

- There should be an approximately equal number of 1s and 0s
- If the keystream is treated as a stream of bytes, then all of the 256 possible byte values should appear approximately equally often

A key length of at least 128 bits is desirable

- The output of the pseudorandom number generator is conditioned on the value of the input key
- The same considerations that apply to block ciphers are valid

With a properly designed pseudorandom number generator a stream cipher can be as secure as a block cipher of comparable key length

- A potential advantage is that stream ciphers that do not use block ciphers as a building block are typically faster and use far less code than block ciphers

RC4

- Designed in 1987 by Ron Rivest for RSA Security
- Variable key size stream cipher with byte-oriented operations
- Based on the use of a random permutation
- Eight to sixteen machine operations are required per output byte and the cipher can be expected to run very quickly in software
- Used in the Secure Sockets Layer/Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers
- Is also used in the Wired Equivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA) protocol that are part of the IEEE 802.11 wireless LAN standard

Strength of RC4

A number of papers have been published analyzing methods of attacking RC4

- None of these approaches is practical against RC4 with a reasonable key length

A more serious problem is that the WEP protocol intended to provide confidentiality on 802.11 wireless LAN networks is vulnerable to a particular attack approach

- The problem is not with RC4 itself, but the way in which keys are generated for use as input
- Problem does not appear to be relevant to other applications and can be remedied in WEP by changing the way in which keys are generated
- Problem points out the difficulty in designing a secure system that involves both cryptographic functions and protocols that make use of them

Entropy Sources

- A true random number generator (TRNG) uses a nondeterministic source to produce randomness
- Most operate by measuring unpredictable natural processes such as pulse detectors of ionizing radiation events, gas discharge tubes, and leaky capacitors
- Intel has developed a commercially available chip that samples thermal noise by amplifying the voltage measured across undriven resistors
- LavaRnd is an open source project for creating truly random numbers using inexpensive cameras, open source code, and inexpensive hardware

Possible Sources of Randomness

RFC 4086 lists the following possible sources of randomness that can be used on a computer to generate true random sequences:

Sound/video input

- The input from a sound digitizer with no source plugged in or from a camera with the lens cap on is essentially thermal noise
- If the system has enough gain to detect anything, such input can provide reasonable high quality random bits

Disk drives

- Have small random fluctuations in their rotational speed due to chaotic air turbulence
- The addition of low-level disk seek-time instrumentation produces a series of measurements that contain this randomness

There is also an online service (random.org) which can deliver random sequences securely over the Internet

Table 7.5

	Pseudorandom Number Generators	True Random Number Generators
Efficiency	Very efficient	Generally inefficient
Determinism	Deterministic	Nondeterministic
Periodicity	Periodic	Aperiodic

Comparison of PRNGs and TRNGs

Skew

- A TRNG may produce an output that is biased in some way, such as having more ones than zeros or vice versa
  - Deskewing algorithms
    - Methods of modifying a bit stream to reduce or eliminate the bias
    - One approach is to pass the bit stream through a hash function such as MD5 or SHA-1
    - RFC 4086 recommends collecting input from multiple hardware sources and then mixing these using a hash function to produce random output
- Operating systems typically provide a built-in mechanism for generating random numbers
  - Linux uses four entropy sources: mouse and keyboard activity, disk I/O operations, and specific interrupts
    - Bits are generated from these four sources and combined in a pooled buffer
    - When random bits are needed the appropriate number of bits are read from the buffer and passed through the SHA-1 hash function

Summary

- Principles of pseudorandom number generation
  - The use of random numbers
  - TRNGs, PRNGs, and PRFs
  - PRNG requirements
  - Algorithm design
- Pseudorandom number generators
  - Linear congruential generators
  - Blum Blum Shub generator
- Pseudorandom number generation using a block cipher
  - PRNG using block cipher modes of operation
  - ANSI X9.17 PRNG
  - NIST CTR_DRBG

- Stream ciphers
- RC4
  - Initialization of S
  - Stream generation
  - Strength of RC4
- True random number generators
  - Entropy sources
  - Comparison of PRNGs and TRNGs
  - Skew
  - Intel digital random number generator
  - DRNG hardware architecture
  - DRNG logical structure

Linear Congruential Method

$X_{n+1} = (aX_n + c) \bmod m$

Examples

1. Consider a = c = 1.
The sequence produced is obviously not satisfactory.

2. Consider a = 7, c = 0, m = 32, and X0 = 1.

This generates the sequence {7, 17, 23, 1, 7, etc.}, which is also clearly unsatisfactory. Of the 32 possible values, only four are used; thus, the sequence is said to have a period of 4.

3. If, instead, we change the value of a to 5, then the sequence is {5, 25, 29, 17, 21, 9, 13, 1, 5, etc.}, which increases the period to 8.

Blub Blub Shub

The procedure is as follows.

1. First, choose two large prime numbers, *p* and *q*, that both have a remainder of 3 when divided by 4. That is,

$p = q = 3 \pmod 4$ it means (p mod 4) = (q mod 4) = 3.

For example, prime numbers 7 and 11 satisfy 7 = 11 = 3(mod 4).

2. Let $n = p * q$.

3. Next, choose a random number *s*, such that *s* is relatively prime to *n*; this is equivalent to saying that neither *p* nor *q* is a factor of *s*.