

Week 1

Overview Lecture

Subject Overview

Lecture 1

Introduction to cryptography.

Lecture 2

Introduction to Numbers

Workshops start from Week 2

Quiz 1

Introduction to cryptography

COMP90043

Lecture 1

Introduction to cryptography



Lecture 1

1.1 Information Security

- Definitions, Role of Cryptography, Cyber Security
- Story of Cryptography since ancient times
- A story of Alice and Bob: terms and notations

1.2 Motivating Examples

- Practical Banking
- A Communication Game:

1.3 Classical example

- Diffie-Hellman Protocol

1.4 Basic Security Objectives

1.1 Information Security

COMP90043

Lecture 1

Information Security

Definitions, Role of Cryptography, Cyber Security

- What is Cryptography?
 - “Secret Writing”
 - Refers to the techniques required for protecting data between authorized parties on information communication technologies in the presence of potentially malicious elements.
 - Refers to a range of techniques such as Encryption, Signature, Hash functions, assuring Privacy, Integrity, and Authentication of data in the digital world.
- What is Information Security?
 - A broad topic of exchange and processing of information on modern computers and networks.
 - Confidentiality, Integrity, and Availability.
- What is Cyber Security?
 - Refers to management of attacks and risks by adversarial and malicious elements on computers and networks that support modern businesses and economy involving business, government, and community.

Information Security

The field of Network and Internet security

- Stallings Take:
 - The field of network and Internet security consists of measures to deter, prevent, detect, and correct security violations that involve the transmission of information.
- Our Approach:
 - Is to study certain basic cryptographic primitives such as symmetric and public key cryptography, hash functions, message authentication and signatures, and use them explore the field of network and Internet security protocols.

Story of Cryptography since ancient times



Alice

Let us
meet in
the
alley
today



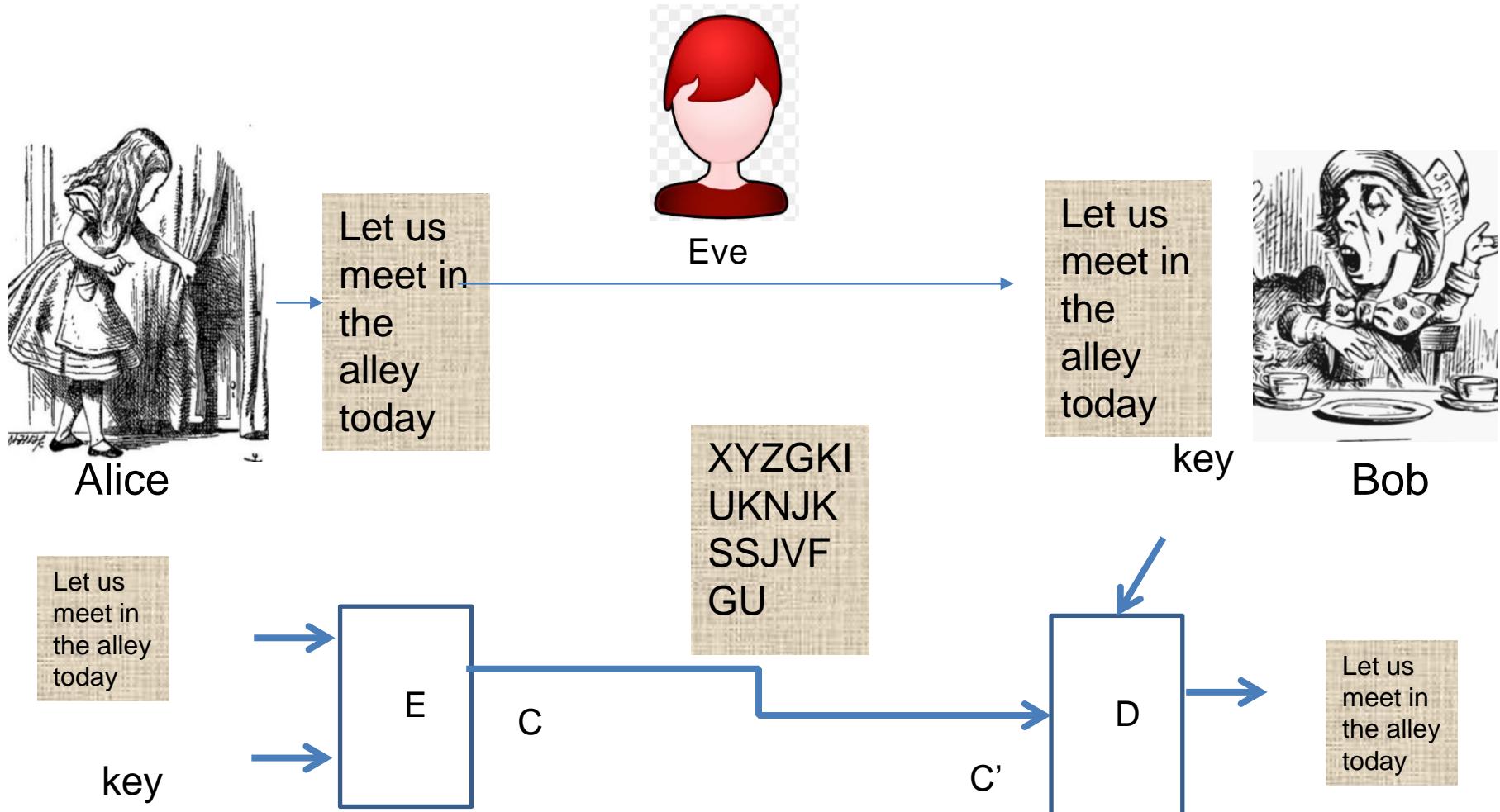
Eve

Let us
meet
in the
alley
today



Bob

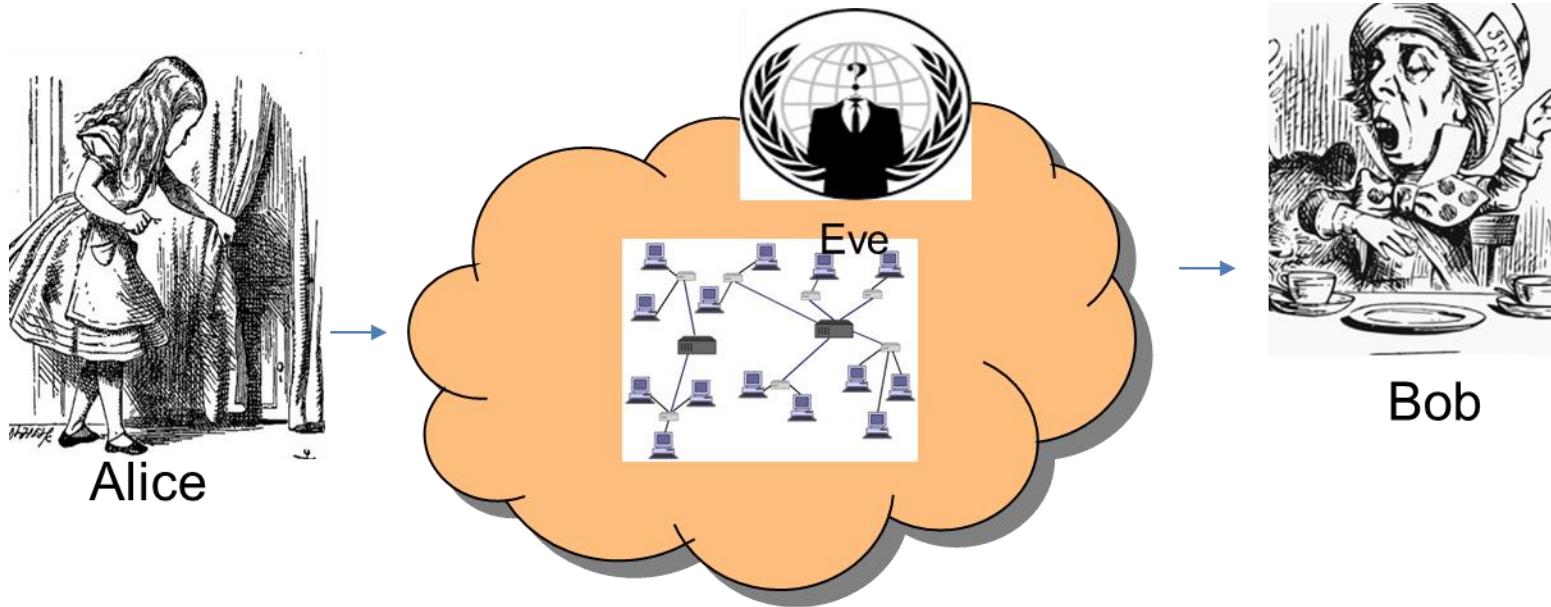
Story of Cryptography since ancient times



How do they agree on the “key”? -Chicken and Egg Problem

© University of Melbourne, Udaya
Parampalli

Fast forward: In Modern times



Alice and Bob cannot meet in advance for every situation

Can Mathematics come to their aid?

The magic tool is the One Way function.

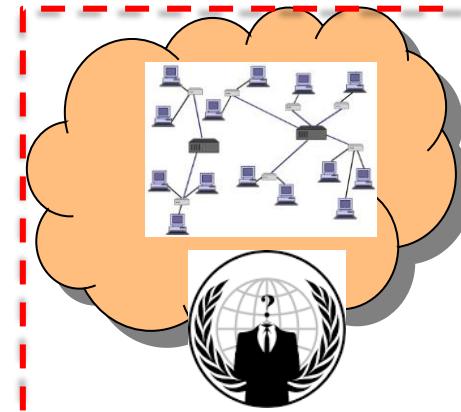
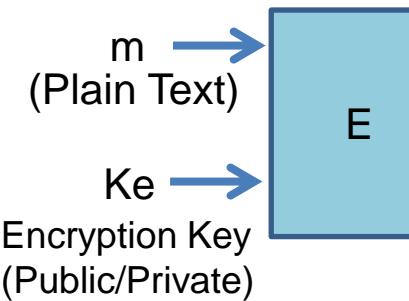
We will consider many such functions based on numbers in the subject

Story of Alice and Bob terms and notations



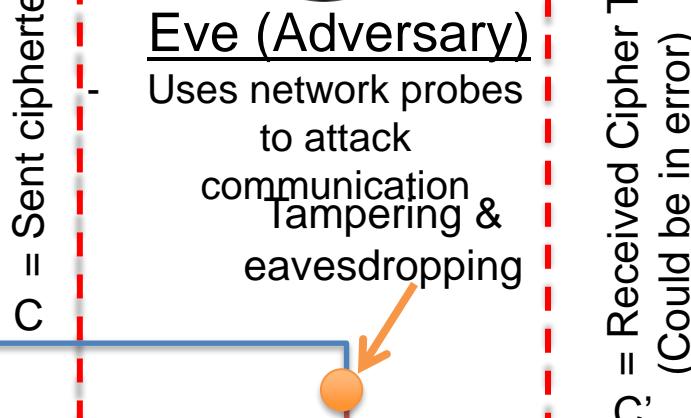
Alice (Sender)

- Uses an Encryption Function (E)



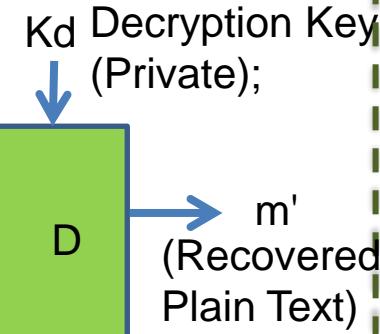
Eve (Adversary)

- Uses network probes to attack
- communication
- Tampering & eavesdropping



Bob (Receiver)

- Uses a Decryption Function (D)



E, D are public; c is the ciphertext, c' is received ciphertext; ideally $m=m'$;

Cryptography involves many conceptual ideas, we look at the basic functions

Differences

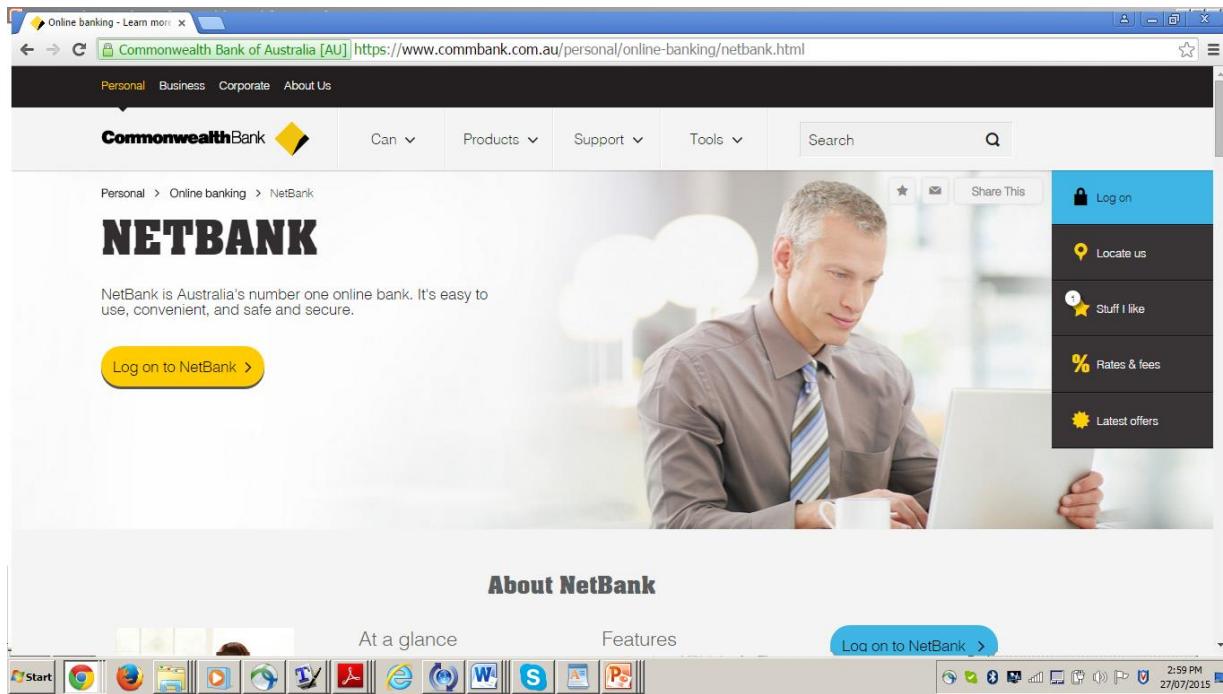
- $Ke = Kd$:Symmetric key also sometimes referred as private key. But we shall call always symmetric key-
 - Known since antiquity.
- $Ke \neq Kd$:Asymmetric or Public Key Cryptography –
 - Fairly recent- since 1974 after the celebrated paper by Diffie-Hellman.
 - Please read this paper. I have added a link to this page in LMS.

1.2 Motivating Examples

COMP90043

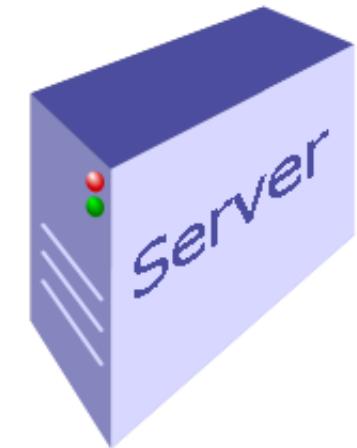
Lecture 1

Motivating examples



A screenshot of the Commonwealth Bank NetBank website. The page features a banner with a man working on a laptop. On the right, there's a sidebar with links like 'Log on', 'Locate us', 'Stuff I like', 'Rates & fees', and 'Latest offers'. Below the banner, there's a section titled 'About NetBank' with tabs for 'At a glance' and 'Features'. A 'Log on to NetBank' button is visible at the bottom.

Comm bank Server

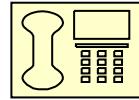


Issues in getting your money from the bank.
Should work over Internet
Think, who is Alice, Bob and Eve here.
What tools Cryptography can provide here?

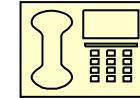
A Communication Game



Alice



Dating Problem!



Bob

Alice and Bob want to spend an evening together.

They want to decide whether to go to Music concert or Cinema

They can resolve either way by tossing a coin.

If they can meet together, it is a simple task.

However, they are in different offices connected by a telephone.

They need to book the program in advance and want to make decision over the phone.

Can you help them?

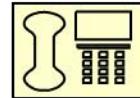
A Cryptographic Solution Using Mathematics!

- Assume we have a magic function with
 - A. For every integer x , it is EASY to compute $f(x)$ from x , however given a value for $f(x)$ it is impossible to find x which is the pre-image of $f(x)$, eg. To decide if x is odd or even
 - A. It is impossible to find a pair of integers with x not equal to y and $f(x) = f(y)$
- Even number x in $f(x)$ denotes EVEN and the other case denotes ODD.

A protocol



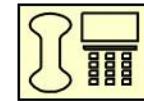
Alice



Choose a random x and compute $f(x)$

Dating Problem!

EVEN: HEADS
ODD: TAILS



Bob

Guesses x is even or Odd

Send x

Verify $x = f(x)$
check if his guess is correct or not

Whoever wins the game decides the venue of the meeting!

Is this protocol correct and fair (unbiased)?
Can you modify so that both Alice and Bob

If the line is not secure: Some questions

- They need to introduce traditional cryptography to secure the line
- Symmetric key or Asymmetric key?
- Or Use Different methods of communication where intruder cannot read the channel.
- We will discuss cryptographic solutions.

Models for Information Security

- Traditional Communication Model:
 - Alice and Bob are connected by an insecure channel. Marvin, an adversary, can listen to their conversation and modify if needed.
- Modern Network Model:
 - Network itself is an adversary. More than two participants. A valid participant also can be an adversary to others. Many models exist.

If the line is not secure: Some questions

- They need to introduce traditional cryptography to secure the line
- Symmetric key or Asymmetric key?
- Or Use Different methods of communication where intruder cannot read the channel.
- We will discuss cryptographic solutions.

If the line is not secure: Some questions

- They need to introduce traditional cryptography to secure the line
- Symmetric key or Asymmetric key?
- Or Use Different methods of communication where intruder cannot read the channel.
- We will discuss cryptographic solutions.

One-Way functions

- Does One Way functions exist?
- This simple question rises lots of philosophical issues. Cryptographers would like believe that they exist and have come up with many practical one-way functions.
- Do they have a clear cut proof for these claims?
- On the other hand, cryptanalysts believe in the opposite and work towards breaking the claims of cryptographers.

1.3 Classical example

COMP90043

Lecture 1

Diffie-Hellman Idea: Basics

- Two users want to share a common secret over a public network, Is this possible? Think!
- For a moment assume that we have a one way function.
- What is one way function?
 - Given x in domain it is easy to compute $f(x)$
 - Given y in range, it is difficult find x in domain such that $f(x)=y$

DH Continued

- Alice can create **x** in a domain (agreed in advance) –keep it secret,
- Compute $f(x)$ – Send it to Bob over public channel
- Bob can create secret **y** in the domain and he also computes $f(y)$ – Send it back to Alice
- Now both of them have $f(x)$ and $f(y)$ -
- If f is such that they can workout a common function of their secrets which others who observed $f(x)$ and $f(y)$ cannot compute, then one can attempt to have a solution to this problem.
- **Diffie-Hellman in their 1974 paper give one such concrete solution!**
Please read it, you will love the idea.

Prime Numbers

- A number is said to be a prime number if $p > 1$ and p has no positive divisors except 1 and p .
- Example: $p = 2, 3, 5, 7, 11, 13$
- The numbers which are not prime numbers are referred as composite numbers.
- For any integer n , $n > 1$, let $Z_n = \{0, 1, 2, \dots, n-1\}$ be a set of numbers. This set is called the set of **residues modulo n** , as the elements are remainders of integers divided by the number n .
- We define the following operations on the set Z_n using the modulo operation.

$$x \oplus_n y = (x + y) \bmod n.$$

$$x \otimes_n y = (xy) \bmod n$$

- Example: $(6 + 7) \bmod 12 = 1$; $5 \times 4 \bmod 12 = 8$;
- In this lecture, n will only be a prime number.

Clock Arithmetic



Modular Inverse

Definition

Let $x \in Z_n$, if there is an integer y such that

$$x \otimes_n y = 1,$$

then we say y is the multiplicative inverse of x . It is denoted by $y = x^{-1}$ usually.

Example: let $n = 5$, 2 is inverse of 3 in Z_5 . Or in other words 2 is inverse of 3 modulo 5.

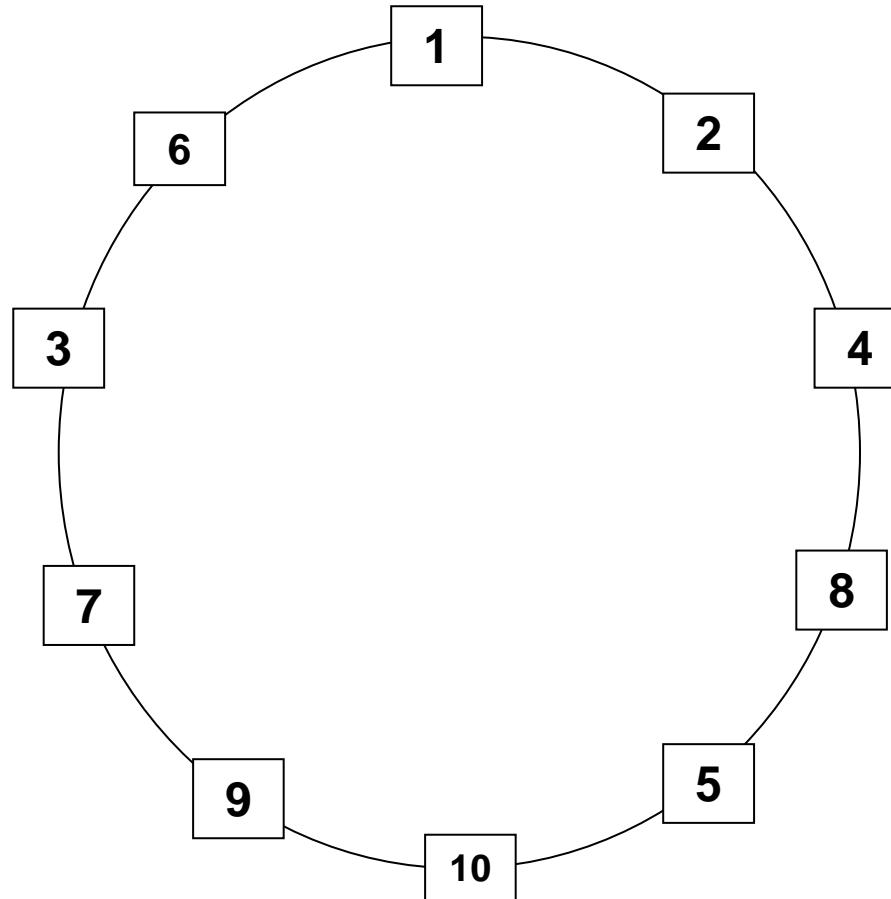
We can now define a cyclic group over nonzero elements of Z_p when p is prime.

Let $Z_p^* = \{1, 2, 3, \dots, (p-1)\}$. Let g be an element of Z_p^* such that

$Z_p^* = \{g, g^2, g^3, \dots, g^{p-1}=1\}$, (*you can always find such an element g)

*We do not cover this idea here, it requires more study; those interested can see the textbook

An example



Example of a Cyclic group modulo $p = 11$

g : generator = 2

26/07/2022

Order(size) of G = 10

© University of Melbourne, Udaya
Parampalli

What power of 2 is 3?

g^i	$g^i \text{ mod } p$	$D\log(g^i)$
2^1	2	1
2^2	4	2
2^3	8	3
2^4	5	4
2^5	10	5
2^6	9	6
2^7	7	7
2^8	3	8
2^9	6	9
2^{10}	1	10

The Example of One Way Function

X	$2^x \bmod 11$
0	1
1	2
2	4
3	8
4	5
5	10 Or -1
6	9
7	7
8	3
9	6
10	1
11	2

Discrete Logarithm Problem (DLP)

Let ‘ g ’ and ‘ h ’ be elements of the group G . Then the discrete logarithm (DL) problem is the problem of finding ‘ x ’ such that $g^x = h$.

For example, the solution to x in the problem:

$$3^x \equiv 13 \pmod{17} \rightarrow x = 4, \text{ because } 3^4 = 81 \equiv 13 \pmod{17}.$$

- o The discrete log problem is believed to be hard. Therefore it has become the basis of several public key schemes, for example: El-Gamal.
- o Next, we will consider the Diffie-Hellman protocol, the first public key algorithm.
- o The protocol is defined over a cyclic group: $Z_p^* = \{g, g^2, g^3, \dots, g^{p-1} = 1\}$,

Diffie-Hellman Key Establishment Protocol

- Alice
- Choose $N_a=2$
- $g^{N_a} = 2^2=4 = M_a$

Bob
Choose $N_b=6$



-
- Compute
- $K_{ab} = M_b^{N_a}$
- $=9^2=4$

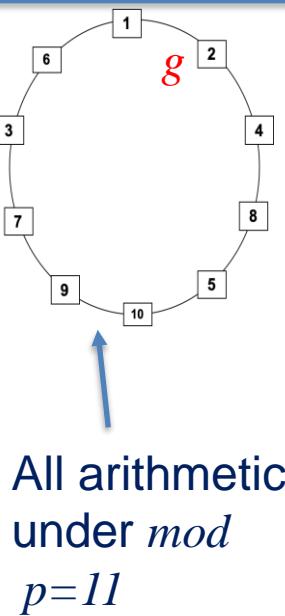


$g^{N_b} = 2^6=9=M_b$

-
-
- $K_{ab} = K_{ba}=4$

Compute
 $K_{ba} = M_a^{N_b} = 4^6=4$

Diffie-Hellman Protocol



Alice
Choose $Na=2$
Choose $Nb=6$

$p=11, g=2$
Eve

$$g^{Na} = 2^2 = 4 = Ma$$

Compute

$$K_{ab} = Mb^{Na} = (g^{Nb})^{Na} \\ = 9^2 = 4$$

$$K_{ab} = K_{ba} = 4 = (g^{NaNb})$$

Bob

$$g^{Nb} = 2^6 = 9 = Mb$$

Compute

$$K_{ba} = Ma^{Nb} = (g^{Na})^{Nb} \\ = 4^6 = 4$$



Whitfield Diffie
and
Martin Hellman

CDH PROBLEM

Problem for Eve in the above protocol

Clearly a solution to DL implies a solution to CDH

Is the converse True?*

* Open Problem

Let G be a cyclic group of size q and g be a generator of the group G . Given g^a and g^b , two arbitrary elements of the group G for some integers a and b in the range: $0 \leq a, b \leq q$, then find g^{ab}
Normally G is a multiplicative group in a suitable finite field.

New directions in Cryptography, IEEE Trans. Inf. Theory 22(6): 644-654 (1976)

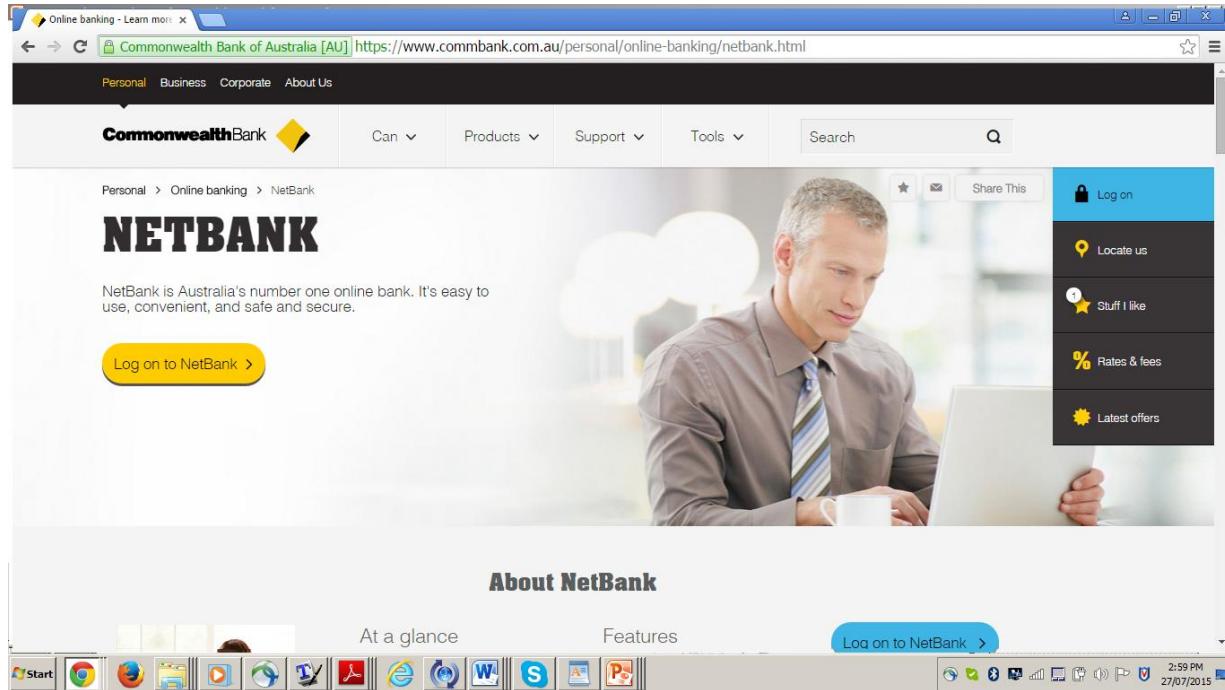
Issues with this Protocol: Secure?

- Exchanged data -only g^{N_a} and g^{N_b}
- So Alice cannot guess N_b nor Bob can guess N_a
- So their secrets are safe from each other
- But also none can guess N_a and N_b for the same reason
- Both Alice and Bob can compute common secret $g^{N_a N_b}$
- It is also believed that $g^{N_a N_b}$ cannot be computed by others who can only see g^{N_a} and g^{N_b}
- **The later problem is known as Computational Diffie-Hellman problem (Hard!)**

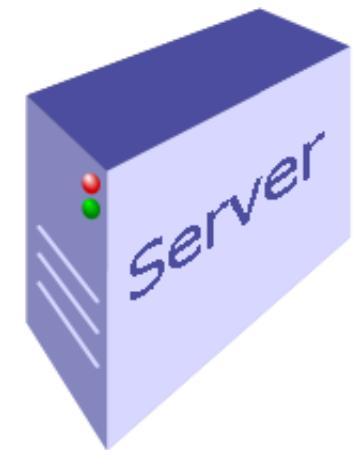
Man in the Middle Attack

- Alice
 - g^{Na}
 - Eve
 - g^{Nm}
 - Bob
 - g^{Nb}
 - g^{NaNm}
 - g^{NmNb}
- Marvin comes in between Alice and Bob, and he can create two secrets one with Alice and the other with Bob.
- This is possible because when Bob receives communication from Alice, there is no way for him to determine if it indeed come from Alice, in other words, the messages are not authenticated.
- A way to solve this problem is by using digital signatures! –We will revisit these ideas when we visit Public Key topics later in the semester.

Comm bank Server



A screenshot of a web browser displaying the Commonwealth Bank of Australia NetBank homepage. The URL in the address bar is <https://www.commbank.com.au/personal/online-banking/netbank.html>. The page features a large image of a man in a suit using a laptop. A sidebar on the right contains links for 'Log on', 'Locate us', 'Stuff I like', 'Rates & fees', and 'Latest offers'. At the bottom, there are links for 'About NetBank' and 'At a glance'.



1.4 Basic Security Objectives

COMP90043

Lecture 1

Three important concerns of Information security



- Confidentiality
 - In simple terms, confidentiality of information or data ensures that the access is given only to authorized individuals.
- Integrity
 - Information integrity ensures that enough safe guarding mechanisms exists so that authorized individuals get the **right** information and any changes to the information by intentional and un intentional means will be detected.
- Availability
 - Information or data availability ensures that the information is authorized available to the users.

From the textbook definitions

OSI Security Architecture

- How to define the requirements for security in networked world and characterizing the approaches to satisfy those requirements?
- Refer to ITU-T X.800 “Security Architecture for OSI”
 - It defines a systematic way of defining and providing security requirements
- Three main aspects:
 - Security attacks
 - Security Mechanisms.
 - Security services.

Security Attack

- *Attack* is any action that compromises the security of information owned by an organization
- *Threat* is a possible potential for violation of security,
- Information security is about how to prevent attacks, or failing that, to detect attacks on information-based systems
- often *threat* & *attack* used to mean same thing (threat is attack in waiting)
- Generally we have a wide range of attacks:
- Some generic types of attacks:
 - passive
 - active

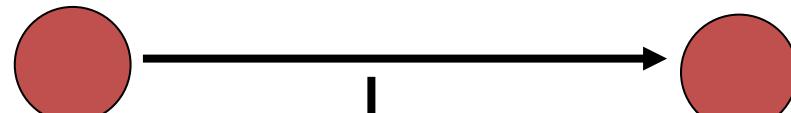
Basics Security Services

We concentrate on Implementation and Mechanism aspects of Information Security.

- Authentication
- Confidentiality
- Integrity
- Nonrepudiation
- Availability

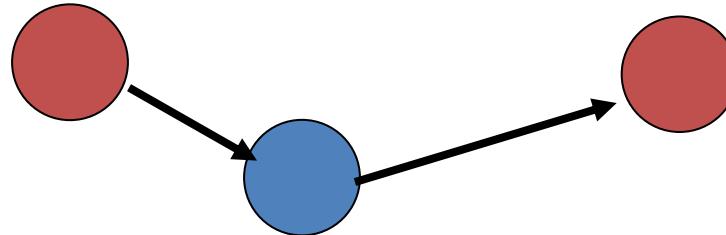
Security Threats in Networked World

- Security services are defined to address or withstand threats



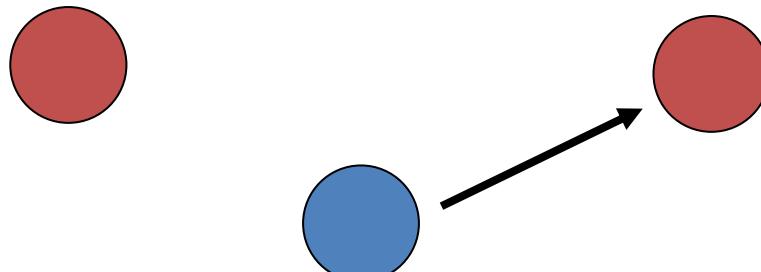
Interception

Confidentiality



Modification

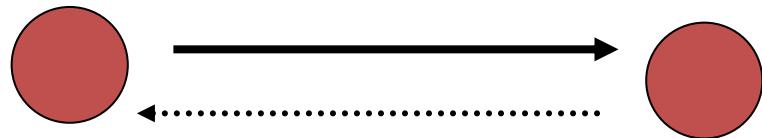
Integrity



Fabrication

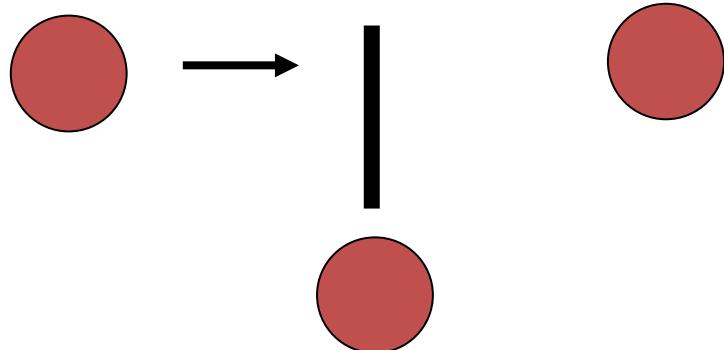
**Confidentiality
Authentication**

Security Threats in Networked World



Non-Repudiation

**Source
Authentication**



Interruption

Network QOS

Model for Network Security (Textbook)

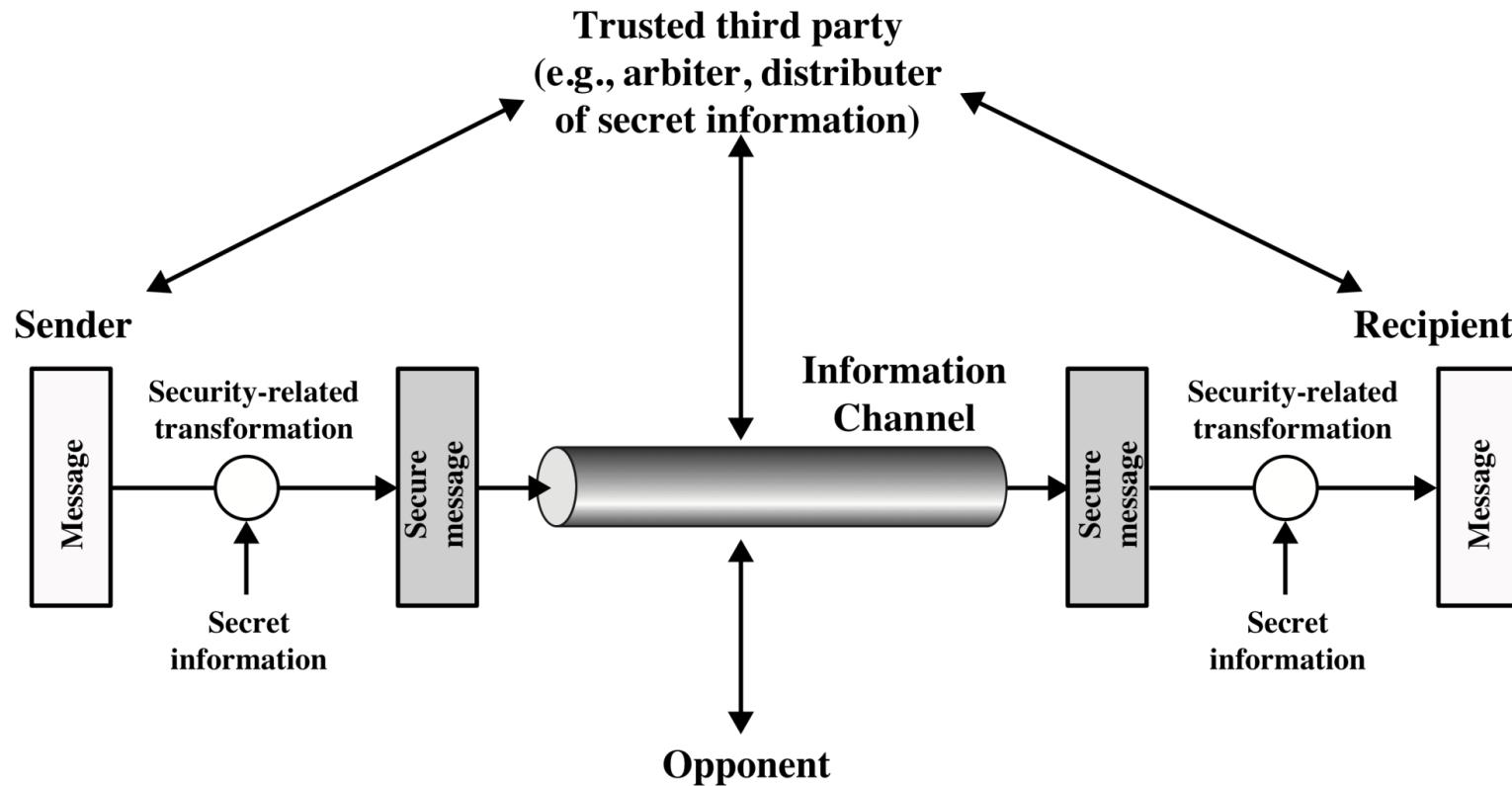


Figure 1.2 Model for Network Security

Source: William Stallings, Cryptograph and Security

Network Access Security Model

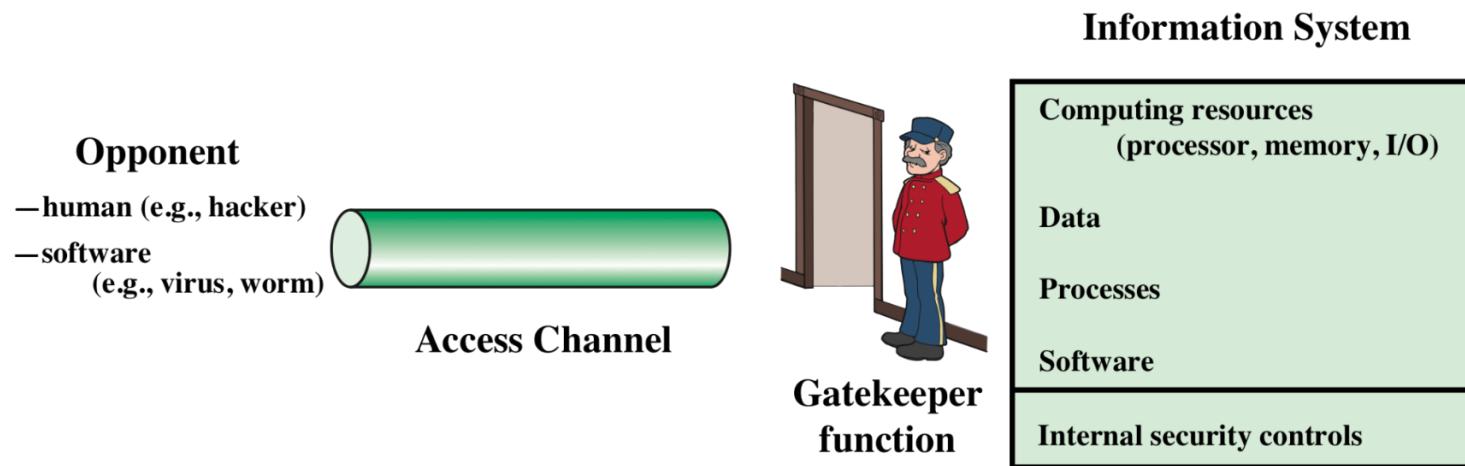


Figure 1.3 Network Access Security Model

Source: William Stallings, Cryptograph and Security

Week 1

Overview Lecture
Subject Overview

Lecture 1
Introduction to cryptography.

Lecture 2
Introduction to Numbers
Workshops start from Week 2
Quiz 1

This Week



Overview Lecture

Subject Overview

Lecture 1

Introduction to cryptography

Lecture 2

Introduction Continued

Workshops start from Week 2

Quiz 1

Subject Overview

Overview Lecture

Instructors and Tutors

Instructors

- Udaya Parampalli, Udaya@unimelb.edu.au

Tutors

- Ms Wenjun Zhou, wenjun.zhou@unimelb.edu.au, Head tutor
- Mr Faxing Wang, faxing.wang@student.unimelb.edu.au, Tutor
- Ms Medha Mishra, mishramedha1110@gmail.com, Tutor
- Mr Patrick Vicky, patr.vick@gmail.com, Marker

A little bit of myself

- Udaya Parampalli,
 - Professor and Reader, Leader-Quantum Computing Research, School of Computing and Information Systems
- Research Interests:
 - Quantum computing and Post Quantum Cryptography
 - Steganography or Information Hiding
 - Cryptography for Networks and Communications
 - Sequence design for Radar and Communications
 - Coding theory for Storage and DNA
- Publications:
 - <http://people.eng.unimelb.edu.au/udaya/>

How to contact me?

- Preferably at the end of lectures
- Email: udaya@unimelb.edu.au (Include the word COMP90043 in subject field)
- Expect 48 hours turn around on occasions!
- Level 2, Melbourne Connect,
- Consultation: Times will be announced on LMS and also by appointment.

Subject Structure

- 12 Weeks of Lectures
 - 2 lectures (maximum of 3 hours per week) + 1 hour of tutorial (in parallel sessions)
- Assessment:
 - 40% Final examination
 - 2 Hour Final examination
 - Mid-Semester Test (10%)[Tentative date: Week 7]
- 50% Project/Assignment
 - 2 Assignments (7.5% each individual work)
 - Weekly Quiz
 - 2 bonus marks for completing 8 out of 10 quizzes (80%).
 - 1 Research project (35% Total) a group project-details will be released soon)
 - Part A: Presentation in Week 10 (10%)
 - Part B: Research Report due in Week 12 (25%)

Bonus applies to the Assignment component not exceeding max cap of 15

Research Project

- Group Project, group size of maximum 3, we would prefer groups of 3 people.
- Project should be based on a topic that involves Cryptography.
 - A list of suggested topics will be available.
 - You should organize your groups preferably with members from same tutorial group
 - Your tutor will be the first point of contact for any discussion on the project
- You need to choose a topic and propose a topic for the research project. The proposal should detail the rough division of work amongst group members.
- The Body of the work:
 - Implementations:
 - Problem Identification
 - Analysis
 - Conclusion
- Marks breakdown:
 - Part A: Presentation in Week 10 or 11 (10%)
 - Part B: Research Report (25%) due in Week 12

Hurdle Requirements

- To pass the subject, students must obtain at least:
 - 50% overall.
 - 50% in the homework assignments
 - Note that by completing 80% of the online quizzes you can earn 2 bonus marks.
- 50% in the research project
- 50% in the end-of-semester written examination
- No hurdle for the mid-semester test component

Intended Learning Outcomes (ILO)

- ILO1: Identify security issues and objectives in computer systems and networks.
- ILO2: Apply various security mechanisms derived from cryptography to computers and computer networks.
- ILO3: Explain the workings of fundamental public key and symmetric key cryptographic algorithms including RSA, ElGamal, Diffie-Hellman schemes and stream ciphers.
- ILO4: Explain the protocols which ensure security in contemporary networked computer systems.
- ILO5: Describe the interaction between the underlying theory and working computer security infrastructure.
- ILO6: Analyze security of network protocols and systems.

Lecture Times



COMP90043: Cryptography and Security

- Two lectures per week, total time maximum of 3 hours*.
 - Tuesday 15.15 to 17.15,
 - Thursday, 17:15 to 18:15 hrs.
- Note that in the subject you are expected to work on programs on departmental servers. There will not be any official laboratory workshops. You will need to work yourselves. We will provide consultations.
- We may have some guest lectures and revisions in some lectures. The contents in some of these guest lectures are examinable

Subject Resources

- Textbook: Cryptography and Network Security: Principles and Practice, 7/E by William Stallings

References:

- Douglas R Stinson, Cryptography, Theory and Practice, Chapman & Hall/CRC, 2006.
- Richard E. Smith, INTERNET CRYPTOGRAPHY, ADDISON WESLEY, 1997.
- Andrew S. Tanenbaum , COMPUTER NETWORKS, Fourth Edition, Prentice-Hall International, Inc, 2002.
- Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, October 1996.
- Wenbo Mao, ``Modern cryptography Theory and Practice'', www.hp.com/hpbooks, Pearson Education, Prentice Hall, 2004.
- Articles from Lecture Notes in Computer Science series covering security and cryptography

Subject Outline

- This subject covers fundamental concepts in information security on the basis of methods from modern cryptography. We will concentrate on topics which are of current interest as well as the more `classic' topics which underlay this discipline.

Topics drawn from:

- symmetric key and public key cryptosystems,
- hash functions,
- authentication
- secret sharing
- Protocols
- Key Management

There will be some guest lectures in specialized topics.

Subject Description

The objective of this subject is for students

- to understand the fundamentals of security principles in modern networks and computer systems,
- to be able to explain the protocols which ensure security in contemporary networked computer systems;
- to study various cryptographic primitives like encryption, hashing and signature functions which are used in theory and practice of network security.

Course Plan (Dates to be Confirmed)

Topics by week:

- 1. Introduction to Cryptography and Security (Ch 1), Introduction to Numbers,
- 2. Symmetric Ciphers, Classical Ciphers., (Group Formation) (Assignment 1 handed out)
- 3. Modern Symmetric Ciphers: Block and Stream Ciphers (Ch 2,3,6,7)
- 4. Basics from Number Theory (Ch 8)
- 4. Public Key Cryptography and RSA (Ch 9) (Assignment 1 due)
- 5. Hash functions (Ch 11) (Project topics confirmation)(Assignment 2 handed out)
- 6. Message Authentication Codes (Ch 12)
- 7. Digital Signatures (Ch 13) (Mid Semester Test) (Assignment 2 due)
- 8. Key management,
- 9. Key management cont., Secret Sharing (Ch 14)
- 10. Guest Lecture/Project Presentations
- 11. Application/Advanced Topics (Part 5)
- 12. Review, Report Due

Generic Skills



- GS1: Ability to undertake problem identification, formulation, and solution.
- GS2: Ability to utilise a systems approach to solving complex problems and to design for operational performance
- GS3: Ability to manage information and documentation
- GS4: Capacity for creativity and innovation
- GS5: Ability to communicate effectively, with the engineering team and with the community at large

Academic Standards

From your canvas home page access **Academic integrity module**

- <https://catalog.lms.unimelb.edu.au/browse/communities/student-induction/>
- Even if you have completed this module before, you should listen to the talks and complete the reflection exercises again. This will take less than 15 minutes. The experience will help you to realize the importance of academic honesty which is the key progress in your studies.

Week 2



Lecture 1

Part -1 Extended GCD Algorithm and Related Computations

Part 2 - Symmetric key Cryptography

Lecture 2

Properties of Numbers,

Workshop 2: Workshops start from this week.

Quiz 2

Symmetric key Cryptography

COMP90043
Lecture 2-Part II

Symmetric key Cryptography

Lecture 2 Part II

1.1 Symmetric Cipher Models

- Basic Terminology
- Model and Logical View
- Basic Requirements and Kerckhoffs's principle

1.2 Security

- Characterization of Symmetric key Encryption
- Attacks on Symmetric key Encryption

1.3 Classical Ciphers

- Substitution Ciphers
 - Caesar and Affine Ciphers
 - Monoalphabetic Substitution Ciphers
- Transposition Ciphers
 - Rail fence cipher
 - Row Transposition Cipher

1.4 Cryptanalysis of Classical Ciphers

- Caesar Cipher
- Affine Cipher
- Monoalphabetic Substitution Ciphers

1.5 Complex Ciphers

Polyalphabetic Ciphers Vigenère Cipher

1.1 Symmetric Cipher Models

COMP90043
Lecture 2-Part II

Symmetric Key Encryption

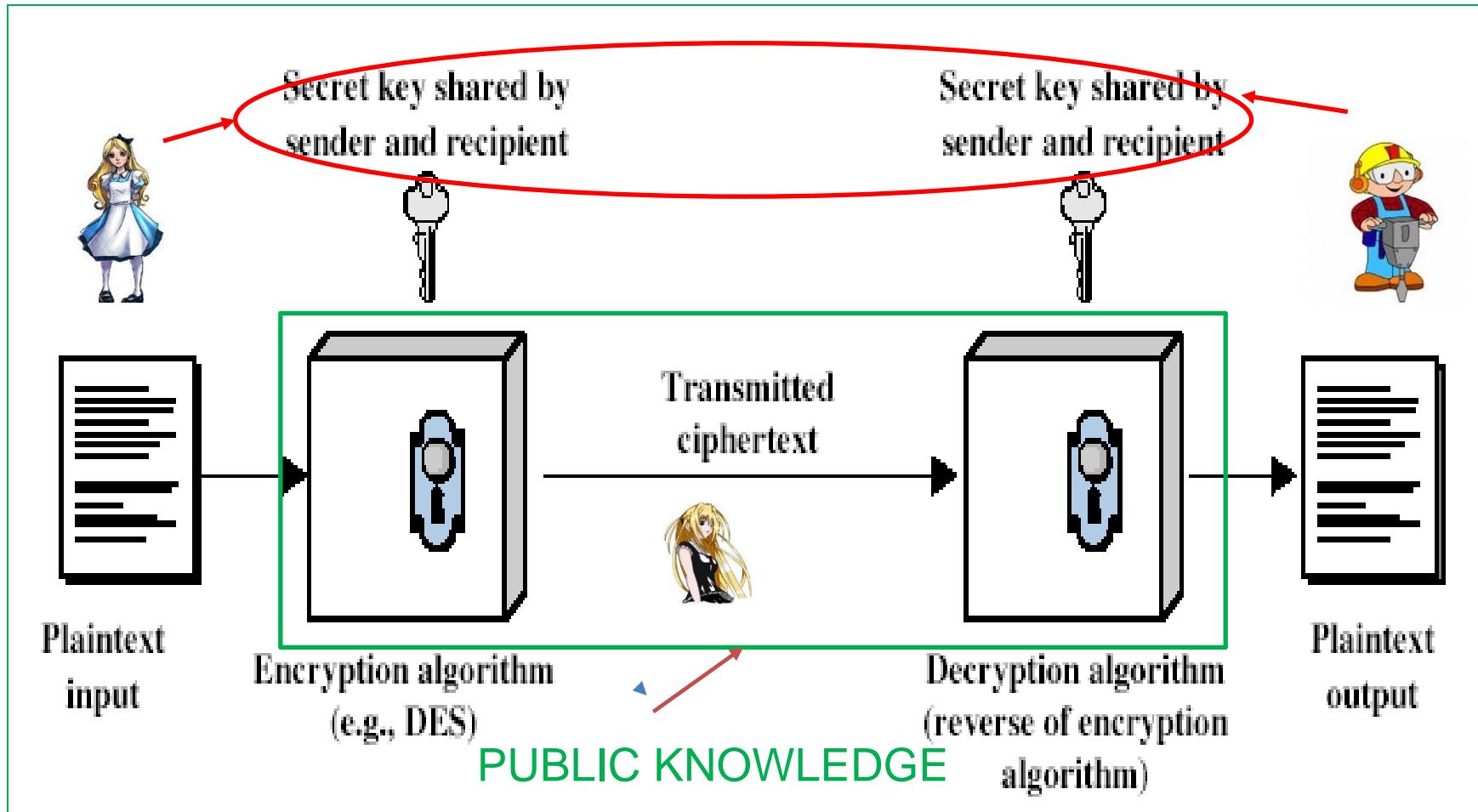
- Conventional encryption since antiquity-known as single key or private key or **symmetric key** systems.
- A same key is used for both encryption and decryption.
- One of the main assumption is that both sender and receiver should have access to the symmetric key used in the encryption.
- Widely used in practice.
- Examples: DES, AES etc.

Terminology

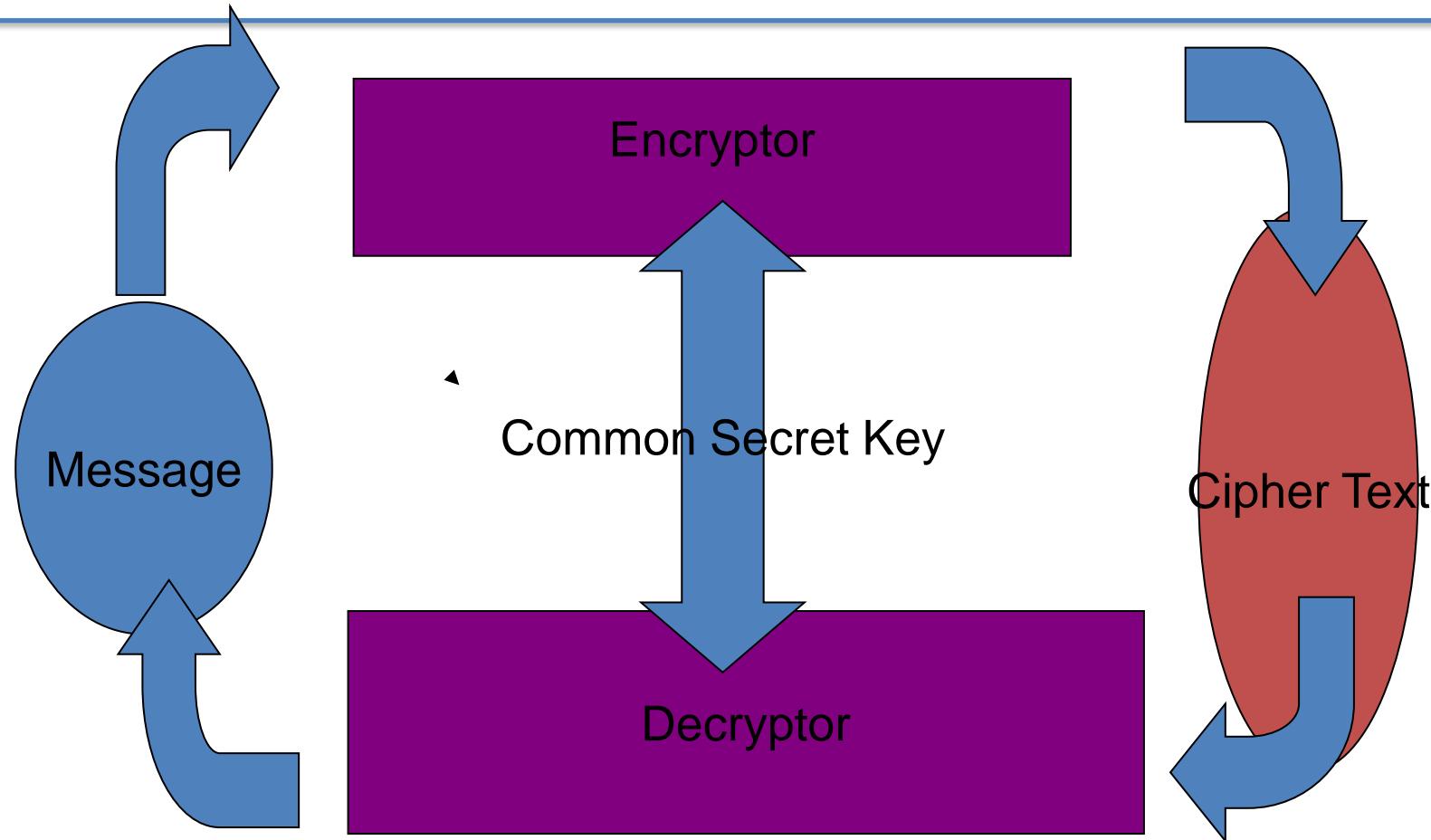
- **Plaintext** - Source message
- **Ciphertext** - Encrypted message
- **Cipher** or Encryption Algorithm- Procedure for transforming plaintext to ciphertext
- **Key** – info or secret used in cipher known only to sender/receiver
- **Encipher (encrypt)** - Converting plaintext to ciphertext
- **Decipher (decrypt)** - Recovering plaintext from ciphertext
- **Cryptography** - Study of encryption principles/methods
- **Cryptanalysis (codebreaking)** - Study of principles/ methods of deciphering Ciphertext *without* knowing key
- **Cryptology** - Field of both cryptography and cryptanalysis

Model for Symmetric Key Cipher

Modified From: Stallings Figure 2.1:



Logical View of Symmetric Key System



Block diagram of a Symmetric Key System: Logical view

Basic Requirements and Kerckhoffs's principle



- If Cipher Algorithms are kept secret from adversaries, will it help achieving security for the sender and receiver?
- Kerckhoffs's principles argues that security through obscurity is not recommended.
- All algorithm details are made public and security should be obtained by using only secrecy of key used in the encryption
- Stallings recommends two essential requirements:
 - A strong encryption algorithm
 - A secret key known only to participants.
- In symmetric key systems security it is a mandatory requirement that keys are to be kept secret between sender and receiver.

1.2 Security

COMP90043
Lecture 2-Part II

Model of Symmetric Key Usage

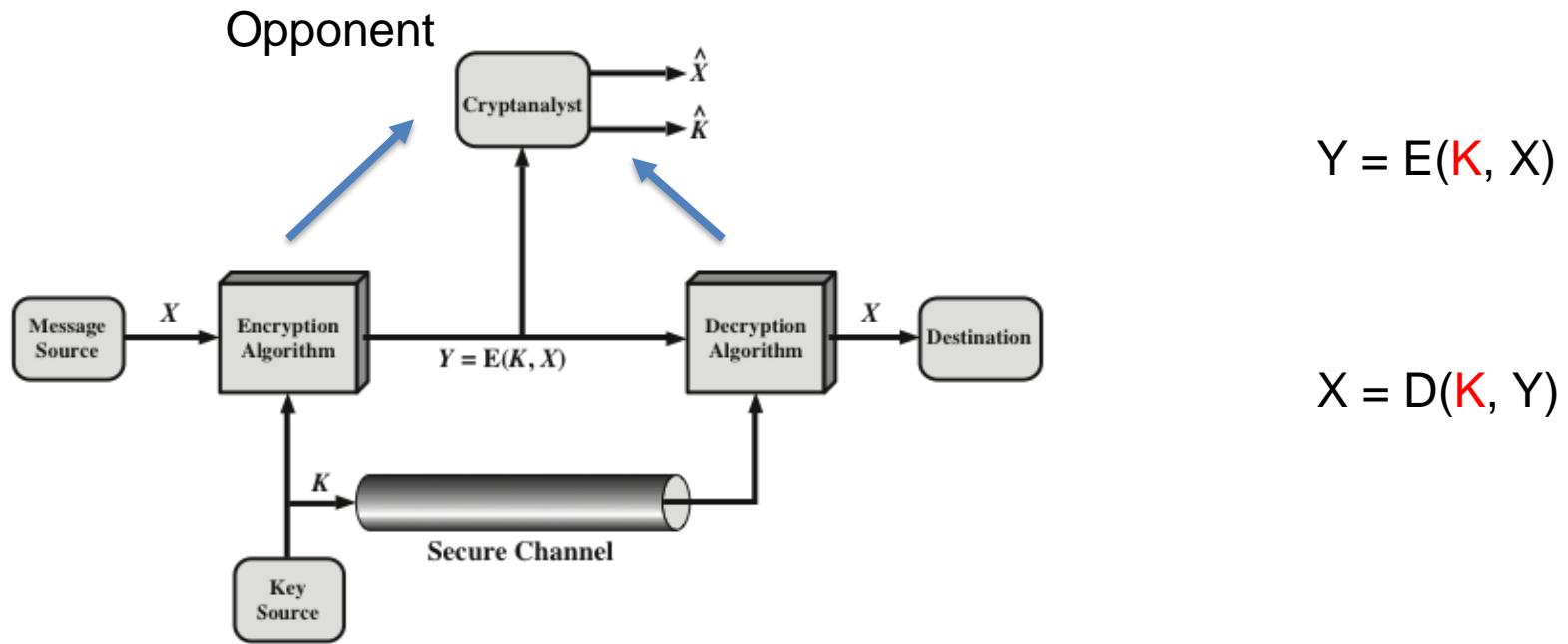


Figure 3.2 Model of Symmetric Cryptosystem

From Stallings Figure 3.2:

Users Perspective:Symmetric key Encryption



Crypto systems have 3 independent Dimensions according to Stallings.

- Algorithm for Ciphers: Transformation details of plaintext to cipher text. They are based on mathematics, heuristics and pragmatic ideas.
- Number of Possible Keys used: Higher the key size better protection.
- Types of Plaintext/Cipher text processing
 - Stream ciphers: plaintexts are streamed to cipher producing stream of ciphertexts element by element.
 - Block Ciphers: plaintext is divided into blocks of data, cipher process one block at a time

Opponent's Perspective: Cryptanalysis

- Main task for him to be able to decrypt ciphertexts without access to keys.
- Usually the objective is to obtain keys by observing plaintext/ciphertext pairs called **Cryptanalysis**. In principle, opponent can get all information about cryptosystem except the key involved.
- Keys should be large enough such that **Brute-force** search is impossible.
- There are types of Cryptanalytic attacks based on capability of opponent's model.
- **Ciphertext only, Known plaintext, Chosen plaintext, Chosen ciphertext, Chosen text:**

Computing View of Security

- The attack models described earlier is based on the model of modern Information and Communication that exists today.
- Adversaries are the entities on communication network who can deploy various services to watch, collect, record and process information that flows at the points that they desire. They can use centralized or distributed architecture.
- Two important definitions are interesting on which much of the cryptologic research of modern times are based.
- **Unconditional Security (Shannon):** The security of the cipher is independent of the computing resource available to the adversaries.
- **Computational Security (Turing):** Adversaries are provided with constrained computing resources and the security of the cipher determined by the size of the computations required to break the cipher.

Implications of Brute-force Attack

- To break a ciphertext $C = E(K, M)$, one could try all possible messages, but that is generally futile as the space is large. And if even we break one ciphertext, one may need to repeat the same steps for every ciphertext. Not a feasible approach.
- Next best thing you can hope is to brute-force on every possible keys.
- You realize immediately that the attack is directly proportional to the size of the key space.
- Of course you assume you have a method to recognize plaintext while trying all possible keys.
- Generally the size of the key space will tell you the complexity of the Brute-force key attack.
- You need at least 128 bit key to protect against this attack in practice based on assumption that adversaries are equipped with classical computing resources.
- We need to increase the key size to protect against Quantum computing attacks (we will deal later)

1.3 Classical Ciphers

COMP90043
Lecture 2-Part II

Classical Ciphers



- Why do we study?
- They are based on simple properties of plaintext alphabets and are known from antiquity.
- Help us to illustrate both encryption and cryptanalysis in a simple language easy to follow.
- The ideas behind methods and analysis of these schemes have parallels in the design and analysis of modern symmetric key schemes.

Types of Classical Ciphers

- Substitution Ciphers
 - Here plaintext symbols are substituted or replaced with other symbols using an unknown key.
 - The substitutions can be performed as sequence of symbols or symbol by symbol.
 - Eg RANDOM NODE- DITNAP TANF
- Transposition Ciphers
 - Here plaintexts are organized as a sequence of plaintext blocks and symbol positions in each block are permuted or transposed using a key. The same permutation is used for every block
 - Eg. RANDOM LETTER -> MORADN RELETT

Caesar Cipher

- Historically attributed to Julius Caesar
- Assumes letter ordering in a language.
- For example, in English
- The alphabet order is: ABCDEFGHIJKLMNOPQRSTUVWXYZ
- Consider plaintext in sequence, each letter is replaced with the letter that stands in a certain secret(key) places further in the alphabet.
- Example: when $k = 3$, can you decrypt this ciphertext:
- PHHW PH DIWHU WKH WRJD SDUWB

Caesar Cipher Mathematically

- $P := \text{Plain Text Space} = \mathbb{Z}_{26}$ Space
- $C := \text{Cipher Text Space} = \mathbb{Z}_{26}$
- $p := \text{plaintext}$ $c := \text{ciphertext}$
- Encryption: $E(k, p) = c = p + k \bmod 26$
- Decryption: $D(k, c) = p = c - k \bmod 26$
- What is the size of the key space?

Affine Cipher

- $P :=$ Plain Text Space $= \mathbb{Z}_{26}$ Space
- $C :=$ Cipher Text Space $= \mathbb{Z}_{26}$

Key space $= (\mathbb{Z}_{26}, \mathbb{Z}_{26})$,

Key $k = (a, b)$, a, b belongs to \mathbb{Z}_{26}

- $p :=$ plaintext $c :=$ ciphertext
- Encryption: $E(k, p) = c = ap + b \text{ mod } 26$
- Can you Determine the decryption function?
- Decryption: $D(k, p) = \text{Inverse}(a)(c - b) \text{ mod } 26$
- What is the size of the key space?

Monalphabetic Cipher

- We considered two simple functions as Caesar and Affine Ciphers before.
- In fact, we can consider a more general key using a general permutation on 26 alphabets.
- Thus, a key is a permutation on the alphabet Z_{26} (plaintext letter maps to a different random ciphertext letter)
- Consider an example: key could be a permutation:
- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- **DKVQFIBJWPESCXHTMYAUOLRGZN**
- Exercise: Complete the Encryption of the following phrase:
- **Message:** newcoronaviruscasescross
- **Ciphertext:** xf.....
- **How many possible keys?**

Transposition Ciphers

- A permutation of plaintext symbols are employed here as opposed to substitution.
- As a result letter are only rearraged for every d positions.
- Divide plain text as sequence of plaintext blocks of certain size say, d .
- Then apply a permutation to every d positions of the plaintext. The permutation is the key.

Rail Fence cipher

- A simpler technique where message is written out diagonally over a depth of certain rows (say d). Then ciphertext is read row by row.
- Example from the textbook:
- eg. write message out as:

m e m a t r h t g p r y
e t e f e t e o a a t
- giving ciphertext

MEMATRHTGPRYETEFETEOAAT
- Such ciphers are easy to break if depth is small.

Row Transposition Ciphers

- A more complex Transposition Cipher is by employing a permutation on blocks of columns, when messages are written row by row.
 - An example from the textbook:
 - Write letters of message out in rows over a specified number of columns
 - Then reorder the columns according to some key before reading off the rows

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: TTNA APTM TSUO AODW COIX KNLY PETZ

- Convention for the key
 - $(1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7)$ Input Order
 - $(3 \ 4 \ 2 \ 1 \ 5 \ 6 \ 7)$ Output Order

1.4 Cryptanalysis of Classical Ciphers

COMP90043
Lecture 2-Part II

Caesar Cipher



- There are only 26 possible keys, In fact, only 25 non-trivial keys.
- You could mount a simple Brute-force attack.
- Try applying shifts on the alphabets in the ciphertext from 1 to 25, when you recognize some meaningful plaintext stop,
- Can you break the ciphertext:
 - GCUA VQ DTGCM
 - I have provided some magma code on the lms try them.

Affine Cipher



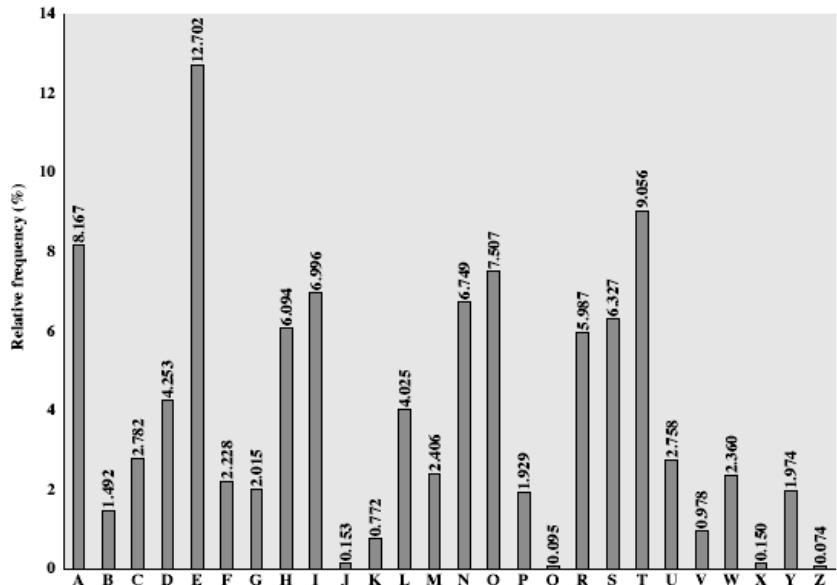
- How many different keys?
- What is the complexity of Brute-force search?
- See a Workshop question next week.

General Monoalphabetic Cipher

- How many different keys?
- 26!
- Brute-force seems impossible. But do you think the cipher is safe?
- Language statistics comes into play.
- In English some letters appear more frequent than others, for example “e” appears more frequently followed by “t” etc, A monoalphabetic cipher always maps a distinct alphabet to another symbol. So the mapping preserves the language statistics. With this one can start guessing which is “e” which is “t” etc.

Language Frequencies

From Stallings Fig. 3.5



The original ideas were developed in 9th Century by Arabian mathematicians.

Refer to the textbook for more discussion

There are now automated methods on Internet

1.5 Complex Ciphers

COMP90043
Lecture 2-Part II

Polyalphabetic Cipher

- How do you make the encryption process more complex so that it is difficult to break?
- Use a set of monoalphabetic ciphers at different time when processing plaintext sequence.
- A key could be used to specify which monoalphabetic cipher to use in a given time context.
- The textbook has some examples, please follow them.

Vigenère Cipher

- This is a simple polyalphabetic substitution cipher.
- Here a set of Caesar ciphers is employed.
- i th plaintext symbol is handled by Caesar cipher with key: $k_{(i \bmod d)}$
- The idea is very simple, a key is a multiple letter word: $K = k_1 k_2 \dots k_d$
- $P = p_1 p_2 \dots p_d p_{d+1} p_{d+2} \dots p_{2d} \dots$
- $C = c_1 c_2 \dots c_d c_{d+1} c_{d+2} \dots c_{2d} \dots$
- Encryption: $E(K, P) = C$, where $c_i = p_i + k_i \bmod 26$
- Decryption: $D(K, C) = P$, where $p_i = c_i - k_i \bmod 26$
- If d is large it offers better security.

Example of Vigenère Cipher

key: deceptivedeceptivedeceptive

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Product Ciphers

- Substitution and Transposition Ciphers are not secure as they are vulnerable to cryptanalysis based on plaintext language characteristics.
- We can think of more general product cipher by applying several substitution and transposition ciphers in succession.
- These ideas were used in German cipher during world war time-see section on Rotar Machines in the textbook.
- This is a link to modern ciphers where more complex substitution and transposition ideas are used.

Week 2



Lecture 1

Part -1 Extended GCD Algorithm and Related Computations

Part 2 - Symmetric key Cryptography

Lecture 2

Properties of Numbers-II

Workshop 2: Workshops start from this week.

Quiz 2

Week 2

Lecture 1 Extended GCD Algorithm Udaya Parampalli

School of Computing and Information Systems
University of Melbourne



Lecture 1

- Part -1 Extended GCD Algorithm and Related Computations
- Part -2 Symmetric key Cryptography

Lecture 2

Properties of Numbers

Workshop 2: Workshops start from this week

Quizz 2

- [**1.1** Extended GCD Algorithm](#)
- [**1.2** Inverse Mod n](#)
- [**1.3** Extended GCD Algorithm: Theorem Proving Version](#)

1.1 Extended GCD Algorithm: A direct version

- Algorithm
- An example.

Extended GCD algorithm

Let us look at the gcd computation again with general numbers a and b with $a > b > 0$. Let $a_0 = a$, $a_1 = b$ and $q_1 = \lfloor a_0/a_1 \rfloor$.

		$gcd(a_0, a_1)$	
a_0	$=$	$q_1 \times a_1 + a_2$	$gcd(a_1, a_2)$
a_1	$=$	$q_2 \times a_2 + a_3$	$q_1 = \lfloor a_0/a_1 \rfloor$
a_2	$=$	$q_3 \times a_3 + a_4$	$gcd(a_2, a_3)$
	\vdots		$q_2 = \lfloor a_1/a_2 \rfloor$
a_{t-2}	$=$	$q_{t-1} \times a_{t-1} + a_t$	$gcd(a_{t-1}, a_t)$
a_{t-1}	$=$	$q_t \times a_t + 0$	$q_{t-1} = \lfloor a_{t-2}/a_{t-1} \rfloor$
			$gcd(a_t, 0)$
			$q_t = \lfloor a_{t-1}/a_t \rfloor$

Table: Computation of $gcd(a, b)$

By using the fact on gcd before, we have

$$\text{gcd}(a, b) = \text{gcd}(a_0, a_1) = \text{gcd}(a_1, a_2) = \cdots = \text{gcd}(a_{t-1}, a_t) = \text{gcd}(a_t, 0)$$

Solving for a_t in the above equations starting from last-but-one to the first, we can express a_t as a linear combination of a_0 and a_1 .

$$\text{gcd}(a, b) = a_t = x \cdot a + y \cdot b.$$

The following example illustrates the above point. A theorem proving version of the algorithm is given at the end of this set of slides.

Extended Euclid's algorithm: Example 1

Consider $\text{gcd}(33, 21)$:

$$\begin{array}{rcl} 33 & = & 1 \times 21 + 12 & \text{gcd}(21, 12) & (\text{A}) \\ 21 & = & 1 \times 12 + 9 & \text{gcd}(12, 9) & (\text{B}) \\ 12 & = & 1 \times 9 + 3 & \text{gcd}(9, 3) & (\text{C}) \\ 9 & = & 3 \times 3 + 0 & \text{gcd}(3, 0) & \end{array}$$

Table: Determine $\text{gcd}(33, 21)$

$$\begin{aligned} 3 &= 12 - 1 \times 9 && \text{From (C)} \\ 3 &= 12 - 1 \times (21 - 1 \times 12) && \text{From (B)} \\ 3 &= 2 \times 12 - 1 \times 21 \\ 3 &= 2 \times (33 - 1 \times 21) - 1 \times 21 && \text{From (A)} \\ 3 &= 2 \times 33 + (-3) \times 21 && \text{Simplification} \end{aligned}$$

1.2 Inverse Mod n

- Definition
- Inverse mod n Computation
- Computation with Magma

Modular Arithmetic

Let a and b be integers and let n be a positive integer.

We say “ a ” is congruent to “ b ”, modulo n and write

$$a \equiv b \pmod{n},$$

if a and b differ by a multiple of n ; i.e ; if n is a factor of $|b - a|$.

Every integer is congruent mod n to exactly one of the integers in the set

$$\mathbb{Z}_n = \{0, 1, 2, \dots, n - 1\}.$$

We can define the following operations:

$$x \oplus_n y = (x + y) \pmod{n}.$$

$$x \otimes_n y = (xy) \pmod{n}$$

When the context is clear we use the above special addition and multiplication symbols interchangeably with their counterpart regular symbols.

Modular Multiplicative Inverse

Definition

Let $x \in Z_n$, if there is an integer y such that

$$x \otimes_n y = 1,$$

then we say y is the multiplicative inverse of x . It is denoted by $y = x^{-1}$ usually.

Example: let $n = 5$, 2 is inverse of 3 in Z_5 . Or in other words 2 is inverse of 3 modulo 5.

Fact

For any integers a and b , there exist integers x and y such that

$$\gcd[a, b] := ax + by.$$

You can determine x and y by modifying Euclid's algorithm for $\gcd(a, b)$. Thus we can say that we can find inverse of a modulo b provided $\gcd(a, b) = 1$.

Computing inverse mod n

If $\gcd(a, n)$ is 1 then we can use extended Euclid's algorithm on a and n and get two integers x and y such that

$$xa + yn = 1.$$

Taking mod n on both sides of the above equation we get

$$xa = 1 \bmod n.$$

Clearly x is the inverse of $a \bmod n$.

Computing inverse mod n

If $\gcd(n, a)$ is 1 then we can use extended Euclid's algorithm on a and n and get two integers x and y such that

$$xn + ya = 1.$$

Taking mod n on both sides of the above equation we get

$$ya = 1 \bmod n.$$

Clearly y is the inverse of a mod n . Note that the inverse is unique. Also it is clear that if $\gcd(n, a) > 1$, then inverse does not exist. **Note:** *The output of the extended gcd algorithm which is the inverse of a given integer depends on the order of the input arguments.*

Extended Euclid's algorithm: Example 2

Consider $\text{gcd}(13, 25)$:

$$\begin{aligned} 25 &= 1 \times 13 + 12 \quad \text{gcd}(13, 12) \quad (A) \\ 13 &= 1 \times 12 + 1 \quad \text{gcd}(12, 1) \quad (B) \\ 12 &= 12 \times 1 + 0 \quad \text{gcd}(1, 0) \end{aligned}$$

Table: Determine $\text{gcd}(13, 25)$

$$\begin{aligned} 1 &= 13 - 1 \times 12 && \text{From (B)} \\ 1 &= 13 - 1 \times (25 - 1 \times 13) && \text{From (A)} \\ 1 &= 2 \times 13 - 1 \times 25 \\ 1 &= 2 \times 13 + (-1) \times 25 && \text{Simplification} \end{aligned}$$

It is easy to see now, 2 is inverse of 13 mod 25.

Magma is a symbolic mathematical software package which can help you to do computations in algebra, number theory and geometry.

<http://magma.maths.usyd.edu.au/magma/>

An online calculator is available here:

<http://magma.maths.usyd.edu.au/calc/>

ExtendedGreatestCommonDivisor(m, n) : RngIntElt, RngIntElt
→ RngIntElt,
RngIntElt, RngIntElt

Xgcd(m, n) : RngIntElt, RngIntElt → RngIntElt, RngIntElt, RngIntElt
XGCD(m, n) : RngIntElt, RngIntElt → RngIntElt, RngIntElt, RngIntElt

The extended GCD of m and n ; returns integers g , x and y such that g is the greatest common divisor of the integers m and n , and $g = x.m + y.n$. If m and n are both zero, g is zero; otherwise g is always positive. If m and n are both non-zero, the multipliers x and y are unique.

1.3 Extended GCD Algorithm: Theorem Proving Version

Theorem

Given two positive integers a and b with $a > b$, let $a_0 = a$, $a_1 = b$ and $q_1 = \lfloor a_0/a_1 \rfloor$. Perform the following matrix equations for $r = 1, 2, \dots, n$:

$$q_r = \lfloor \frac{a_{r-1}}{a_r} \rfloor,$$

$$\begin{bmatrix} a_r \\ a_{r+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -q_r \end{bmatrix} \begin{bmatrix} a_{r-1} \\ a_r \end{bmatrix}$$

until $a_{n+1} = 0$, where n is an integer. Then a_n is the GCD of a and b .

Proof: You can convince that the termination of the algorithm is well defined since $a_{r+1} < a_r$. So eventually, for some n , $a_{n+1} = 0$.

- hence we can write the recursion as the following matrix equation:

$$\begin{bmatrix} a_n \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -q_n \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -q_{n-1} \end{bmatrix} \cdots \begin{bmatrix} 0 & 1 \\ 1 & -q_1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}.$$

Hence, we have

$$\begin{bmatrix} a_n \\ a_{n+1} = 0 \end{bmatrix} = \left\{ \prod_{l=n}^1 \begin{bmatrix} 0 & 1 \\ 1 & -q_l \end{bmatrix} \right\} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix},$$

Where \prod , is the symbol for multiplication. Then, consider only the first row of the above matrix equation, you get $a_n = A_{1,1} a_0 + A_{1,2} a_1$, where A is the matrix in the RHS of the above equation. Thus any divisor of both $a_0 = a$ and $a_1 = b$ divides a_n . Hence, greatest common divisor $\gcd(a, b)$ also divides a_n .

- Further observe that,

$$\begin{bmatrix} 0 & 1 \\ 1 & -q_r \end{bmatrix}^{-1} = \begin{bmatrix} q_r & 1 \\ 1 & 0 \end{bmatrix}$$

and hence by inverting the matrix equation recursively, we get

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \left\{ \prod_{l=1}^n \begin{bmatrix} q_l & 1 \\ 1 & 0 \end{bmatrix} \right\} \begin{bmatrix} a_n \\ 0 \end{bmatrix}.$$

So a_n must divide both $a_0 = a$ and $a_1 = b$ and hence divides $\gcd(a, b)$.

Thus $a_n = \gcd(a, b)$.

Some implications of the theorem. Let

$$A^r = \left\{ \prod_{l=r}^1 \begin{bmatrix} 0 & 1 \\ 1 & -q_l \end{bmatrix} \right\} = \begin{bmatrix} 0 & 1 \\ 1 & -q_r \end{bmatrix} A^{r-1}.$$

Theorem

For any integers a and b there exist integers X and Y such that $\gcd(a, b) = X a + Y b$.

Proof

From Theorem 1, we have

$$\begin{bmatrix} a_n \\ 0 \end{bmatrix} = A^n \begin{bmatrix} a \\ b \end{bmatrix}.$$

Hence $\gcd(a, b) := a_n = A_{11}^n a + A_{12}^n b$.

Similarly prove the following theorem.

Theorem

The matrix elements A_{21}^n and A_{22}^n satisfy

$$a = (-1)^n A_{22}^n \ gcd(a, b)$$

$$b = (-1)^n A_{21}^n \ gcd(a, b).$$

Lecture 1

- Part -1 Extended GCD Algorithm and Related Computations
- Part -2 Symmetric key Cryptography

Lecture 2

Properties of Numbers

Workshop 2: Workshops start from this week

Quizz 2

Week 2

Lecture 2 Properties of Numbers II Udaya Parampalli

School of Computing and Information Systems
University of Melbourne



Lecture 1

- Part -1 Extended GCD Algorithm and Related Computations
- Part -2 Symmetric key Cryptography

Lecture 2

Properties of Numbers

Workshop 2: Workshops start from this week

Quizz 2

- 2.1 More on Inverse Modulo n
- 2.2 Euler's Phi Function
- 2.3 How can you use Euler's Phi to compute inverses?

2.1 More on Inverse Modulo n

Modular Arithmetic

Let a and b be integers and let n be a positive integer.

We say “ a ” is congruent to “ b ”, modulo n and write

$$a \equiv b \pmod{n},$$

if a and b differ by a multiple of n ; i.e ; if n is a factor of $|b - a|$.

Every integer is congruent mod n to exactly one of the integers in the set

$$\mathbb{Z}_n = \{0, 1, 2, \dots, n - 1\}.$$

We can define the following operations:

$$x \oplus_n y = (x + y) \pmod{n}.$$

$$x \otimes_n y = (xy) \pmod{n}$$

When the context is clear we use the above special addition and multiplication symbols interchangeably with their counterpart regular symbols.

Definition

Let $x \in Z_n$, if there is an integer y such that

$$x \otimes_n y = 1,$$

then we say y is the multiplicative inverse of x . It is denoted by $y = x^{-1}$ usually.

Example: let $n = 5$, 2 is inverse of 3 in Z_5 . Or in other words 2 is inverse of 3 modulo 5.

Fact

For any integers a and b , there exist integers x and y such that

$$\gcd[a, b] := ax + by.$$

You can determine x and y by modifying Euclid's algorithm for $\gcd(a, b)$. Thus we can say that we can find inverse of a modulo b provided $\gcd(a, b) = 1$.

2.2 Euler's Phi Function

Euler Phi function

Definition

Two numbers a and b are relatively prime if $\gcd(a, b)$ is 1.

Definition

Euler phi function(or Euler totient function): For $n \geq 1$, let $\phi(n)$ denote the number of integers less than n but are relatively prime to n .

Definition

Reduced set of residues mod n : For $n \geq 1$, the reduced set of residues, $R(n)$ is defined as set of residues modulo n which are relatively prime to n .

Example: $\phi(6) = 2$: Observe, $\gcd(1, 6) = 1$, $\gcd(2, 6) = 2$, $\gcd(3, 6) = 3$, $\gcd(4, 6) = 2$, $\gcd(5, 6) = 1$. Then $R(6) = \{1, 5\}$. Hence $\phi(6) = 2$.

Some Relations

Fact

$$\phi(p) = p - 1, \text{ for any prime } p.$$

This is easy and follows from definition of a prime number.

Fact

$$\phi(p^a) = p^a - p^{a-1} = p^{a-1}(p - 1),$$

for any prime p and any integer $a \geq 1$.

Consider numbers from 0 to $p^a - 1$, then only numbers which have some common divisor with p^a are those numbers which are multiple of p . There are exactly p^{a-1} such numbers including the number 0. All other numbers are relatively prime to p^a . Hence, $\phi(p^a) = p^a - p^{a-1} = p^{a-1}(p - 1)$ as needed.

Example: $\phi(8) = 4$, the numbers which are multiple of 2 are $\{2, 4, 6, 8\}$ and hence the relatively prime numbers are all odd numbers up to 7, i.e $R(8) = \{1, 3, 5, 7\}$.

Some Relations, cont.

Fact

$$\phi(pq) = (p - 1)(q - 1), \text{ for any pair of primes } p \text{ and } q.$$

Proving this result is trickier than before but still not difficult to visualize. Again consider numbers from 1 to pq . Like before, we can exclude all those numbers which are multiple of p and q to form $R(pq)$. Then can we say the following?

$$|R(pq)| = pq - ((pq)/q) - ((pq)/p) = (pq - p - q)$$

In the above counting, we have excluded multiple of pq twice, once while excluding the multiples of p and again while excluding the multiples of q . So we need to make the following change

$$\phi(pq) = |R(pq)| = pq - p - q + 1 = (p - 1)(q - 1).$$

Example: $\phi(15) = 8$, the relatively prime numbers are
1, 2, 4, 7, 8, 11, 13, 14.

Euler Phi function is multiplicative

Fact

If a and b are relatively prime numbers ($\gcd(a, b) = 1$), then,

$$\phi(ab) = \phi(a)\phi(b).$$

This is not directly obvious with whatever we have studied so far. But take this as a fact. You can prove this using some elementary number theory results.

Using the above fact, we can derive a general result about Euler's ϕ function. We know that any number has a unique factorization:

$$n = \prod_{i=1}^{\tau} p_i^{a_i} = p_1^{a_1} p_2^{a_2} \cdots p_{\tau}^{a_{\tau}} ,$$

where τ is a positive number, p_i are primes and $a_i \geq 1$ and \prod is the symbol for product. Find $\phi(n)$ for this case. Example: What is $\phi(200) = \phi(2^3 5^2)$?

Euler Phi function for general n

Using the multiplicative property of ϕ , we can simplify $\phi(n)$ as follows:

$$\phi(n) = \phi(\prod_{i=1}^{\tau} p_i^{a_i}) = \phi(p_1^{a_1} p_2^{a_2} \cdots p_{\tau}^{a_{\tau}}),$$

From the fact on $\phi(p^a)$ given before we can write,

$$\phi(n) = \prod_{i=1}^{\tau} p_i^{a_i-1} (p_i - 1)).$$

Example: What is $\phi(200) = \phi(2^3 5^2) = \phi(2^3)\phi(5^2) = 80$.

2.3 How can you use Euler's Phi to compute inverses?

Inverse Mod n again

We have seen how Extended GCD Algorithm to compute $\text{inverse}(a) / \text{mod } n$ before.

We will prove the following result later, but let us state it now. let \mathbf{Z}_n^* be set of numbers from 1 to $n - 1$ but are relatively prime.

Theorem

If $a \in \mathbf{Z}_n^*$, then $a^{\phi(n)} = 1 \pmod{n}$.

Now, how can you use the above theorem for computing inverse of $a \pmod{n}$?

inverse(a) mod n

Given a a number less than n but relatively prime to n

```
Function(a, n)
inva :=  $a^{\phi(n)-1} \pmod{n}$ .
Return(inva);
end function;
```

Lecture 1

- Part -1 Extended GCD Algorithm and Related Computations
- Part -2 Symmetric key Cryptography

Lecture 2

Properties of Numbers

Workshop 2: Workshops start from this week

Quizz 2

Week 3

Lecture 2 Properties of Numbers III Udaya Parampalli

School of Computing and Information Systems
University of Melbourne



Lecture 1 Modern Symmetric key Ciphers

Lecture 2 Properties of Numbers III

Workshop 3: Workshops based on Lectures in Week 2

Quizz 3

- [**2.1 Euler's and Related Theorems**](#)
- [**2.2 Groups, Rings and Fields**](#)
- [**2.3 Functions and Chinese Remainder Theorem**](#)

- Numbers, Divisibility, Mod Operation, GCD, Extended GCD
- Inverse Mod n
- Properties Euler's Phi (ϕ)Function
- $\phi(p) = p - 1$, for any prime p .
- $\phi(p^a) = p^{a-1}(p - 1)$, for any prime p and any integer $a \geq 1$.
- $\phi(pq) = (p - 1)(q - 1)$, for any two primes p and q .
- In fact, $\phi(mn) = \phi(m)\phi(n)$, for any two numbers which are relatively prime.

let \mathbf{Z}_n^* be set of numbers from 1 to $n - 1$ but are relatively prime.

Theorem

If $a \in \mathbf{Z}_n^*$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

Using Extended GCD Algorithm

Function(a, n)

g,x,y:=XGCD(a,n);

If g eq 1 then Return(x)

else Return(“The Inverse Does not Exist”), end if;

end function;

Using Euler's Phi Function Result

Function(a, n)

inva := $a^{\phi(n)-1} \pmod{n}$.

Return(inva);

end function;

The later function works only if a is relatively prime to n .

2.1 Euler's and Related Theorems

Euler's Theorem

Definition

Remainders mod n: For $n \geq 1$, the set of remainders obtained by dividing integers by n , precisely these are elements of $\mathbf{Z}_n = \{0, 1, \dots, n - 1\}$.

However, not all elements of \mathbf{Z}_n can be inverted. We define further the set of invertible numbers in \mathbf{Z}_n .

Definition

Reduced set of residues mod n: For $n \geq 1$, the reduced set of residues, $R(n)$ is defined as set of residues modulo n which are relatively prime to n .

Sometimes, $R(n)$ is also represented as $\mathbf{Z}^*(n)$. In fact

$\phi(n) = \#R(n)$, the cardinality(size) of the set $R(n)$.

Example: $\phi(15) = 8$, because $\phi(15) = \phi(5 \times 3) = (4 \times 2) = 8$.

$\phi(37) = 36$, as 37 is a prime number.

Next we consider Euler's theorem.

Theorem

If $a \in \mathbb{Z}_n^*$, then $a^{\phi(n)} = 1 \pmod{n}$.

Proof: Let $R(n) = \{r_1, r_2, \dots, r_{\phi(n)}\}$, be reduced set of residues modulo n . Now consider the set $aR(n) = \{a r_1, a r_2, \dots, a r_{\phi(n)}\}$. Since a is relatively prime to n , the set $aR(n)$ is identically equal to $R(n)$. Note that the process of multiplying a only rearranges the residues in $R(n)$. Hence we can multiply all the elements in $R(n)$ and equate with the multiplication of all the elements of $aR(n)$. Hence we can write:

$$r_1 \times r_2 \cdots \times r_{\phi(n)} = (ar_1) \times (ar_2) \cdots \times (ar_{\phi(n)}).$$

Note that r_i s are relatively prime to n and hence we can cancel r_i in the above equation by multiplying r_i^{-1} , $i = 1 \cdots \phi(n)$, to both the side of the equation. Then the above equation simplifies to

$$1 = a^{\phi(n)}. \text{ Hence the result.}$$

Euler's Theorem example when $n = pq$

When $n = pq$, p and q are primes, then $\phi(n) = (p - 1)(q - 1)$.

Theorem

If $a \in \mathbf{Z}_{pq}^*$, then $a^{(p-1)(q-1)} = 1 \pmod{pq}$.

The above result will be used in next week lectures.

Example: $n = 35$, $\phi(35) = 24$, because

$$\phi(35) = (\phi(7) \times \phi(5)) = (6 \times 4) = 24.$$

2 is relatively prime to 35

$$2^{24} \pmod{35} = 1$$

Fermat's Theorem

Theorem

Let p be a prime number, then if $\gcd(a, p) = 1$, then

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

This is the particular case of Euler's Theorem when n is prime.

Fermat's Little Theorem

Theorem

Let p be a prime number,

$$a^p = a \pmod{p}, \text{ for any integer } a.$$

When a is relatively prime, the theorem follows from the Fermat's theorem. When a is multiple of p , the result is trivially true.

- When p is a prime number, we learn that all nonzero numbers less than p are relatively prime and hence they are closed modulo p .
- In otherwords, all nonzero elements are invertible in \mathbf{Z}_p .
- They are closed under addition modulo p .
- Hence \mathbf{Z}_p is closed under addition and multipliaction mod p .
- In fact, \mathbf{Z}_p is a finite field, a structure extensively used in Cryptography.

2.2 Groups, Rings and Fields

Recap of Group, Ring, and Field

Let us visit a few concepts that we have learnt already. A *Group* is a set G together with a binary operation \cdot on G such that the following three properties hold:

- \cdot is *associative*; that is, for any $a, b, c \in G$

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

- There is an *identity* element e in G such that for all $a \in G$,

$$a \cdot e = e \cdot a = a$$

- For each $a \in G$, there exists an *inverse* element $a^{(-1)} \in G$ such that

$$a \cdot a^{-1} = a^{-1} \cdot a = e$$

- If the group also satisfies

For all $a, b \in G$,

$$a \cdot b = b \cdot a$$

then the group is called *abelian* (or *commutative*).

A *Ring* $(R, +, \cdot)$ is a set R , together with two binary operations, denoted by $+$ and \cdot , such that:

- R is an abelian group with respect to $+$.
- \cdot is associative; that is, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all $a, b, c \in R$.
- The *distributive laws* hold; that is, for all $a, b, c \in R$ we have $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(b + c) \cdot a = b \cdot a + c \cdot a$

We note that the set $\mathbf{Z}_p = \{0, 1, \dots, p - 1\}$, where p is a prime number, satisfies axioms of a field.

- The set is closed under addition.
- Since p is prime number, any nonzero element in \mathbf{Z}_p has an inverse (Use Extended Euclidean algorithm).
- you can verify that additions and multiplications are distributive.

In \mathbf{Z}_p , unlike in Integers, p times any element in the field is zero in the field. This leads to a concept called “characteristic” of a field.

We also denote \mathbf{Z}_p^* as a set of non-zero elements of \mathbf{Z}_p .

Definition

Let F be a field with the multiplicative identity 1 and the additive identity 0. The characteristic of F , sometimes written as $\text{char}(F)$, is the smallest integer $n \geq 0$ such that addition of the 1 with itself n times results in 0. i.e $n(1) = 0$.

Note that for real and complex fields you cannot find a positive integer n satisfying the above criteria. Hence, the characteristic of real and complex fields is 0.

In contrast for residue class rings \mathbf{Z}_n , the characteristic is n .

When n is prime, \mathbf{Z}_p is a field and accordingly the characteristic of \mathbf{Z}_p is p . One of the consequences of the above property is that $p = 0$ in the field for any α in the field.

\mathbf{Z}_p is the main source of prime fields. Another class of finite fields are those whose size is a power of prime, we will consider this class later.

2.3 Functions and Chinese Remainder Theorem

Definition: A function is defined by a triplet $\langle X, Y, f \rangle$, where
 X : a set called domain; Y : a set called range or codomain and
 f : a rule which assigns to each element in X precisely one element
in Y .

It is denoted by $f : X \rightarrow Y$

Example: Let $X = Y = \mathbf{Z}_5$, Then $f : X \rightarrow Y$ given by
 $f(x) = 2 * x$ is a function.

Image : If $x \in X$, the image of x in Y is an element $y \in Y$ such that $y = f(x)$.

Pre-image : If $y \in Y$, then a Pre-image of y in X is an element $x \in X$ such that $f(x) = y$.

Image of a function f ($Im(f)$): A set of all elements in Y which have at least one Pre-image.

$$Im(f) = \bigcup_{x \in X} \{f(x)\} \quad (1)$$

One-to-one (injective) Function

A function is one-to-one (injective) if each element in the codomain Y is the image of **at most** one element in the domain X . In other words, each element in x in X is related to different y in X , never two different elements in X map to a same element in Y . We can say that $|X| \leq |Y|$. An alternate definition would be, a $f : X \rightarrow Y$ is one-to-one (injective), provided

$$f(x_1) = f(x_2) \text{ implies } x_1 = x_2.$$

Examples: Let $X = Y = \mathbf{Z}_4$, Then $f : X \rightarrow Y$ given by $f(x) = 3 * x$ is a one-to-one function. However $f(x) = x^2$ is not a one-to-one function.

Onto (surjective) Function

A function is Onto (surjective) if each element in the codomain Y is the image of **at least** one element in the domain X .

A function $f : X \rightarrow Y$ is onto if $Im(f) = Y$

We can say that, if f is onto then $|Y| \leq |X|$.

Example: Let $X = Y = \mathbf{Z}_5$, Then $f: X \rightarrow Im(f)$ given by

$f(x) = x^2$ is a onto function.

Bijection: A function which is both one-to-one and onto.

In this case, we have $|X| \leq |Y|$ and $|Y| \leq |X|$. This implies $|X| = |Y|$.

If $f: X \rightarrow Y$ is one-to-one then $f: X \rightarrow Im(f)$ is a bijection.

If $f: X \rightarrow Y$ is onto and X and Y are finite sets of the same size then f is a bijection.

Bijection

Let m and n are relatively prime number, $X = \mathbf{Z}_{mn}$, $Y = \mathbf{Z}_m \times \mathbf{Z}_n$.
Then the mapping

$$f : X \rightarrow Y, f(x) = ((x \bmod m), x \bmod n),$$

is a bijection.

Example: $X := \mathbf{Z}_6$, $Y = \mathbf{Z}_2 \times \mathbf{Z}_3$. The function f given below is a bijection:

$X = \mathbf{Z}_6$	\rightarrow	$\mathbf{Z}_2 \times \mathbf{Z}_3$
0	\rightarrow	(0, 0)
1	\rightarrow	(1, 1)
2	\rightarrow	(0, 2)
3	\rightarrow	(1, 0)
4	\rightarrow	(0, 1)
5	\rightarrow	(1, 2)

Table: $f : \mathbf{Z}_6 \rightarrow \mathbf{Z}_2 \times \mathbf{Z}_3$

Chinese Remainder Theorem (CRT)

Let n_1, n_2 be pair-wise relatively prime integers, the system of simultaneous congruences

$$x \equiv a_1 \pmod{n_1},$$

$$x \equiv a_2 \pmod{n_2},$$

has a unique solution modulo $n = n_1 n_2$.

Note that the mapping $f : \mathbf{Z}_{n_1 \ n_2} \rightarrow \mathbf{Z}_{n_1} \times \mathbf{Z}_{n_2}$ given by $f(x) \rightarrow x \text{ mod } n_1, x \text{ mod } n_2$ is a bijection.

The proof has two points. First show that the function is one-to-one. If there exists two elements x and y such that

$$x \text{ mod } n_1 = y \text{ mod } n_1,$$

and

$$x \text{ mod } n_2 = y \text{ mod } n_2,$$

then $x - y$ is divisible by both n_1 and n_2 . Since n_1 and n_2 are relatively prime, $x - y$ is divisible by $n_1 \ n_2 = n$. Hence x and y are identical equal modulo n . This proves that the function is one-to-one. In the next slide, we give an explicit construction for the inverse function which proves that the map is onto. Hence the f is bijection.

In fact, Chinese Remainder theorem gives a construction method to obtain the inverse function. Let

$$N_1 = n/n_1 = n_2, N_2 = n/n_2 = n_1.$$

Choose

$$M_1 = (N_1)^{-1} \pmod{n_1}$$

and

$$M_2 = (N_2)^{-1} \pmod{n_2}$$

Then the solution to the simultaneous congruences is given by

$$x = a_1 (N_1 M_1) + a_2 (N_2 M_2) \pmod{n}.$$

You can immediately verify that x determined as above satisfies the congruences (This is because $N_1 \pmod{n_2} = 0$ and $N_2 \pmod{n_1} = 0$)

Chinese Remainder Theorem (CRT)

If n_1, n_2, \dots, n_k are pair-wise relatively prime integers, k being a positive integer, the system of simultaneous congruences

$$x \equiv a_1 \pmod{n_1},$$

$$x \equiv a_2 \pmod{n_2},$$

$$x \equiv a_3 \pmod{n_3},$$

...

$$x \equiv a_k \pmod{n_k},$$

has a unique solution modulo $n = n_1 n_2 \dots n_k$.

Let

$$N_i = n/n_i$$

for $i = 1, 2, \dots, k$.

Choose

$$M_i = (N_i)^{-1} \pmod{n_i},$$

for $i = 1, 2, \dots, k$.

Then the solution is given by

$$x = \sum_{i=1}^k a_i N_i M_i \pmod{n}.$$

Lecture 1 Modern Symmetric key Ciphers

Lecture 2 Properties of Numbers III

Workshop 3: Workshops based on Lectures in Week 2

Quizz 3

COMP90043: Cryptography and security:

Week 3 Activity

Activity: Working of Fiestel's algorithm for encryption and decryption.

Both encryption and decryption iteratively runs sixteen rounds of the same inner algorithm. The only difference is that the decryption rounds use a reversed key order. Verify that the decryption is the inverse operation of the encryption.

Plaintext: **LE₀** || **RE₀**

Output of the 16th round (Encryption): **LE₁₆** || **RE₁₆**

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

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

$$LD_1 = RD_0 = \underline{\hspace{2cm}}$$

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

$$= RE_{16} \oplus F(\underline{\hspace{2cm}})$$

$$= \underline{\hspace{2cm}} \oplus F(\underline{\hspace{2cm}}) \oplus F(\underline{\hspace{2cm}})$$

$$LD_1 = \text{ and } RD_1 =$$

Output of 1st round of decryption :

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

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

$$RE_{i-1} = \underline{\hspace{2cm}}$$

$$LE_{i-1} = \underline{\hspace{2cm}}$$

Output of the 16th round (Decryption): **RE₀** || **LE₀**

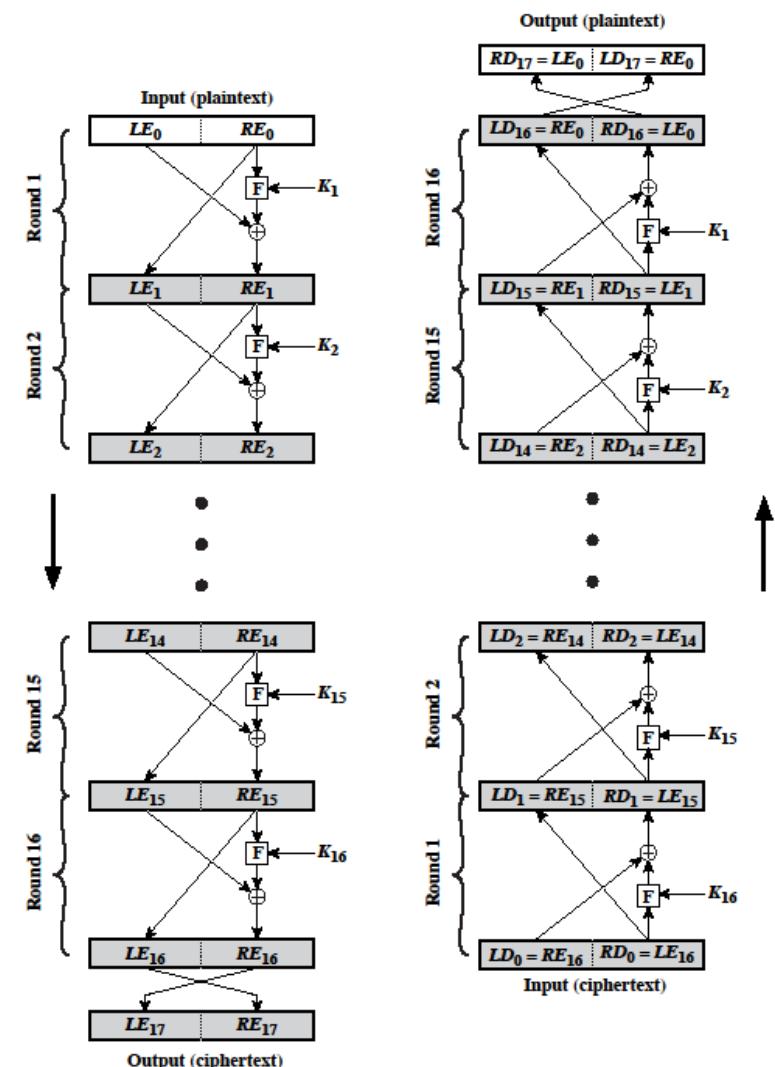


Figure 3.3 Feistel Encryption and Decryption (16 rounds)

Week 4

Lecture 2 Properties of Numbers IV Proof of RSA Encryption Udaya Parampalli

School of Computing and Information Systems
University of Melbourne



Lecture 1

Public Key Cryptography: Diffie-Hellman Protocol and RSA

Lecture 2

Properties of Numbers IV

Workshop 3: Workshops based on Lectures in Week 3

Quizz 4

2.1 Proof of RSA Encryption

- Numbers, Divisibility, Mod Operation, GCD, Extended GCD
- Inverse Mod n
- Properties Euler's Phi (ϕ)Function
- $\phi(p) = p - 1$, for any prime p .
- $\phi(p^a) = p^{a-1}(p - 1)$, for any prime p and any integer $a \geq 1$.
- $\phi(pq) = (p - 1)(q - 1)$, for any two primes p and q .
- In fact, $\phi(mn) = \phi(m)\phi(n)$, for any two numbers which are relatively prime.

let \mathbf{Z}_n^* be set of numbers from 1 to $n - 1$ but are relatively prime.

Theorem

If $a \in \mathbf{Z}_n^*$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

Recap:Fermat's Theorem

Theorem

Let p be a prime number, then if $\gcd(a, p) = 1$, then

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

This is the particular case of Euler's Theorem when n is prime.

Fermat's Little Theorem

Theorem

Let p be a prime number,

$$a^p = a \pmod{p}, \text{ for any integer } a.$$

When a is relatively prime, the theorem follows from the Fermat's theorem. When a is multiple of p , the result is trivially true.

Recap: Chinese Remainder Theorem (CRT)

Let n_1, n_2 be pair-wise relatively prime integers, he system of simultaneous congruences

$$x \equiv a_1 \pmod{n_1},$$

$$x \equiv a_2 \pmod{n_2},$$

has a unique solution modulo $n = n_1 n_2$.

2.1 Proof of RSA Encryption.

Before starting any transactions, Alice(A) and Bob (B) will set up the following key initializations.

Alice will do the following:

- ① Generate two large and distinct primes p_A and q_A of almost equal size.
- ② Compute $n_A = p_A q_A$ and $\phi_A = (p_A - 1)(q_A - 1)$.
- ③ Select a random integer e_A , such that $GCD[e_A, \phi_A] = 1$.
- ④ Compute the integer d_A such that

$$e_A d_A \equiv 1 \pmod{\phi_A}.$$

(Use Extended Euclidean Algorithm).

- ⑤ **Alice's Public key is (n_A, e_A) .**
Alice's Private key is d_A .

Similarly, Bob will also initialize the key parameters. Let
Bob's Public key be (n_B, e_B) and
Bob's Private key be d_B ,

Here we assume that Bob wants to send a message to Alice.

Encryption at B

- ① Get A's Public Key (n_A, e_A).
- ② Choose a message M as an integer in the interval $[0, n_A - 1]$.
- ③ Compute $c = M^{e_A} \pmod{n_A}$.
- ④ Send the cipher text c to A.

Decryption at A

- ① To recover m compute $M = c^{d_A} \pmod{n_A}$ using the secret d_A .

Proof of RSA Decryption

Since $e_A d_A \equiv 1 \pmod{\phi_A}$, by the extended Euclidean algorithm it is possible to find k such that

$$e_A d_A = 1 + k\phi_A = 1 + k(p_A - 1)(q_A - 1).$$

(Run Extended Euclidean algorithm on $(e_A, \phi(n_A))$ or $(d_A, \phi(n_A))$.)
From Fermat' theorem we get,

$$M^{p_A-1} \equiv 1 \pmod{p_A}.$$

Hence,

$$M^{e_A d_A} \equiv M^{1+k(p_A-1)(q_A-1)} \equiv M (M^{(p_A-1)})^{(q_A-1)} \equiv M \pmod{p_A}.$$

Similarly,

$$M^{e_A d_A} \equiv M^{1+k(p_A-1)(q_A-1)} \equiv M (M^{(q_A-1)})^{(p_A-1)} \equiv M \pmod{q_A}.$$

Since, p_A and q_A are distinct primes, it follows from Chinese Remainder Theorem that

$$M^{e_A d_A} \equiv M \pmod{n_A}.$$

This implies,

$$c^{d_A} = (M^{e_A})^{d_A} \equiv M \pmod{n_A}.$$

More serious proof of RSA Decryption

Note that we need to prove

$$(M^{e_A})^{d_A} = M^{e_A \cdot d_A} = M \bmod n_A.$$

If M is relatively prime to n_A , then this implies

$(M, p_A) = (M, q_A) = 1$. Then the arguments in the previous slides prove the result.

You can also see this as an application of Eulers's theorem. Note that,

$$e_A d_A = 1 + k\phi_A = 1 + k(p_A - 1)(q_A - 1). \quad (1)$$

Then

$$M^{e_A \cdot d_A} = M^{1+k\phi_A} = M \cdot M^{k\phi_A} = M \cdot (M^{\phi_A})^k = M$$

as $M^{\phi_A} = 1 \bmod n_A$ (Eulers's theorem).

However, again note that to be able to use Fermat's or Euler's theorem, we need $(M, n_A) = 1$.

What if M is not relatively prime to n_A ?

Note that the probability that M is not relatively prime to n_A is very small ($1/p_A + 1/q_A - 1/(p_A q_A)$). If we just ignore this possibility we are done. But, if you are serious and want to prove the RSA result for all $M < n_A$, then see the following.

Case when M is not relatively prime to n_A .

In this case M is divisible by either p_A or q_A . If it is divisible by both p_A and q_A , then $M = 0 \text{ mod } n_A$ and hence the RSA result is trivially true. Then with out loss of generality assume that p_A divides M and hence we can write $M = c \ p_A$. Then we must have $(M, q_A) = 1$ (Otherwise, M is also multiple of q_A and hence identically equal to $0 \text{ mod } n_A$).

Now we can use Fermat's theorem

$$M^{(q_A-1)} = 1 \text{ mod } q$$

Then taking $(k(p_A - 1))^{th}$ power on either side of the above equation, we get,

$$M^{k(p_A-1)(q_A-1)} = 1 \text{ mod } q_A,$$

where k is as in (1). This implies

$$M^{k(p_A-1)(q_A-1)} = 1 + k' q_A,$$

for some k' . Multiplying each side by $M = cp_A$, we get

$$M^{k(p_A-1)(q_A-1)+1} = (1 + k' q_A)M = M + k' (c p_A) q_A = M + k'' n_A.$$

Taking mod n_A on both sides gives the result.

Lecture 1

Public Key Cryptography: Diffie-Hellman Protocol and RSA

Lecture 2

Properties of Numbers IV

Workshop 3: Workshops based on Lectures in Week 3

Quizz 4

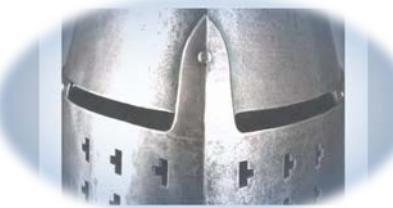
Week 3



Additional Reading for Week 3:

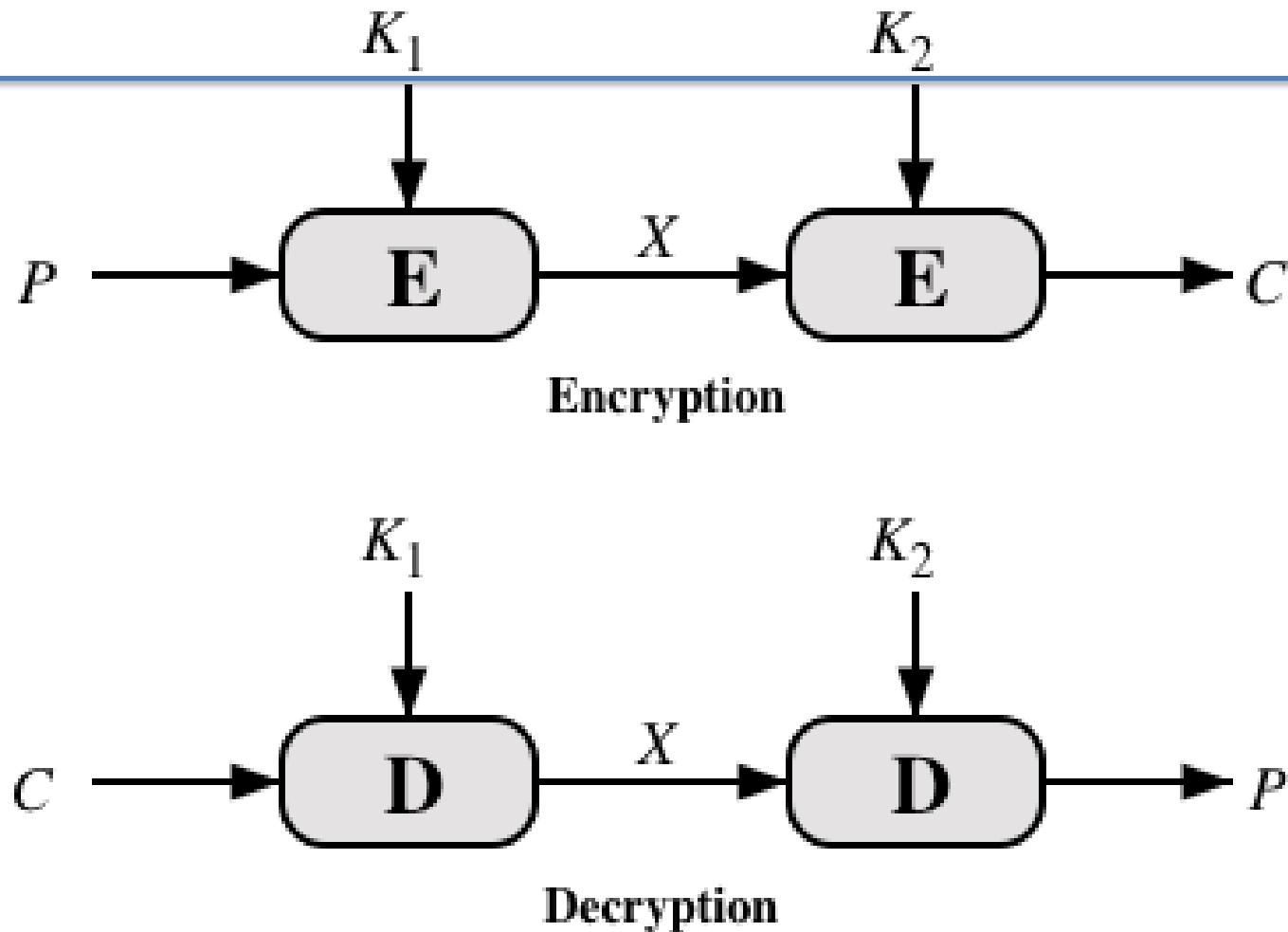
These slides are taken directly from Chapter 7 of the textbook by William Stallings.

You are required to study for this for next week's workshop.



Chapter 7

Block Cipher Operation



(a) Double Encryption

Figure 7.1 Multiple 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 $56 \times 3 = 168$ 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

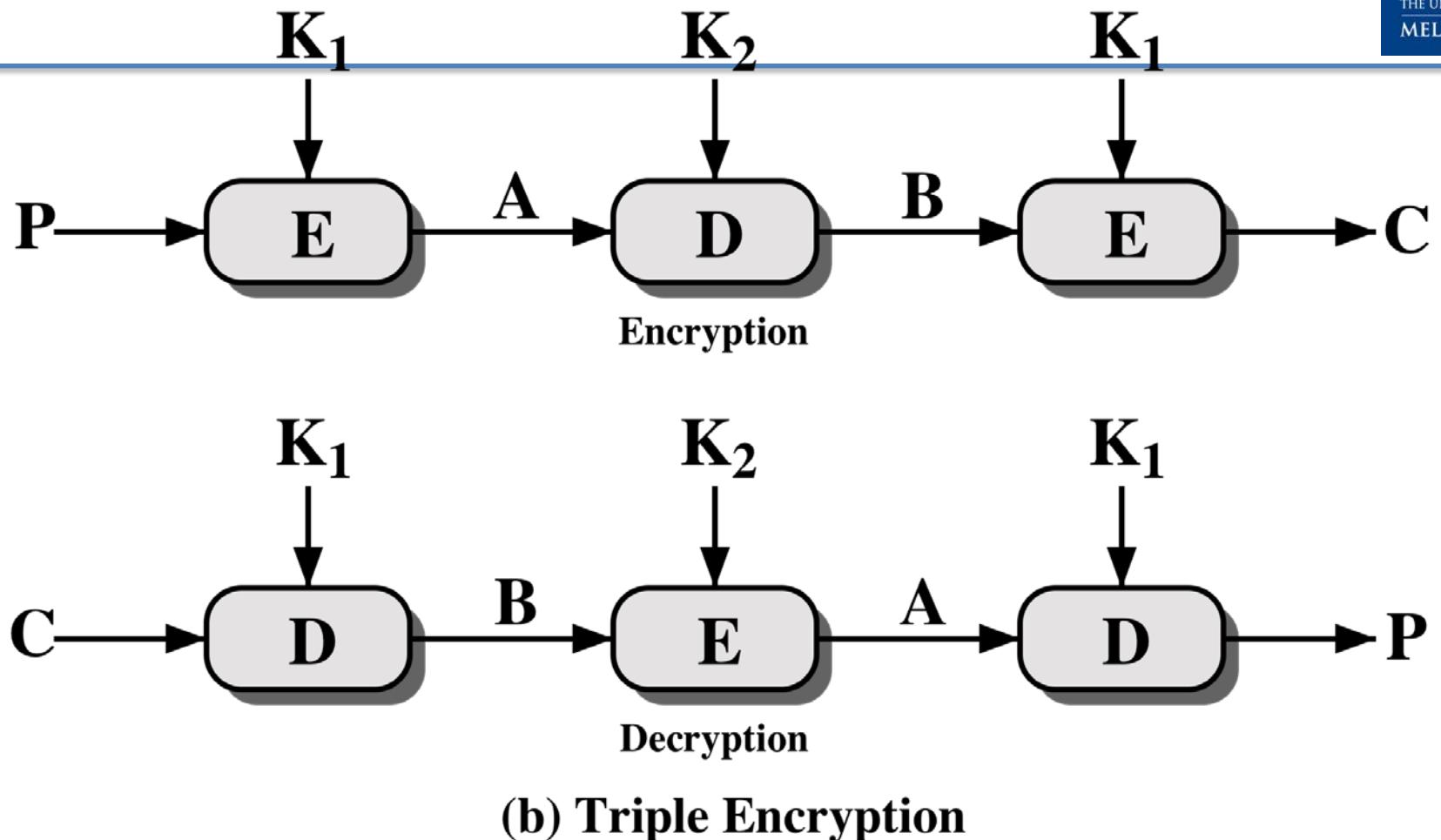
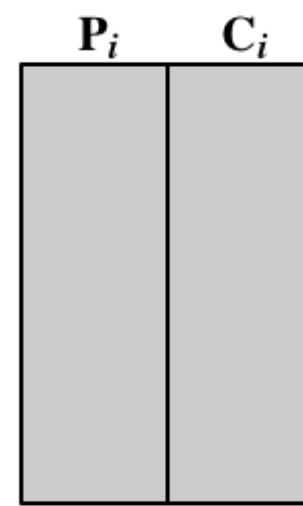
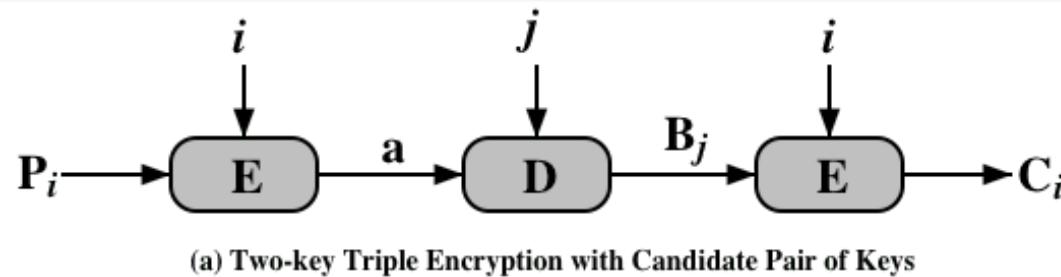
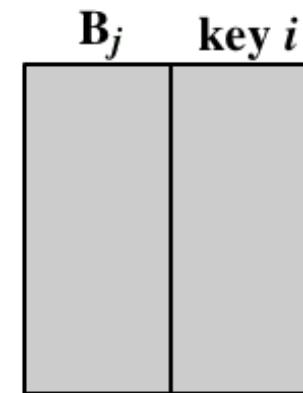


Figure 7.1 Multiple Encryption



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



(c) Table of intermediate values and candidate keys

Figure 7.2 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:

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

Backward compatibility with DES is provided by putting:

$$\bullet K_3 = K_2 \text{ or } K_1 = K_2$$

- A number of systems 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 7.1 Block Cipher Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of plaintext bits is encoded independently using the same key.	<ul style="list-style-type: none">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.	<ul style="list-style-type: none">General-purpose block-oriented transmissionAuthentication
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.	<ul style="list-style-type: none">General-purpose stream-oriented transmissionAuthentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	<ul style="list-style-type: none">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.	<ul style="list-style-type: none">General-purpose block-oriented transmissionUseful for high-speed requirements

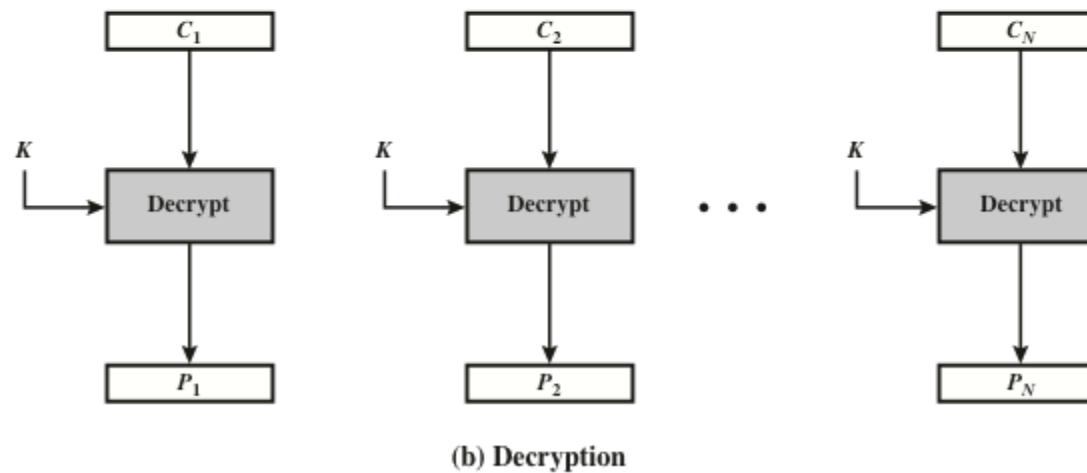
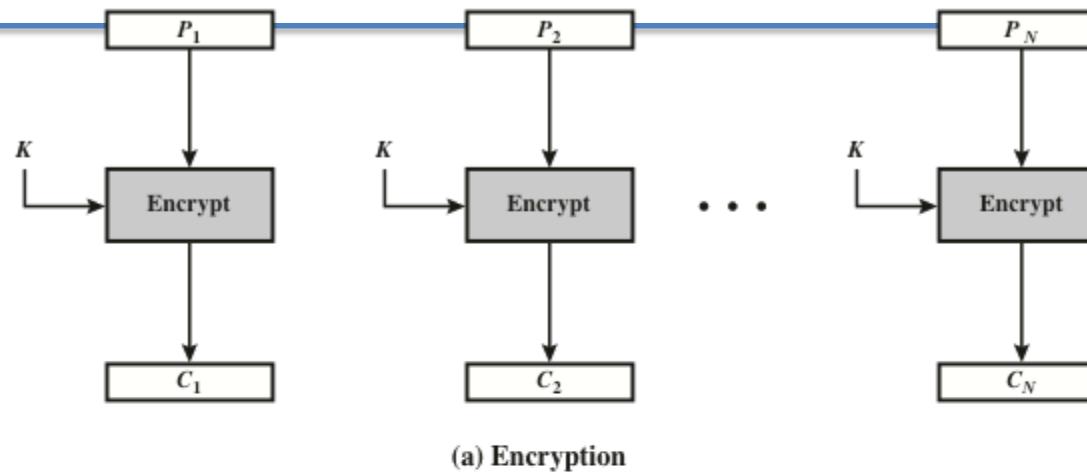


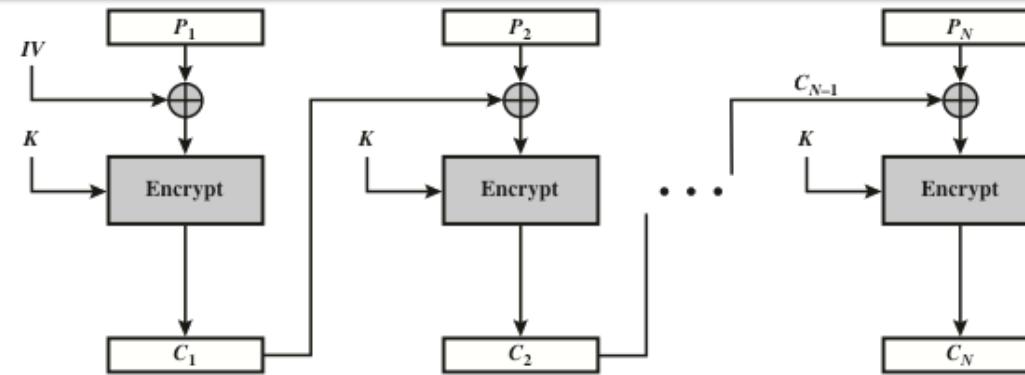
Figure 7.3 Electronic Codebook (ECB) Mode



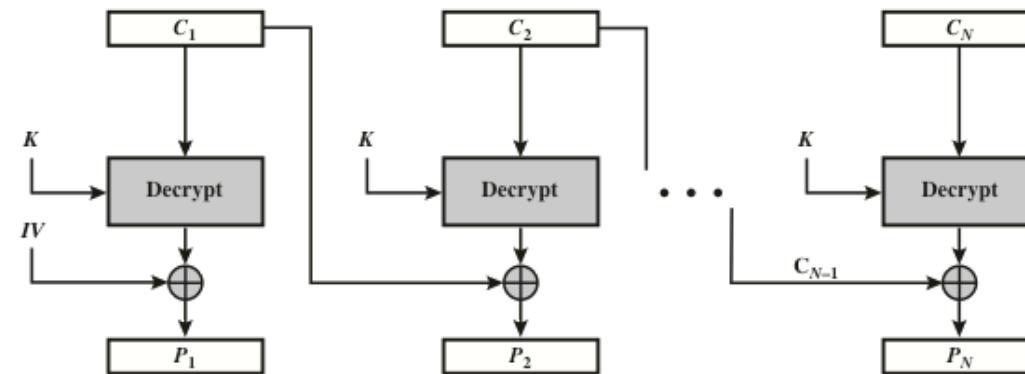
Criteria and properties for evaluating and constructing block cipher modes of operation that are superior to ECB:

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





(a) Encryption



(b) Decryption

Figure 7.4 Cipher Block Chaining (CBC) Mode



Cipher Feedback Mode

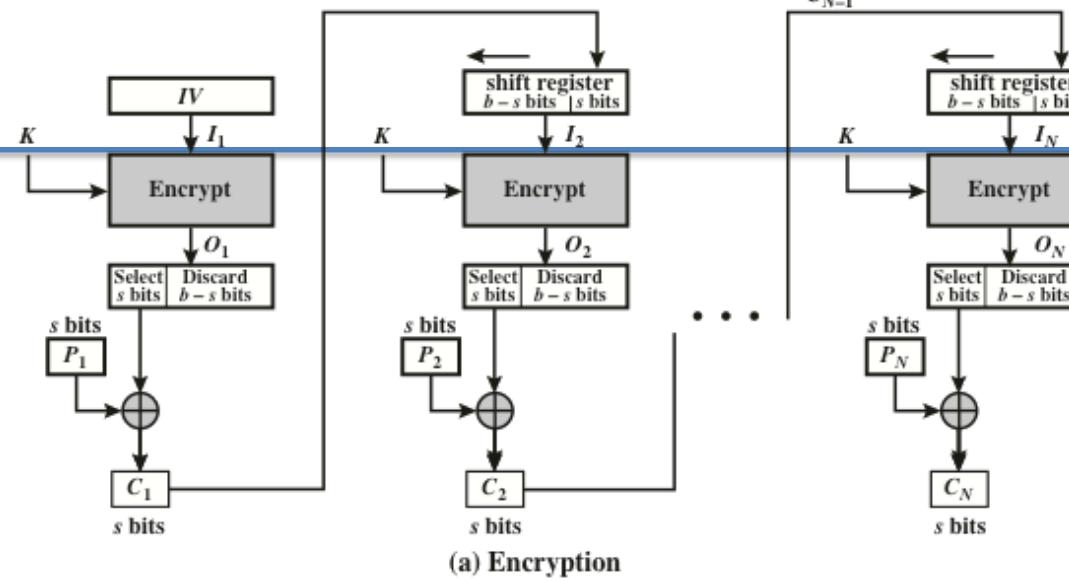
- For AES, DES, or any block cipher, encryption is performed on a block of b bits
 - In the case of DES $b = 64$
 - In the case of AES $b = 128$

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

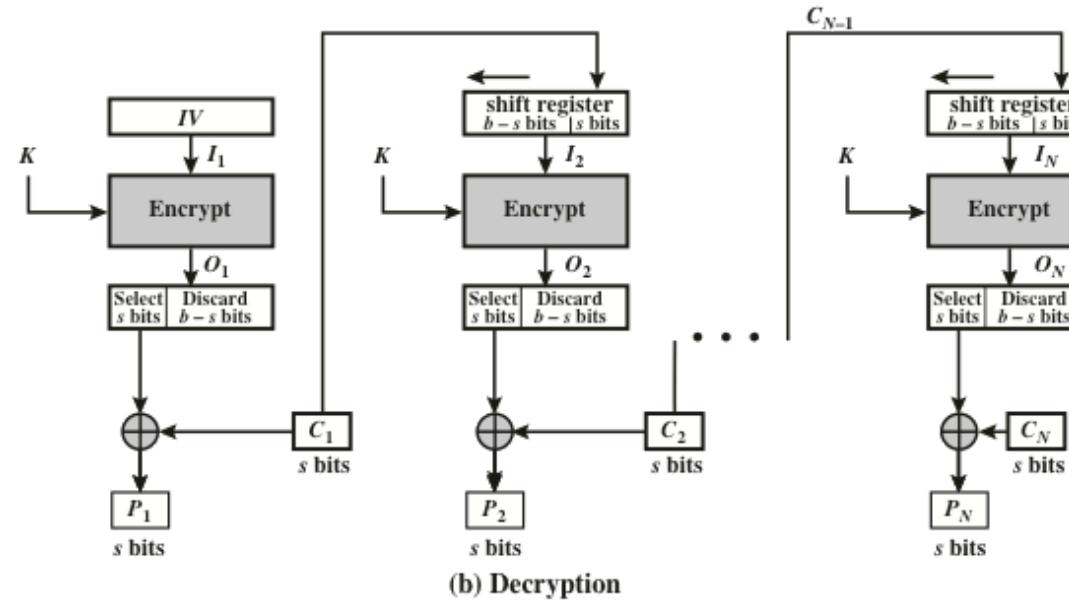
Cipher feedback (CFB) mode

Output feedback (OFB) mode

Counter (CTR) mode



(a) Encryption



(b) Decryption

Figure 7.5 *s*-bit Cipher Feedback (CFB) Mode

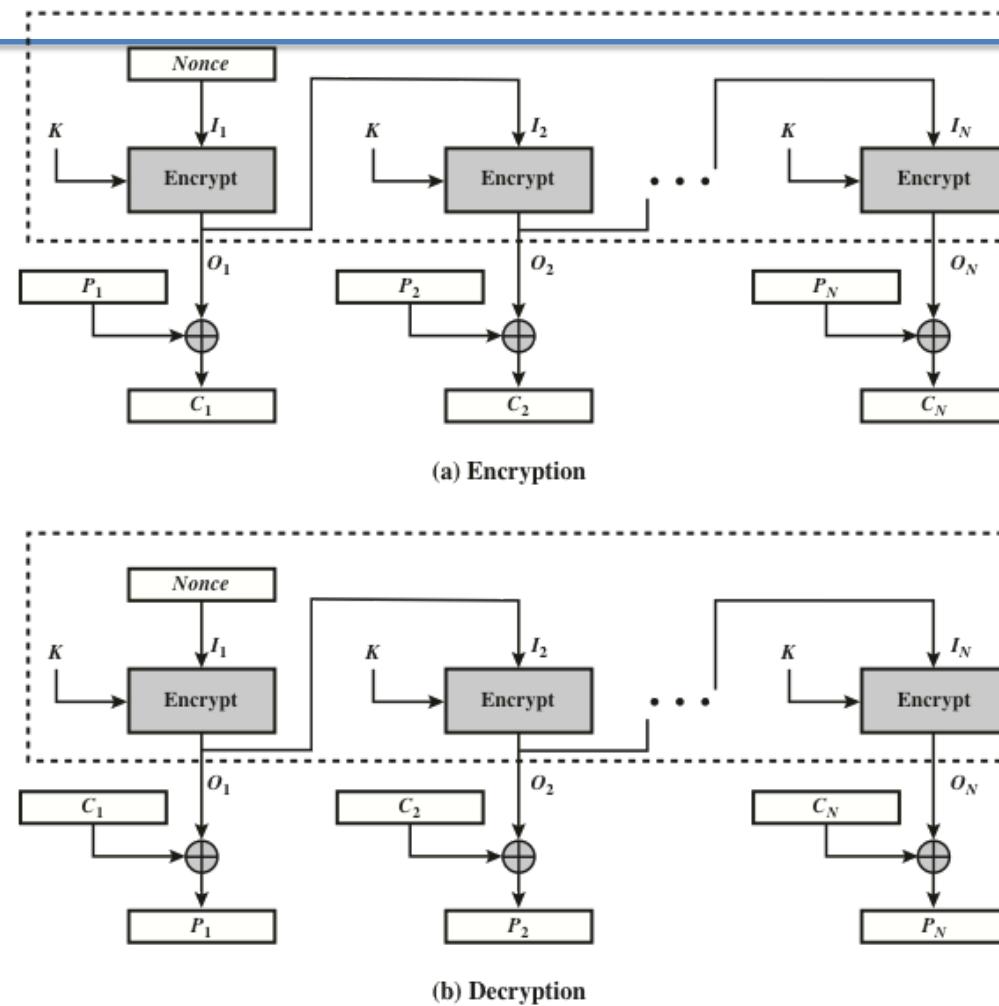
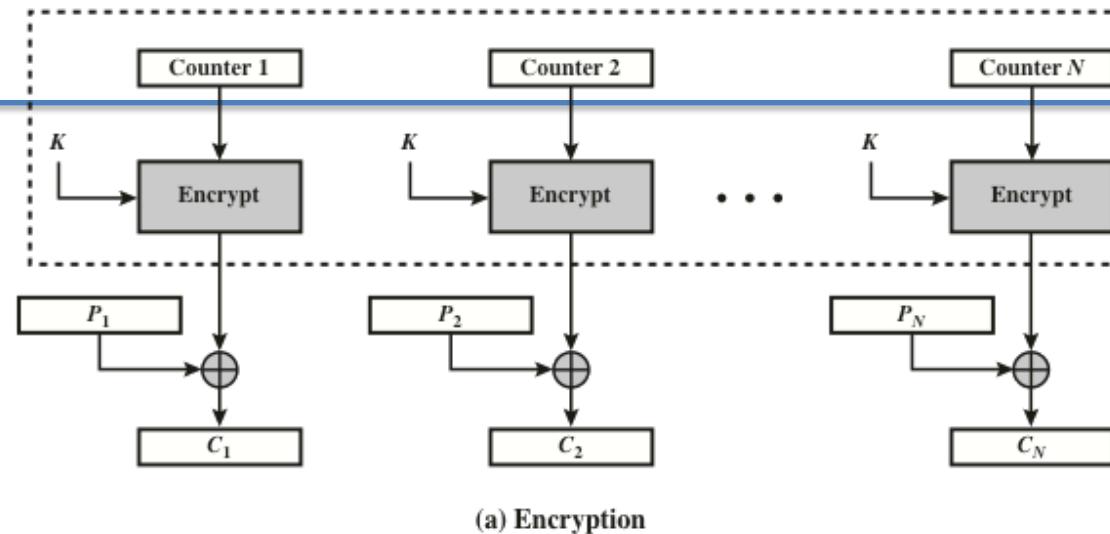
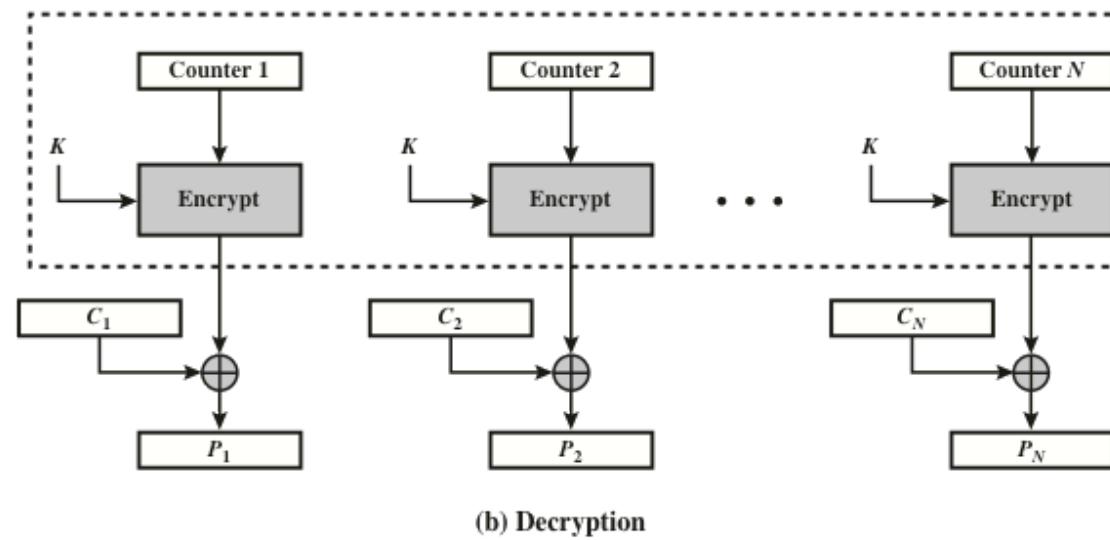


Figure 7.6 Output Feedback (OFB) Mode



(a) Encryption



(b) Decryption

Figure 7.7 Counter (CTR) Mode



Advantages of CTR

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



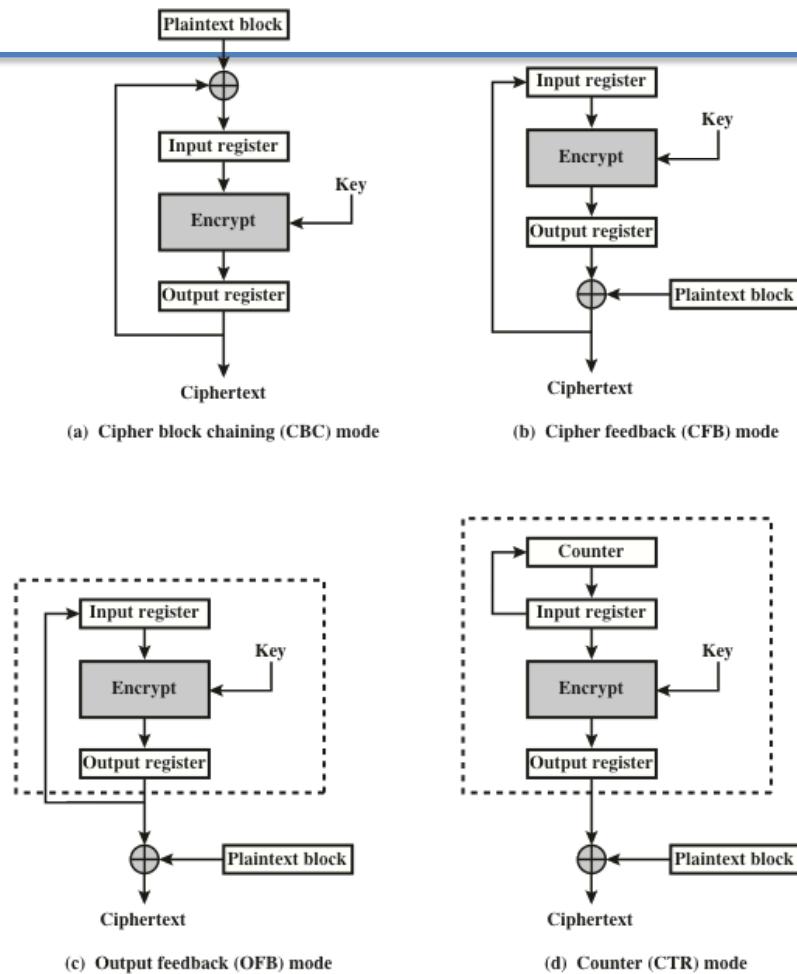


Figure 7.8 Feedback Characteristic of Modes of Operation

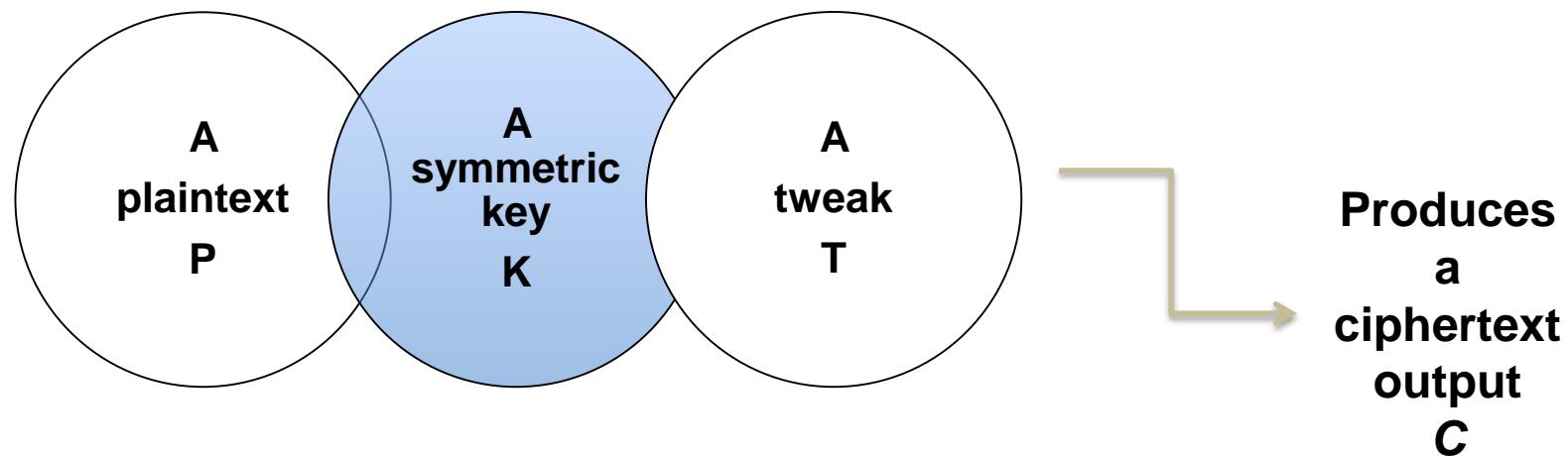


XTS-AES Mode for Block-Oriented Storage Devices

- 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

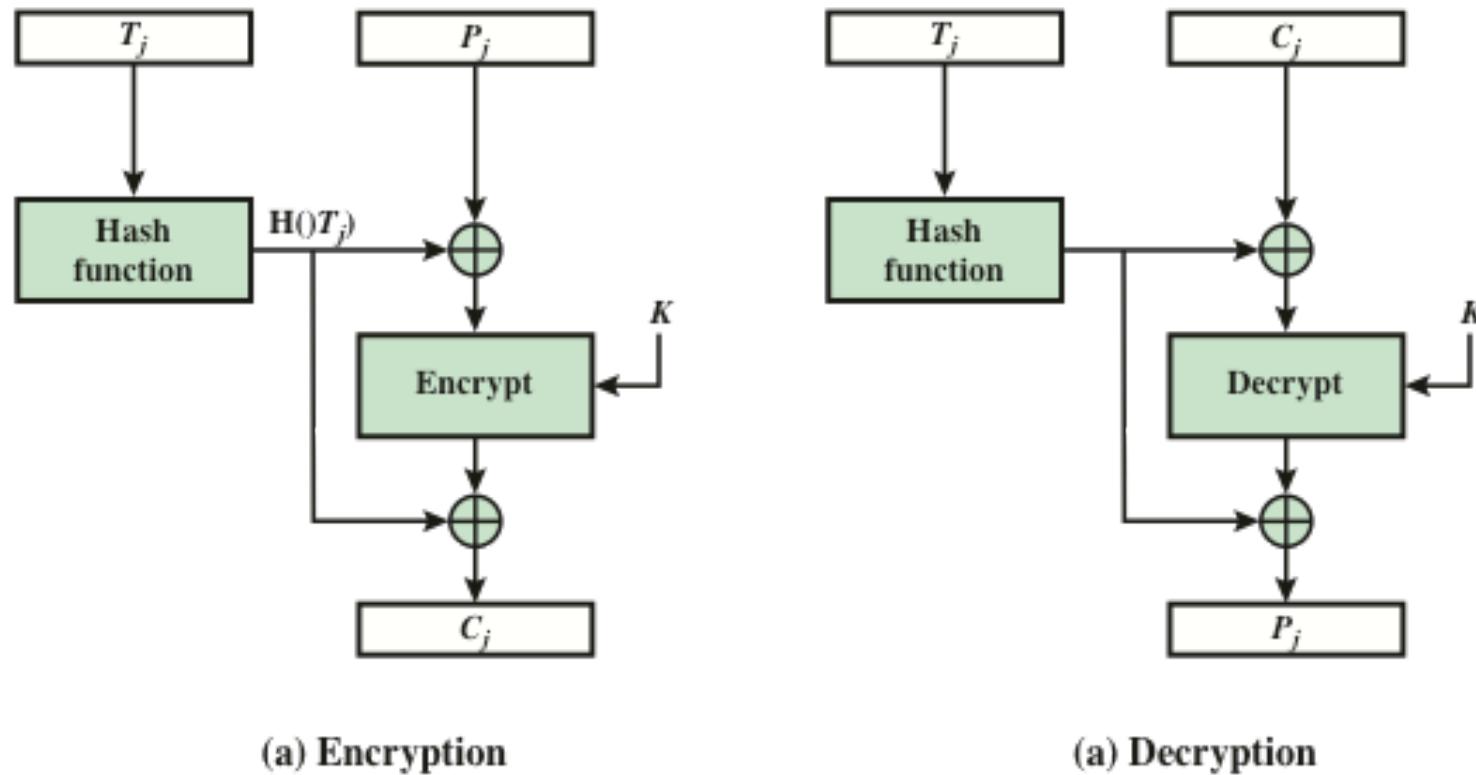


Figure 7.9 Tweakable Block Cipher

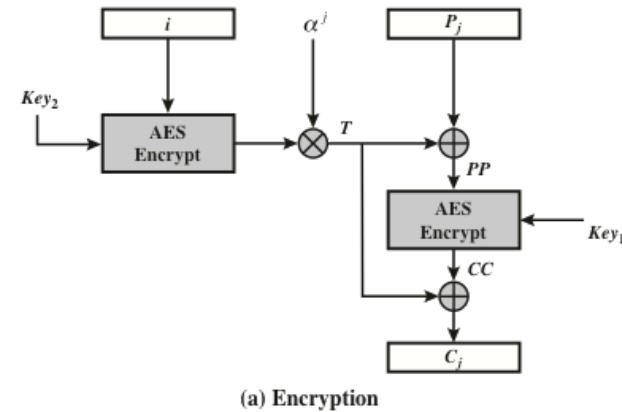


Storage Encryption Requirements

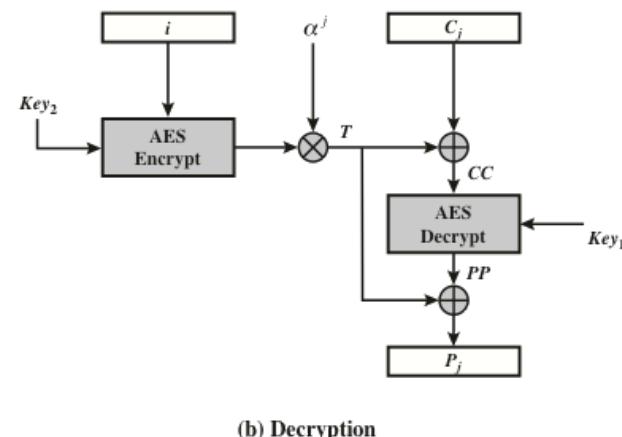
- The requirements for encrypting stored data, also referred to as “data at rest”, differ somewhat from those for transmitted data
- The P1619 standard was designed to have the following characteristics:
 - The ciphertext is freely available for an attacker
 - The data layout is not changed on the storage medium and in transit
 - Data are accessed in fixed sized blocks, independently from each other
 - Encryption is performed in 16-byte blocks, independently from each other
 - There are no other metadata used, except the location of the data blocks within the whole data set
 - The same plaintext is encrypted to different ciphertexts at different locations, but always to the same ciphertext when written to the same location again
 - A standard conformant device can be constructed for decryption of data encrypted by another standard conformant device



XTS-AES Operation on Single Block



(a) Encryption



(b) Decryption

Figure 7.10 XTS-AES Operation on Single Block

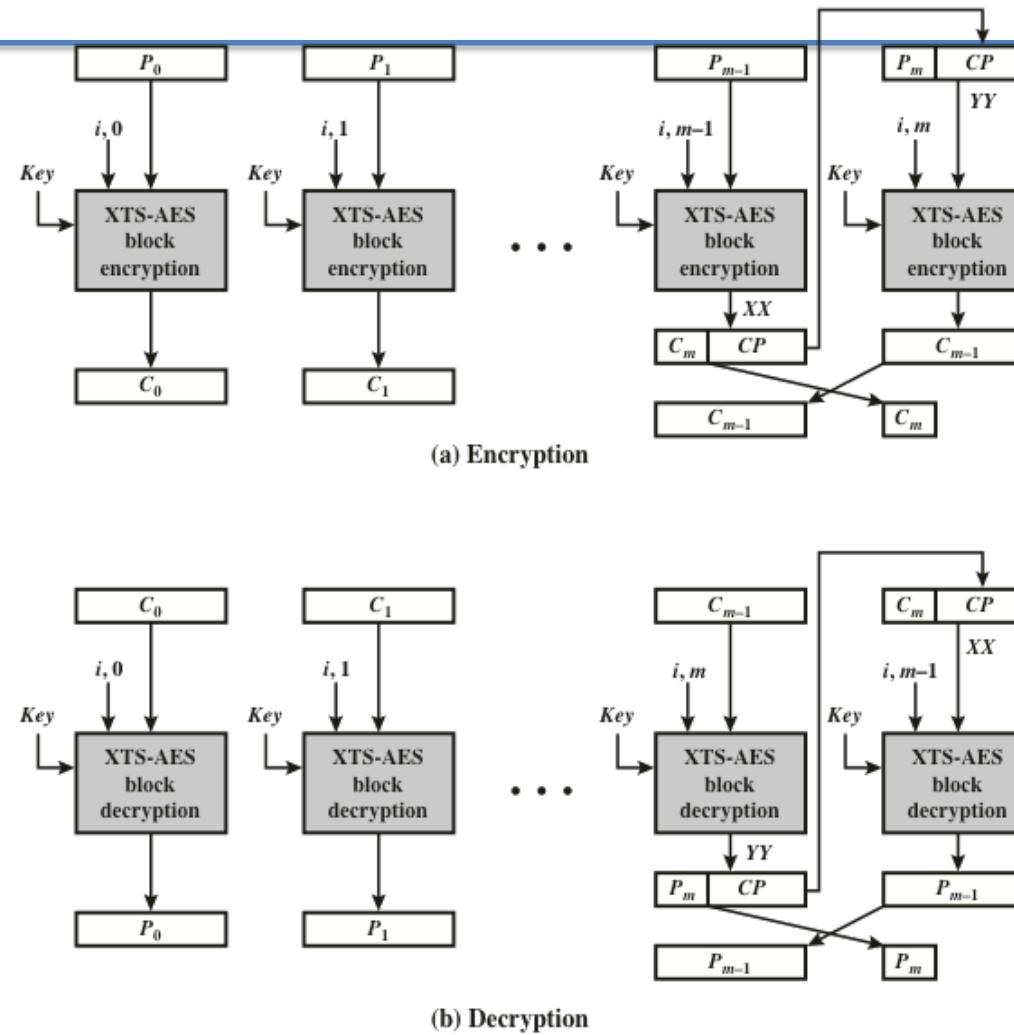


Figure 7.11 XTS-AES Mode

Format-Preserving Encryption (FPE)



- Refers to any encryption technique that takes a plaintext in a given format and produces a ciphertext in the same format
 - For example: credit cards consist of 16 decimal digits. An FPE that can accept this type of input would produce a ciphertext output of 16 decimal digits. (Note that the ciphertext need not be, and in fact is unlikely to be, a valid credit card number.) But it will have the same format and can be stored in the same way as credit card number plaintext.





Table 7.2

Comparison of Format-Preserving Encryption and AES

	Credit Card	Tax ID	Bank Account Number
Plaintext	8123 4512 3456 6780	219-09-9999	800N2982K-22
FPE	8123 4521 7292 6780	078-05-1120	709G9242H-35
AES (hex)	af411326466add24 c86abd8aa525db7a	7b9af4f3f218ab25 07c7376869313afa	9720ec7f793096ff d37141242e1c51bd



Motivation

FPE facilitates the retrofitting of encryption technology to legacy applications, where a conventional encryption mode might not be feasible because it would disrupt data fields/pathways

FPE has emerged as a useful cryptographic tool, whose applications include financial-information security, data sanitization, and transparent encryption of fields in legacy databases

The principal benefit of FPE is that it enables protection of particular data elements, while still enabling workflows that were in place before FPE was in use

- No database schema changes and minimal application changes are required
- Only applications that need to see the plaintext of a data element need to be modified and generally these modifications will be minimal

Some examples of legacy applications where FPE is desirable are:

- COBOL data-processing applications
- Database applications
- FPE-encrypted characters can be significantly compressed for efficient transmission



Difficulties in Designing an FPE

- A general-purpose standardized FPE should meet a number of requirements:
 - The ciphertext is of the same length and format as the plaintext
 - It should be adaptable to work with a variety of character and number types
 - It should work with variable plaintext length
 - Security strength should be comparable to that achieved with AES
 - Security should be strong even for very small plaintext lengths

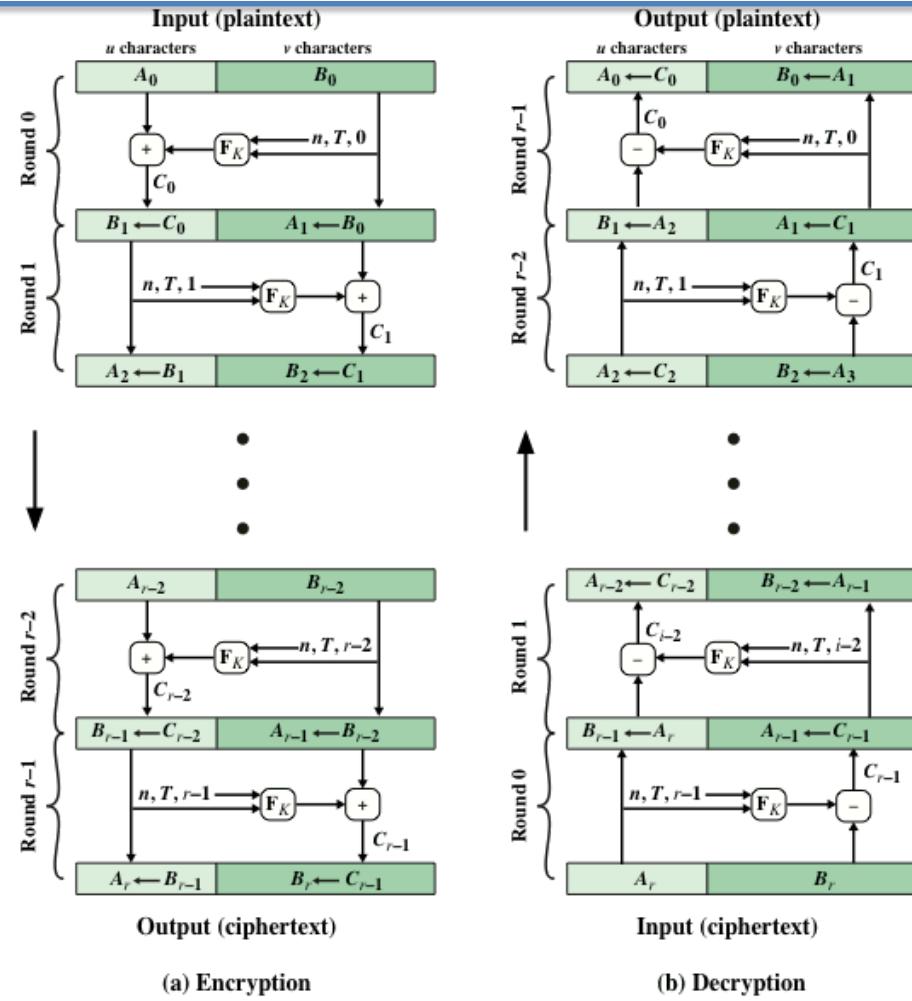


Figure 7.12 Feistel Structure for Format-Preserving Encryption



Character Strings

- The NIST, and the other FPE algorithms that have been proposed, are used with plaintext consisting of a string of elements, called *characters*
- A finite set of two or more symbols is called an *alphabet*
- The elements of an alphabet are called *characters*
- A *character string* is a finite sequence of characters from an alphabet
- Individual characters may repeat in the string
- The number of different characters in an alphabet is called the *base* (also referred to as the *radix*) of the alphabet

**Table 7.3 Notation and Parameters Used in FPE Algorithms****(a) Notation**

$[x]^s$	Converts an integer into a byte string; it is the string of s bytes that encodes the number x , with $0 \leq x < 2^{8s}$. The equivalent notation is $\text{STR}_2^{8s}(x)$.
$\text{LEN}(X)$	Length of the character string X .
$\text{NUM}_{\text{radix}}(X)$	Converts strings to numbers. The number that the numeral string X represents in base radix , with the most significant character first. In other words, it is the non-negative integer less than $\text{radix}^{\text{LEN}(X)}$ whose most-significant-character-first representation in base radix is X .
$\text{PRF}_K(X)$	A pseudorandom function that produces a 128-bit output with X as the input, using encryption key K .
$\text{STR}_{\text{radix}}^m(x)$	Given a nonnegative integer x less than radix^m , this function produces a representation of x as a string of m characters in base radix , with the most significant character first.
$[i .. j]$	The set of integers between two integers i and j , including i and j .
$X[i .. j]$	The substring of characters of a string X from $X[i]$ to $X[j]$, including $X[i]$ and $X[j]$.
$\text{REV}(X)$	Given a bit string, X , the string that consists of the bits of X in reverse order.

Table 7.3

Notation and Parameters Used in FPE Algorithms

(b) Parameters

<i>radix</i>	The base, or number of characters, in a given plaintext alphabet.
<i>tweak</i>	Input parameter to the encryption and decryption functions whose confidentiality is not protected by the mode.
<i>tweakradix</i>	The base for tweak strings
<i>minlen</i>	Minimum message length, in characters.
<i>maxlen</i>	Maximum message length, in characters.
<i>maxTlen</i>	Maximum tweak length



t

Prerequisites:

Approved, 128-bit block cipher, CIPH;

Key, K , for the block cipher;

Input:

Nonempty bit string, X , such that $\text{LEN}(X)$ = is a multiple of 128.

Output:

128-bit block, Y

Steps:

1. Let $m = \text{LEN}(X)/128$.
2. Partition X into m 128-bit blocks X_1, \dots, X_m , so that $X = X_1 \parallel \dots \parallel X_m$
3. Let $Y_0 = [0]^{16}$
4. For j from 1 to m :
 - 4.i let $Y_j = \text{CIPH}_K(Y_{j-1} \oplus X_j)$.
6. Return Y_m .

Figure 7.13 Algorithm PRF(X)

**Prerequisites:**

Approved, 128-bit block cipher, CIPH;
Key, K , for the block cipher;
Base, $radix$, for the character alphabet;
Range of supported message lengths, $[minlen .. maxlen]$;
Maximum byte length for tweaks, $maxTlen$.

Inputs:

Character string, X , in base radix of length n such that $n \in [minlen .. maxlen]$;
Tweak T , a byte string of byte length t , such that $t \in [0 .. maxTlen]$.

Output:

Character string, Y , such that $\text{LEN}(Y) = n$.

Steps:

1. Let $u = \lfloor n/2 \rfloor$; $v = n - u$.
2. Let $A = X[1 .. u]$; $B = X[u + 1 .. n]$.
3. Let $b = \lceil \lceil v \log_2(radix) \rceil / 8 \rceil$; $d = 4 \lceil b/4 \rceil + 4$
4. Let $P = [1]^1 \parallel [2]^1 \parallel [1]^1 \parallel [radix]^3 \parallel [10]^1 \parallel [u \bmod 256]^1 \parallel [n]^4 \parallel [t]^4$.
5. For i from 0 to 9:

- i. Let $Q = T \parallel [0]^{(t-b-1) \bmod 16} \parallel [j]^1 \parallel [\text{NUM}_{radix}(B)]^b$.
- ii. Let $R = \text{PRF}_K(P \parallel Q)$.
- iii. Let S be the first d bytes of the following string of $\lceil d/16 \rceil$ 128-bit blocks:
 $R \parallel \text{CIPH}_K(R \oplus [1]^{16}) \parallel \text{CIPH}_K(R \oplus [2]^{16}) \parallel \dots \parallel \text{CIPH}_K(R \oplus [\lceil d/16 \rceil - 1]^{16})$.
- iv. Let $y = \text{NUM}_2(S)$.
- v. If i is even, let $m = u$; else, let $m = v$.

- vi. Let $c = (\text{NUM}_{radix}(A) + y) \bmod radix^m$.
- vii. Let $C = \text{CTR}_{radix}^m(c)$.
- viii. Let $A = B$.
- ix. Let $B = C$.

6. Return $Y = A \parallel B$.

Figure 7.14 Algorithm FF1 (FFX[Radix])



Approved, 128-bit block cipher, CIPH;
Key, K , for the block cipher;
Base, $tweakradix$, for the tweak character alphabet;
Range of supported message lengths, $[minlen .. maxlen]$
Maximum supported tweak length, $maxTlen$.

Inputs:

Numerical string, X , in base $radix$, of length n such that $n \in [minlen .. maxlen]$;
Tweak numerical string, T , in base $tweakradix$, of length t such that $t \in [0 .. maxTlen]$.

Output:

Numerical string, Y , such that $\text{LEN}(Y) = n$.

Steps:

1. Let $u = \lfloor n/2 \rfloor$; $v = n - u$.
2. Let $A = X[1 .. u]$; $B = X[u + 1 .. n]$.
3. If $t > 0$, $P = [radix]^1 \parallel [t]^1 \parallel [n]^1 \parallel [\text{NUM}_{tweakradix}(T)]^{13}$;
else $P = [radix]^1 \parallel [0]^1 \parallel [n]^1 \parallel [0]^{13}$.
4. Let $J = \text{CIPH}_K(P)$
5. For i from 0 to 9:
 - i. Let $Q \leftarrow [i]^1 \parallel [\text{NUM}_{radix}(B)]^{15}$
 - ii. Let $Y \leftarrow \text{CIPH}_J(Q)$.
 - iii. Let $y \leftarrow \text{NUM}_2(Y)$.
 - iv. If i is even, let $m = u$; else, let $m = v$.
 - v. Let $c = (\text{NUM}_{radix}(A) + y) \bmod radix^m$.
 - vi. Let $C = \text{STR}_{radix}^m(c)$.
 - vii. Let $A = B$.
 - viii. Let $B = C$.
6. Return $Y = A \parallel B$.

Figure 7.15 Algorithm FF2 (VAES3)



Approved, 128-bit block cipher, CIPH;
Key, K , for the block cipher;
Base, $radix$, for the character alphabet such that $radix \in [2 .. 2^{16}]$;
Range of supported message lengths, $[minlen .. maxlen]$,
such that $minlen \geq 2$ and $maxlen \leq 2\lceil \log_{radix}(2^{96}) \rceil$.

Inputs:

Numerical string, X , in base $radix$ of length n such that $n \in [minlen .. maxlen]$;

Tweak bit string, T , such that $\text{LEN}(T) = 64$.

Output:

Numerical string, Y , such that $\text{LEN}(Y) = n$.

Steps:

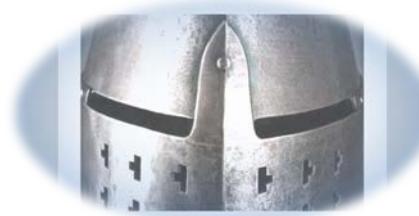
1. Let $u = \lceil n/2 \rceil$; $v = n - u$.
 2. Let $A = X[1 .. u]$; $B = X[u + 1 .. n]$.
 3. Let $T_L = T[0..31]$ and $T_R = T[32..63]$
 4. For i from 0 to 7:
 - i. If i is even, let $m = u$ and $W = T_R$, else let $m = v$ and $W = T_L$.
 - ii. Let $P = \text{REV}([\text{NUM}_{radix}(\text{REV}(B))]^{12}) \parallel [W \oplus \text{REV}([i]^4)]$.
 - iii. Let $Y = \text{CIPH}_K(P)$.
 - iv. Let $y = \text{NUM}_2(\text{REV}(Y))$.
 - v. Let $c = (\text{NUM}_{radix}(\text{REV}(A)) + y) \bmod radix^m$.
 - vi. Let $C = \text{REV}(\text{STR}_{radix}^m(c))$.
 - vii. Let $A = B$.
 - viii. Let $B = C$.
5. Return $A \parallel B$.

Figure 7.16 Algorithm FF3 (BPS-BC)



Summary

- Multiple encryption and triple DES
 - Double DES
 - Triple DES with two keys
 - Triple DES with three keys
- Electronic codebook
- Cipher block chaining mode
- Format-preserving encryption
 - Motivation
 - Difficulties in designing
 - Feistel structure
 - NIST methods
- Cipher feedback mode
- Output feedback mode
- Counter mode
- XTS-AES mode for block-oriented storage devices
 - Tweakable block ciphers
 - Storage encryption requirements
 - Operation on a single block
 - Operation on a sector



Week 3



Lecture 1

Modern Symmetric key Ciphers

Lecture 2

Properties of Numbers III,

Workshop 3: Workshop based on Lectures in Week2

Quiz 3

Modern Symmetric key Ciphers

COMP90043

Lecture 1

Modern Symmetric key Cryptography

Lecture 1

1.1 Modern Symmetric Ciphers

- Model and Design Principles
- Stream Ciphers and Block Ciphers

1.2 One-Time Pad Encryption

- Vernam Cipher
- One-Time Pad
- Perfect Secrecy

1.3 Fiestel Cipher

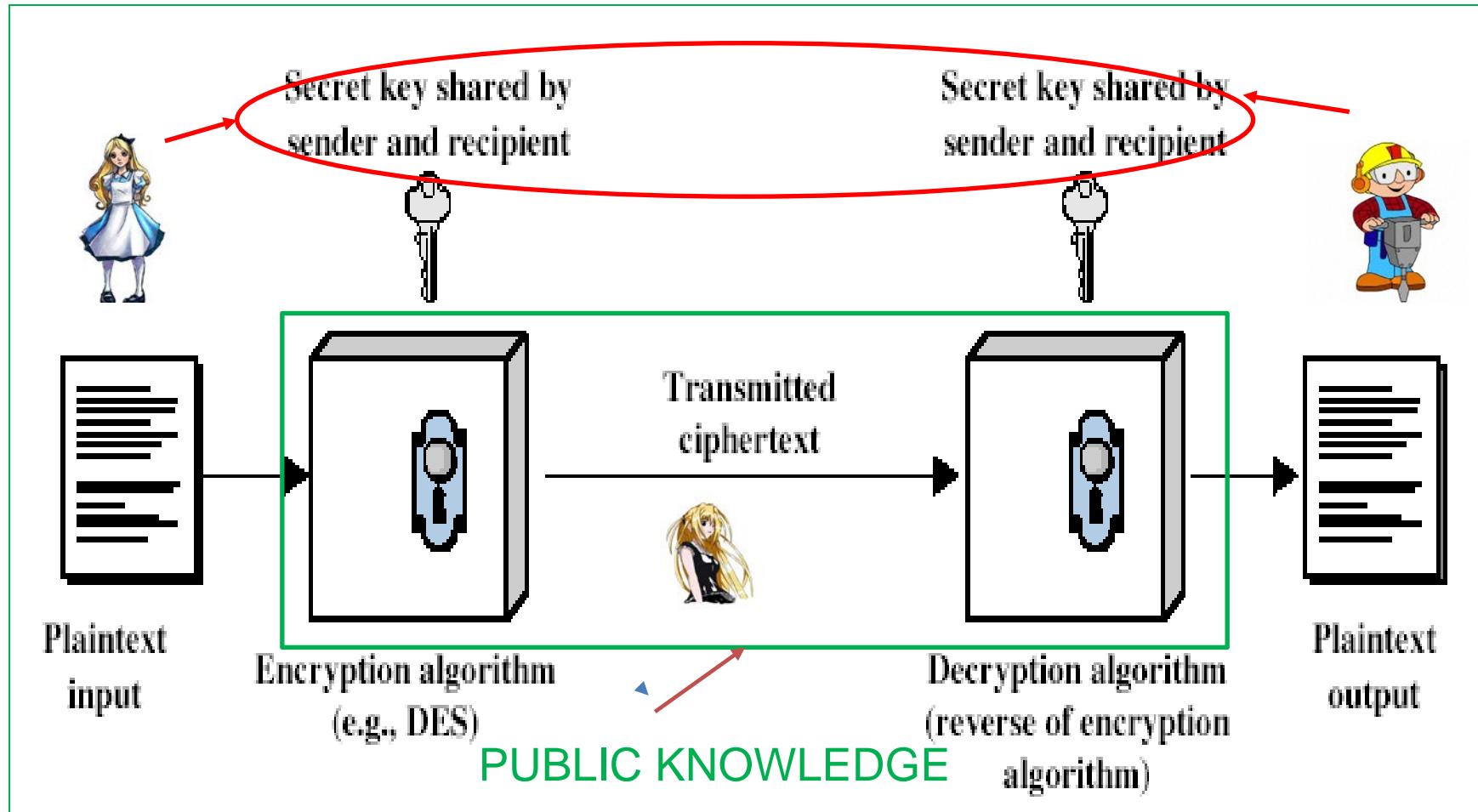
- Motivation and General ideas
- Cipher Terms and Structure
- Data Encryption Standard
- A worksheet

1.4 Modes of Block Ciphers

- Codebook Mode
- Cipher Block Chain
- Stream Cipher modes

Recap: Symmetric Key Cyptosystems

Modified From: Stallings Figure 2.1:



Recap (Week 2)



- 1.1 Symmetric Cipher Models
 - Basic Terminology
 - Model and Logical View
 - Basic Requirements and Kerckhoffs'sprinciple
- 1.2 Security
 - Characterization of Symmetric key Encryption
 - Attacks on Symmetric key Encryption
- 1.3 Classical Ciphers
 - Substitution Ciphers Caesar and Affine Ciphers
 - Monoalphabetic Substitution Ciphers
 - Transposition Ciphers Rail fence cipher
 - Row Transposition Cipher
- 1.4 Cryptanalysis of Classical Ciphers
 - Caesar Cipher
 - Affine Cipher
 - Monoalphabetic Substitution Ciphers
- 1.5 Complex Ciphers
 - Polyalphabetic Ciphers, Vigenère Cipher

Numbers, gcd, primes,
Extended GCD algorithm
Inverse mod n
Euler Phi function

1.1 Modern Symmetric Ciphers

COMP90043

Lecture 1

Design Principles

- These are two major kinds of ciphers, which differ in the way the plaintexts are encrypted.
- **Block Cipher:** A block cipher takes a fixed length plain text message block (for example, 64 or 128 bits) and a key, and produces a cipher text block of the same length as the original message.
 - DES (56), Triple-DES (168), IDEA (128), Blowfish() and AES (128)
- **Stream Cipher:** Takes a key of fixed size and generates a key stream in a pseudo random fashion with large period; this key stream is then combined with the plain text message stream on a bit by bit basis to form a cipher text stream.
 - RC4, A5, BlueTooth cipher etc.

Stream and Block Ciphers

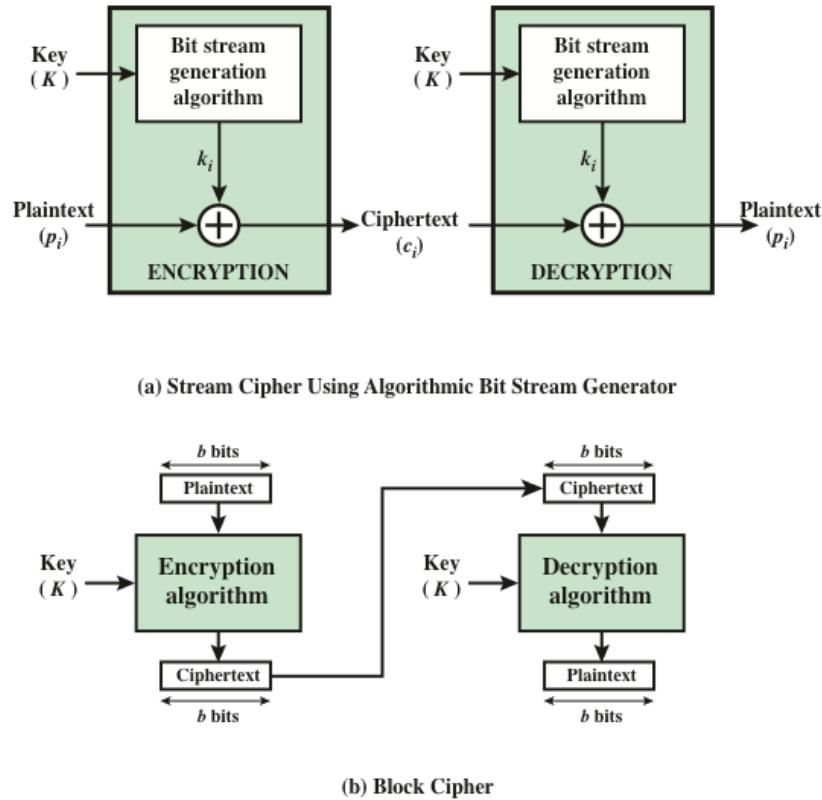


Figure 4.1 Stream Cipher and Block Cipher

From:Stallings Figure 4.1:

- Unit of stream operation can be “bit by bit” or “byte by byte” or “symbol by symbol”, it encrypts one unit of plain text stream at a time. Useful for processing stream-based data such as voice, connection-oriented traffic etc.
- Unit of operation is always of a block of information, generally n-bit blocks. Useful in many situations of data traffic.

.

1.2 One Time Pad Encryption

COMP90043

Lecture 1

Vernam Cipher

- We looked at Vigenere Cipher, a simple polyalphabetic substitution cipher.
- i^{th} plaintext symbol is handled by Caesar cipher with key: $k_{(i \bmod d)}$
- The idea is very simple, a key is a multiple letter word: $K = k_1 k_2 \dots k_d$
- $P = p_1 p_2 \dots p_d p_{d+1} p_{d+2} \dots p_{2d} \dots$
- $C = c_1 c_2 \dots c_d c_{d+1} c_{d+2} \dots c_{2d} \dots$
- Encryption: $E(K, P) = C$, where $c_i = p_i + k_i \bmod 26$
- Decryption: $D(K, C) = P$, where $p_i = c_i - k_i \bmod 26$
- Here we extend the size of the key to be equal to the message ($d = n$). The resulting cipher is Vernam.
- The scheme can be defined over any alphabet ($\bmod m$).
- It is also called as One-Time-Pad.

One-Time Pad Definition

- Defined over binary messages.
- Let \oplus denote exclusive or symbol. Let $[0,1]$ be binary alphabet.
 - $0 \oplus 0 = 1 \oplus 1 = 0$;
 - $0 \oplus 1 = 1 \oplus 0 = 1$.
- We will extend the operation naturally (point wise) to any sequence over $[0,1]$.
- If A, B, C are vectors, is point-wise vector XOR then
 - $A \oplus B = C$; then $B = A \oplus C$; $A \oplus A = 0$; $B \oplus B = 0$, $C + C = 0$
- Suppose Alice wishes to send a message $M = 0110111$ to Bob and they have previously established a shared secret key: $K = 1011011$.
The cipher text is formed by exclusive-oring the message with the key:
$$C = M \oplus K = 1101100.$$

Decryption is trivial: the message could be obtained by the same process, i.e. by addition of K to C .

$$M = C \oplus K = 0110111.$$

One-Time Pad Properties

- An extension of Vernam Cipher for binary messages.
- Here the key is as long as the message.
- For each message you need a distinct random key.
- Encryption and Decryption operation are exactly same, XOR with the key.

Perfect Secrecy

- What does it mean for an encryption scheme to be perfectly secure?
- Let us look at the approach taken by Shannon to answer this question.
- An encryption scheme has the property of **unconditional security** if the cipher text generated by the algorithm does not reveal sufficient information to break the scheme, even with access to an unlimited amount of computational power.
- In other words, the adversary cannot obtain any knowledge to reverse the encryption by watching any amount of cipher text without access to the key. Shannon in his seminal paper* in 1949 showed that one-time pad encryption is perfectly secure.

* C.E. Shannon. Communication in presence of noise. IEEE, 37:1021, 1949.

Probability Basics

- Let S be a sample space of events.
- $S = \{x_1, x_2, x_3, \dots, x_n\}$
- An event A is a subset of S , probability of A satisfies:
- $0 \leq P(A) \leq 1$.
- $P(S) = 1, P(\emptyset) = 0$.
- If $E \subset F, E, F \in S$, then $P(E) \leq P(F)$
- $P(E) + P(E^c) = 1$, where $E^c = S \setminus E$.
- Conditional Probability: If $A, B \in S$ are any events in S and $P(B) = 0$, then the conditional probability relative to the event B is given by
- $P(A | B) = P(A \cap B) / P(B)$

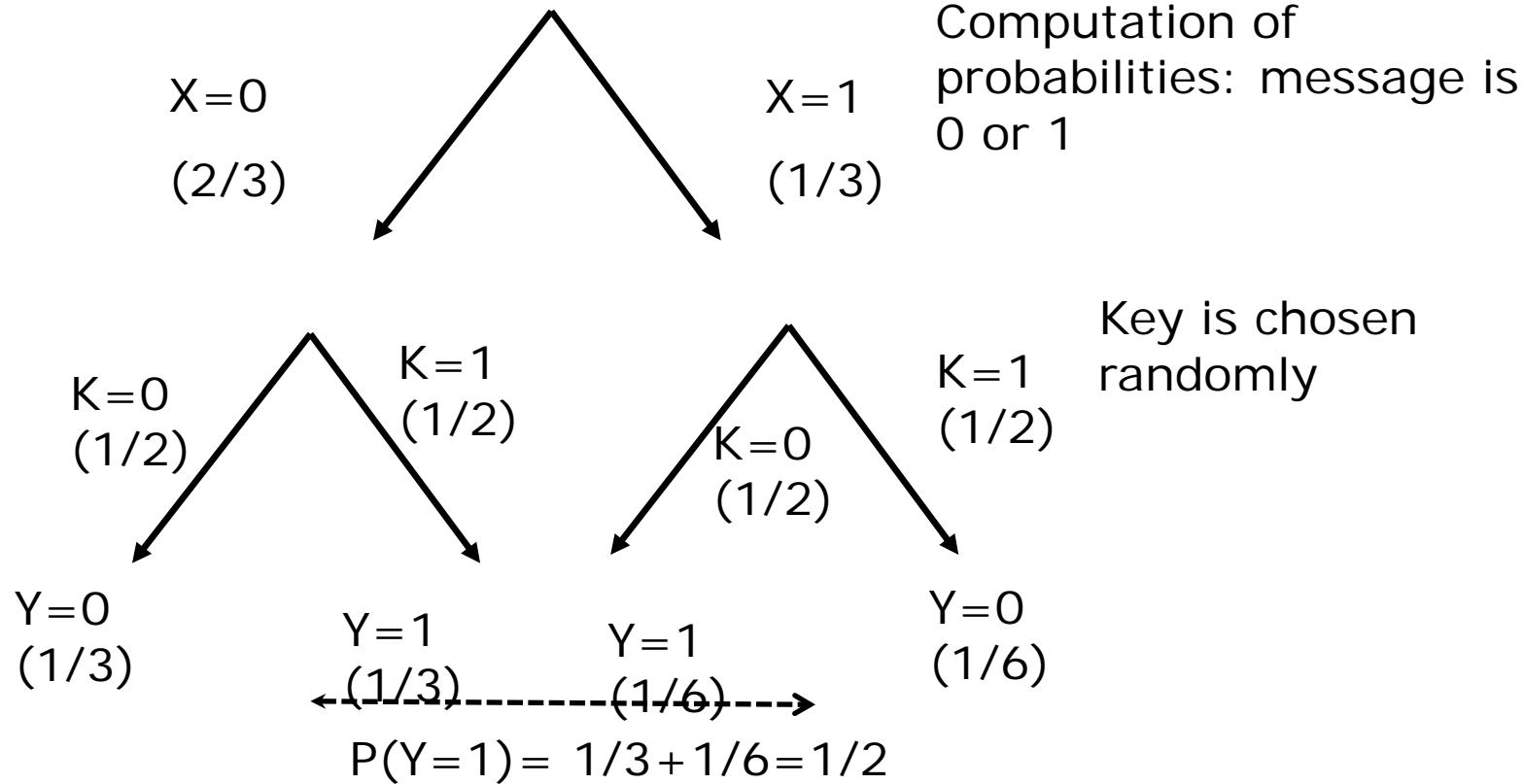
Perfect Secrecy

- Let x : input, y : output
- Perfect Security implies: $P_{X|Y}(x|y) = P_X(x)$
- The one-time pad offers perfect secrecy. Let us make it more precise what this means.
- Let us assume that the message space is binary (0 or 1) and key space is also binary. Assume that A chooses message 0 quarter of the time, i.e Probability that the message is 0 is equal to $1/4$, $P(M=0) = 1/4$. Perfect secrecy means knowing this fact, any adversary (E) should not get more information by observing the cipher message ($C = M \oplus K$). i.e. The condition probability, $P(M = 0 | C = 1)$ should not be different from apriori probability $P(M=0)$.
- This means that seeing the cipher text C does not increase the adversary's knowledge about the message

Another example

- Let message space be 0 or 1, i.e $X = 0$ or 1.
- Assume that the Adversary a priori knows that probability that ($X = 0$) is $2/3$.
- i.e, $P(X=0) = 2/3$, then $P(X=1) = 1/3$.
- Suppose $Y = 1$ was observed at the output of the cipher.
- We want to prove $P(X=0|Y=1) = P(X=0)$.
- **This equivalent to : Seeing the cipher text does not increase the adversaries knowledge about the underlying message.**

Graph of one bit encryption



$$P(X=0|Y=1) = P(X=0 \wedge Y=1) / P(Y=1) = ((2/3)(1/2)) / (1/2) = 2/3 = P(X=0)$$

General Result



- When X and Y are long sequences of 1's and 0's of length n.
- Theorem: $P(X=m|Y=c) = P(X=m)$.
- Proof depends critically on the fact that K is generated according to uniform distribution,
- i.e, $P(K=k_1) = 1/2^n$,

Implications

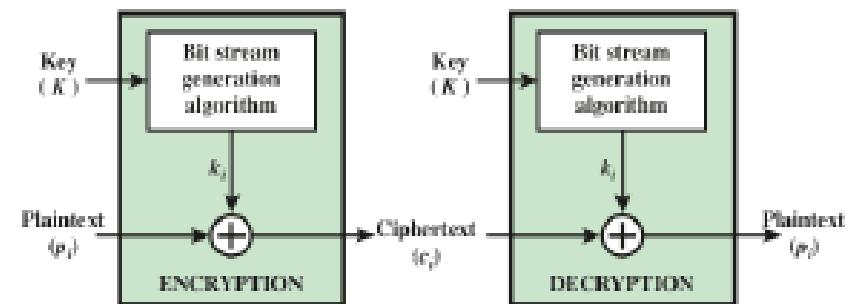
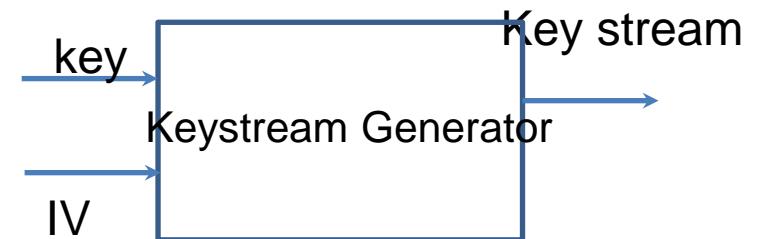
- In practice, messages may be biased; could be observed by the adversaries.
- Requirement: Encryption transformation should distribute messages to cipher space fairly uniformly irrespective of known apriory statistics of the messages.
- One-time pad analysis tells us that if we choose a random secret key pad at least the size as the message, we can achieve the perfect secrecy.
- Basically, the random key, which is as long as the message, hides the message completely leading to the perfect “confusion” to the adversary by perfectly “diffusing” the statistical structure of the plain text to the entire ciphertext.
- However, one-time pad is not practical.

Two-time pad is Dangerous

- One-time pad is not practical. It demands a key as long as the message.
- What happens if we reuse the one-time pad used in the encryption?
- $C_1 = M_1 \oplus K; C_2 = M_2 \oplus K$; then
- $C_1 \oplus C_2 = M_1 \oplus M_2 \oplus K \oplus K = M_1 \oplus M_2$.
- Even though $M_1 \oplus M_2$ may not direct meaning, it still leaks information about both M_1 and M_2 . Also, in a cryptanalysis setting if one of the messages M_1 or M_2 is available to the adversary, then he/she can get the other.
- This attack implies that you need a new key for every message.
- The idea is used in attacking Vigenere cipher (same key-pad is added many times).
- This type of analysis helped Allied in World Wars in 20th Century. Germans made this mistake in the war times!. Turing led Allied team made use of such vulnerability during initial key broadcast by Germans, which eventually helped to crack the master key used for the day.

Stream Ciphers

- How do we define a practically useful One-Time pads?
- An idea is to generate a long stream based on a short key and use it as a keystream in One-Time pad scheme. The resulting cipher is “stream Cipher”.
- Stream cipher in general takes a key and a random nonce (Initial Vector(IV)) as input and outputs a keystream of arbitrary size. The keystream is then xored with the plaintext to obtain a ciphertext.



Modified From: Stallings Figure 4.1a:

Modern Stream Ciphers

- Stream ciphers are extensively employed in modern communication networks.
- They are of the algorithm of choice in Light Weight Cryptographic applications.
- eSTREAM: ECRYPT Stream Cipher Project: An European stream cipher project in the last decade gave impetus to the development of the subject.
- They are every where: BlueTooth, Phones, browsers etc.
- We will revisit this idea when we study Block Ciphers in Stream Cipher mode.

1.3 Modern Block Ciphers

COMP90043

Lecture 1

Block Ciphers



- Encrypts blocks of n characters/bits of plain text simultaneously outputting blocks of cipher texts.
- Same key is used for many different message blocks.
- Fundamental building blocks for many cryptographical functions.
- Examples include hash functions, pseudorandom generators, message authentication codes etc.
- **Confusion and diffusion principles:**
 - **Diffusion** dissipates statistical structure of plaintext over bulk of ciphertext.
 - **Confusion** makes relationship between ciphertext and key as complex as possible.
 - Generally diffusion is created by permutations and confusion is created by substitution.

Product Ciphers and Fiestel Ciphers

- A **product cipher** combines two or more transformations so that resulting cipher is more secure than the individual components by making use of confusion and diffusion principles.
- A **substitution-permutation cipher** is a product cipher made up of number of stages each involving substitution and permutation. The operations of substitution and permutation are responsible for effecting the confusion and diffusion respectively.
- An **iterated block cipher** is a block cipher involving sequential repetition of an iterated function called a round function.
- The parameters of iterated block ciphers are r : number of rounds; n : block length; k : bit-size of key, K from which r subkeys (round keys) k_i 's are derived.
- **Fiestel Cipher** is an example of an iterated block cipher.

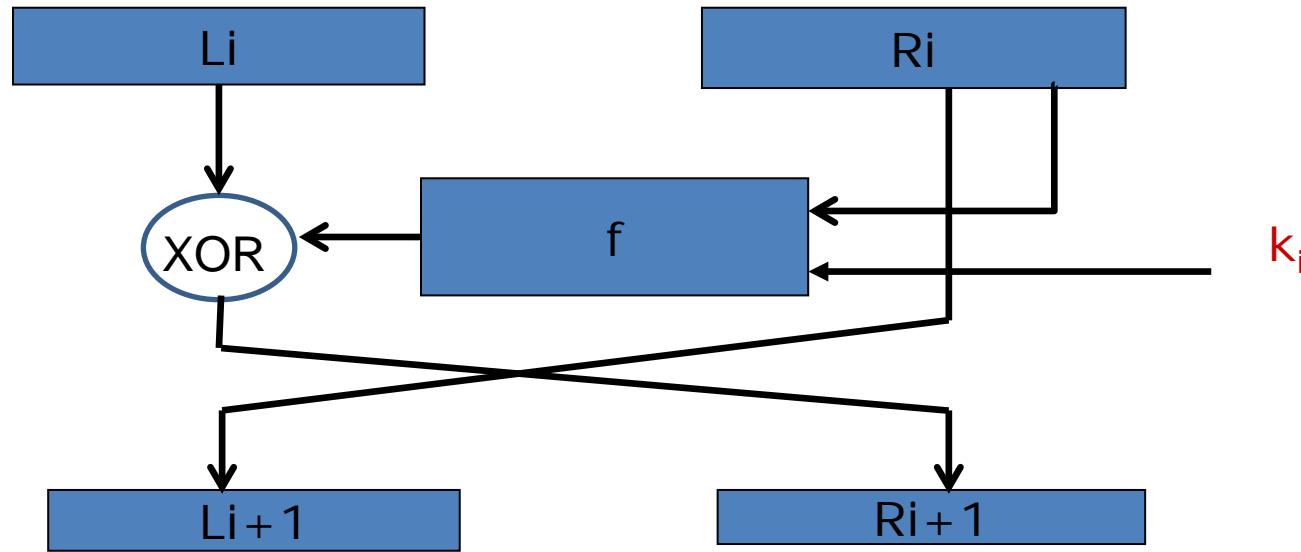
Fiestel Block Cipher

- **Fiestel** ciphers are iterative ciphers; they repeat a given operation several times in rounds.
- Each round will have the following distinct operations:
- **Substitution:** Each plaintext bit or group of bits in a block are replaced with a corresponding cipher text bit or group of bits.
- **Permutation:** A certain perimutation is effected to each transformed ciphertext bits.
- The above round operations are repeated certain number of rounds.

Fiestel Block Cipher, cont.

For such a cipher, the input key is used to produce round keys k_1, k_2, \dots, k_r . The message is initially divided into two parts, namely left and right halves, L and R.

For each of r rounds, the following operations are executed.



After r rounds, the final left and right halves are swapped and concatenated to form the cipher text.

The design of a good function f is partly ``ART'' and partly ``SCIENCE''.

Data Encryption Standard (DES)

- IBM's 1974 submission for a standard.
A Fiestel cipher
Block size: $n = 64$,
keysize = $k = 56$ bits.

The key is specified with 64 bits containing 8 bits of parity.
Number of rounds = 16.

Strengthening DES:

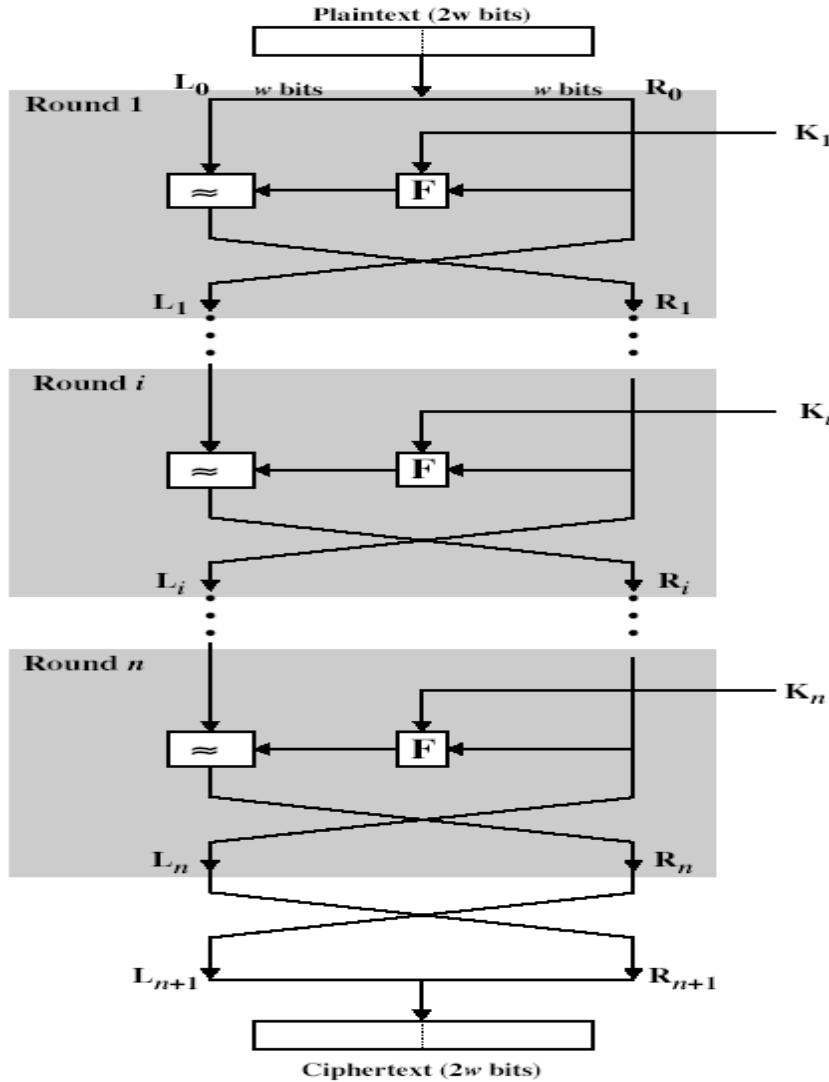
DESX: Apart from 56 bit key K, choose two new 64 bit keys K_I and K_O, then we encrypt

$$C = K_O \oplus \text{DES}(K, M \oplus K_I)$$

This method increases effective key length to $199-t$, where t is a quantity related to adversaries' cryptanalytic assumptions where the adversary is able to collect 2^t matching input-output pairs.

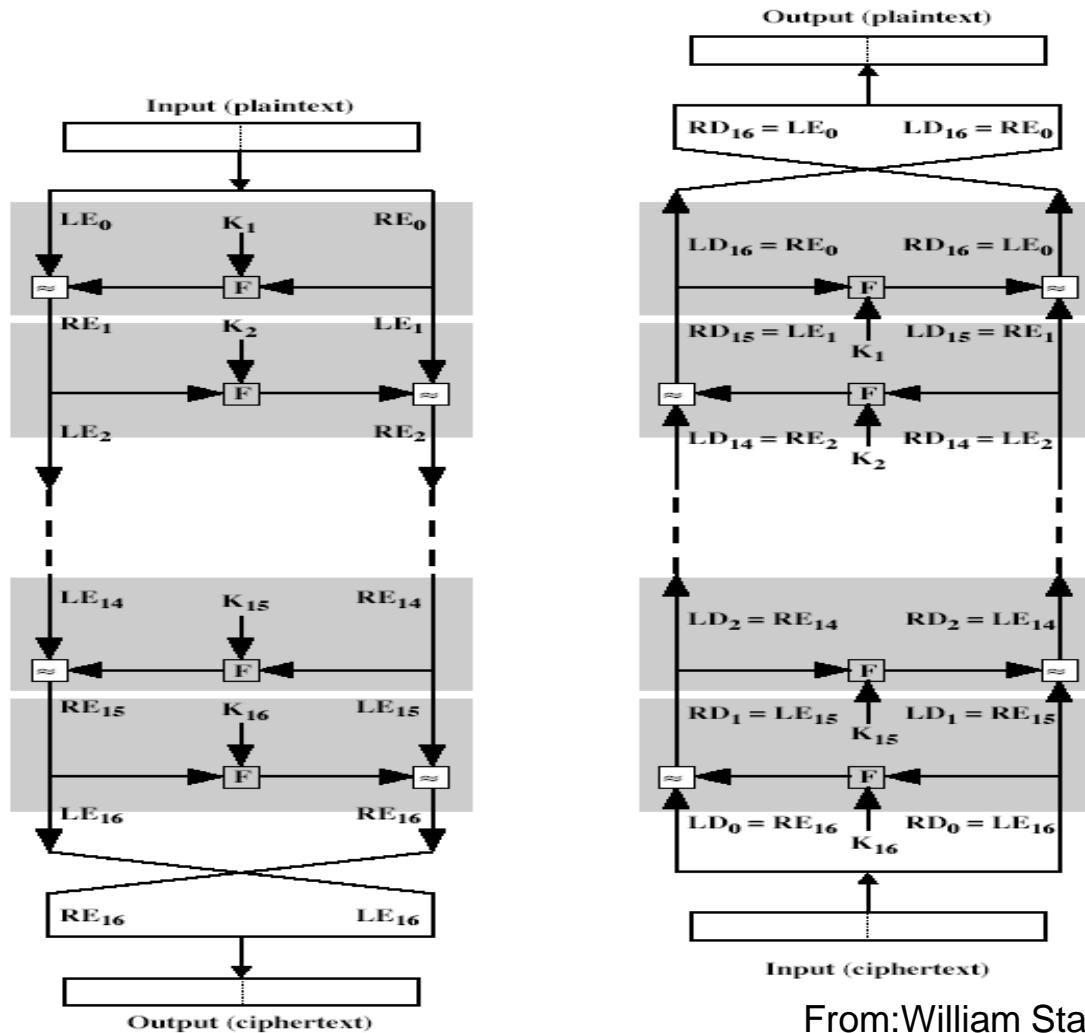
- Read the textbook for more details on DES.

Feistel Cipher Structure



From: William Stallings 5th Edition:

Feistel Cipher Decryption



From: William Stallings 5th Edition:

Strengths, properties and attacks on DES

- Each bit of cipher text depends on all bits of the key and all bits of the plain text.
- No statistical relationship between plain and cipher visible.
- Altering a key bit or a plain text bit should alter each cipher bit with probability close to half.
- Altering a cipher bit should result in unpredictable change in plain text block.

Cryptanalysis of DES

- Empirically it is found that DES is safe.
- Exhaustive search -- Brute force. 2^{56} computations.

Differential cryptanalysis

- Chosen plain text attack,
- Not realistic -complexity 2^{47} computations.

Linear cryptanalysis

- Complexity : 2^{43} computations.
- The main drawback is limited key space.
- The new standard for encryption now is AES which has key space $\geq 2^{128}$.

Advanced Encryption Standard (AES)

- DES is not recommended as it has small key space and have known theoretical attacks.
- Financial Systems still use a modification of DES such as Triple-DES, which also has significant drawbacks (Slow and have small block size)
- So, NIST worked with crypto community to develop to develop an Advanced Encryption Standard (AES) in 1997.
- In October 2000, NIST accepted Rijndael as the AES in Oct-2000.
- It is proposed by cryptographic researchers: Dr. Joan Daemen and Dr. Vincent Rijmen.

AES Algorithm

- Stallings discusses AES algorithm in detail.
- It is not a Fiestel cipher, but still iterative.
- Main design requirements:
 - Should withstand all known attacks
 - It should have flexible implementation, to be able to run on varieties of platforms and CPUs.
 - It should have a simple design features.



How do you make Encryption more complex?

- One can increase block size n and also look for different functions for encryption.
- In practice, data comes in many forms. We can modify the function for different modes.
- These practical modes are developed by people working on using encryption. More on Chapter 7 of the textbook

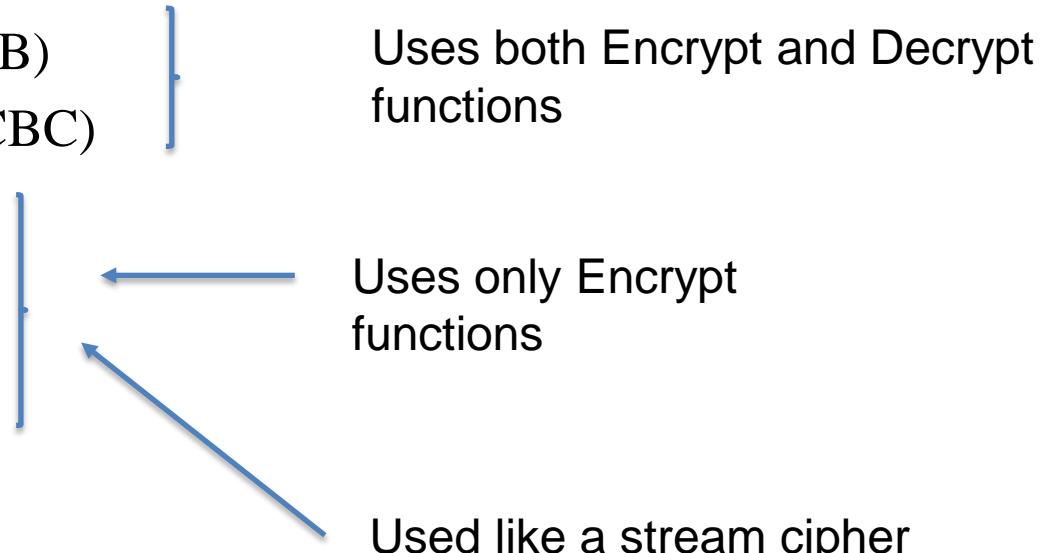
1.4 Modes of Block Ciphers

COMP90043

Lecture 1

Modes of Operations

- NIST defined five basic modes of usage of block cipher.
- They are generic: can be used with any block cipher.
- Five modes:
 - Electronic Codebook (ECB)
 - Cipher Block Chaining (CBC)
 - Cipher Feedback (CFB)
 - Output Feedback (OFB)
 - Counter (CTR)



Mode of Operations

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of plaintext bits is encoded independently using the same key.	<ul style="list-style-type: none"> 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.	<ul style="list-style-type: none"> 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.	<ul style="list-style-type: none"> 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.	<ul style="list-style-type: none"> 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.	<ul style="list-style-type: none"> General-purpose block-oriented transmission Useful for high-speed requirements

From: Stallings Table 7.1:

You will learn more from the textbook.

Week 3



Lecture 1

Modern Symmetric key Ciphers

Lecture 2

Finite Field mathematics,

Workshop 3: Workshop based on Lectures in Week2

Quiz 3

Week 4

Lecture 1

Recap of Week 3 lectures and workshops

Modes of Encryption Chapter 7

DES EXAMPLES

Public Key Cryptography: Diffie-Hellman Protocol and RSA

Lecture 2

Proof of RSA Encryption + Chinese Remainder Theorem, Continued from Week 3 Lecture 2

Workshop 3: Workshop based on Lectures in Week 3

Quiz 4

Public Key Cryptography: Diffie-Hellman Protocol and RSA

COMP90043

Lecture 1

Lecture 1

- 1.1 Concept of Public Key
 - Limitations of Symmetric key system
 - Notations for Public key
- 1.2 Diffie-Hellman Protocol
 - Motivation
 - The protocol and Implications
 - Man in the Middle Attack
- 1.3 RSA Idea
 - Informal Idea
 - RSA Algorithm
 - Attacks on RSA

Recap from Week 1-3

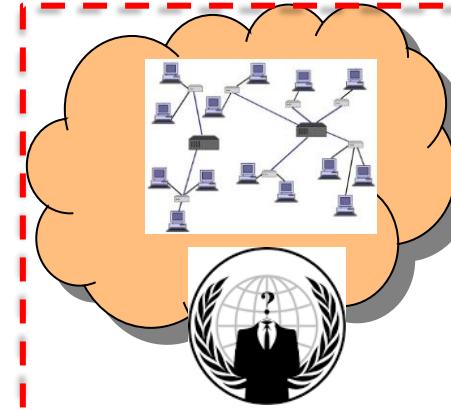
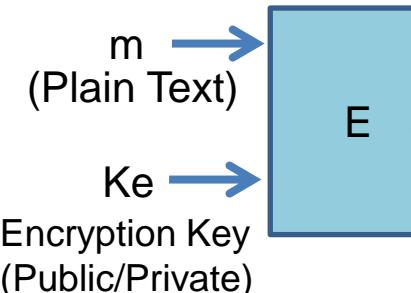
- First we will revisit some of the concepts we dealt in Week 1-3.

Story of Alice and Bob terms and notations



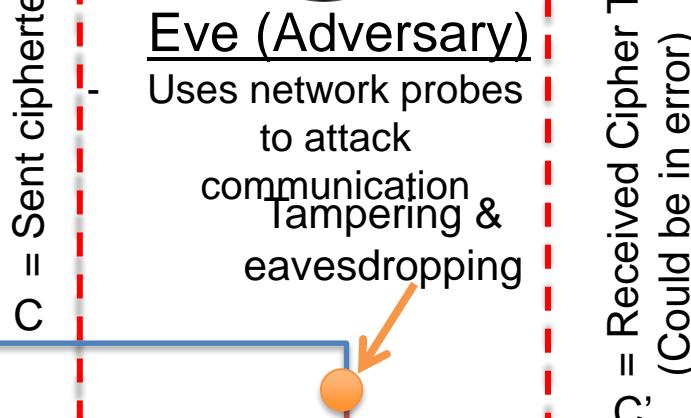
Alice (Sender)

- Uses an Encryption Function (E)



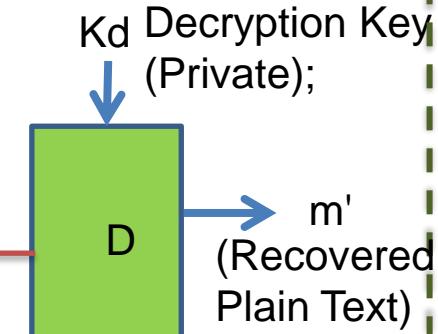
Eve (Adversary)

- Uses network probes to attack
- communication
- Tampering & eavesdropping



Bob (Receiver)

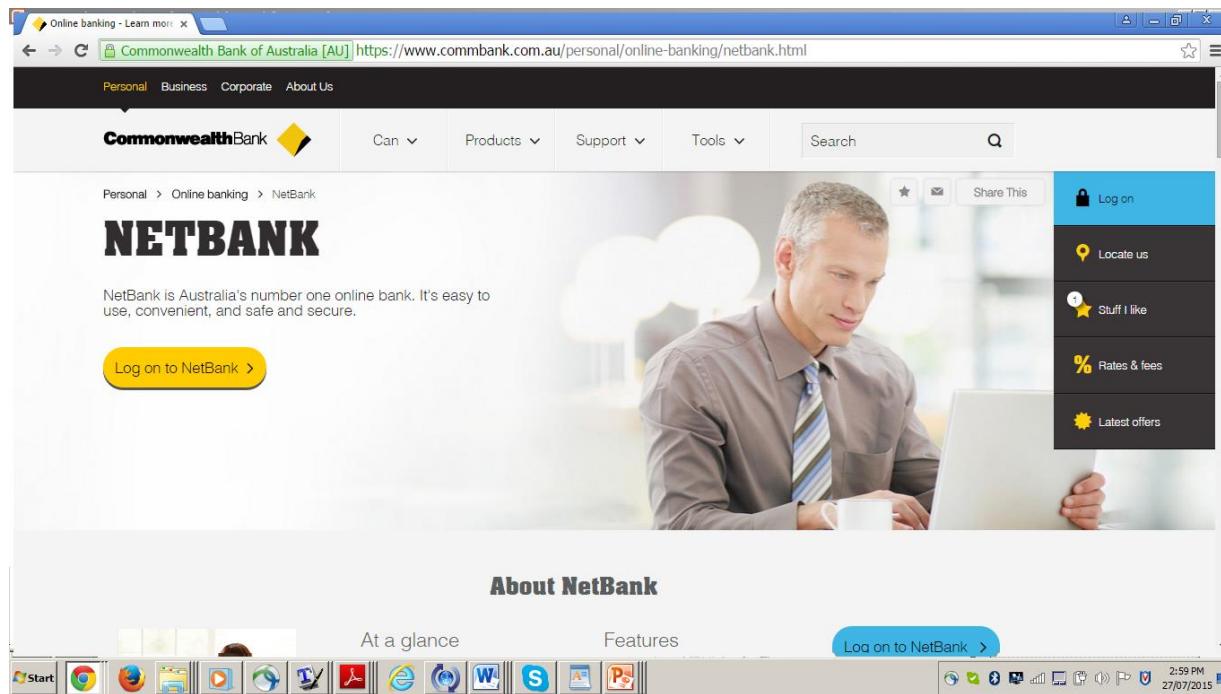
- Uses a Decryption Function (D)



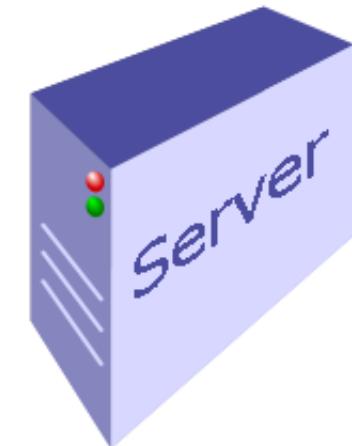
E, D are public; c is the ciphertext, c' is received ciphertext; ideally $m=m'$;

Cryptography involves many conceptual ideas, we look at the basic functions

Recap Motivating examples



Comm bank Server

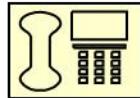


Issues in getting your money from the bank.
Should work over Internet
Think, who is Alice, Bob and Eve here.
What tools Cryptography can provide here?

A protocol



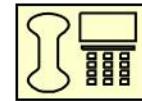
Alice



Choose a random x and compute $f(x)$

Dating Problem!

EVEN: HEADS
ODD: TAILS



Bob

Guesses x is even or Odd

Send x

Verify $x = f(x)$
check if his guess is correct or not

Whoever wins the game decides the venue of the meeting!

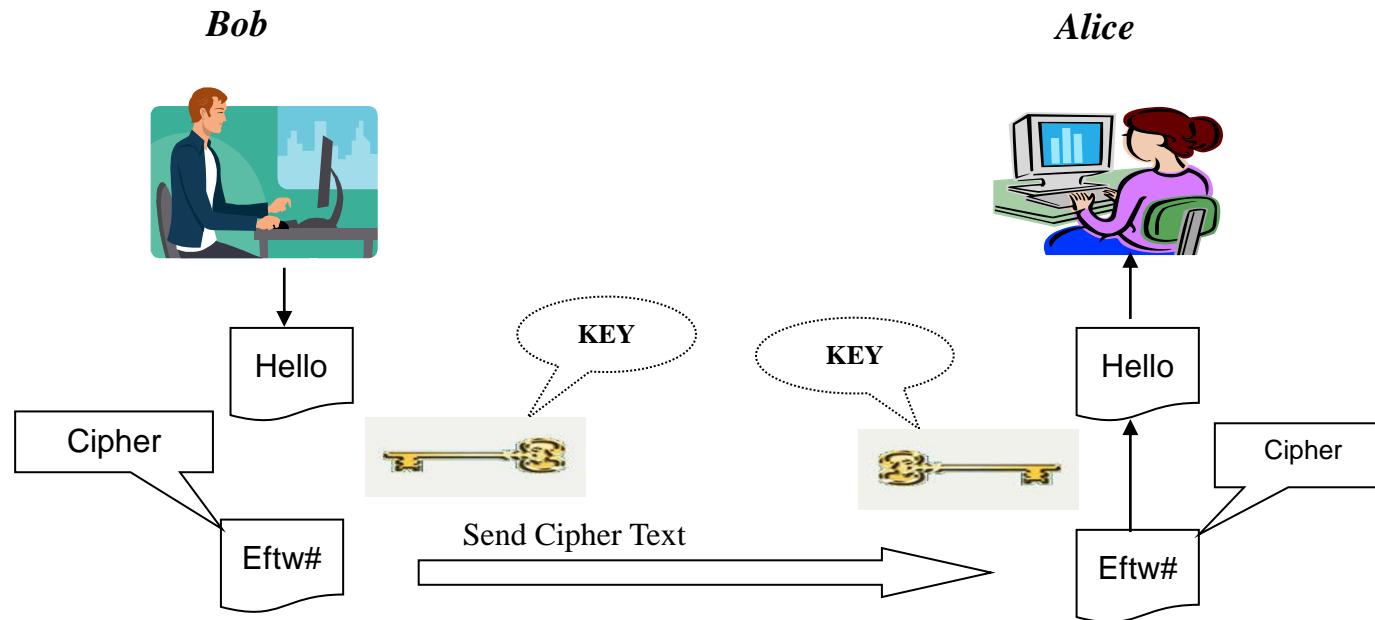
Is this protocol correct and fair (unbiased)?
Can you modify so that both Alice and Bob

1.1 Concept of Public Key

COMP90043

Lecture 1

Limitations of Symmetric Key Systems



- Symmetric key is fast and provides in built in Authentication by virtue of users sharing the key.
- Sharing the key is a huge problem.
- They definitely provide confidentiality, but never **protect** against each other.

Disadvantages of Symmetric key Systems

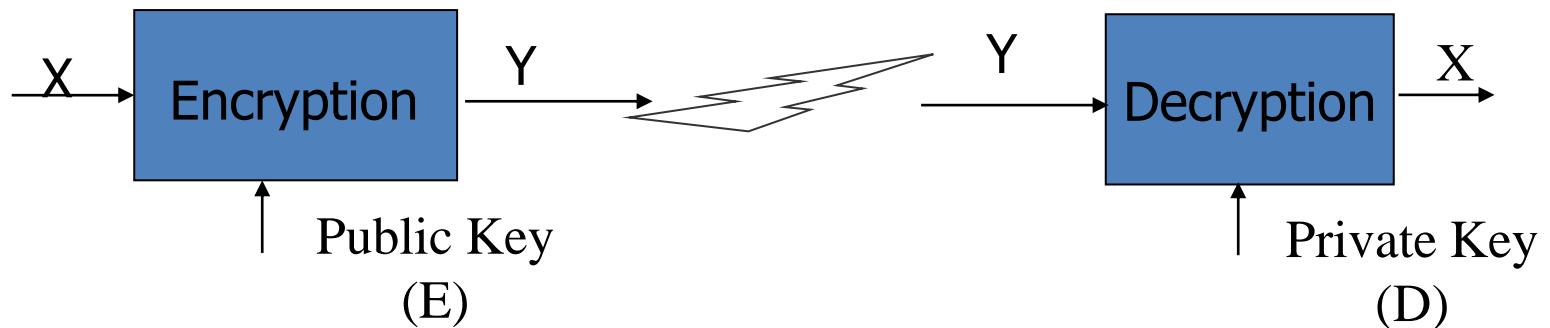
- One key is used for both encryption and decryption.
- Further the key need be shared by both sender and receiver.
- If the key is disclosed, the scheme is compromised.
- Non-repudiation is impossible as sender and receiver are equal. One party can forge other party's data. Hence it does not protect the sender from a receiver forging a message and then claiming that it is sent by the sender.
- In networked situation, the requirement for the key storage grows quadratic in n , the numbers of users. The number of common keys is $n(n-1)/2$.

Birth of Public Key Cryptography

- We discussed about Diffie-Hellman protocol in the Introduction lecture. Many consider this development a historical significant. Why?
- Symmetric key system may seem secure, but if keys are compromised, fails completely.
- Another problem is about authentication, can we have useful equivalent of hand written signatures for electronic transactions? We will discuss this concept later in the course, but public key cryptography achieves this property efficiently.
- Please read the original paper by Diffie-Hellman.

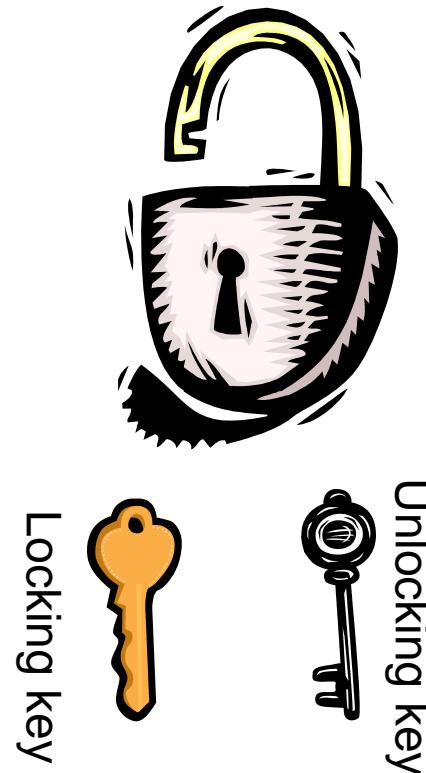
Asymmetric Cryptography

- Communication parties are not equal, a precondition for having accountability for their inputs into the conversation.
- Uses two keys; a public and a private key.
- From Shannon's analysis of perfect cipher: Encryption transformation should distribute messages to cipher space fairly uniformly. Diffie-Hellman gave a concrete realization of this property without using any secret. This heralded the birth of **public key cryptography**.



Asymmetric Cryptography, Continued

- Modern cryptography; The paper of Diffie-Hellman in 1976 (December 1975 to be precise).
- In a networked situation, the requirement for the key storage grows linearly in n , the number of users.
- Uses two keys; a public and a private key.
- Non-Repudiation is possible, leading to natural accountability to the transactions.
- Mechanisms differ from the way you lock and unlock.
- Eg: Secure staff mail box in the department office



Picture from General Internet Resources

The Figure Illustrates the notations
And use of Public Key functions;
We will use this notation
throughout this semester.

Public key of B : PU_b
Private key of B : PR_b

Encryption and Decryption by A

$$\downarrow$$

$$Y = E(PU_b, X)$$

$$X := D(PR_b, Y)$$

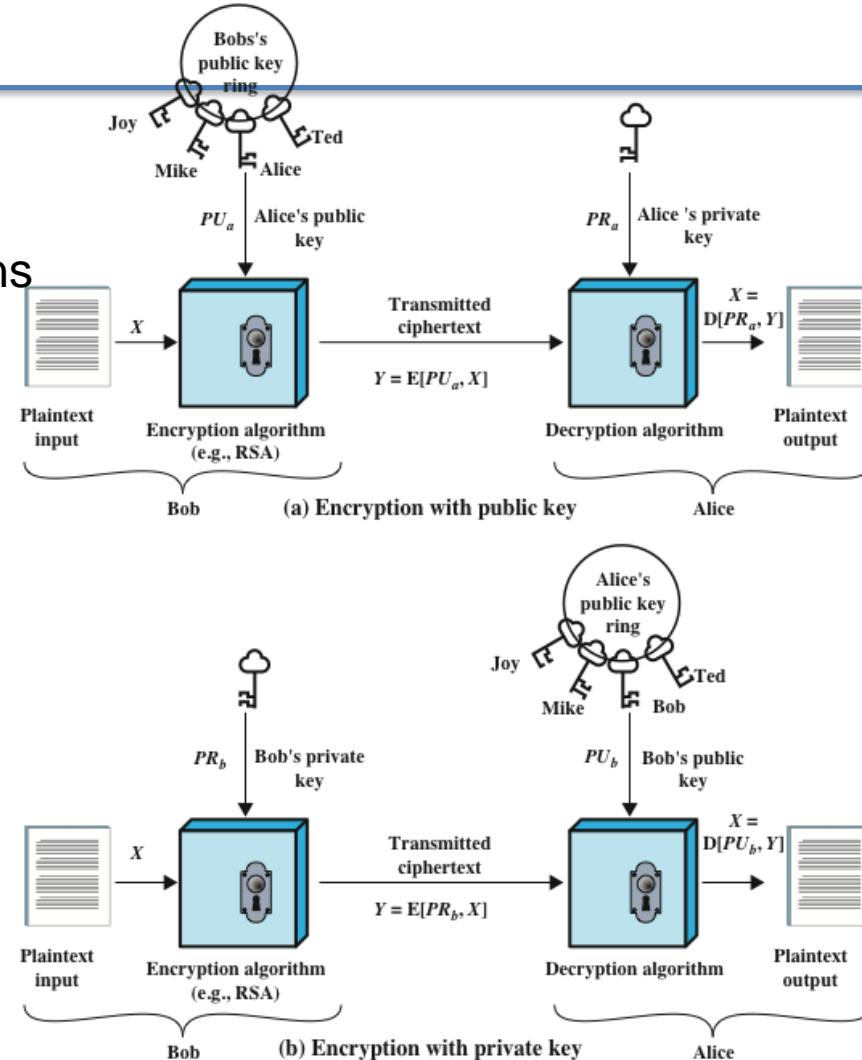
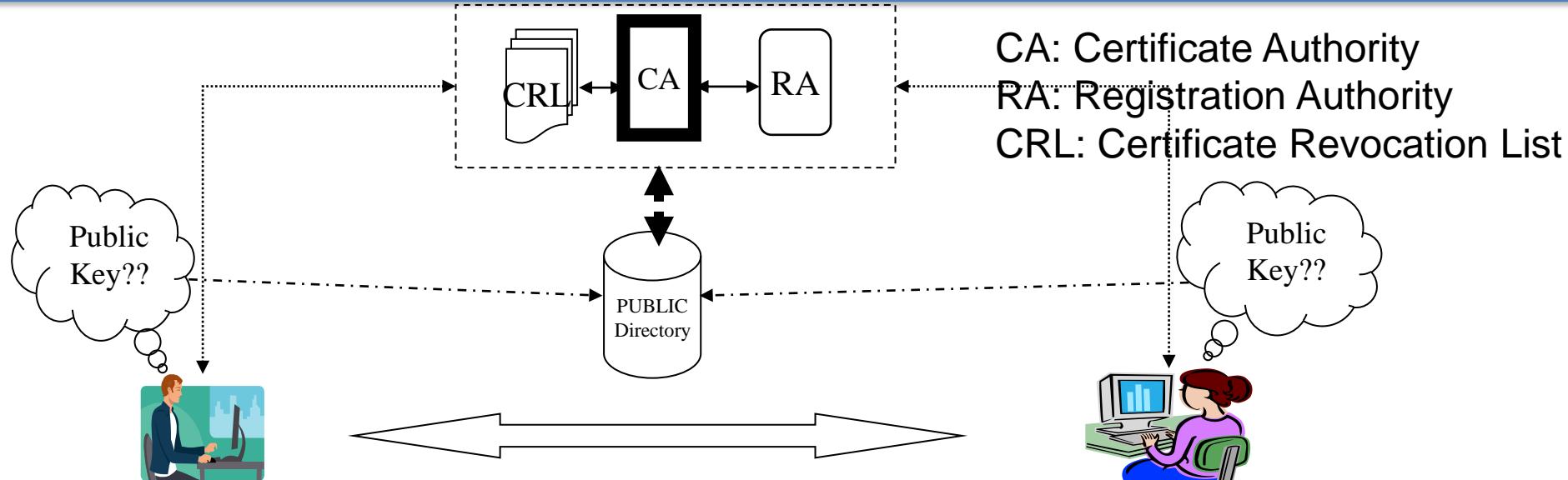


Figure 9.1 Public-Key Cryptography

Table 9.2 from the textbook

Conventional Encryption	Public-Key Encryption
<p><i>Needed to Work:</i></p> <ol style="list-style-type: none"> 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. <p><i>Needed for Security:</i></p> <ol style="list-style-type: none"> 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. <p>From Stalling's Textbook</p>	<p><i>Needed to Work:</i></p> <ol style="list-style-type: none"> 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). <p><i>Needed for Security:</i></p> <ol style="list-style-type: none"> 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.

Traditional or Conventional PKC



- How will Bob locate Alice's Public Key?.
- If he trusts a directory, then he could directly read from it, similar to one gets telephone numbers from White Pages. (White pages now has moved to online)
- An adversary can compromise the public directory. How will you handle such attacks?
- In practice, we need a elaborate arrangement of Public-Key Infrastructure, we will study later (In Chapter 14).

Typical Uses of Public Key Encryption

Encryption:

- Generally, it involves the use of two keys:
 - A public-key, which may be known by anybody and can be used to encrypt messages.
 - A private-key, known only to the recipient, used to decrypt messages.

Signatures

- Generally, it involves the use of two keys:
 - A private-key, known only to the signer is used to sign messages.
 - A public-key, which may be known by anybody and can be used to verify messages.
- The above methods are asymmetric, because those who encrypt messages or verify signatures cannot decrypt messages or create signatures.

1.2 Diffie-Hellman Public Key Protocol

COMP90043

Lecture 1

Idea

- We gave a general introduction and motivation for the subject in the first week.
- Recall that the main difficulty with symmetric key scheme is that the key management is going to be hard.
- Public key promises to simplify the key management-any two users who have not met before still be able to obtain a common secret using only Public information.
- However, public key systems also bring in new key management issues-will require a trusted system to distribute public keys. We will study this later.

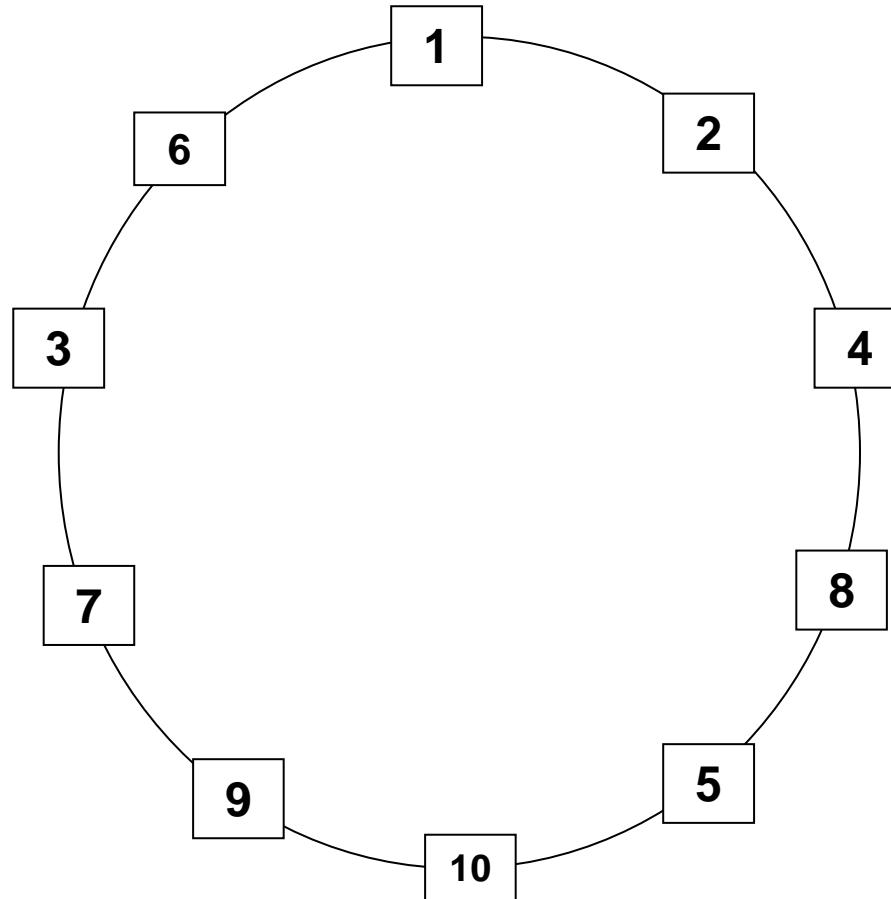
One Way Function

- Let f be a function defined over integers modulo a large number (can be a prime or product of two large primes)
- Computing $f(x)=y$ given ‘ x ’ is easy;
- Given $y=f(x)$, computing ‘ x ’ from ‘ y ’ is difficult or hard
- Issues :How hard?
 - Generally the best-known algorithm for inverting the function is sub-exponential in number of bits used to represent the elements in function domain or range.

Discrete Logarithm Problem

- Let ‘g’ and ‘h’ be elements of the group G. Then discrete logarithm (DL) problem is the problem of finding ‘x’ such that $g^x = h$.
 - For example, the solution to the problem
 - $3^x \equiv 13 \pmod{17}$ is 4, because
 - $3^4 \equiv 81 \equiv 13 \pmod{17}$.
- The discrete log problem is believed to be difficult. Therefore it has become the basis of several public key schemes, for example: El-Gamal.

An example



Example of a Cyclic group modulo $p = 11$

g : generator = 2

15/08/2022

Order(size) of G = 10

g^i	$g^i \text{ mod } p$	$D\log(g^i)$
2^1	2	1
2^2	4	2
2^3	8	3
2^4	5	4
2^5	10	5
2^6	9	6
2^7	7	7
2^8	3	8
2^9	6	9
2^{10}	1	10

Example mod 11

X	$2^x \text{ mod } 11$	$3^x \text{ mod } 11$
0	1	1
1	2	3
2	4	9
3	8	5
4	5	4
5	10 Or -1	1
6	9	3
7	7	9
8	3	5
9	6	4
10	1	1
11	2	3

- 2 is a primitive element.
- 3 is not a primitive element
- Given any power of 2, the exponent can be obtained from reading the corresponding index in the table
- In practice a large modulus is used and hence finding the exponent is difficult. This is one of the important one way functions used in modern cryptography.
- In general finding primitive element is also an interesting problem. We use the groups where we can easily find generating elements.

Diffie-Hellman Protocol

- Alice
- Choose $N_a=2$
- $g^{N_a} = 2^2=4 = M_a$

Bob
Choose $N_b=6$



-
- Compute
- $K_{ab} = M_b^{N_a}$
- $=9^2=4$

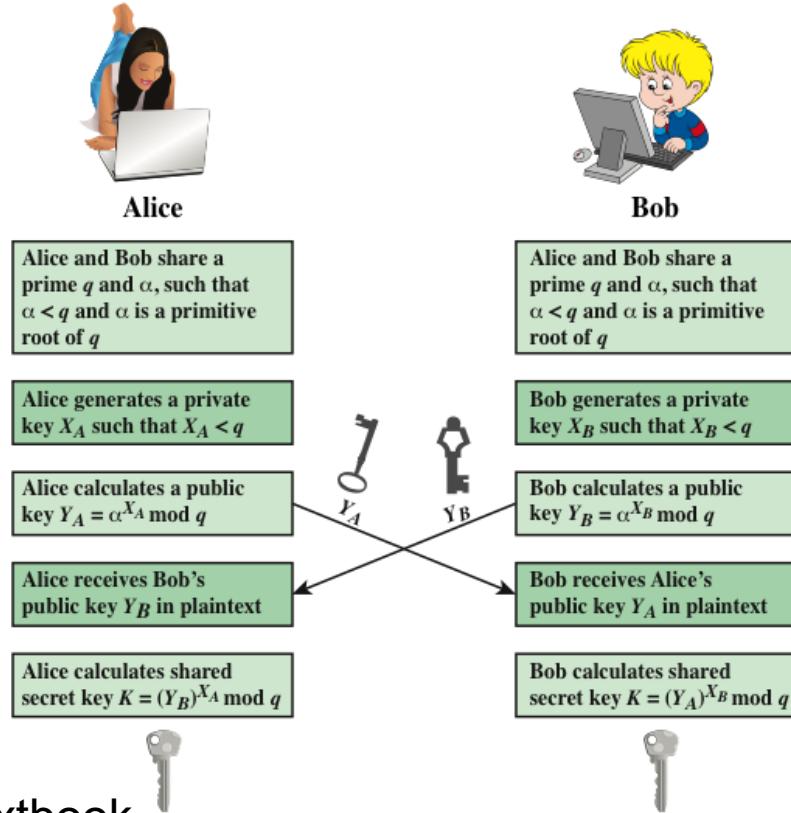
$g^{N_b} = 2^6=9=M_b$



-
-
- $K_{ab} = K_{ba}=4$

Compute
 $K_{ba} = M_a^{N_b} = 4^6=4$

Diffie-Hellman Protocol



From Stalling's textbook

Figure 10.1 Diffie-Hellman Key Exchange

Computational DH problem

- Let G be a cyclic group of size q and g be a generator of the group G .
- Given g^a and g^b , two arbitrary elements of the group G for some integers a and b in the range of $0 \leq a, b \leq q$, then find

$$g^{ab}$$

Normally G is a **multiplicative** group in a suitable **finite field**.

If someone comes with an efficient algorithm for this problem, the protocol is broken!

- Clearly a solution to DLOG implies a solution to DH.
- Is the converse true?
- This is one of the open problems.

Problems with DH Key Exchange

- The protocol has a new problem,
- When Alice and Bob exchanging information, how do they know that they are indeed talking to the right individuals?
- With the nature of Internet, someone could masquerade as Alice or Bob and try to fool Bob or Alice. This is because, the public information they exchange is not authenticated.
- The protocol is vulnerable to Man in the Middle Attack.

Man in the middle Attack

- Alice
- Choose N_a
- g^{N_a}

Malice

Bob

Choose N_m g^{N_m} Choose N_b Gets g^{N_m} Computes $(g^{N_m})^{N_a}$ Computes $(g^{N_m})^{N_b}$ Malice shares $k_1 = g^{(N_m N_a)}$ with AliceMalice shares $k_2 = g^{(N_m N_b)}$ with Bob

Also study the version of the
Protocol in the textbook

How do we overcome the MITM Attack

- Main reason for the attack is because of lack of authentication.
- We need Digital Signatures and related concept to tackle this attack.
- We will study this later in the subject.

1.3 RSA Crypto System

COMP90043

Lecture 1

RSA

The first Public key
encryption algorithm



Ron Rivest, Adi Shamir and Leonard Adleman

Based on the assumption that factoring an integer which has an alleged factorization as a product of two prime numbers is a hard problem;

In other words, given an integer n which is constructed by two secret primes p and q , finding the factors is a hard problem.

Message and cipher text belong to Z_n

How to construct a crypto system using the above hard problem?

Basic Facts Again

Definition: A cryptosystem is a five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where the following conditions are satisfied:

\mathcal{P} : a set of possible plaintexts;

\mathcal{C} a set of possible ciphertexts;

\mathcal{K} , the space of keys, a finite set of possible keys;

For each k in \mathcal{K} , there is an encryption rule e_k in \mathcal{E} and a corresponding decryption rule d_k in \mathcal{D} . Each

- $e_k: \mathcal{P} \rightarrow \mathcal{C}$ and $d_k: \mathcal{C} \rightarrow \mathcal{P}$

are functions such that

- $d_k(e_k(x)) = x$ for every plaintext x in \mathcal{P} .

How do we create public key encryption?

- Let us try to recreate questions that came to the creators of RSA encryption.
- Encryption function should be publicly available, eg. from a directory.
- Anyone should be able to encrypt: We need a **one way** function f .
- Only the designated user should be able to decrypt
 - The user needs to invert f somehow – f is one-way.
 - idea is to create a **trapdoor** function which should enable to get the encrypted message.
 - Any public key encryption should have the above features.

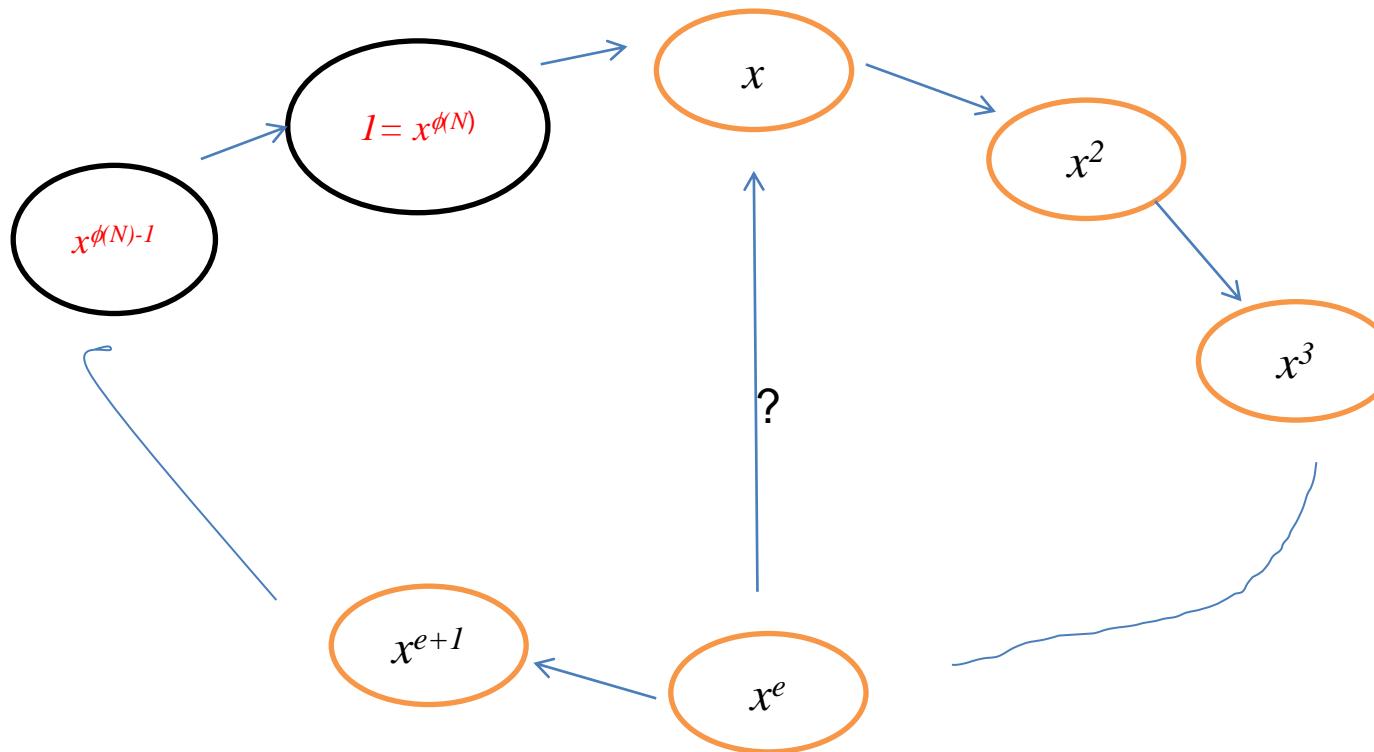
New One way functions

- We looked at Discrete Logarithm problem before.
- We considered a cyclic group G with a generator g
 - Let G be multiplicative group of a large order q with a generator g ,
$$G = \{g^0=1, g^1, \dots g^q=1\}.$$
 - Given a random t in G , computing g^t is easy.
 - However, given an arbitrary y in G , it is computationally hard to obtain Discrete Log of y ; i.e it is hard to find t such that $g^t = y$.
- Are there any other groups whose order could be secret!
- RSA is one such scheme.
- RSA relies on a group of numbers modulo n , which is a product of two large primes.
- I will explain this idea informally. We will also explain the idea from mathematical results that we have introduced.

RSA Idea

The basic RSA idea begins as follows:

- Alice claims that she knows the factorization of $n = pq$; p, q Large Primes.
- Currently it is impossible for anyone to get p, q from n : Factorization is a hard problem.
- Let us work with some random $x \text{ mod } n$.
- We will assume that $\gcd(x, n) = 1$.
- Consider the group generated by $x \text{ mod } n$.
- We can show that $x^{\phi(n)} = 1 \text{ mod } n$.
- Alice needs to create a public encryption function that anyone can encrypt, but only she can decrypt.

$x^{\phi(n)} = 1 \text{ mod } n : \text{how it works}$


The operations are in the group of numbers modulo n under multiplication

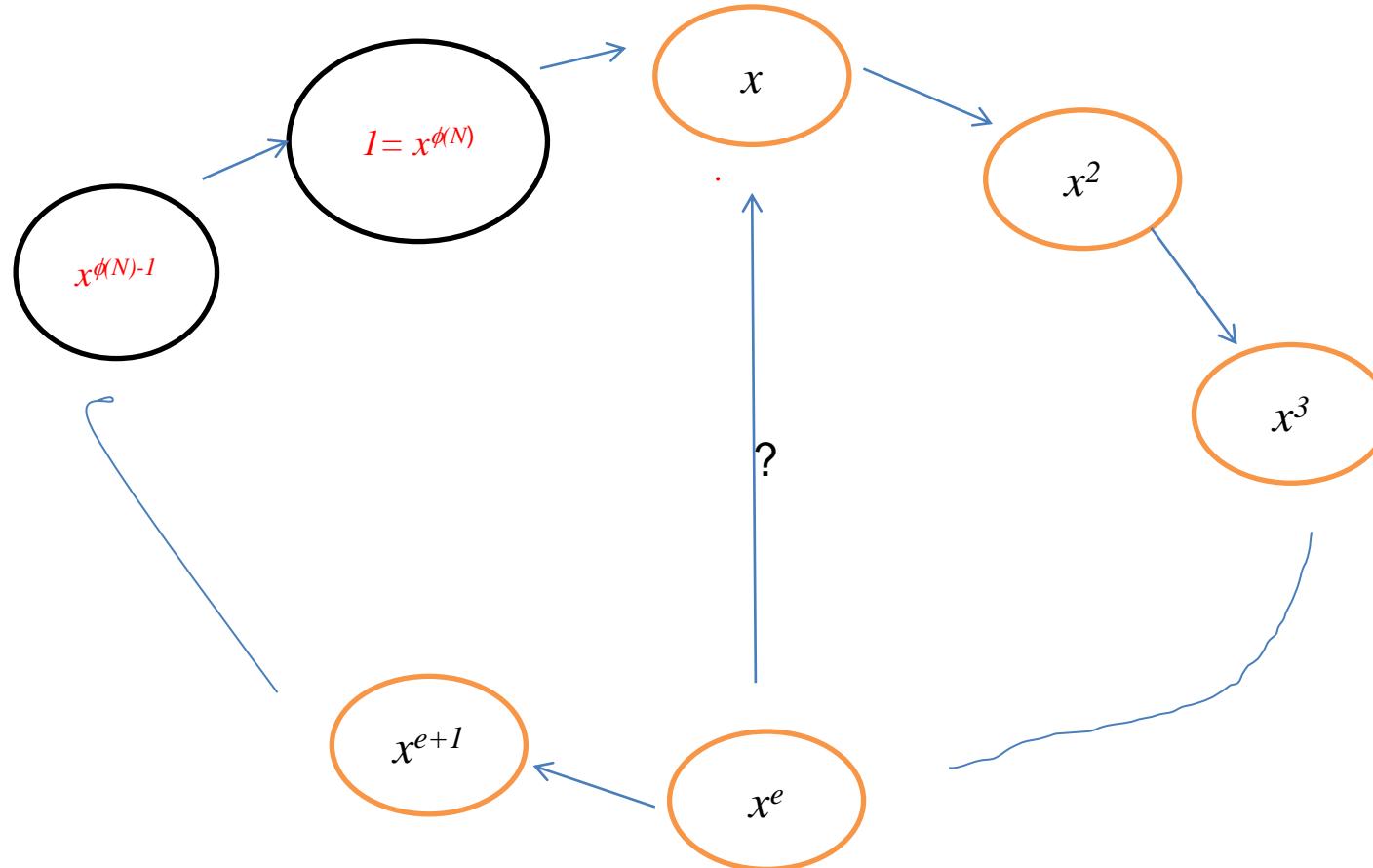
The order of the group is $\phi(n)$ = number of integers less than n and relatively prime to $n= (p-1)(q-1)$.

$$\begin{aligned}
 n - p - q + 1 \\
 &= pq - p - q + 1 \\
 &= (p-1)(q-1)
 \end{aligned}$$

RSA Idea Cont

- Alice will choose e a random number between 1 and $\phi(n)$ and make it public.
- So, Bob can take (e, n) and compute:
 - > $x^e \text{ mod } n$, as his encryption.
- No one else can work backwards from x^e to x because it is another hard problem-finding e^{th} root mod n (also known as RSA problem).
- But how does Alice recover x ?
 - > She will create a trapdoor as follows.
 - > She will compute d such that $e \times d \equiv 1 \text{ mod } \phi(n)$.
 - > $(x^e)^d \text{ mod } n = x$;

Why does it work? Alice has a trapdoor



The operations are in the group of numbers modulo n under multiplication

The order of the group is $\phi(n)$ = number of integers less than n and relatively prime to $n= (p-1)(q-1)$.

$$\begin{aligned}
 n - p - q + 1 \\
 &= pq - p - q + 1 \\
 &= (p-1)(q-1)
 \end{aligned}$$

RSA PKC (1978)

- Let $n = p \times q$; p, q are primes. Let the plain text and cipher text belong to integers modulo n and let (e, d) pair be computed such that

$$e \times d \equiv 1 \pmod{\phi(n)}$$

$(\phi$:Euler's totient function)

- For the RSA key parameter set $K = (n, p, q, e, d)$, define

$$E_k(x) = x^e \pmod{n}$$

And

$$D_k(y) = y^d \pmod{n},$$

where $(x, y \text{ in } Z_n)$. The values (n, e) are termed the **public key**, and the values p, q and d form the private key.

RSA Example

- Let $n = 91$; $p=13$, $q=7$ are primes. Let the plain text and cipher text belong to Z_{91} (residue Integers modulo 91). $\phi(n) = 12 \times 6 = 72$.
- For $K = (n=91, p=13, q=7, 5, 29)$, define

$$E_k(x) = x^e \bmod n$$

And

$$D_k(y) = y^d \bmod n,$$

- Verify $5 \times 29 = 145 \bmod 72 = 1$
- Message $x = 11$
- $E_k(11) = C = 11^5 = 72$
- $D_k(72) = 72^{29} = 11$

Real-World RSA

We only illustrated some toy examples so far.

Let us look at more realistic RSA parameters.

- RSA-768: a 768-bit RSA modulus with 232-digit decimal representation:

$n =$
1230186684530117755130494958384962720772853569595334792197322452151726
40050726
3657518745202199786469389956474942774063845925192557326303453731548268
50791702
6122142913461670429214311602221240479274737794080665351419597459856902
143413.

- RSA Laboratories had issued a challenge to factor the above modulus.

- In 2009, this was broken!*

$n = 3347807169895689878604416984821269081770479498371376856891$
 $2431388982883793878002287614711652531743087737814467999489 \times$
3674604366679959042824463379962795263227915816434308764267
6032283815739666511279233373417143396810270092798736308917

- The current key size on Internet for secure operations is greater than 4000 bits.

* Thorsten Kleinjung, et. al, **Factorization of a 768-Bit RSA Modulus.** [CRYPTO 2010: 333-350](#)

RSA is a encryption function

- You need to convince yourself that the RSA decryption function is a one way trapdoor function. If you know d , you can decrypt, otherwise it is impossible. We will prove this fact in the coming lecture.
- It is known that given $n, e, c = M^e \pmod{n}$, it is impossible to determine M . This problem is called RSA problem and also known as determining e^{th} root of c mod n . In general this problem is hard.
- If you determine d from only public parameters, then also you can break RSA. This problem can be solved if you can solve integer factorization problem.

Security of RSA

Brute Force attacks

Mathematical attacks

- Clearly, the security depends on the hardness of the **factorization problem**.
- If someone can obtain factors p or q, then they can find out $\phi(n)$ and can determine the decryption exponent itself.
- As a consequence of RSA encryption a new problem emerges called the **RSA problem**.
- It is stated as follows: Given $(n, e, c = M^e)$ determine e^{th} root of $c \text{ mod } n$.
- This problem is also considered to be hard. The complexity is sub exponential on the key size.
- Quantum computing can help to factor n efficiently, however it may take some years before they are developed.

Complexity of Factorization

- In general, the factorization is hard.
 - Brute force Attack: (infeasible given size of numbers) Brute force algorithm is exponential in b , where b is number of bits in the representation of the number n to be factored.
- Complexity of the best known algorithm for factorization:
$$\exp((c+O(1)b^{1/3} \log^{2/3}(b))),$$
for some integer $c < 2$
- May be quantum computers come to our rescue; earlier people were thinking it might thousands of years. But with rapid development of Quantum computing, the risk has been moved from “long term” to “medium term”.

Summary

Hard Problems on which RSA is based:

- **1. Integer Factorization problem:** Given a large positive integer n , find its prime factorization. (Every number n can be expressed as $p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$, where the p_i 's are distinct primes and each $e_i > 1$). In particular, if a number n is constructed as a product of two large primes, it is difficult to factor n .
- **2. RSA problem:** Given a positive integer n that is a product of two distinct odd primes p and q , ($n=pq$) and a positive integer e such that $\gcd(e,(p-1)(q-1)) = 1$, and an integer c , find an integer m such that
$$m^e \equiv c \pmod{n}.$$

Comment on Security of Known Schemes

- Almost all modern cryptosystems are based on more than one hard problems in mathematics (eg. Discrete logarithms, factorization, RSA problem etc).
- In fact there are no theoretical proofs available stating that these problems are hard.
- On the other hand, there are many instances where the so-called hard problems are easy to perform.
- We should ensure that the practical implementation do not use such pathological cases. Hence, we have to address security against any known vulnerability of these hard problems.
- Such attacks based on specific vulnerability of instances of hard mathematical algorithms can be considered as Mathematical attacks. We look for active attacks next.

Security Notions

- The security of a cryptosystem is defined with respect to the attacks it can withstand.
- The attacker will not be given private or secret information of the cryptographic key whose public cryptosystem he is attacking.
- There are three types of active attacks:
 - **Chosen-plaintext attack(CPA)**
 - Encryption box is available to the attacker before the attack.
 - Here the attacker can obtain cipher texts corresponding any chosen plain texts. The goal is to weaken the crypto system with the obtained plaintext-ciphertext pairs
 - **Chosen-ciphertext attack(CCA)**
 - Decryption box is available to the attacker before the attack.
 - **Adaptive Chosen-ciphertext attack(CCA2)**
 - Decryption box is available to the attacker except for the challenged ciphertext.
 - Here attacker can obtain plaintexts corresponding any chosen ciphertexts. This means the attacker gets decryption assistance for any chosen ciphertext. The goal for the attacker is to obtain any part of the plaintext after the decryption assistance is terminated.

Efficient Computation

- RSA requires an algorithm for exponentiation in mod n.
- I will give you an extended workshop sheet where we workout some mathematical results pertaining to RSA operations.
- You will have an opportunity to work on RSA key generation exercises in next week workshop.
- Can you write your own fast algorithm for RSA encryption and decryption?

- How do you choose primes for RSA?
- You need to know a bit more mathematics to understand the theory. We will not study in this topic in this subject.

Week 4

Lecture 1

Recap of Week 3 lectures and workshops

Modes of Encryption Chapter 7

DES EXAMPLES

Public Key Cryptography: Diffie-Hellman Protocol and RSA

Lecture 2

Proof of RSA Encryption + Chinese Remainder Theorem, Continued from Week 3 Lecture 2

Workshop 3: Workshop based on Lectures in Week 3

Quiz 4

Week 5



Lecture 1

Part 1: Public Key Cryptography: RSA Digital Signature

Part II: Security of RSA

Lecture 2

Revisiting Modes of Encryption and any left-over mathematics.

Workshop 3: Workshop based on Lectures in Week5

Quiz 5

Public Key Cryptography: Diffie-Hellman and RSA



Lecture 1

Part I

- 1.1 RSA Digital Signature
 - Digital Signature Introduction.
 - Different Versions RSA Signatures.
 - Signature Algorithm in Practice
- 1.2 Mathematical Attacks
 - Security of RSA
 - Elementary Attacks
 - Status of Factorization problem

Part II Security of RSA

- 1.3 Security Notions and Attack Models
 - Security Notions
 - CCA Attack
 - Timing Attacks

1.3 Security Notions and Attack Models

COMP90043

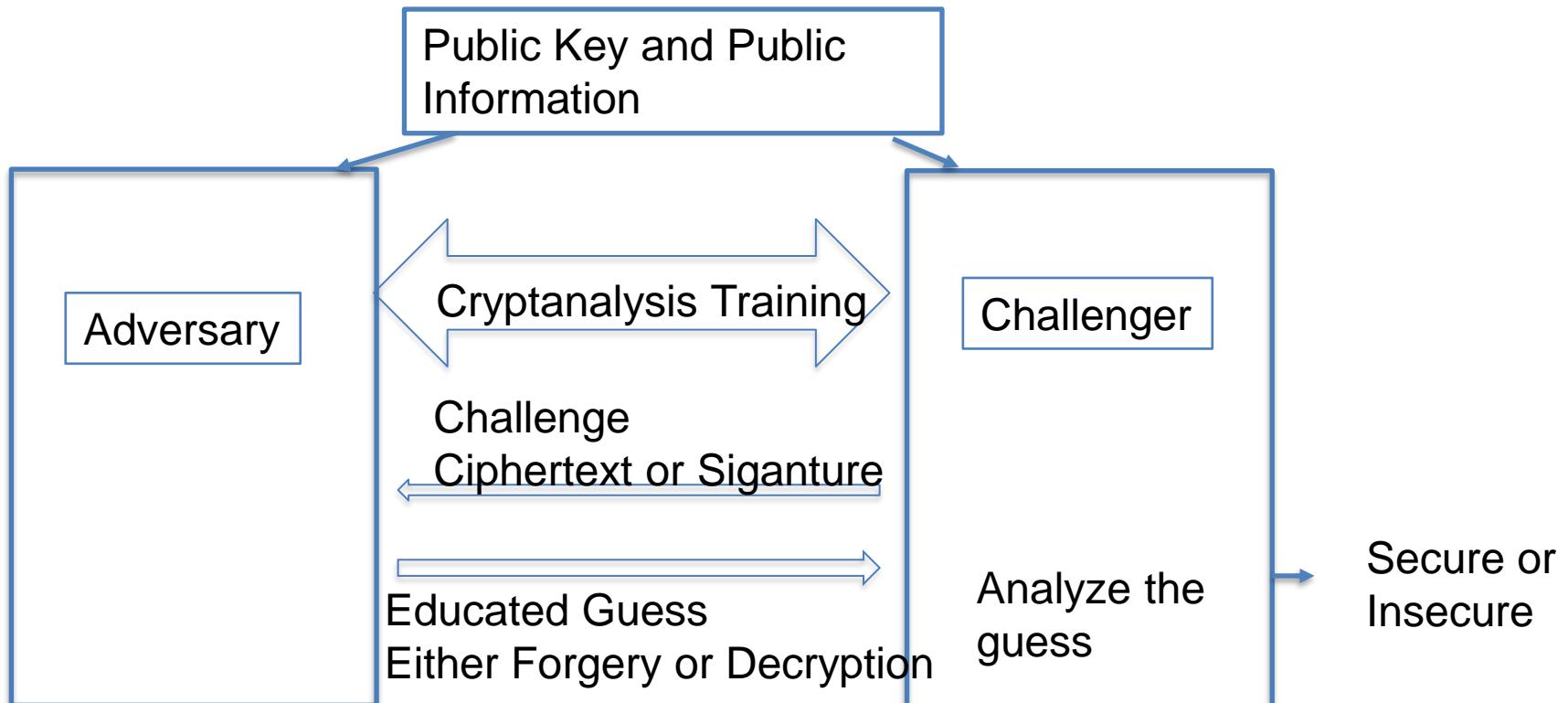
Lecture 1

Security Notions

- We introduced security notions for public key cryptography in the previous weeks.
 - The security of a cryptosystem is defined with respect to the attacks it can withstand.
 - The attacker will not be given private or secret information of the cryptographic key whose public cryptosystem he is attacking.
- There are three types of active attacks:
 - **Chosen-plaintext attack(CPA)**
 - Encryption box is available to the attacker before the attack.
 - Here the attacker can obtain cipher texts corresponding any chosen plain texts. The goal is to weaken the crypto system with the obtained plaintext-ciphertext pairs
 - In Public Key Cryptography, attacker can create as many public keys as he can to study its security. Interesting attacks are in fact with breaking decryption, i.e CCA attacks:
 - **Chosen-ciphertext attack(CCA)**
 - Decryption box is available to the attacker before the attack.
 - **Adaptive Chosen-ciphertext attack(CCA2)**
 - Decryption box is available to the attacker except for the challenged ciphertext.
 - Here attacker can obtain plaintexts corresponding any chosen ciphertexts. This means the attacker gets decryption assistance for any chosen ciphertext. The goal for the attacker is to obtain any part of the plaintext after the decryption assistance is terminated.

Attack Model

- We create a game involving Challenger and Adversary.



Justification for the Attack Model

- The framework need to capture the practical realities when analysing the strength of cryptosystems.
- In reality, sometime encryption box or decryption box are available to practical adversaries.
- The analysis will help to evaluate the security of the system.
- Main goal is capture all types of practical attacks, which is not easy in general.
- The textbook RSA is naturally not suited work in this framework.
- In the next slide, we show how RSA is vulnerable to CCA.

CCA Security

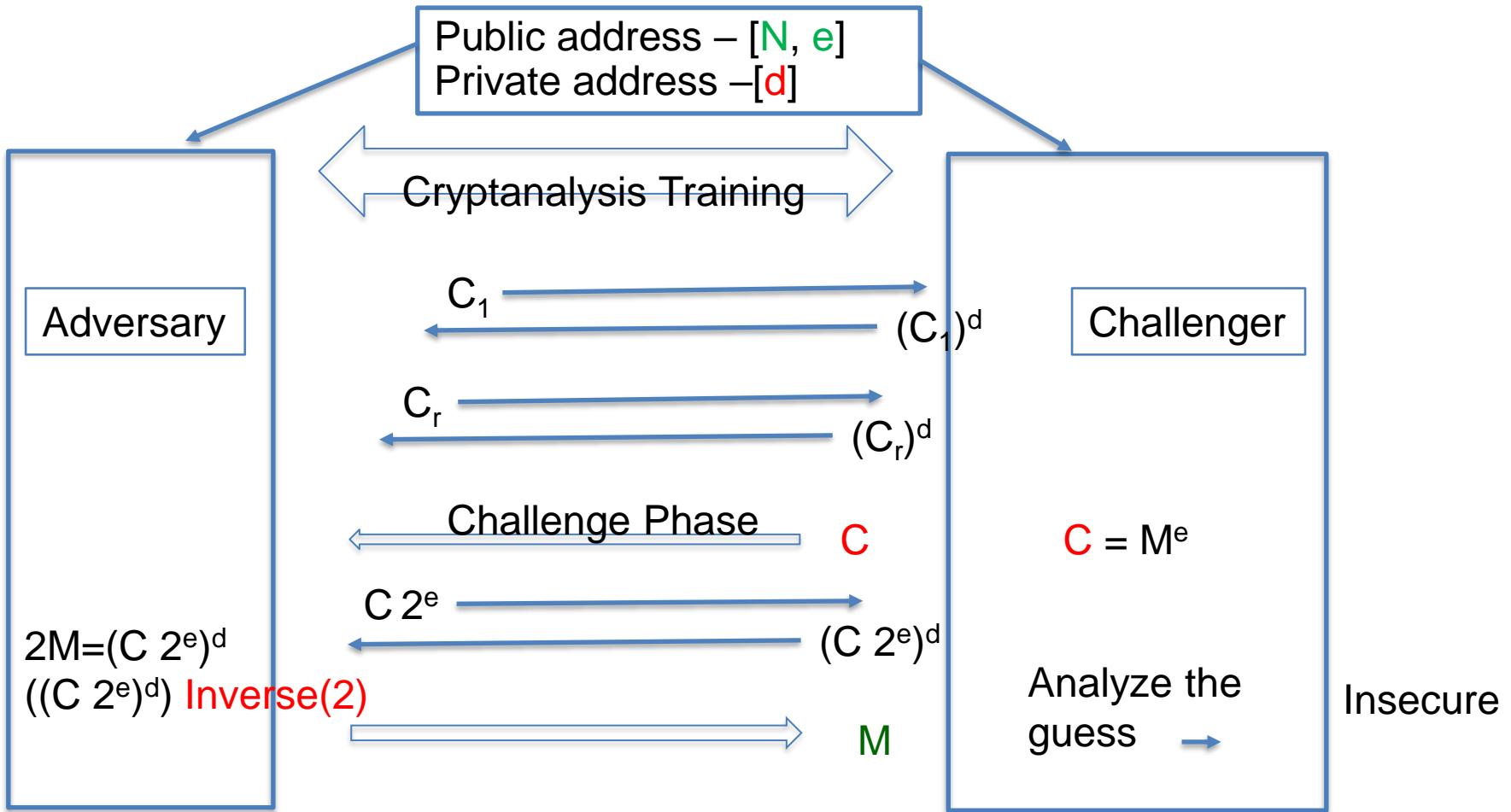


- The basic RSA algorithm is vulnerable to a chosen ciphertext attack (CCA).
- In this scenario, the adversary gets decryption of a number of ciphertexts of his choice.
- Adversary will then be given a challenge cipher text for which he has to produce the decryption (without having access to the private key).
- This is because of the multiplicative Property of the RSA Algorithm:

$$(M_1 \times M_2)^e = M_1^e \times M_2^e = (M_1 \times M_2)^e$$

$$(C_1 \times C_2)^d = C_1^d \times C_2^d = (C_1 \times C_2)^d$$

Attack



Adversary can choose any value b (instead of s) to blind the ciphertext
But $(b, n) = 1$

Practical RSA: Use RSA with OAEP

- To overcome the previous attack, you need somehow break the multiplicative property of the scheme.
- In practice message is introduced with a specific format, which removes the multiplicative property.
- OAEP is one such formatting method.
Please follow Fig 9.10 of the textbook

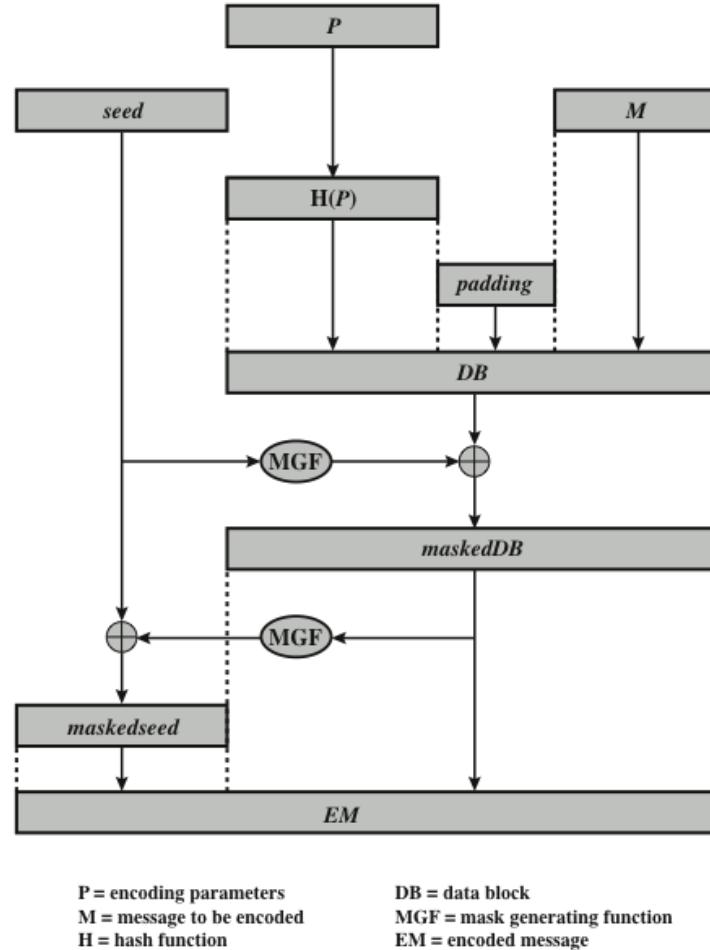


Figure 9.10 Encryption Using Optimal Asymmetric Encryption Padding (OAEP)

Figure: © 2017 Pearson Education, Inc., Hoboken, NJ. All rights reserved.

Timing Attacks

- Timing attacks are slightly different to the previous attacks. There is a surprise element to it.
- Here the attacker will observe the behaviour of the Cryptographic algorithms to different inputs and use the experience to break the secret directly.
- It can be devastating especially because adversary only needs ciphertexts.
- The attack is applicable wide range of cryptographic algorithms.
- If you observe variability in any aspects of the crypto algorithm, you may be able to convert into an attack. The generalizations of this attack include power analysis attack and fault based attack. The later, a certain faults are introduced deliberately and attacker studies the algorithm.
- Please refer Chapter 9 for more details.

Counter measures for Timing attacks

- **Constant time:** One way is to make sure that your algorithm takes a constant time for all inputs. This approach requires you to estimate the longest delay in advance and use appropriate idle time when results take less than the worst case time. However, this method may still leak power profile. In general performance decreases in efficiency.
- **Random delay:** You will add a random delay to algorithm execution to ensures that the relationship between key and the execution time is uncorrelated.
- **Blinding:** You can use the blinding technique introduced earlier. With this, the algorithm takes a random amount time and assures that the relationship between key and the execution time is uncorrelated.

Week 5



Lecture 1

Part 1: Public Key Cryptography: RSA Digital Signature

Part II: Security of RSA

Lecture 2

Revisiting Modes of Encryption and any left-over mathematics.

Workshop 3: Workshop based on Lectures in Week5

Quiz 5

Week 5



Lecture 1

Part 1: Public Key Cryptography: RSA Digital Signature

Part II: Security of RSA

Lecture 2

Revisiting Modes of Encryption and any left-over mathematics.

Workshop 3: Workshop based on Lectures in Week5

Quiz 5

Public Key Cryptography: RSA Digital Signature

COMP90043

Lecture 1

Public Key Cryptography: Diffie-Hellman and RSA



Lecture 1

Part I Public Key Cryptography: RSA Digital Signature

- 1.1 RSA Digital Signature
 - Digital Signature Introduction.
 - Different Versions RSA Signatures.
 - Signature Algorithm in Practice
- 1.2 Mathematical Attacks
 - Security of RSA
 - Elementary Attacks

Part II Security of RSA

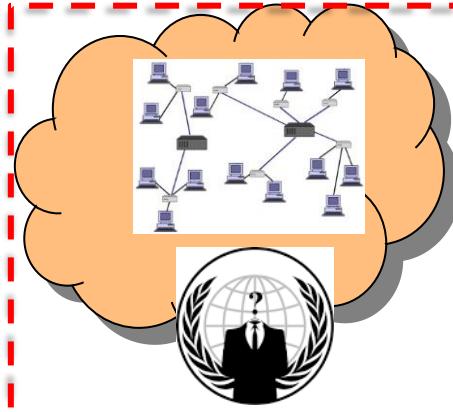
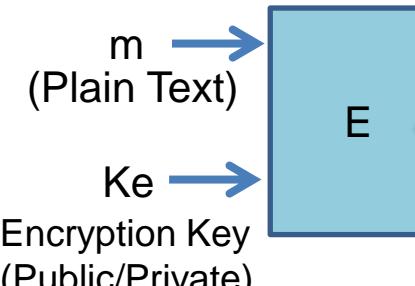
- 1.3 Security Notions and Attack Models
 - Security Notions
 - CCA Attack
 - Timing Attacks

Recap: Story of Alice and Bob terms and notations



Alice (Sender)

- Uses an Encryption Function (E)



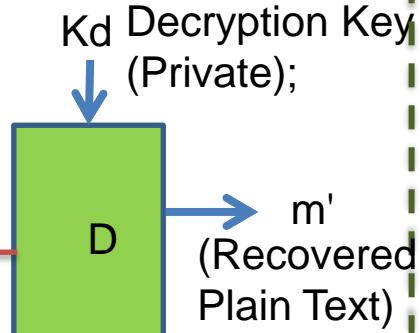
Eve (Adversary)

- Uses network probes to attack communication Tampering & eavesdropping



Bob (Receiver)

- Uses a Decryption Function (D)



E, D are public; c is the ciphertext, c' is received ciphertext; ideally $m=m'$;

Cryptography involves many conceptual ideas, we look at the basic functions

The Figure Illustrates the notations
And use of Public Key functions;
We will use this notation
throughout this semester.

Public key of B : Pu_b
Private key of B : PR_b

Encryption and Decryption by A

$$\downarrow$$

$$Y = E(PU_b, X)$$

$$X := D(PR_b, Y)$$

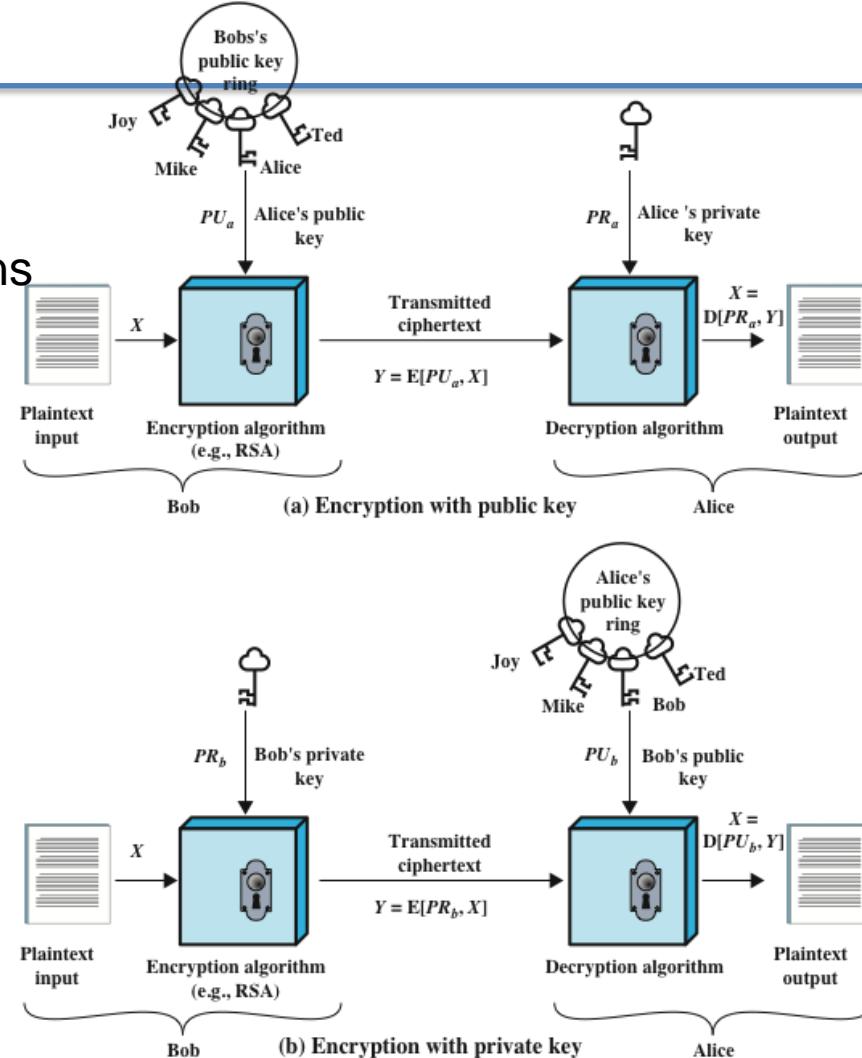


Figure 9.1 Public-Key Cryptography

1.1 RSA Digital Signature

COMP90043

Lecture 1

Digital Signature

- We studied Public Encryption last week. We saw how RSA encryption and decryptions work as functions.
- What is a digital signature?
- An electronic tag as a sequence of bits created by a sender on a message sequence, enabling a verifier to conclude that the tag was indeed created by the Sender for the message, providing authentication and assurance that the contents of the transmitted bits are not modified before the verification.
- Main goal for signature is to bind the message to the sender with assurance that the signature is not forged by anyone other than the sender.
- Can we use RSA to create signatures?
- We will tackle this problem this week.

RSA PKC (1978)

- Let $n = p \times q$; p, q are primes. Let the plain text and cipher text belong to integers modulo n and let (e, d) pair be computed such that

$$e \times d \equiv 1 \pmod{\phi(n)}$$

(ϕ :Euler's totient function)

- For the RSA key parameter set $K = (n, p, q, e, d)$, define

$$E_k(x) = x^e \pmod{n}$$

And

$$D_k(y) = y^d \pmod{n},$$

where $(x, y \text{ in } Z_n)$. The values (n, e) are termed the **public key**, and the values p, q and d form the private key.

RSA Example

- Let $n = 91$; $p=13$, $q=7$ are primes. Let the plain text and cipher text belong to Z_{91} (residue Integers modulo 91). $\phi(n) = 12 \times 6 = 72$.
- For $K = (n=91, p=13, q=7, 5, 29)$, define

$$E_k(x) = x^e \bmod n$$

And

$$D_k(y) = y^d \bmod n,$$

- Verify $5 \times 29 = 145 \bmod 72 = 1$
- Message $x = 11$
- $E_k(11) = C = 11^5 = 72$
- $D_k(72) = 72^{29} = 11$

RSA Signature

- The purpose of this discussion is to understand how RSA signature works and how it is different from RSA encryption we studied earlier.
- RSA signature is a public key digital signature scheme.
- In general, a signature is a means for a trusted third party (Network Security Manager) to bind the identity of a user to a public key.
- Why do we need a trusted third party? It becomes clear when you try to answer the following?
- Can we conceive digital signature using symmetric key cryptosystem?
- Answer: Technically yes, but you require a trusted centre who provide key management service for creators and verifiers of the digital signature. We will not study this concept in this subject.
- Do we need trusted third party for Public key signatures?
- Yes, it is mainly to address the authentication of public addresses, we will look at this issue later.

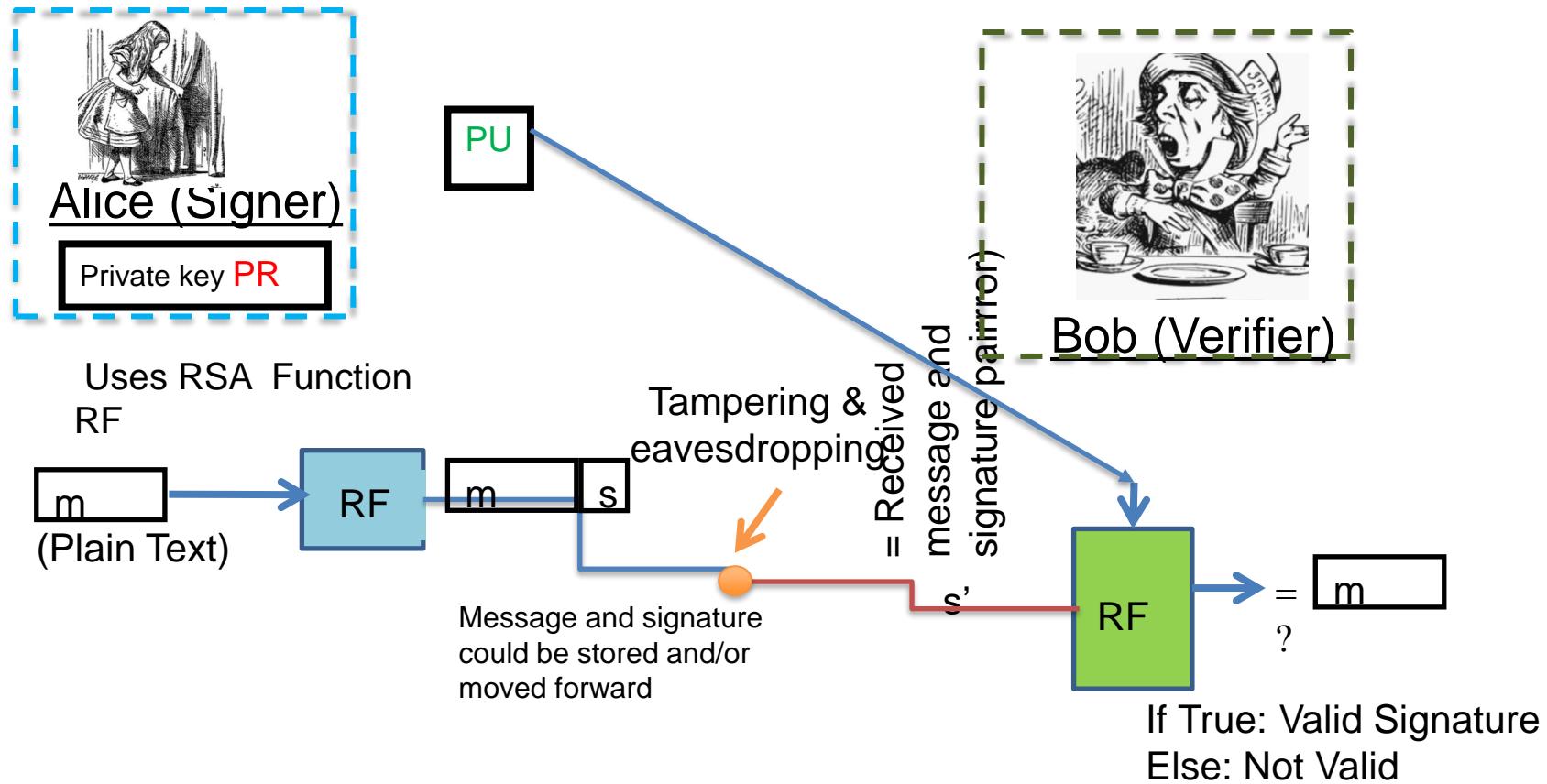
RSA signature

- RSA signature is a complement of RSA public encryption.
- In RSA Signature, the owner for the private key signs messages; Anyone with the public key can verify the signatures.
- $\text{Sig} = \text{RF}(\text{PR}, \text{Message})$; Verification is a $\text{RF}(\text{PU}, \text{Message}, \text{Sig})$, which outputs 1 if it is a True Signature and 0 if invalid.
- In RSA encryption, anyone with the public key can encrypt messages and the owner of the private key can decrypt the messages.
- $\text{Ciphertext} = \text{RF}(\text{PU}, \text{Message})$, Message can be decrypted by $\text{RF}(\text{PR}, \text{Ciphertext})$ which also gives additional output of 1 if correctly decrypted or 0 if erroneous.

What is the Signature Structure

- Let M be a message, PR and PU are private and Public keys of Alice.
- Signature is obtained by a
 - $\langle M, \text{Sig} = \text{RF}(PR, M) \rangle$, where RF is RSA function,
- Verification is essentially checking
- $\text{RF}(PU, \text{Sig})$ is equal to M or not?
- Let us try to create Signature process using RSA construction.

RSA Digital Signature



First Version of RSA Signature

- $N = P \times Q$; P, Q Large Primes,
- Choose Public key e and private key d such that $e * d \equiv 1 \pmod{\phi(N)}$
- Public address – $[N, e]$
- Private address – $[d]$
- Signature Generation:
- Message $0 < M < N$;
- Compute: $s = M^d \pmod{N}$;
 - Signature— $[M, s]$
- Verification – if $s^e \pmod{N} =?= M$ then “Signature Valid”
- Else “Signature Invalid”

First Version of RSA Signature

- $N = P \times Q$; P, Q Large Primes,
- Choose Public key e and private key d such that $e * d \equiv 1 \pmod{\phi(N)}$
- Public address – $[N, e]$
- Private address – $[d]$
- Signature Generation:
- Message $0 < M < N$;
- Compute: $s = M^d \pmod{N}$;
 - Signature— $[M, s]$
- Verification – if $s^e \pmod{N} =?= M$ then “Signature Valid”
- Else “Signature Invalid”

Issues with the First Version of the Scheme

Multiplicative property of RSA signature

$$(M_1 \times M_2)^d = M_1^d \times M_2^d = (M_1 \times M_2)^d$$

i.e. if s_1 = Signature of M_1 ;
 s_2 = Signature of M_2 ;

Then $(s_1 \times s_2)$ is the signature of $(M_1 \times M_2)$. Follows from exponential law.

This property leads to a possibility of forgery of signature!

This is in fact an example of existential forgery.

Further,

- What if Message is very long?
- You may need to split a long message into several messages of size less than N and sign the parts one by one.
- Also, a problem called blinding, which we will look in the next slide.

Blinding

- Alice's Public address – $[N, e]$
- Alice's Private address – $[d]$
- You want to get Alice sign a message M , which Alice may not be ready to do.
- Choose a random x – in the range $[0..N-1]$
- Form a blinded message -- $M_b = x^e M \bmod N$
- Alice may agree to sign this blinded message M_b (assume),
- Alice then signs the message M_b as $s_b = M_b^d \bmod N$
- This blinded signature can be used to compute the signature for M using the multiplicative property:
- Now you can compute signature for M as
- $s = s_b / x \bmod N$
- This is true because, let us apply the verification rule, Note
- $s^e = s_b^e / x^e = (M_b)^{d \cdot e} / x^e = (M_b) / x^e = x^e M / x^e = M$
- Hence s is the signature of M . So we have produced a forgery!

The issues

- The multiplicative property makes it impossible to the textbook RSA in practical application because of possibility of blinding.
- Also there is a possibility of existential forgery!
- So, for use as digital signature we need to some way to break the multiplicative rule of the RSA function.
- But note that this multiplicative property itself will be useful in some other situation as creating electronic cash analogous to the printed cash!

Second Version of RSA Signature

- $N = P \times Q$; P, Q Large Primes,
- Choose Public key e and private key d such that $e * d \equiv 1 \pmod{\phi(N)}$
- Public address – $[N, e]$
- Private address – $[d]$
- Signature Generation:
- Message $0 < M < N$;
- Find $M_1 = R(M)$; where R is a redundancy function and $1 < R(M) < N$.
- Compute: $s = M_1^d \pmod{N}$;
Signature— $[M, s]$
- Verification – $M_1 = R(M)$; if $s^e \pmod{N} =? M_1$ then “Signature Valid”
Else “Signature Invalid”

RSA Signature in Practice

- A practical signature scheme should take care of the two problems discussed before.
- Messages are generally long
- RSA signature scheme needs a redundancy function to avoid existential forgery attacks.
- Also repeated messages carry same signature.
- In practice, generally a suitable cryptographic hash function is applied to the message (which could be arbitrarily large); and sign the hash.
- We will learn about Hash functions later.

Third (Better) Version of RSA Signature

- $N = P \times Q$; P, Q Large Primes,
- Choose Public key e and private key d such that $e * d \equiv 1 \pmod{\phi(N)}$
- Public address – [N, e]
- Private address – [d]
- Signature Generation:
- Let M be a Message be of arbitrary length: $M = [.....]$; Find $H(M)$, where H is a Hash function
- Format $M_1 = [H(M), \text{Identity Information}, \text{Random Number}]$, such that $1 < M_1 < N$.
- Compute: $s = M_1^d \pmod{N}$;
Signature—[M, s]
- Verification – Extract M_1 by computing $s^e \pmod{N}$;
- If Any formatting violations – Reject the Signature.
- Further Verify $H(M) = H(M)$ on M_1

Signature Algorithm in Practice

- The previous discussion is about the issues related to RSA signature.
- In practice, standard signature algorithm proposed by NIST Digital Signature Algorithm (DSA) and Elliptic Curve DSA(ECDSA) are generally used, which we will discuss later.
- The 2019 version of the standard (FIPS186) include a version based on RSA. Please read Section 13.6 for further details.

1.2 Mathematical Attacks

COMP90043

Lecture 1

Security of RSA



- Brute force Attack: (infeasible given size of numbers)
- Attack by making use of loopholes in Key distribution.
- Mathematical attacks (Factoring and RSA problem)
- Elementary attacks
- Advanced Factorization methods
- Network attacks

Mathematical attacks

- The RSA function is one way. This is an assumption and we do not have a proof, but it is considered as one-way by researchers.
- The problem
- Given $n, e, c = M^e \pmod{n}$,
 - Can we determine M ?
 - Do we have an algorithm to find the e^{th} root of $c \pmod{n}$?
 - Can we find d such that $de = 1 \pmod{\Phi(n)}$?
- Can we factor n ?

Factorization Problem

- In general the factorization is hard.
- Brute force algorithm is exponential in b , where b is number of bits in the representation of the number n to be factored.
- Complexity of the best known algorithm for factorization:
$$\exp((c+O(1)b^{1/3} \log^{2/3}(b))),$$
for some integer $c < 2$
- It is not worth thinking of factoring.
- Maybe quantum computers come to our rescue; but not in immediate future!

Elementary attacks

- Can we use common modulus for more than one user?
- Facts:
- Knowing e and d such that
$$ed = 1 \pmod{\Phi(n)}$$
Is equivalent to factoring.
- Knowing n, $\Phi(n)$ is also equivalent to factoring.

Common modulus

- Every user chooses same modulus $n=pq$ set up by a trusted central authority. But each user chooses their own private and public key pairs
- User $i \cdots (e_i, d_i)$
- So using the facts in previous slide, any user can factor common modulus n and can find the private information of other user by using only the public information.
-
- Hence it is extremely important that every entity chooses its own RSA modulus n .
- Again the modulus need to be create using random primes.

Broadcast Problem

A group of entities may all have a same encryption exponent but should have different modulus. Further, to improve the public encryption, let the public key be small, say $e=3$.

- A wishes to send a common message m to three entities with modulus n n_1, n_2 and n_3 .
- The cipher text for three entities are given by
- $c_1 = m^3 \pmod{n_1}$
- $c_2 = m^3 \pmod{n_2}$
- $c_3 = m^3 \pmod{n_3}$.

Broadcast Problem

- Then, to recover the message m solve,
- $x = c_1 \pmod{n_1}$,
- $x = c_2 \pmod{n_2}$,
- $x = c_3 \pmod{n_3}$,
- You can use CRT and Then obtain an unique
- $x = m^3 \pmod{n_1 n_2 n_3}$
- m can then be obtained by taking the cube root of x . Finding a cube root in integers is not a hard problem.

Status of Factorization

- The textbook gives a detailed account of development in the factorization algorithm for integers.
- Please revisit Table 9.5 and Fig. 9.9.
- Currently RSA modulus should be as long as 2000 to 4000 bits.

Week 5



Lecture 1

Part 1: Public Key Cryptography: RSA Digital Signature

Part II: Security of RSA

Lecture 2

Revisiting Modes of Encryption and any left-over mathematics.

Workshop 3: Workshop based on Lectures in Week5

Quiz 5

Week 6



Lecture 1

Un Keyed Cryptography: Hash Functions

Lecture 2

Message Authentication Codes or Keyed Hash Function/ Finite Fields

Workshop 6: Workshop based on Lectures in Week 5

Quiz 6

Unkeyed Cryptography: Hash Functions

COMP90043
Lecture 1

Lecture 1

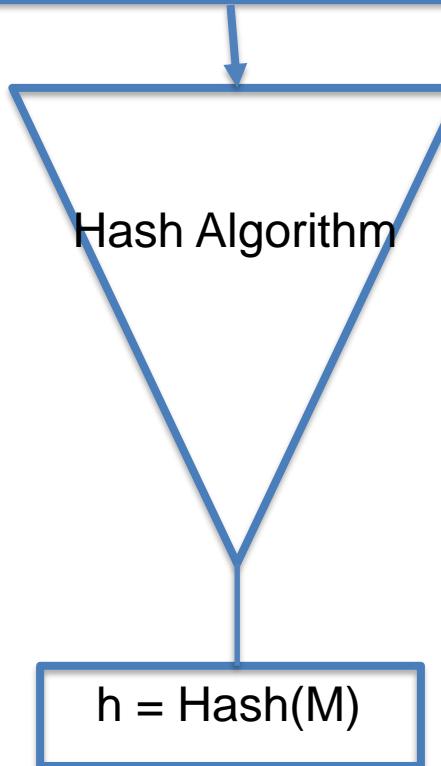
- 1.1 Preliminary Concepts
 - Hash function and message integrity
 - Uses of Hash functions
- 1.2 Requirements and Clarifications
 - Classification and definitions
 - Requirements
 - Analysis and simple constructions
 - A general construction
- 1.3 Attacks on Hash functions
 - Birthday Attack
 - Block Ciphers as hash functions
 - Practical Hash functions

What is an Hash function

- You would have used hash function while building Hash table data structure, where the purpose is to distribute keys evenly in the Hash table. The function you normally use is $(key) \bmod p$, a prime number.
- Cryptographic hash function has different requirements.
- In general, the function takes a variable-length data block as input and produces a fixed length tag or digest satisfying certain properties.
- The main objective is to obtain data integrity.
- It is referred as unkeyed primitive as does not require any key.
- As assumed in the other cryptographic functions, the definition of Hash function is also public.
- Hash is also referred to as message digest.

Cryptographic Hash Function

M = Arbitrary long message of L bits, with appropriate padding



It is a compression function

Integrity

- How does Hash function provide Integrity?
- They are called as Modification Detection Codes (MDC).
- The function Hash has a property that a small change in the message introduces unpredictable changes in the hash value, $h = \text{Hash}(M)$.
- If a message is changed while in transit, then running Hash function at the received message tells you how the value is deviated from the hash value computed at the source, thus assuring integrity with high probability.
- You would have studied CRC (Cyclic Redundancy Check) in the data networks. For integrity objective, they provide similar guarantees. The size of the CRC hash is small 32 bits.
- Cryptographic hash is much longer than CRC but also provide except that that it has many more properties, which we will study here.



Integrity in Active Setting.

Any intermediate adversary could change the data and compute the hash again. Thus the modification done by Malice cannot be detected.

As usual, the algorithm for the Hash function is public.

Hash function for Authentication?

- A simple answer is “No”. However, understanding of it requires some serious introspections.
- When used without a secret, it can best detect integrity violations due to natural errors that might occur during the message transmission.
- In active setting, an adversary can recompute hash for the things that he/she changed.
- To provide authentication with Hash function, you need to involve other encryption techniques, for example, Alice and Bob need to protect the hash value using encryption techniques.
- We explain issues with the scenario shown in Stallings Fig 11.3 and 11.4.

Hash for Message Authentication

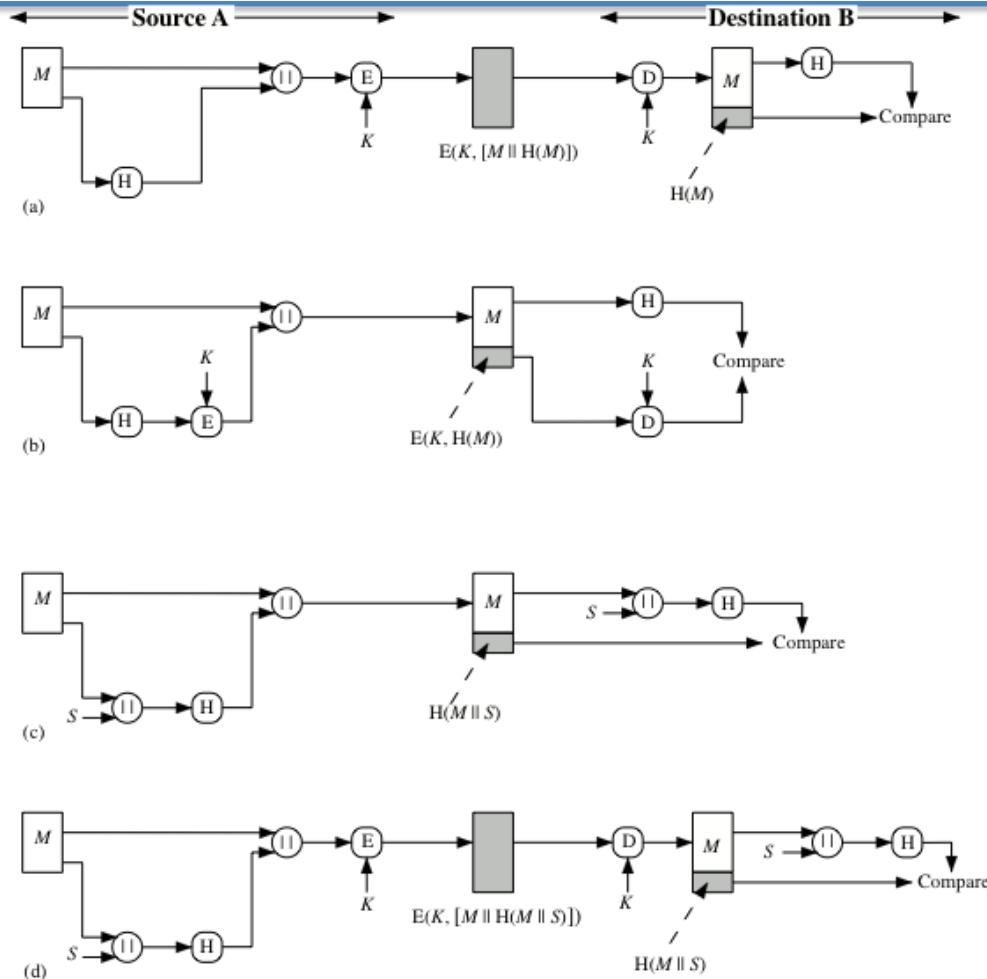
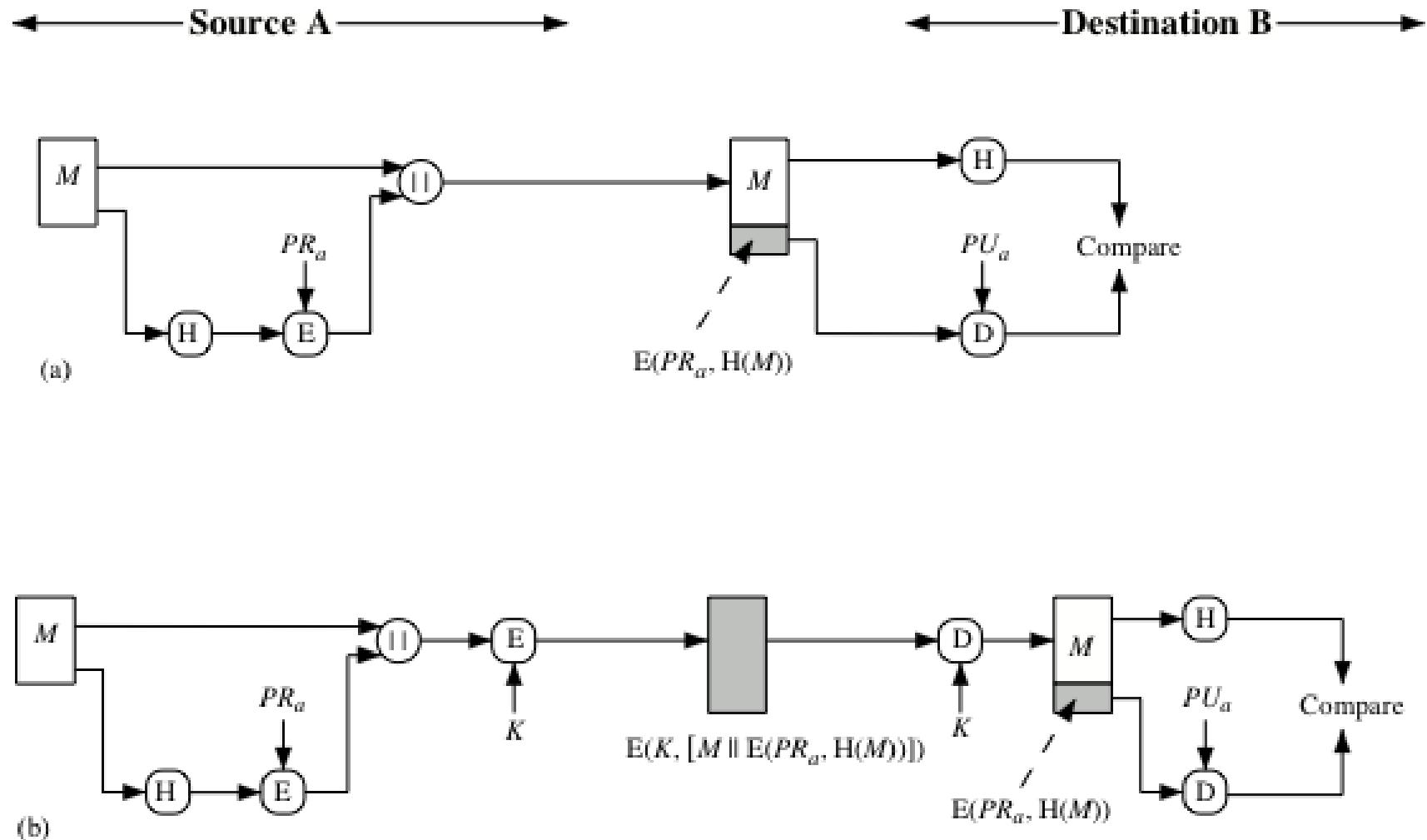


Figure 11.3 Simplified Examples of the Use of a Hash Function for Message Authentication

Use with Digital Signature



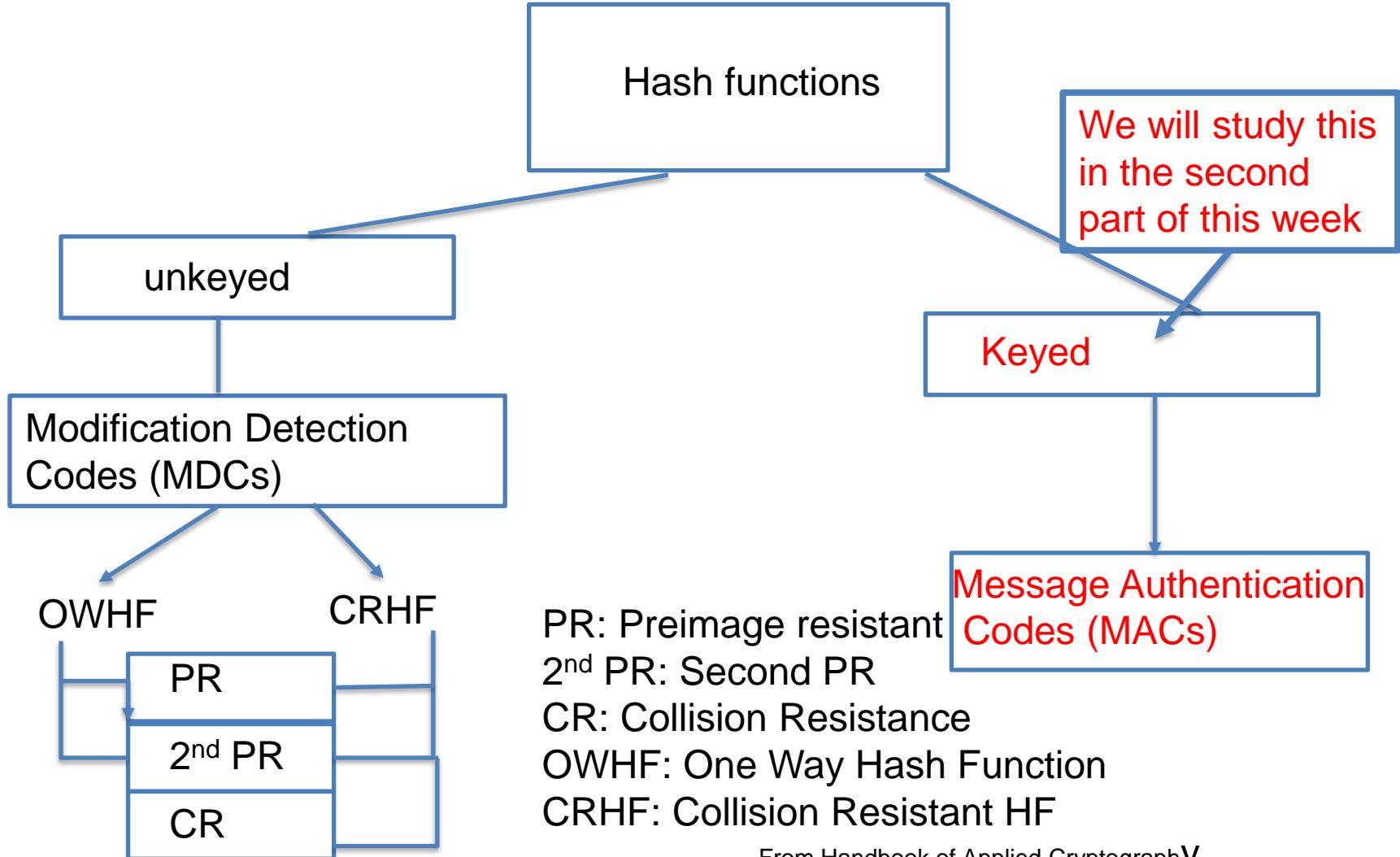
Source: William Stallings, Cryptograph and Security

Digital Signatures

- Hash functions are essential to Digital Signatures.
- Here, a hash value is encrypted using public key of the receiver, so the receiver with private key can decrypt and verify the hash value, thereby providing message authentication.
- If you need origin authentication, then the hash value need to be signed as well.
- Please read the textbook for studying other uses of hash functions.
- Signcryption is another public key primitive which does these two operations in one go.
- Some of these issues are the topic of discussion in later part of the course.

- Let us consider the classification of Hash functions, next to understand the issues.

Hash Functions Classification



Definitions

- We will assume the domain as message space (arbitrary long bit strings F_2^*) and range as the hash space, generally a fixed binary space (eg F_2^n).
- Hash function we look for is a mapping from F_2^* to F_2^n , $n = 128, 256, 512$ etc.
- Hash function should be “easy” to compute.

- “Easy” generally means that an efficiently computable, for example computing with polynomial time complexity.
- “Hard” generally means that computationally infeasible for example having only exponential time complexity.

Definitions

- PR (Preimage Resistance): Give an element in the hash space, it is computationally infeasible to workout an input in the message space that results the hash value as output. In other words, given y in F_2^n , it is not feasible to workout a preimage x' in F_2^* such that $H(x') = y$.
- 2nd PR (Second Preimage Resistance): Given a message and hash pair $(x, H(x))$, it is computationally infeasible to workout an another input in the message space that results the hash value as output. In other words, given x in F_2^* it is not feasible workout another preimage x' in F_2^* such that $H(x') = H(x)$.
- CR (Collision Resistance): It is computationally infeasible to find two different input messages in the message space that results in the same hash value as output. In other words, it is not feasible to workout two messages x and x' such that $H(x') = H(x)$.

Hash Function Requirements

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 11.1 from the textbook

Hash Function Relationships

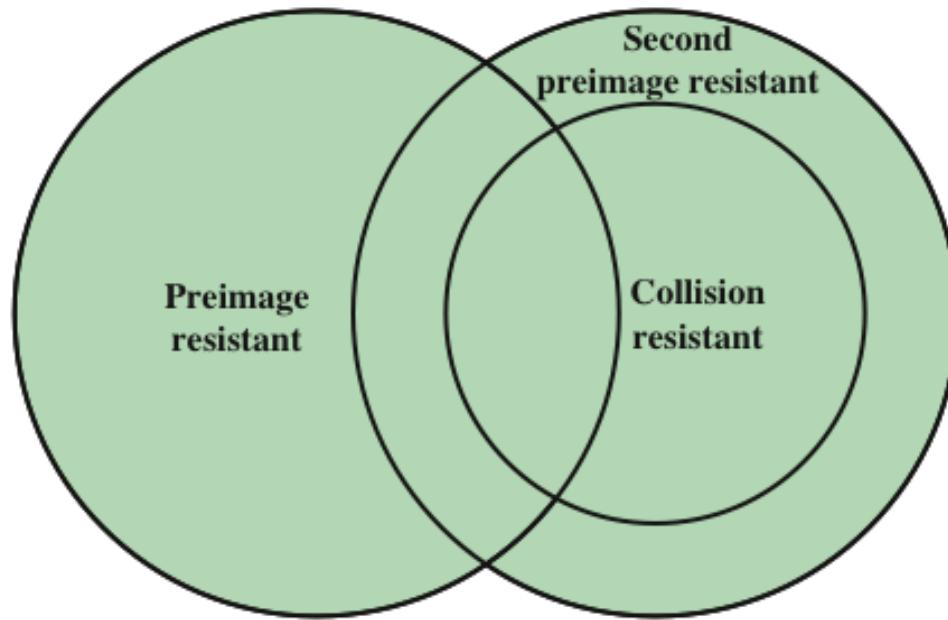


Figure 11.6 Relationship Among Hash Function Properties

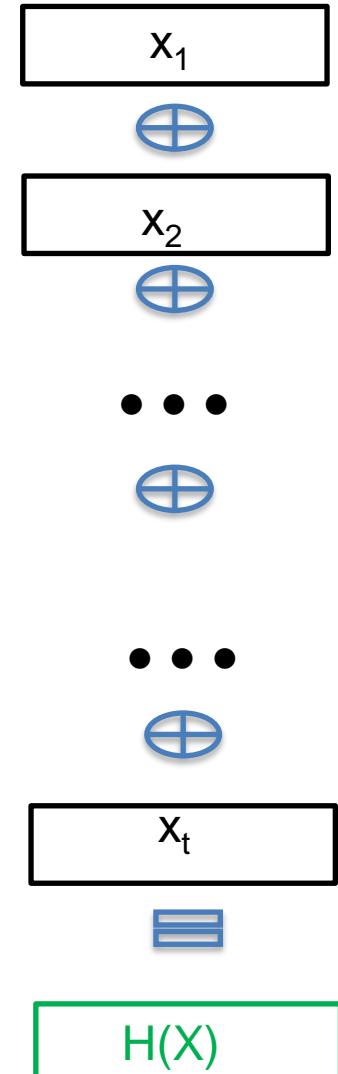
Fig 11.6 from the textbook

Analysis

- CR implies Second Pre Image Resistance(Second PIR).
 - Let H be CR but not PIR, then
 - Given $m, H(m)$, one can find a second pre image of $h(m)$ as m' . Then you discovered m and m' having the same hash contradicting the fact that H is CR. So this situation cannot happen, Hence CR implies Second PIR
- However, the converse is not necessarily true.

How to Construct Hash functions?

- Main requirements, the function should work for arbitrary input, i.e. F_2^* .
- It should have a fixed length output, of length n , i.e F_2^n .
- One way to arrange input as sequence of n bit blocks, in that case if the input is not multiple of n then add sufficient padding.
- Let $[X = x_1 \parallel x_2 \parallel \dots \parallel x_t]$, and Hash defined as $H(X) = x_1 \oplus x_2 \oplus \dots \oplus x_t$
- It can act as MDC, this is a simple parity check code.
- But is this Hash secure?
 - It does not have a preimage resistance, not PR.
 - Consequently it does not have Collision Resistance also, not CR.



An improvement to Parity Check Code

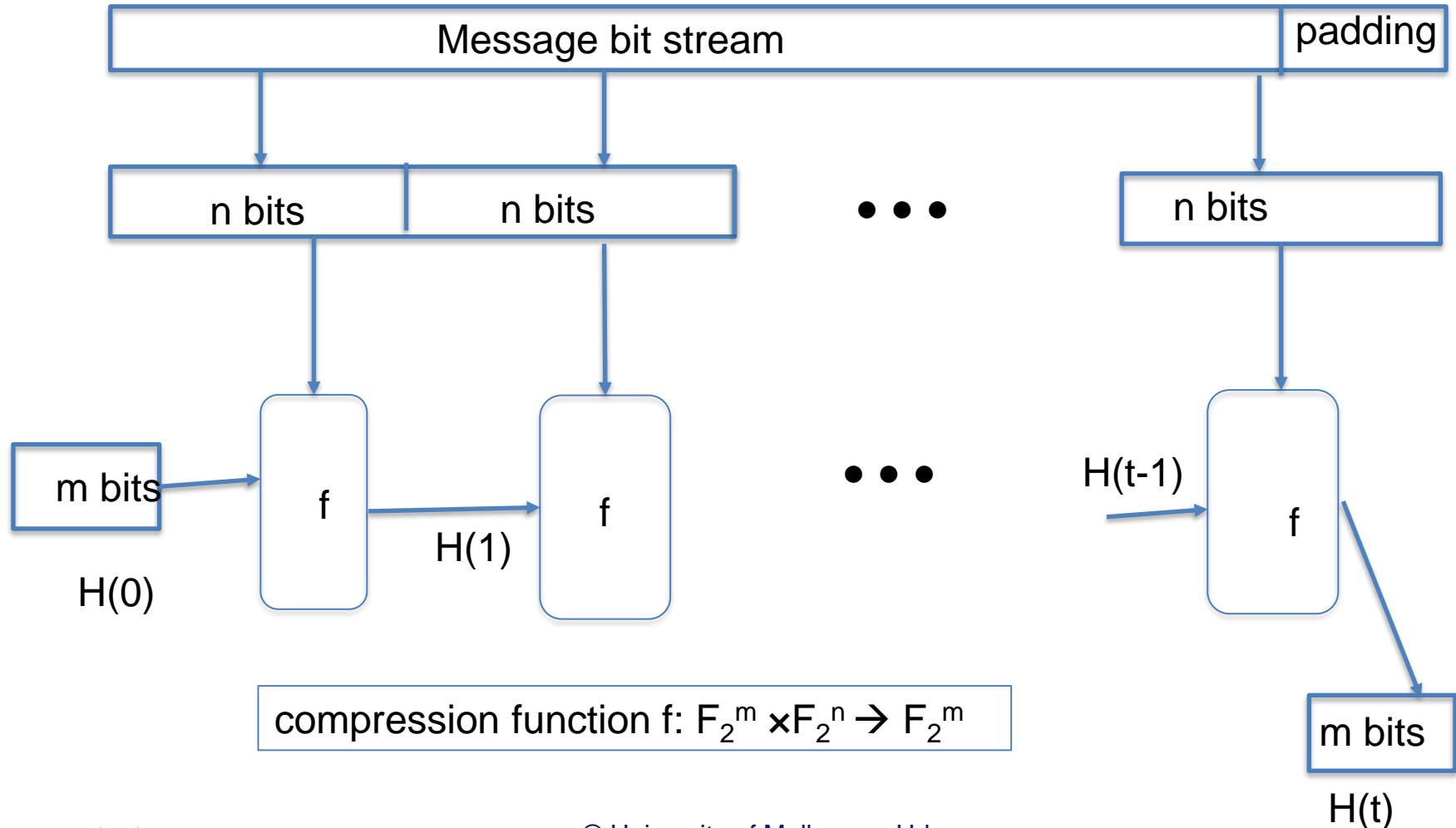
- For the previous construction, make the following changes:
- Add one circular shift after processing each block.
- Let $[X = x_1 \parallel x_2 \parallel \dots \parallel x_t]$, and Hash defined as
- Initialize $h = x_1$
- For each successive block
 - Rotate h to left by 1 bit and xor with the block
- Let T be an left shift operator, then
- $T[x_1(1), x_1(2), \dots, x_1(n)] = [x_1(2), x_1(3), \dots, x_1(n), x_1(1)]$
- $H(X) = T(\dots T(T(x_1) \oplus x_2) \oplus x_3) \dots \oplus x_t$
- For MDC this is a better function than the previous PCC,
- But is it good for security?
- Workout yourself how you can break the PR property.

In general



- Hash Construction methods and Attacks on the constructions goes hand in hand.
- Cryptographers work hard to find good methods; Cryptanalysts will work to break them. It is a competition.
- NIST invited researchers to come up with Hash proposals.
- Having a secure Hash function is still open question for researchers.
- Here I will illustrate a simple method based on the Block cipher constructions mode CBC.
- It is also sometime referred as Merkle–Damgård construction.
- Heart of the construction is a compression function $f: F_2^m \times F_2^n \rightarrow F_2^m$
- Messages are arranged as n bit blocks with possibly the final block padded to make the message length multiple of n.
- The size of the hash is m.

Merkle–Damgård Construction



Attacks on Hash function

- The function should resist brute-force attacks and regular cryptanalysis.
- Attack against PR: Given a random hash value, determine y such that $H(y)$ equals to the hash value.
- Attack against CR: The task is to determine any two messages whose hashes are same, i.e determine x, y such that $H(x) = H(y)$.
- What should be the size of hash value?

- Next we will illustrate the Birthday attack and its implications on the size.

What is the Birthday Attack?

- Consider a set of events of N different outcomes (think of this as numbers from 1 to N). Assume that they are equally likely. $P(i) = 1/N$ for all i ;
- Question: What is the probability that after n trials at least two of the outcomes will be the same?
- Example: Consider 128 bit hash values that are used in the digital signature algorithm for messages of size 1000 bits. There are 2^{1000} possibilities for the messages. The hash values are of length 128 bits and it is drawn uniformly from the space of 0; 1; 2^{128} ; N here is 2^{128} .
- How many messages you draw from the message space such that any two of them have the same hash?
- Is it 2^{128} ? The birthday attack tells us you do not have to wait that long, you will see collision with much less messages to the order of 2^{64} !

Birthday Attack

- This is not a surprising result, it follows from elementary probability theory.
- Consider random numbers from 1 to N using uniform distribution.
- Then, the probability that any two numbers are same exceeds 0.5 after roughly about Square root of N trials. [Please read the textbook]
- So for m bit hash function, there are 2^m possible hash values, $\text{Sqreroot of } 2^m = 2^{(m/2)}$.
- Thus, $2^{(m/2)}$ determines strength of hash code

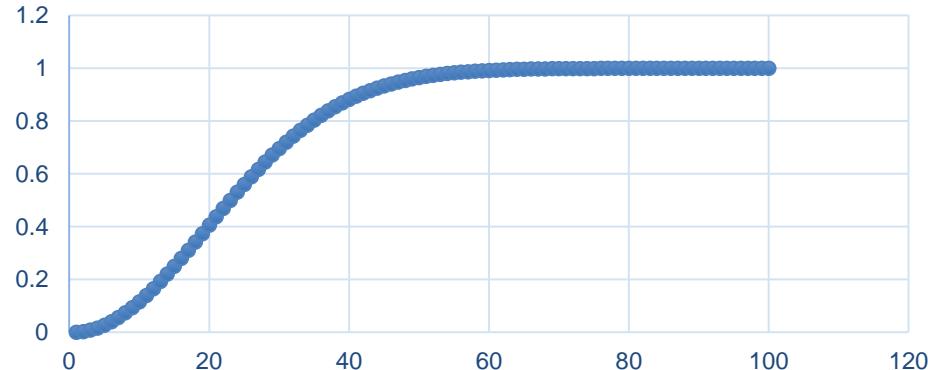
Birthday Attack

From Probability theory, The probability that after n trials at least two outcomes will be the same is at least ε

$$1 - e^{\frac{-1(n-1)n}{2N}}.$$

There are 365 days, $N = 365$, so plot of ε as a function n (trials) is below;
You only need around 23 trials before you see a collision!

Probability of two people having
same birthday as a function of n
trials



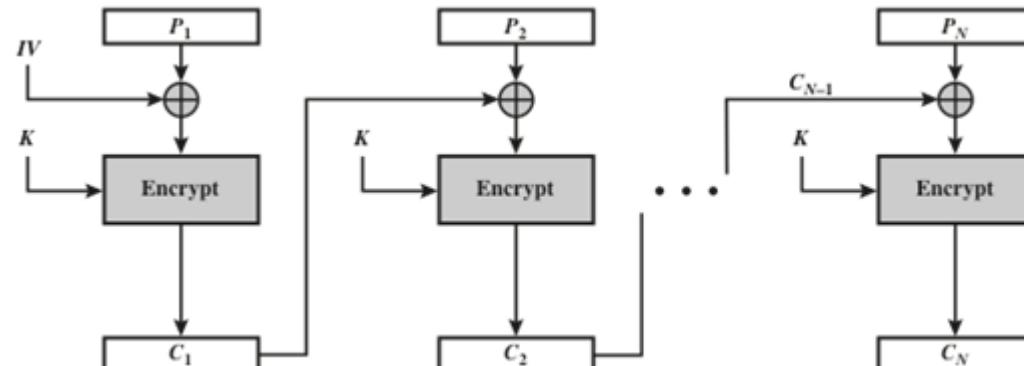
Magma code:
for $n := 1$ to 100 do
[$n, 1 - \text{Exp}(-n*(n-1)/(2*N))$];
end for;

An example from text

- Yuval's strategy of using the above attack for creating collisions.
- Please read 11.3 of the textbook.
- In a scenario, a given user is prepared to sign a valid message x
- An adversary generates $2^{(m/2)}$ variations x' of x , all with essentially the same meaning, and saves them
- The adversary then generates $2^{(m/2)}$ variations y' of a desired fraudulent message y
- Notice that the two sets of messages are compared to find pair with same hash, you will find them with probability > 0.5 by the attack.
- Now adversary can have user sign the valid message, then in the signature replace the corresponding message that resulted in collision, you have a forgery!
- What do we learn from this? The size of the hash should be large enough that finding collisions are impracticable. You need 128 to 512 bits for hash in practice.

Block Ciphers as Hash functions

- You can use Block ciphers in CBC mode to create hash functions.
 - Initialize $H_0=0$ and zero-pad of final block
 - compute: $H_i = E_{M_i}[H_{i-1}]$
 - and use final block as the hash value
 - similar to CBC but without a key
- If DES function is used the resulting hash is too small (64-bit) and hence vulnerable to birthday attack.
- Also, there is a variant of birthday attack called “meet in the middle”, please go through the textbook.



CBC mode encryption
Fig from the textbook

More practical Hash functions

- Secure Hash Algorithm is a proposal designed by NIST in 1993 and was published as a standard (FIPS180).
- There are several revisions to this and a lot of activities happened in breaking these schemes. SHA-1's CR property was broken!
- The latest standard is SHA-3 which is considered to be safe.

Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Week 6



Lecture 1

Un Keyed Cryptography: Hash Functions

Lecture 2

Message Authentication Codes or Keyed Hash Function/ Finite Fields

Workshop 3: Workshop based on Lectures in Week 5

Quiz 6

Week 6



Lecture 1

Un Keyed Cryptography: Hash Functions

Lecture 2

Message Authentication Codes or Keyed Hash Function

Workshop 3: Workshop based on Lectures in Week5

Quiz 6

Message Authentication Codes

COMP90043

Lecture 1

Lecture 2

- 1.1 Message Authentication
 - Issues in Practice
 - Message Encryption-Symmetric and Public key approach
- 1.2 Message Authentication Code
 - Internal and External Error Control
 - MAC in networks
 - Properties and Attacks on MAC
- 1.3 Pseudorandom number generation
 - Using MAC and Hash

Recap: Hash Function Requirements

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 11.1 from the textbook

Hash Function Relationships

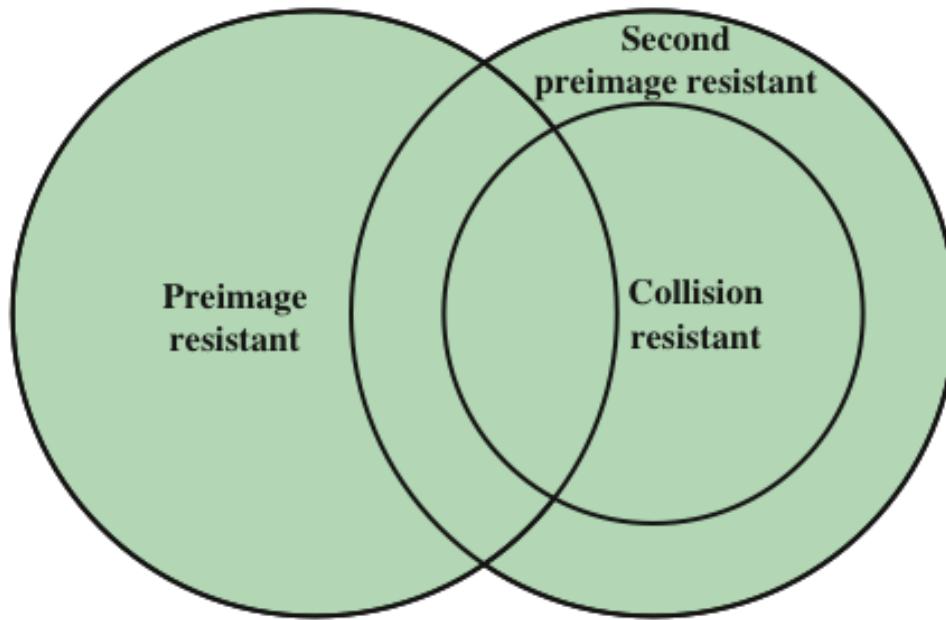


Figure 11.6 Relationship Among Hash Function Properties

Fig 11.6 from the textbook

Message Authentication

- Let us look at message authentication issue in practice.
- What is it concerned with?
 - To address message authentication
 - A dedicated primitive based on symmetric key cryptography
- Issues for message authentication-
 - Message integrity
 - Validation of originator's identity
 - Non-repudiation of the message origin
- Three ways of achieving authentication
 - Message Encryption
 - Hash functions (we looked at it in the previous lecture)
 - Message Authentication Code (MAC) (this lecture)

How do we create message authentication

- We need to separate message authentication function and the protocol that helps us to integrate the message authentication in the application.
- At a basic level, we can create a message authentication code using a secret key.
- At a higher level, the keys are carefully managed to obtain higher level guarantees on the exchanged message including source authentication.

Security Requirements

- Stallings discussed the security issues that can arise in the networked systems and consider following requirements:
 - disclosure
 - traffic analysis
 - masquerade
 - content modification
 - sequence modification
 - timing modification
 - source repudiation
 - destination repudiation
- Please read Section 12.1 for details.

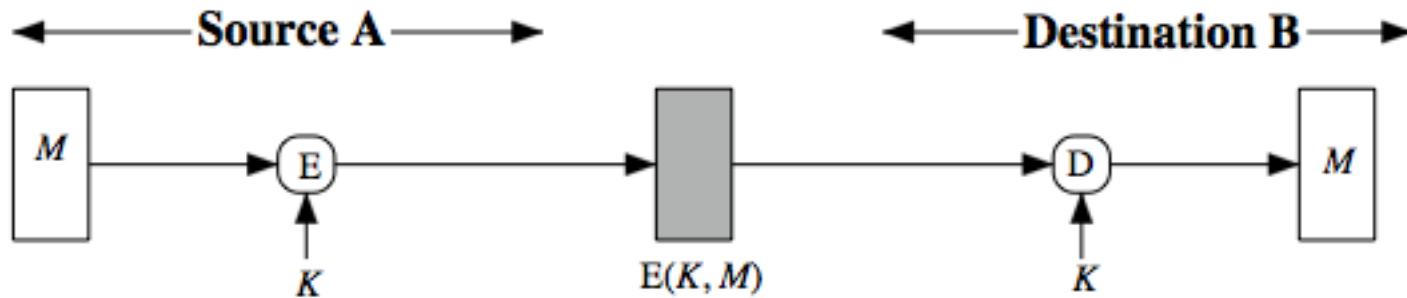
Message Encryption

- First let us understand how message encryption itself provides authentication.
- The issues are different to symmetric and public key methods.
- Note that with public key encryption, anyone could encrypt based on public key of the receiver and if you want source authentication, the sender needs to use signature.
- But symmetric key assumes that sender and receiver share a secret and encryption naturally provides authentication.
- Stallings Section 12.2 gives an account of these discussions.
- Let us first consider Symmetric Encryption.

Symmetric key Encryption

- How authentication is obtained?
 - Since they share the key, receiver is sure that the message was created by the sender.
 - By relying on format and structure of the messages, they can detect any modification,

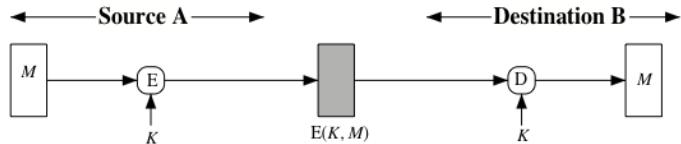
Next, we consider other situations:



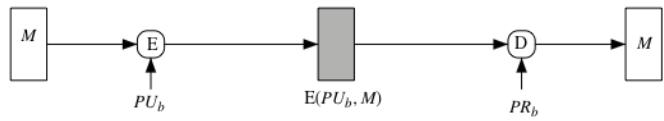
(a) Symmetric encryption: confidentiality and authentication

Fig 12 (a) from the textbook

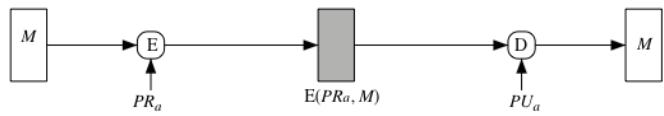
Basic Use of Encryption



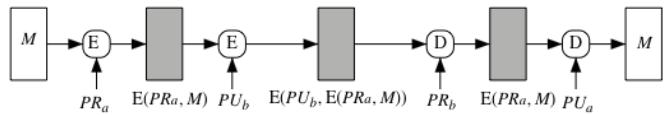
(a) Symmetric encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



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

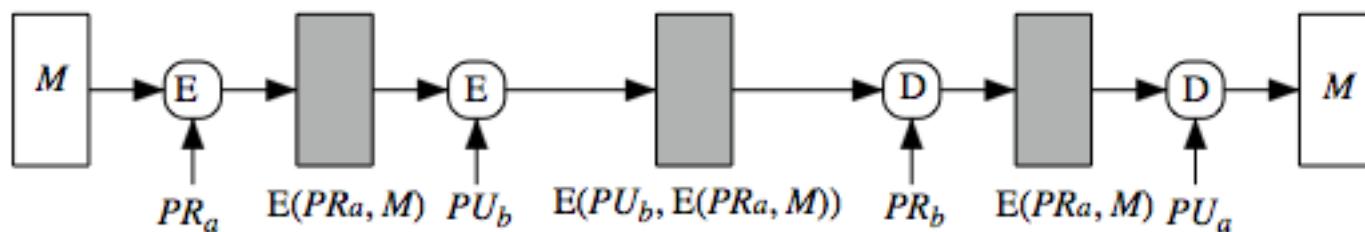
Figure 12.1 Basic Uses of Message Encryption

Read the discussion around Fig. 12.1 in the textbook

Fig 12 from the textbook

Public Key Encryption

- Public key by nature, anyone can use.
- Does not provide any guarantee for the sender.
- To provide authentication, a sender needs to sign as well (use private key) which can be verified by others using the public key.
- How do we decide if the message stream is corrupted or not?
- You need some general formatting rules.



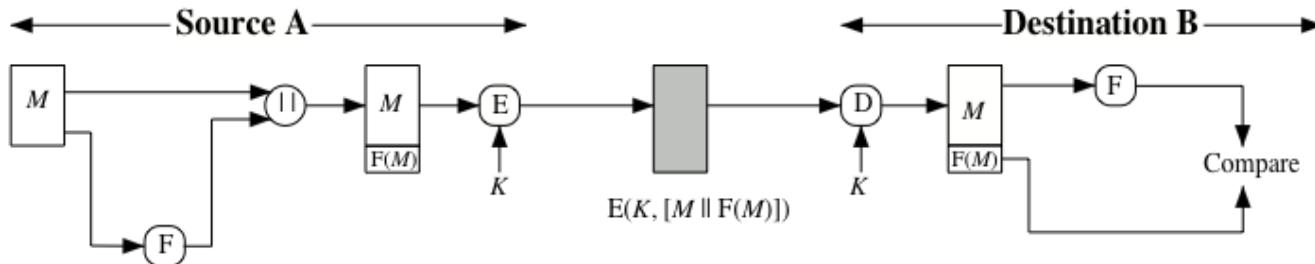
(d) Public-key encryption: confidentiality, authentication, and signature

Fig 12.1from the textbook

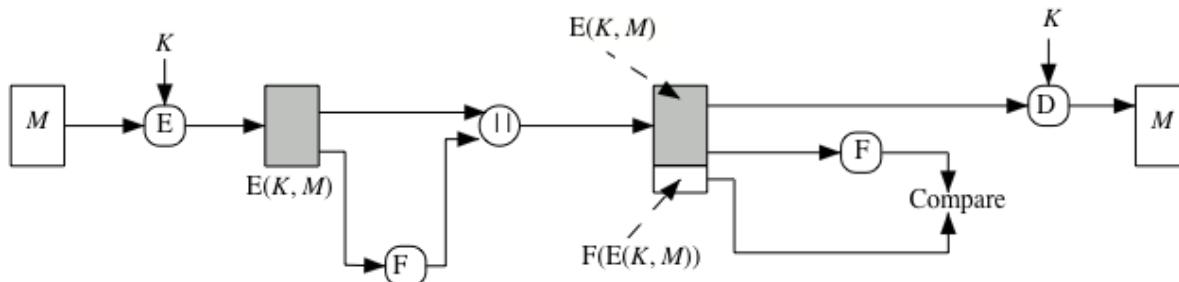
Message Authentication Code (MAC)

- A dedicated primitive to address mainly authentication using a key.
- The output of an algorithm can act as a signature.
- Only the receiver with the key can verify the code by running the same algorithm, thus assuring the integrity of the message from the sender.
- There are two ways of using the message authentication code:
 - Internal Error Control
 - External Error Control

Different Error Controls



(a) Internal error control



(b) External error control

Figure 12.2 Internal and External Error Control

Fig 12.2 from the textbook

MAC use in Practice

- A pair TCP hosts shares a secret key and all exchanges between the hosts use the same key,
- Leads to simple encryptions between hosts-all IP packets between them can be encrypted except the header.

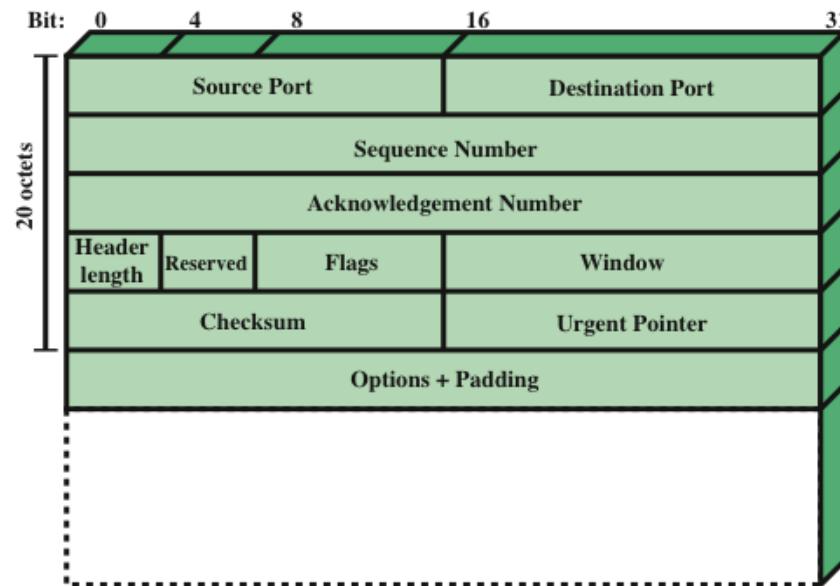


Figure 12.3 TCP Segment

Fig 12.3 from the textbook

Message Authentication Codes

- So formally, MAC is a dedicated symmetric key primitive aimed at providing authentication.
- With encryption it can be easily integrated to provide secrecy also.
- They are useful when in some applications you only need authentication.
- There are many situations where the property of authentication requires longer than confidentiality: authenticated sessions where only at times you may exchange secret information.
- MAC is different to Signatures,

Properties of MAC

- MAC has many properties similar to Hash.
- $\text{mac} := \text{MAC}(\text{Key}, \text{message})$.
- You can treat it as a cryptographic checksum/digest: It takes a arbitrary length message as input and outputs a fixed length authenticator using a key.
- Like hash functions, it is many-to-one function with Preimage resistance (PR).
- For every key, it satisfies hash function properties.
- So sometimes, MAC is referred to as a family of Hash functions.

Attacks on MAC

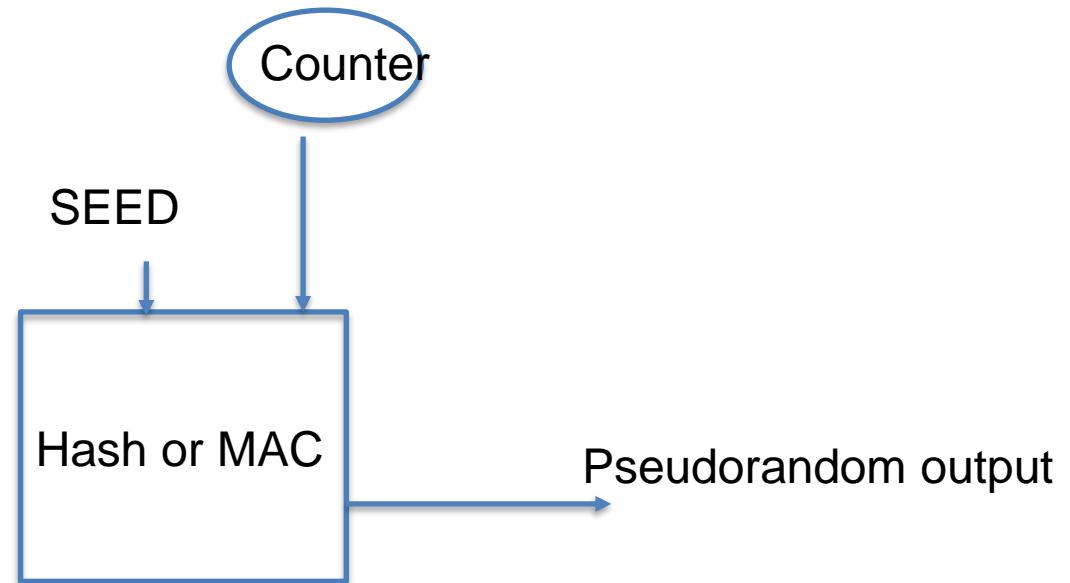
- Brute-force attack: Here the objective is to find a collision.
- For cryptanalysis, there are two approaches:
- Attacker may first determine the key, then he can produce MAC value for any message.
- Sometimes, he may just try to determine a valid tag for a given message.
- Similar to Hash functions, you realize that MAC has to have a certain length to defeat brute-force attacks.
- In general you try to create new MAC functions using existing Hash functions.

MACs Based on Hash Functions: HMAC

- We do not study constructions of MAC in detail.
- It is sufficient to think of it as a keyed hash function.
- MAC based on Hash functions are popular in practice.
- A simple proposal:
- $\text{KeyedHash} = \text{Hash}(\text{Key} \parallel \text{Message})$
- Some weaknesses were discovered using the simple proposal which led to development of HMAC.
- HMAC is thoroughly studied in literature. The textbook explains the concept with some detail. Please go through the discussion in Section 12.5 of the textbook.

Pseudorandom Generation

- As opposed to random numbers, pseudo random number generator takes a seed value as input and generates a sequence of digits.
- Like hash, for the same seed value it generates the same sequence.
- We briefly look at the topic and consider some Pseudorandom proposals based on hash and mac.



Week 6



Lecture 1

Un Keyed Cryptography: Hash Functions

Lecture 2

Message Authentication Codes or Keyed Hash Function

Workshop 5: Workshop based on Lectures in Week5

Quiz 6

Week 7



Lecture 1

Key Management

Lecture 2

MST

Workshop 7: Workshop based on Lectures in Week 6

Quiz 7

Key Management

COMP90043

Lecture 1

Lecture 1

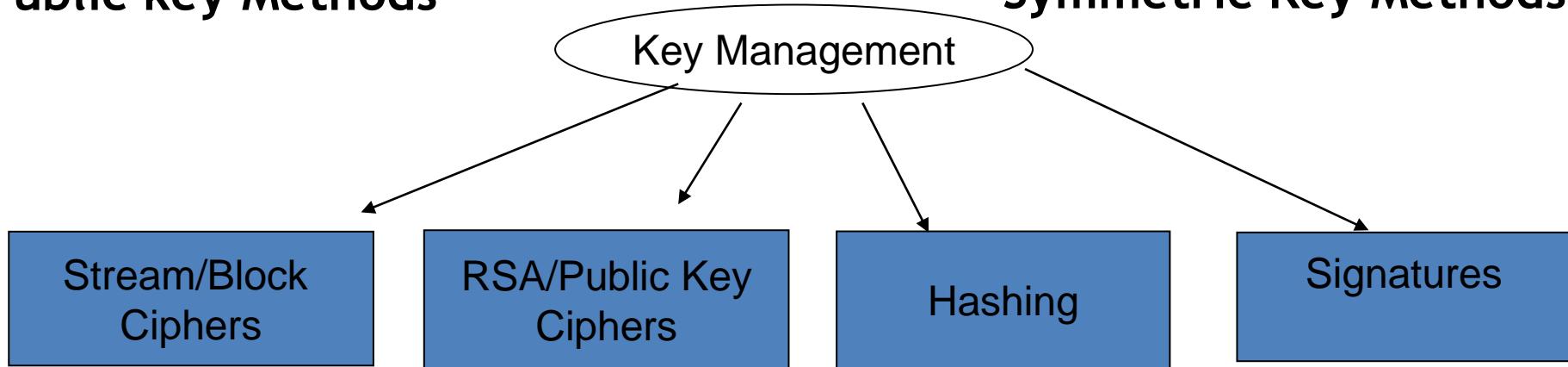
- 1.1 Key Management basics.
- 1.2 Symmetric Key Distribution
 - Key Distribution using a key Server
 - Key Hierarchy
 - Key Distribution Scenario
 - Merkle Key Distribution
 - Merkle's Scheme
 - Hybrid Scheme
- 1.3 Random Numbers
 - Pseudorandom Number Generators
 - Linear Congruential Generator
 - Using Block Ciphers as PRNGs
 - Blum Blum Shub Generator
 - Natural Random Noise

Key Management and Distribution

- Until now, we discussed many cryptographic primitives with different security objectives.
- When used in networked and computer systems, we need a plan to distribute appropriate keys to the relevant programs that use the primitives.
- This lecture will focus on symmetric key distribution methods.
- The issues are complex as it would involve multiple domains of engineering and computer science. These methods need to be included into protocols, operating systems etc.
- Most cryptographic algorithms except Hash functions involve keys.
- Key distribution is all about generation, distribution, storage, archival, of keys (symmetric and public).
- Without proper and secure key management, cryptographic algorithms are useless.

Key Management

Public key Methods



Symmetric Key Methods



Keys in General

- Keys are to be formed using purely random sources, but in practice, they are usually pseudo random based on some secret seeds or random seeds obtained from physical means.
- Long life keys should be generated from a truly random source, examples: Thermal noise, time between key strokes etc.
- **Basic idea of Pseudo-random Bit Generators:**
- **Extend truly generated random number of length k to length t, where $t > k$.**

Keys and Attacks

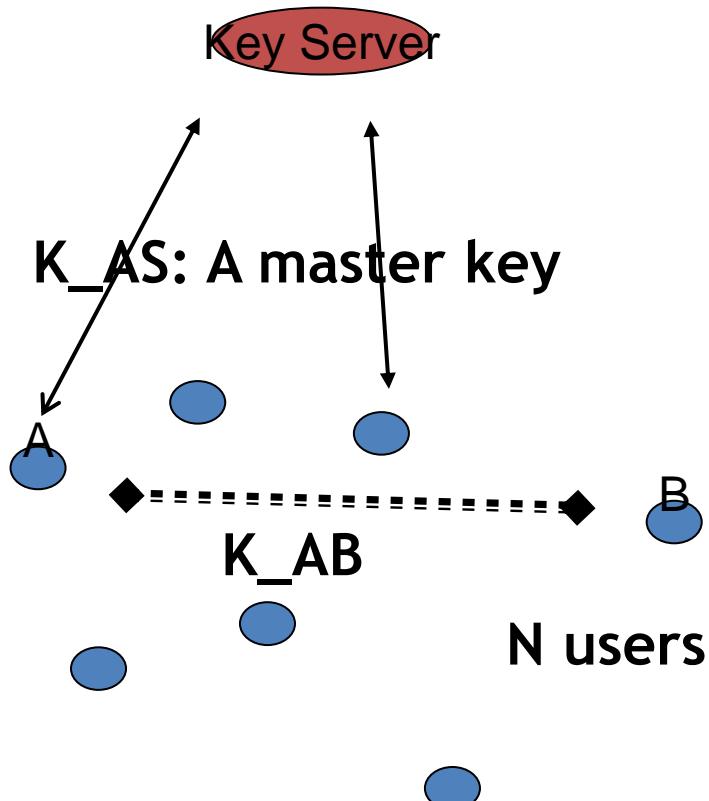


- key's life cycle has many functions:
 - Generation or Creation
 - Distribution
 - Storage and Maintenance
 - Revocation/ Disposal
- Each of the above stages is an opportunity for attackers.
- We focus on symmetric key management in this lecture.

Symmetric Key Distribution

- Consider symmetric key encryption system involving Alice and Bob.
- We studied that the security of Symmetric key systems is based on the following requirements:
 - The encryption algorithm does not have any weakness
 - The secret key is private to the sender and receiver.
- What are the methods of key distribution?
- The users can meet in advance, but this is impracticable as the users are heterogenous in communication networks.
- So we need some sort of key distribution framework as the keys need to be private between users and their access must be denied to others in the network.
- Also the keys need to be frequently changed to minimize the risk of attack.
- In practice, failures in secure system are result of vulnerabilities in key management rather than the weaknesses of encryption functions.

Symmetric Key Distribution



- Main Question:
- A and B do not share a secret channel.
- How to distribute a secret session key K_{AB} to parties A and B?
-
- Requirement:
- A new session key is needed for each new session and for each new security function
- (authentication, data confidentiality, integrity)

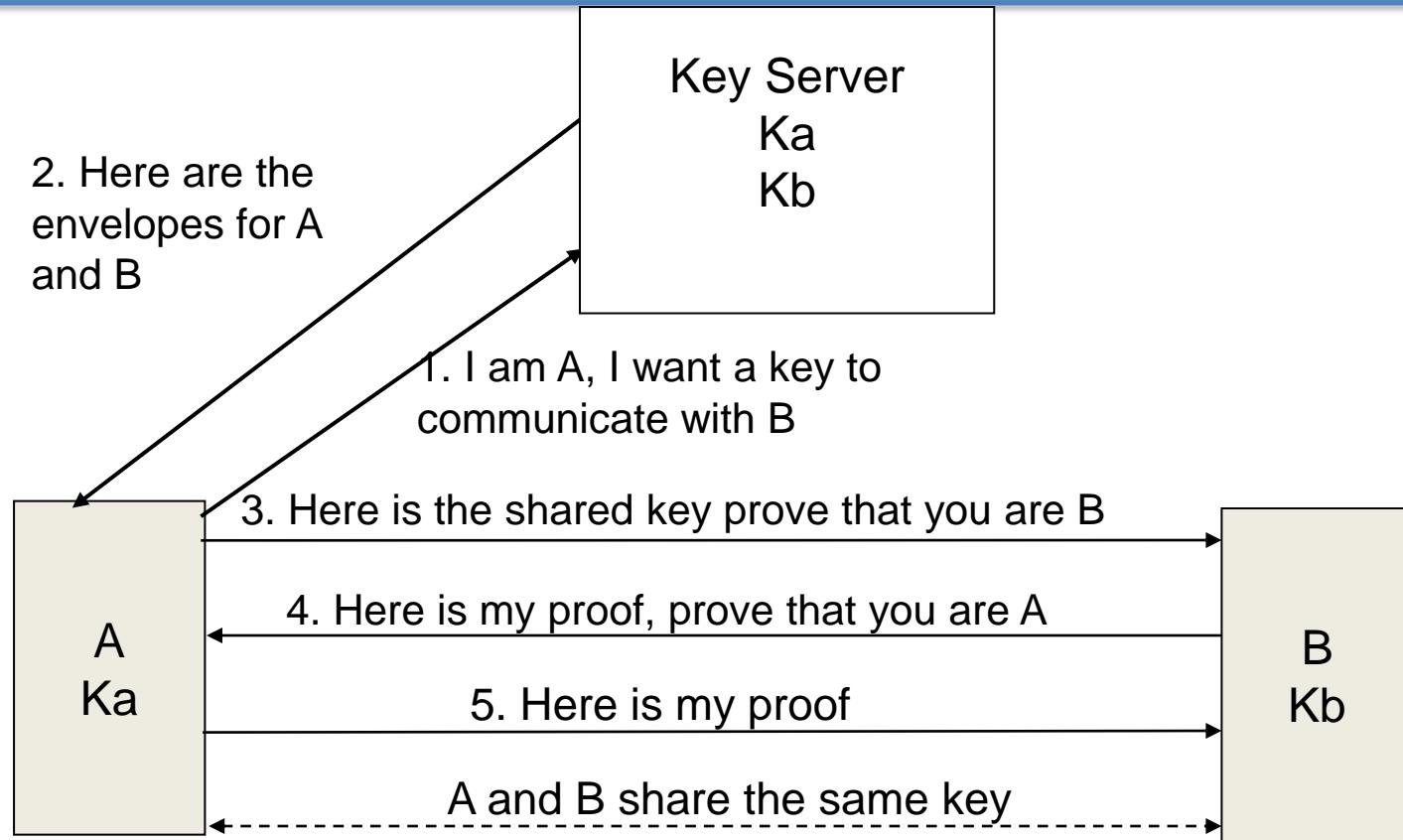
Different Methods

- N users
- Direct sharing between nodes : Complexity of initial key installation : $O(N^2)$
- Public Key Algorithmic Method (Diffie-Hellman Protocols or using RSA) : $O(N)$
- Using a Key server ($O(N)$)

Methods of Key Distribution

- Stallings lists the following four alternatives:
 1. A can select key and physically deliver to B
 2. A third party can select & deliver key to A & B
 3. If A & B have communicated previously can use previous key to encrypt a new key
 4. If A & B have secure communications with a third party C, C can relay key between A & B

Key Distribution using a key Server



A and B need to trust the key server:

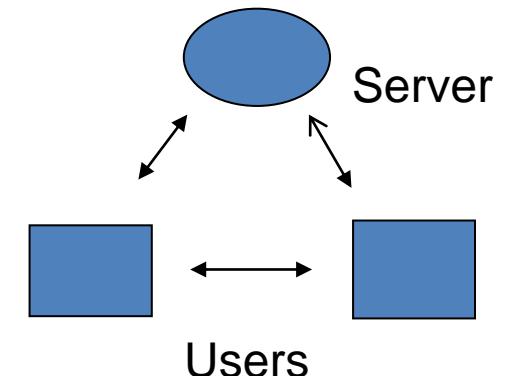
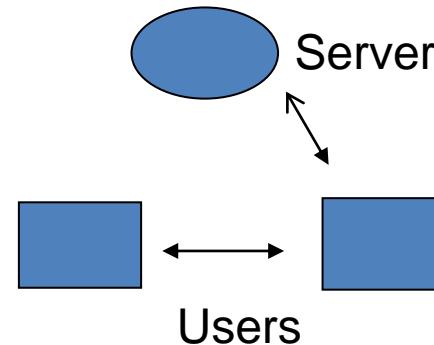
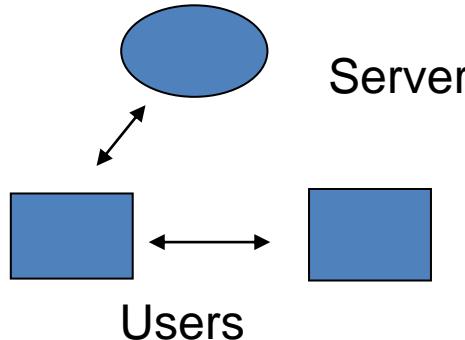
- security of session keys
- entity authentication

Server Based protocols

- Needham-Schroeder Protocol
 - Problems with freshness of session keys
- Otway-Rees Protocol
 - Use active authentication to avoid problems with using old keys

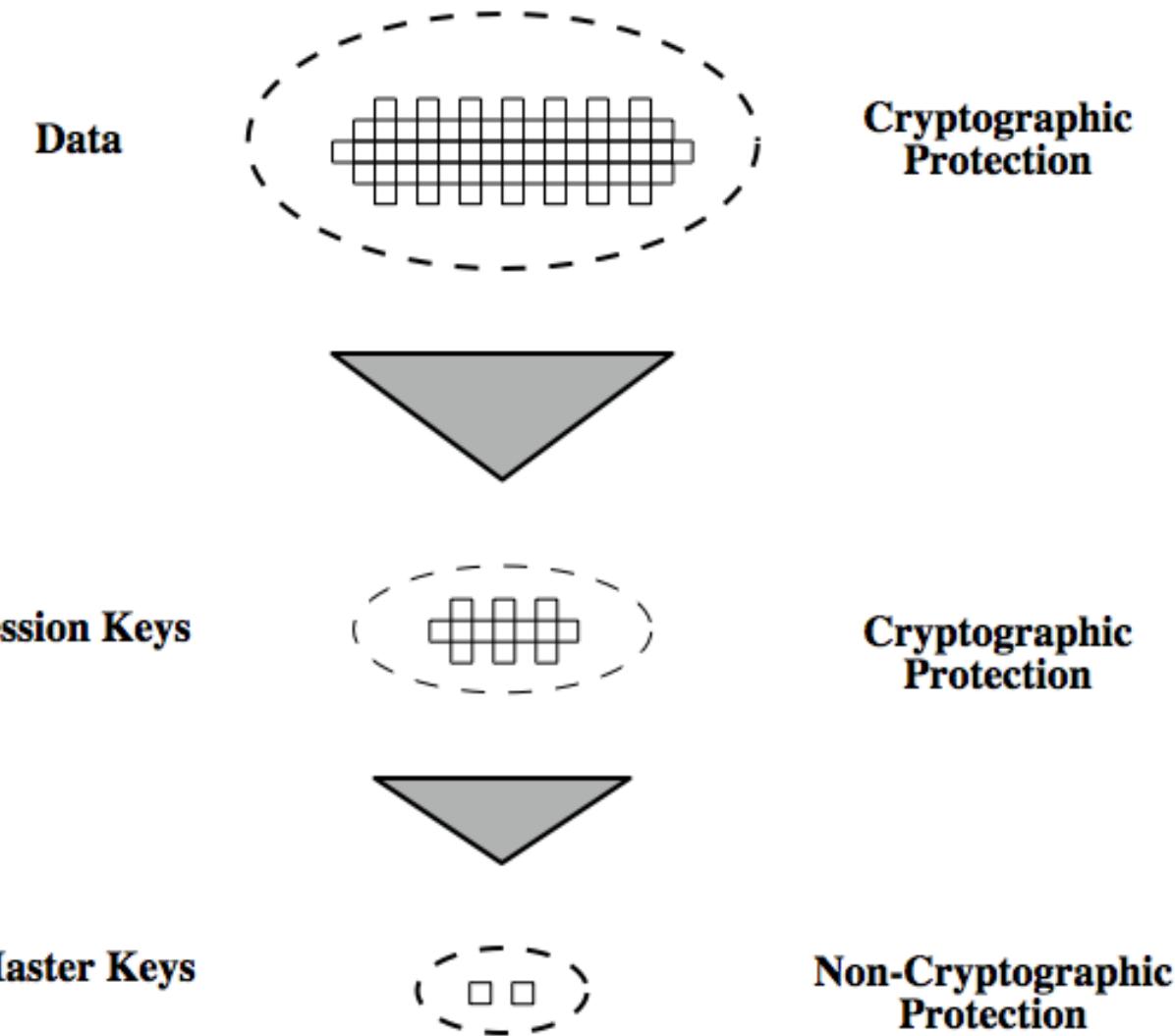
We will only follow the protocols given in the textbook.

Handbook of Applied Cryptography by Menezes et. Al. gives different possibilities of Key Server and user interactions:

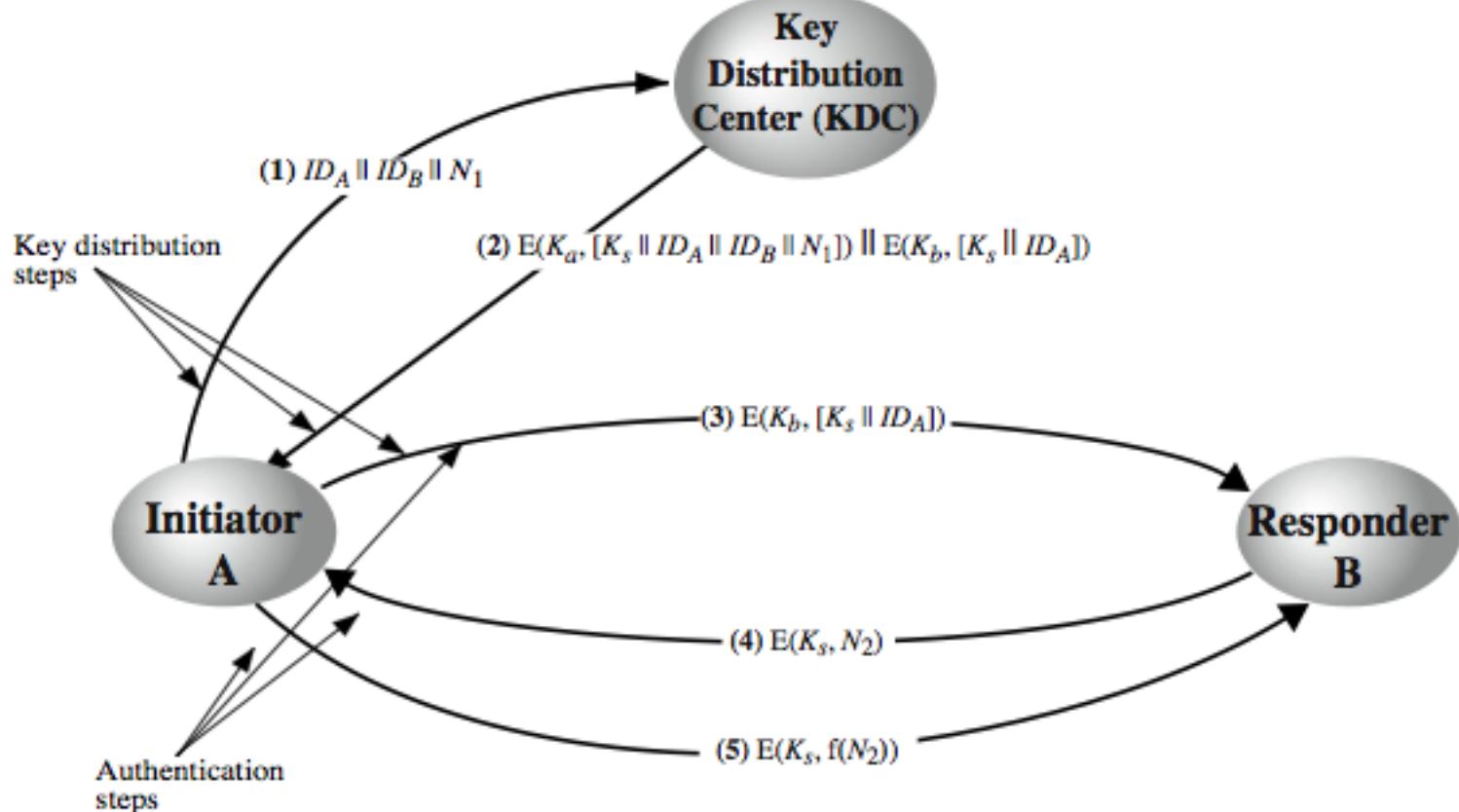


Key Hierarchy

- **Session key**
 - temporary key
 - used for encryption of data between users
 - for one logical session then discarded
- **Master key**
 - used to encrypt session keys
 - shared by user & key distribution centre



Key Distribution Scenario

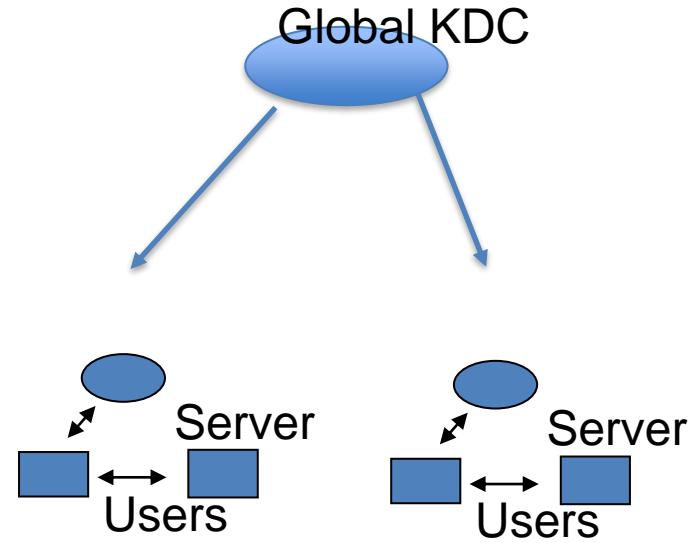


This is a version of Needham-Schroeder Protocol
used by Stallings

From the textbook a version of Fig 14.3

Hierarchical Key Control

- Key distribution method can be extended to multiple KDCs, a local KDC and a global KDC.
- You can have a hierarchy of KDCs.
- Users within a same local domain are supported by the local server,
- Users in two different domain will need involve global KDC to exchange keys.
- Hierarchy of keys minimizes complexity of key distribution. Also localizes the risk of fault or compromise within a local domain.



Session Key Lifetime

- How often session keys should be changed?
- There are two considerations: frequent changes
 - increases security
 - but adds delay to the communication and hence reduces network capacity.
- The textbook explains issues with respect to connection oriented protocols in detail, please read Section 14.1
- Key control is an interesting practical topic.

Decentralized Key Control

- In the previous method we need to trust KDC which needs a protection from being compromised.
- A fully decentralized approach may require every node establish a master key with every other node, thus needs $n(n-1)/2$ keys for n end point system.
- A session key algorithm is explained in Fig 14.5 of the textbook.

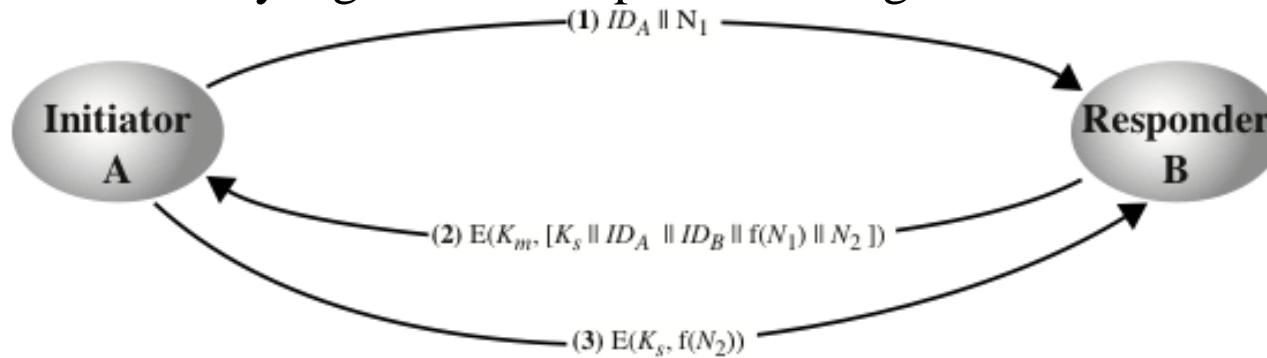


Figure 14.5 Decentralized Key Distribution

From the textbook a version of Fig 14.5

Controlling Key Usage

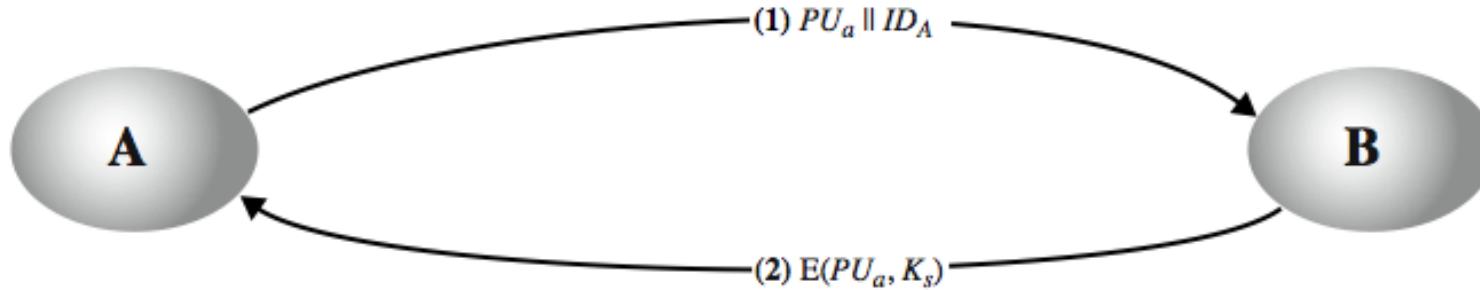
- Having some control on the nature of keys will help to manage them effectively.
- It may help to identify different types of keys:
 - Data Encrypting keys (DEK) for general communication across networks
 - PIN-encrypting Key for PINS and Electronic transfer applications
 - File Encrypting Key, for accessing files in public locations.
- The textbook suggests a method based on DES keys using 8 reserved bits, but tag length is limited.
- A more flexible scheme is depicted in Fig 14.6 of the textbook.

Using Asymmetric Encryption

- Public key was introduced with a promise to reduce the key usage.
- Can we use public key system to exchange session keys?
- Yes, you could do. But only problem is that the end points need to have the authentic public key of the other node.
- Also, in general public key encryptions are slower than symmetric key counterparts and hence they are not used for direct encryption. In general, always a hybrid scheme is employed-first use public key to obtain a session key and then use the session key in a symmetric encryption for efficiency.
- Let us consider a simple idea due to Merkle.

Merkle Key Distribution

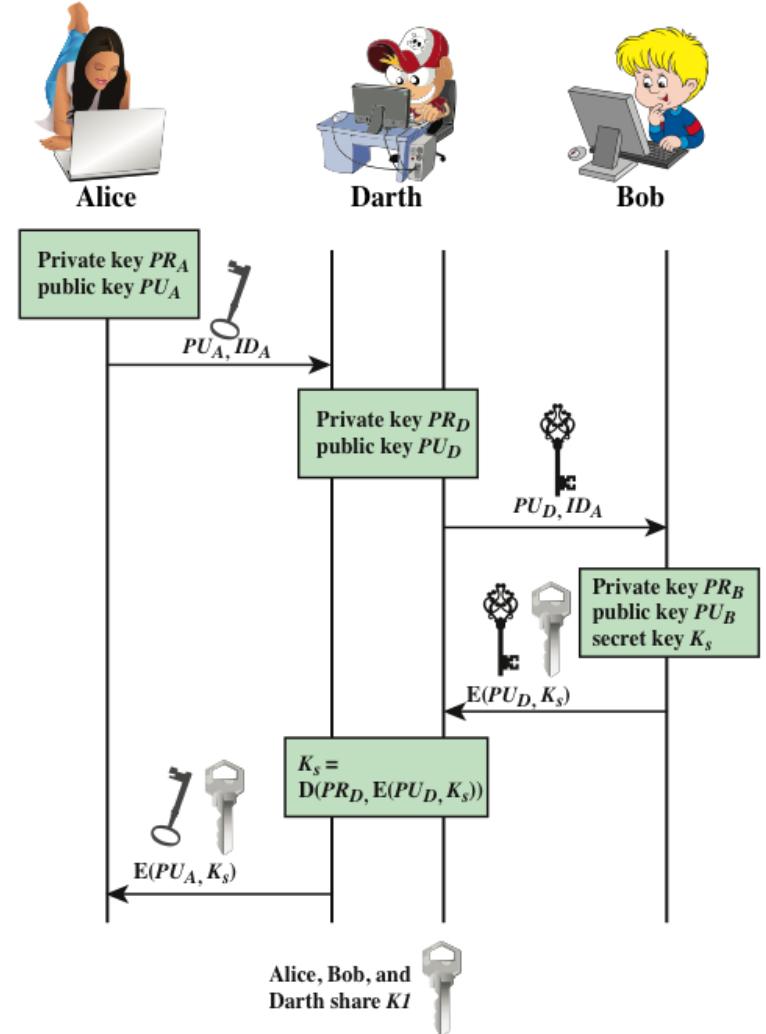
1. A generates a public/private key pair $[PU_a, PR_a]$ and transmits a message to B consisting of PU_a and an identifier of $[A, ID_A]$
2. B generates a secret key, K_s , and transmits it to A, encrypted with A's public key.
3. A computes $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s .
4. A discards PU_a and PR_a and B discards PU_a .



From the textbook a version of Fig 14.7

Merkle's Scheme

- Was the previous Scheme Secure?
- Main issue is authentication of user's public key, vulnerable to Main in the Middle Attack.
- To stop such attacks, users need a method for authentication of public keys.



From the textbook a version of Fig 14.8

Figure 14.8 Another Man-in-the-Middle Attack

Hybrid Scheme

- IBM main frames use this method based on public key.
- The scheme retains a KDC that shares a master key with each user.
- A public key scheme is used to distribute the master keys.
- The consideration is mainly on performance and backward compatibility.

Random Numbers

- Where **random numbers** are used in cryptography?
 - Session keys
 - Authentication protocols to prevent replay
 - Public key generation-SSL
 - Keystream for a one-time pad
- What are the properties you are looking for in these random numbers?
 - Having Uniform distribution, Statistically random, independent
 - Unpredictability of future values from previous values

Pseudorandom Number Generators (PRNGs)

- Deterministic algorithm techniques of generating random like objects from a seed or another random source.
- They are NOT truly random, but satisfies many tests for statistical randomness.
- They are thus called Pseudorandom numbers.

Linear Congruential Generator

- Most common method used in practice, part of many OS implementations.
- The sequences follow a linear relation:
$$X_{n+1} = (aX_n + c) \bmod m,$$
where X_0 is the seed and a and c are constants.
- If choose appropriate a and c and X_0 , you obtain a long sequence which looks random.
- Very useful in practice- Main criteria:
 - function generates a full-period
 - generated sequence should appear random
 - efficient implementation with 32-bit arithmetic
- The method is not secure since an attacker can reconstruct sequence given a small number of values.

Using Block Ciphers as PRNGs

- Stallings discussed many techniques based on block ciphers to generate random numbers.

- Popular ones:

- Counter Mode

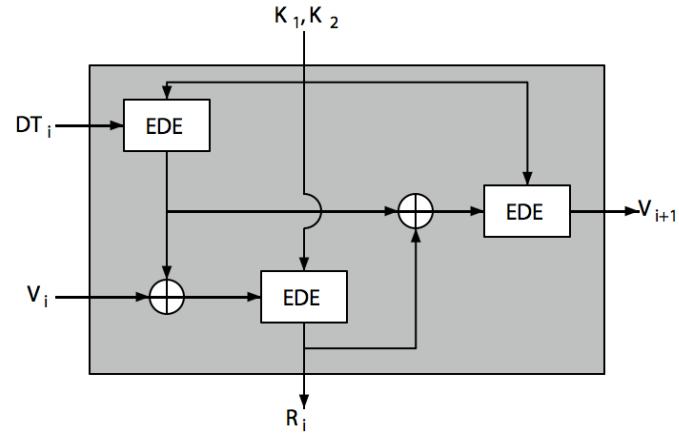
$$X_i = E_{Km}[i],$$

- Output Feedback Mode

$$X_i = E_{Km}[X_{i-1}]$$

ANSI X9.17 PRG

- DT_i - Date/time value at the beginning of ith generation stage
- V_i - Seed value at the beginning of ith generation stage
- R_i - Pseudorandom number produced by the ith generation stage
- K₁, K₂ - DES keys used for each stage
- Then compute successive values as:
- $R_i = EDE([K_1, K_2], [V_i \text{ XOR } EDE([K_1, K_2], DT_i)])$
- $V_{i+1} = EDE([K_1, K_2], [R_i \text{ XOR } EDE([K_1, K_2], DT_i)])$



From the textbook

Blum Blum Shub Generator

- This is an interesting generator based on public key cryptography.
- You start with a seed mod n and the sequence are obtained as:
- $x_i = \text{LSB}(x_{i-1}^2) \bmod n$, where $n=p \cdot q$, and primes $p, q \equiv 3 \pmod{4}$, LSB: Least Significant Bit.
- Interestingly this algorithm has high level of security,
- If someone can break the “next-bit” test, then you can use him to break the factorization of n, which is considered hard-so it has strongest public proof of its strength.
- However it is very slow.

Natural Random Noise

- One can use known natural randomness in real-world to your advantage.
- There are many events which look random: radiation counters, radio noise, audio noise, thermal noise in diodes, leaky capacitors, mercury discharge tubes etc.
- Please read Section 8.6 of the textbook.

Week 7



Lecture 1

Key Management

Lecture 2

MST

Workshop 7: Workshop based on Lectures in Week 6

Quiz 7

Week 8

Lecture 1

Key Management (Public Key)

Lecture 2

Finite Fields and ElGamal Encryption

Workshop 8: Workshop based on Lectures in Week 7

Quiz 8

ElGamal Encryption (Public Key)

COMP90043

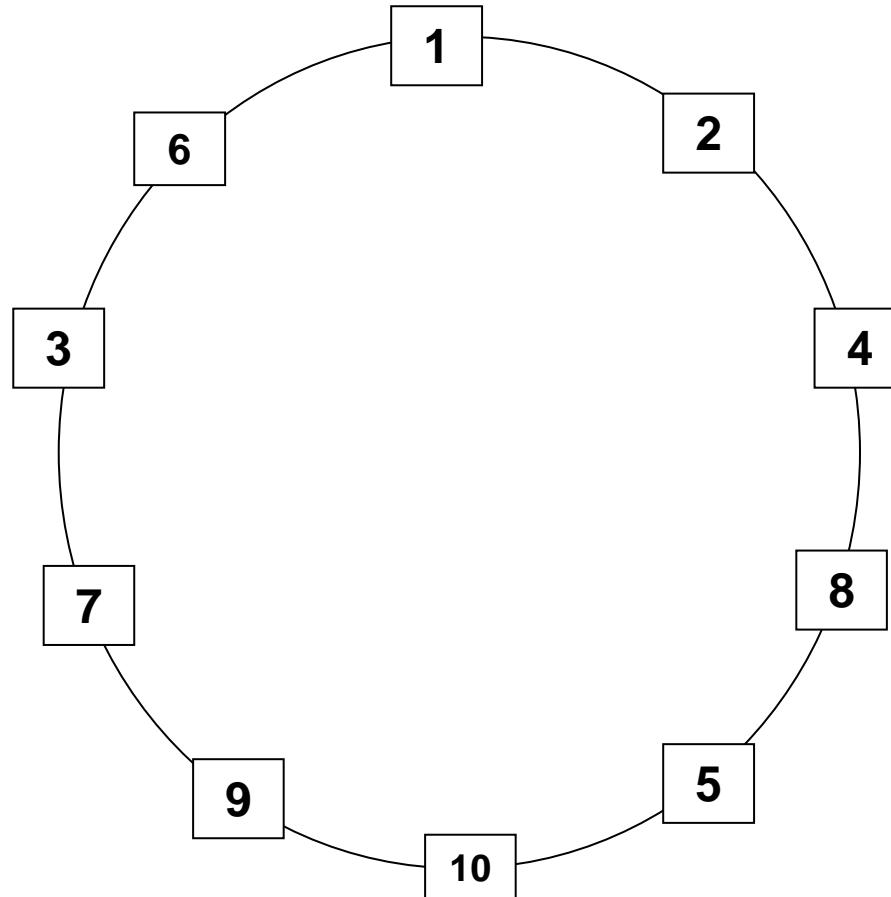
Lecture 1

Lecture 2

1.1 ElGamal Encryption

- DH Protocol to Encryption
- Basic Idea
- Example
- Security Properties

Recap: Cyclic Groups



Example of a Cyclic group modulo $p = 11$

g : generator = 2

Order(size) of G = 10

14/09/2022

g^i	$g^i \text{ mod } p$	$D\log(g^i)$
2^1	2	1
2^2	4	2
2^3	8	3
2^4	5	4
2^5	10	5
2^6	9	6
2^7	7	7
2^8	3	8
2^9	6	9
2^{10}	1	10

Cyclic Groups

- Z_n : Integers modulo n, n is a positive integer, under multiplication
- Z_p : Integer modulo p, p is a prime number, under multiplication
- Residues of Polynomials over Z_p .
- Elliptic Curves over Z_p .
- Some examples of cyclic groups present inside a bigger groups:
 - $C_8 : \{2,4,8,16,15,13,9,1\}$ of size (cardinality) 8, 2 is the generator, operation mod 17,
 - $C_{30} : \{2,4,8,16,1\}$ of size (cardinality) 5, 2 is the generator, operation mod 31

Order of Cyclic Groups

- What is the maximum size of cyclic groups obtained from Z_p , p , a prime number?
- $(p-1)$
- What is the maximum size of cyclic groups obtained from Z_n ?
- $\phi(n) = \text{Numbers of integers } < n \text{ but relatively prime to } n.$
- What is the maximum size of cyclic groups obtained from $Z_p[x] \bmod m(x)$, $\deg(m(x)) = k$?
- $P^k - 1$.
- In fact, we can have groups whose size divides the sizes mentioned above.

Cyclic subgroups mod p

The powers of each element generate a cyclic subgroup mod 11:

$$\langle 1 \rangle = \{1\}$$

$$\langle 2 \rangle = \{2, 4, 8, 5, 10, 9, 7, 3, 6, 1\}$$

$$\langle 3 \rangle = \{3, 9, 5, 4, 1\}$$

$$\langle 4 \rangle = \{4, 5, 9, 3, 1\}$$

$$\langle 5 \rangle = \{5, 3, 4, 9, 1\}$$

$$\langle 6 \rangle = \{6, 3, 7, 9, 10, 5, 8, 4, 2, 1\}$$

$$\langle 7 \rangle = \{7, 5, 2, 3, 10, 4, 6, 9, 8, 1\}$$

$$\langle 8 \rangle = \{8, 9, 6, 4, 10, 3, 2, 5, 7, 1\}$$

$$\langle 9 \rangle = \{9, 4, 3, 5, 1\}$$

$$\langle 10 \rangle = \{10, 1\}$$

Choosing a generator

x	$2^x \pmod{11}$	$3^x \pmod{11}$	$10^x \pmod{11}$
0	1	1	1
1	2	3	10
2	4	9	1
3	8	5	10
4	5	4	1
5	10	1	10
6	9	3	1
7	7	9	10
8	3	5	1
9	6	4	10
10	1	1	1
11	2	3	10

- 2 is a primitive element.
 - It generates the entire group
- 3,10 are not primitive elements
 - They generate subgroups of size 5, 2
- Typically, in cryptography, the generator of a large prime-order subgroup is used
- Easiest case: use a **safe prime** $p = 2q+1$ for a large prime q
- Every element either generates a group of order q , or 2 elements
- Often the modulus p and generator g are chosen carefully and published

The discrete log problem

X	$2^x \bmod 11$
0	1
1	2
2	4
3	8
4	5
5	10
6	9
7	7
8	3
9	6
10	1
11	2

- For $p=11$ the problem is trivial (brute force the table here)
- For larger p , generic algorithms exist which beat brute force
 - "Baby step giant step"
 - Pollard's Rho
- Specific algorithms work even better when working mod p
 - Index Calculus
- As a result, p must be > 1024 bits
 - 1024-bit might be attacked by a nation-state adversary with precomputation
- The industry has largely moved to elliptic curves (EC)
 - Different cyclic group, same DL problem
 - Secure at much smaller sizes

Recap: Diffie-Hellman Protocol

Public Parameters: g: generator, order of the cyclic group: (p-1), prime: p

Alice

Choose $N_a = 2$

$$g^{N_a} = 2^2 = 4 = M_a$$

Compute

$$K_{ab} = M_b^{N_a} = 9^2 = 4$$

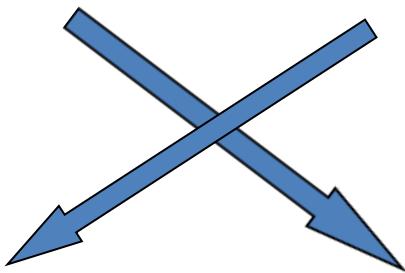
Bob

Choose $N_b = 6$

$$g^{N_b} = 2^6 = 9 = M_b$$

Compute

$$K_{ba} = M_a^{N_b} = 4^6 = 4$$



$$K_{ab} = K_{ba} = 4$$

The Diffie-Hellman problem

Computational (CDH):

- Given p , g , g^a , g^b , can you compute g^{ab} ?

Decisional (DDH):

- Given p , g , g^a , g^b , h , is $h=g^{ab}$?

Both problems *reduce* to solving discrete logs (DLP)

DDH *reduces* to CDH

But an algorithm might exist that solves these, but not DLP

ElGamal Cryptosystem

- Invented by ElGamal in 1985.
- Closely related to Diffie-Hellman (DH).



Taher ElGamal

- Simple observation: do Alice, Bob both need to be online for DH?
 - What if one party publishes a share in advance?
 - Is it safe to re-use a share across multiple exchanges?

ElGamal setup

- We will use the notations as in the Stallings textbook
- Assume that one party in the DH protocol is fixed in advance.
- Assume computations mod q , q is a prime. “ a ”: generator of the group.
- Alice generates her key pair in advance
 - chooses her **secret key**: $1 < x_A < q-1$
 - compute her **public key**: $y_A = a^{x_A} \text{ mod } q$
- Bob somehow learns this in advance, as we discussed last lecture

ElGamal's variant of Diffie-Hellman

Bob:

- Choose a random k and compute $a^k \bmod q$
- **Send $a^k \bmod q$ to Alice**
- Using Alice's public key y_A , compute $y_A^k = a^{k x_A}$
- Encrypt the message for Alice as $C = M a^{k x_A}$

Alice:

- Using her private key x_A , compute $(a^k)^{x_A} = y_A^k$
- Decrypt message $M = C / a^{k x_A}$

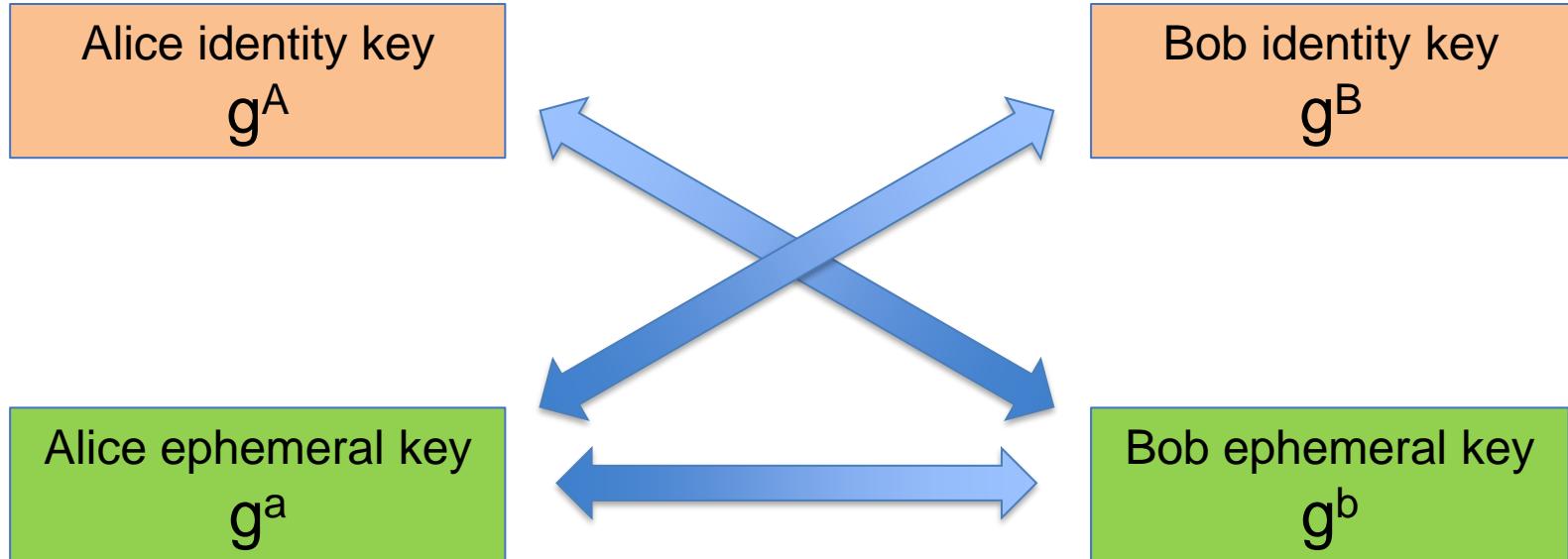
ElGamal in more detail

- Another user (eg Bob) can encrypt a message to send to A by computing the steps below:
 - Represent message M in range $0 \leq M \leq q-1$
 - Choose random integer k with $1 \leq k \leq q-1$
 - Compute one-time key $K = y_A^k \bmod q$
 - Encrypt M as a pair of integers (C_1, C_2) where
 - $C_1 = a^k \bmod q$; $C_2 = KM \bmod q$
- Alice can then perform decryption as follows:
 - Recovering the key K as $K = C_1^{x_A} \bmod q$
 - Compute M as $M = C_2 K^{-1} \bmod q$
- What if Bob re-uses k for multiple messages?

Why random keys are important

- Let $(M_1, C_1 = [C_{11}, C_{12}])$ and $(M_2, C_2 = [C_{21}, C_{22}])$ be two message and ciphertext pairs using the same randomization parameter k .
- This implies that: $C_{11} = a^k \bmod q = C_{21} = a^k \bmod q$
- If the adversary knows M_1 , he can recover $C_{11} = C_{12}/M_1$
- Using C_{11} , he can then recover $M_2 = C_{22}/C_{11}$ 
- Hence k should be random and independent for all encryptions.

Modern case study: 3DH in Signal



Each side computes shared secret:

$$K = H(g^{ab} \parallel g^{aB} \parallel g^{Ab})$$

Modern case study: 3DH in Signal

$$K = H(g^{ab} \parallel g^{aB} \parallel g^{Ab})$$

Properties:

- **Mutual authentication**-both parties convinced other party knows long-term identity secret key
- **Forward secrecy**-both parties can delete ephemeral keys immediately
- **Deniability**-anybody can *simulate* a handshake between Alice, Bob by picking a, b (don't need to know long-term secrets A, B)
- **Asynchronous delivery**-achieved by caching "prekeys" for one side
- **Efficiency**- uses elliptic curves (Curve25519)

Week 8

Lecture 1

Key Management (Public Key)

Lecture 2

Finite Fields and ElGamal Encryption

Workshop 8: Workshop based on Lectures in Week 7

Quiz 8

The scheme ElGamal Cryptography

- Now we give the actual ElGamal Public-key cryptosystem.
- Main tool is exponentiation in a cyclic group where the DLOG is hard.
- As you will see, the security is directly related to difficulty of computing discrete logarithms.
- As before, each user (eg. Alice) generates their key
 - chooses a secret key (number): $1 < x_A < q-1$
 - compute their **public key**: $y_A = a^{x_A} \text{ mod } q$
- NOTE: a is the generator here.

Key Features

- DH protocol can be formulated over any cyclic group where computing discrete logarithm over the group is hard.
- What is the main objective?
 - Two users connected over insecure channel arrive at a common secret by using only public parameters.
 - In our case, they arrive at $g^{(ab)}$, g is a generator of the group; a, b are random secrets chosen by the participants respectively.

Week 8



Lecture 1

Key Management (Public Key)

Lecture 2

Finite Fields and ElGamal Encryption

Workshop 8: Workshop based on Lectures in Week 7

Quiz 8

Key Management (Public Key)

COMP90043

Lecture 1

Lecture 1

1.1 Public Key Management

- Public Key Address and Distribution
- Four different methods
 - Public Announcement
 - Public Key Authority
 - Public Key Certificates and Revocation.
 - Public Key Infrastructure

1.2 Modern Developments

- Mobile Public Key Management.

Recap

The Figure Illustrates the notations
And use of Public Key functions;
We will use this notation
throughout this semester.

Public key of B : Pu_b
Private key of B : PR_b

Encryption and Decryption by A

$$Y = E(PU_b, X)$$

$$X := D(PR_b, Y)$$

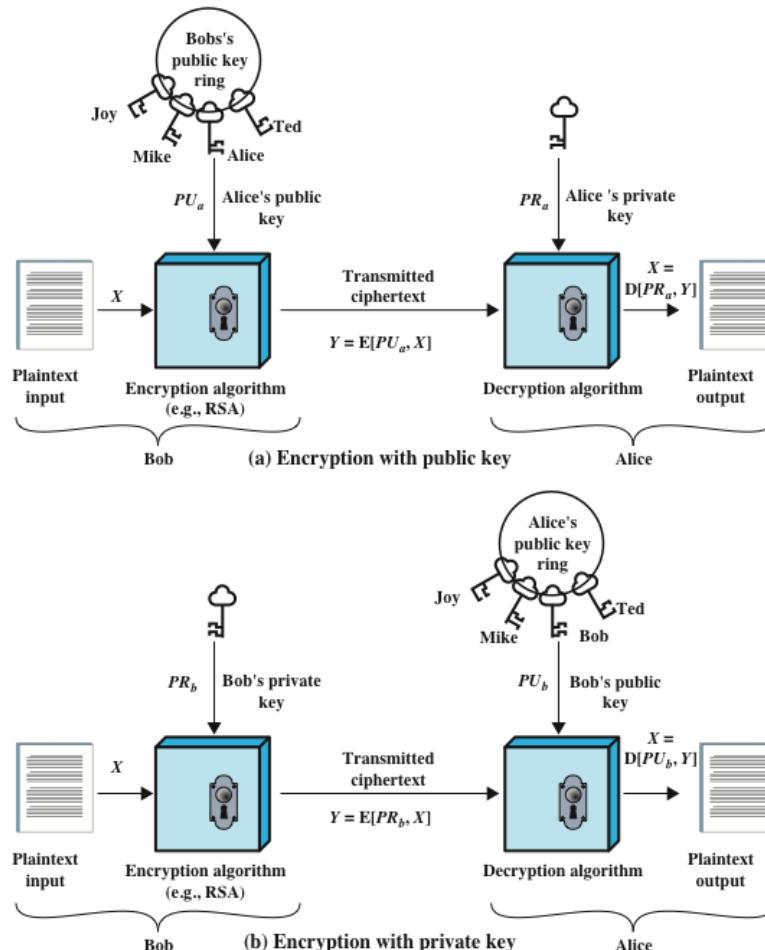
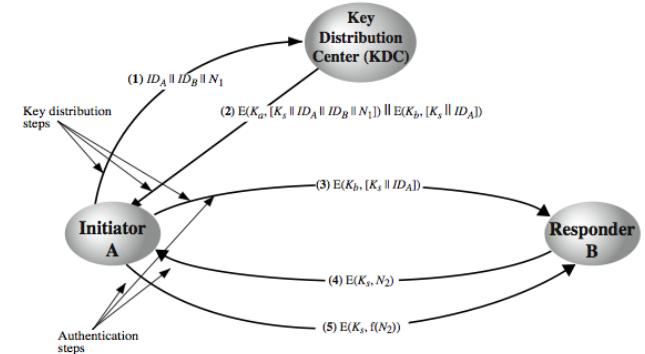


Figure 9.1 Public-Key Cryptography

Recap: Symmetric Key Distribution

- Using Symmetric Key Encryption:
 - Physically share keys between Alice and Bob
 - Third party delivering keys to Alice and Bob
 - Use previously agreed secret to start new secret communication
 - A trusted third-party (Key Distribution Centre (KDC) relaying keys when needed (Needham Schroeder protocol)
- Using Public Key encryption
 - Merkle Key Distribution
 - Problem: Active adversary can modify transmitted key



Recap: Middleperson attacks

- Both users convinced they ran Merkle's protocol with each other
- Middle person (Man-in-the middle) can read all traffic
- This requires an **active adversary**
- Fix: digital signatures

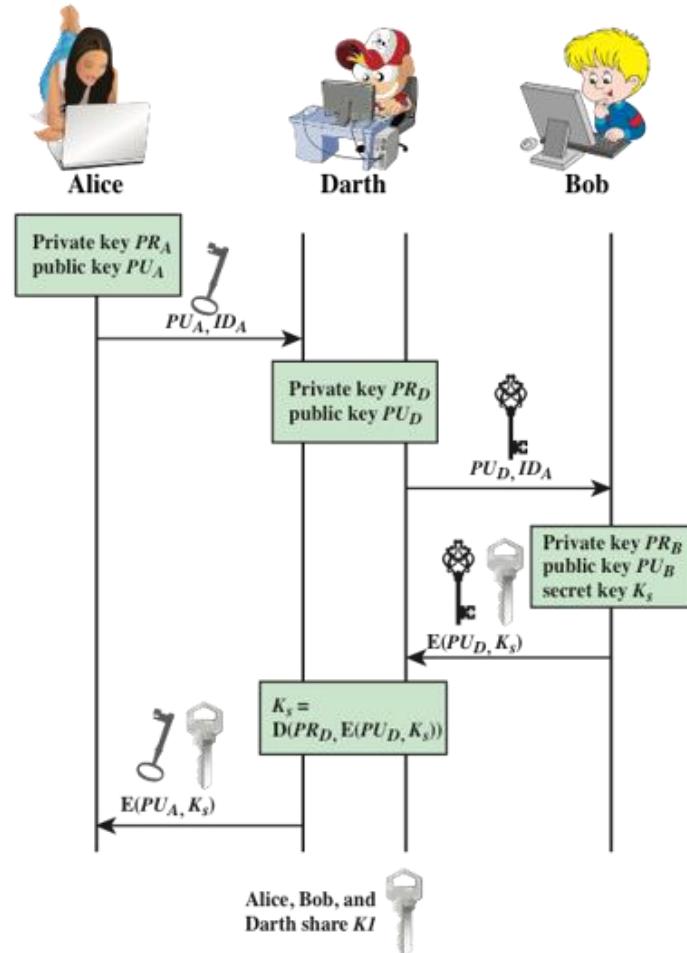


Figure 14.8 Another Man-in-the-Middle Attack

Recap: General Digital Signatures

- We mainly discussed RSA signature scheme, where the encryption and signature uses the same primitive Exponential Modulo n. In general,
- A Digital signature scheme has the following structure
 - **Keygen()** (public key, private key)
 - **Sign(private key, message)** signature
 - **Verify(message, signature, public key)** yes/no
- The security usually comes from certain assumptions:
 - RSA-based (difficulty of factorization)
 - DSA/Schnor (difficulty of discrete logs)
- In practice, You always sign the hash of the actual message for efficiency.

Key distribution with signatures

- There are several variants of the scheme:
- Option 1: Hybrid encryption
 - Alice chooses secret key k_{A-B}
 - Alice encrypts k_{A-B} using Bob's public key PK_B
 - Alice signs the result using her private signature key
 - Alice sends $\text{Sign}(K_A, \text{Encrypt}(PK_B, k_{A-B}))$

Key distribution with signatures

- Option 2: Diffie-Hellman with signatures
 - Alice, Bob each choose DH share g^a, g^b
 - Alice sends $\text{Sign}(K_A, g^a)$, Bob sends $\text{Sign}(K_B, g^b)$
 - Both compute $k_{A-B} = \text{KDF}(g^{ab})$
- In some scenarios, only one side signs (e.g. web browsing)
- Advantage: You obtain what is known as *forward secrecy*

Forward secrecy

- What is *forward secrecy*?
- Consider an adversary eavesdropping on Alice, Bob's key distribution
 - Hybrid encryption: adversary sees $\text{Sign}(K_A, \text{Encrypt}(PK_B, k_{A-B}))$
 - DH exchange: adversary sees $\text{Sign}(K_A, g^a)$, $\text{Sign}(K_B, g^b)$
- The latter is secure even if the adversary later breaks K_A/K_B
 - Assuming both parties discard secrets a, b
- Disadvantage: Both the parties must be online.

Nature of Public Key Address

- We saw an example of an RSA key:
- RSA-768: a 768-bit RSA modulus with 232-digit decimal representation:
 $n = 123018668453011775513049495838496272077285356959533479219732245215172640050726$
 $365751874520219978646938995647494277406384592519255732630345373154826850791702$
 $6122142913461670429214311602221240479274737794080665351419597459856902143413.$
- $e =$
 $1130495357977008075797356098754367899954235678098766789999557326303453731548268507917097747708653$
 $459798904775356794697870813$
- Can you make out if this belongs to an identity that you are familiar?
- In fact, the value looks random and we cannot conclude anything.
- In practice, let us look at the structure of the public key formatting we considered so far.
- If Alice has a public key PU, then we can represent as
- Alice : $\langle \text{IDA}=\text{Alice}, \text{Pu}_a = (n,e) \rangle$.
- If this is in public domain, anyone can make a modification: for eg:
- Trudy : $\langle \text{IDA}=\text{Trudy}, \text{Pu}_t = (n,e) \rangle$.
- How do you ascertain the correct identity?
- This is the authentication problem. This lecture will look into these issues.

Public Key Distribution Problem

- How does Alice advertise her public key so that Bob and others can use it to encrypt information to her?
 - Alice : < IDA=Alice, Pu_a = (n,e)>.
- Note that the above format may appear specific to RSA, but we can extend the idea by including explicit information about public parameters
- Alice : < IDA=Alice, Pu_a = (n,e), Algorithm Public Parameters>.

- Even if Alice signs the public address, as long as the public address is authenticated no one can believe that it belongs to Alice.

- Because as we saw before, any one can replace with a new public key and signature and masquerade as Alice.
 - Alice : < IDA=Alice, Pu_a = (n',e')>.

Public Key Distribution

- Stallings discusses four important methods:
 - Through Public announcement
 - By Using publicly available directory
 - With Public-key authority
 - Using Public-key certificates
- Most of the existing methods can be mapped to one of the above.

Notation



- We use the conventions associated with RSA schemes while explaining public key protocols.
- Public Address: **PU** Private Address: **PR**
- Public Key Encryption/Decryption:
 - Encryption: $E(PU, M) = C$;
 - Decryption: $M = E(PR, C)$
- Public Key Signature/Verification
- Signing:
 - $s = E(PR, M)$; (M, s) is a signature pair
- Verification
 - $M \text{ eq } E(PU, s)?$

NOTE: the notation $E(key, message)$ is used for symmetric key encryption also; the meaning depends on the context.

Through Public announcement

- A simple strategy, users distribute to those who need by any means (broadcasting or email etc)
- Example: PGP keys
- Main issue is that they can be easily forged as we explained before.



Figure 14.10 Uncontrolled Public Key Distribution

From the textbook Fig 14.10

Pretty Good Privacy

- One of early applications of Public key cryptography.
- Users can exchange public keys through email or other communication means.
- Using Fingerprints is a useful technique. Here a cryptographic hash function is applied to generate a short fingerprint corresponding to a long public key.

Key fingerprints make distribution easier

- Public keys are long and not human-friendly
- Example: (HTTPS key for unimelb.edu.au):
 - N =
0xb0b75c9b3580b81ed3bcddaa51a1a4852be7c471c278fbbae4d222c09d34d227afeb97c7e9fce1223d9a276085dd4860b2fd999d87505052c491d44906e639b7c98f7edca8f8af685970f228ee1ced048825219581970abbd08abb24620b3b4b325a3c9adc1f725b47f3b05c1e7551c0994aee0425db8a2392b0e67f6f7b01c68f3c1da46a004745ca05bef6150c0ab065022a961ec63ffaa2a2d966b30aa87b371a542803e29aa8500a01fd2d7d0e9687b9ea5519e80237a503d8c60aa3ddb2d61541ee936feb0d6a5314a54196e5eb1bce13023bb4f7e32295ac31031c5247e6c6b82ef329e7382b8501c9a975c85c6c0cbeaa2ace39348de93f581eee2ff11
 - e = 0x10001
- *Key fingerprint*: hash of public key, often truncated, formatted for humans
0xd339c2a9f78713b6d4b5aa6f33e9f13f5dc04396a6a08037aa8ab78c519a18be
- There are ways to simplify it to make it easy to compare by mapping to space of group of numbers,
- Also you may use third party trusted software for verification. Eg Cloudflare

Can we make key fingerprints memorable?

- In general, no. Otherwise anybody could create a key for any name
- *Identity-based encryption*: Public keys are short strings like "Alice" or "alice.com.au"
 - Issue: Only designated key-generating authority can find corresponding private keys
- *Vanity addresses*: search for key with fingerprint starting with some prefix

 facebookwkhpilnemxj7asaniu7vnjjbiltxjqhye3mhbshg7kx5tfyd.onion

Address

1LitecoiniwdBzUR84opxNeDggcT

By Using publicly available directory

- A directory service is established,
- Each user contacts the directory through secure means and places his public address to be downloaded by other users.
- Each user can update his public key and details. Think, why do you need this feature?
- Sometime keys may be compromised.
- Users can contact the directory electronically.
- Security is better than the previous method, but still vulnerable..

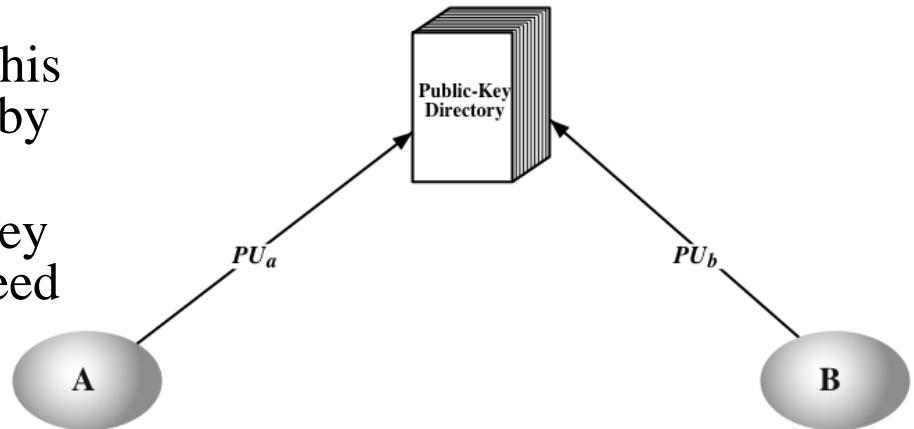


Figure 14.11 Public Key Publication

From the textbook Fig 14.11

With Public-key authority

- This method is a further improvement to the directory service. It has following properties:
- The authority server is always online with tight control over the distribution and maintenance of keys.
- Authority also has a public and private key: $\langle \text{PU}_{\text{auth}}, \text{PR}_{\text{auth}} \rangle$
- Users will contact the authority whenever they need key service.
- Example: OpenPGP Keyserver
- Issues:
 - Server needs to be online always.
 - Still there is a possibility of tampering and attacks.
- Next, we discuss the protocol as in the textbook:

Public-Key Authority: A simple scenario

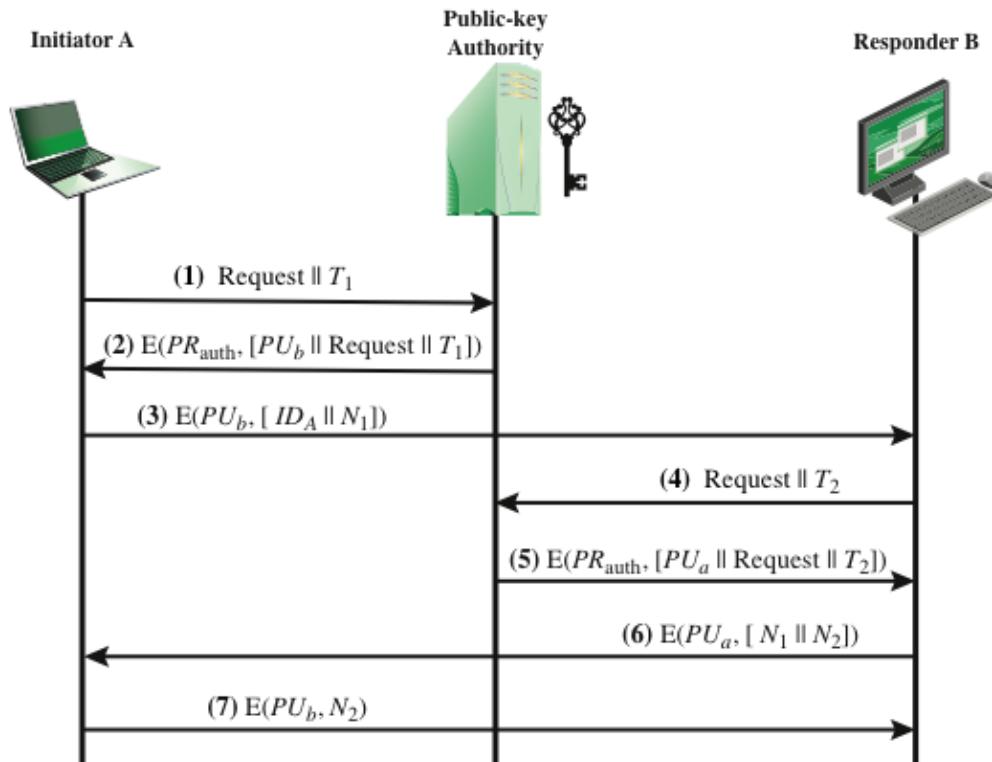


Figure 14.12 Public-Key Distribution Scenario

From the textbook Fig 14.12

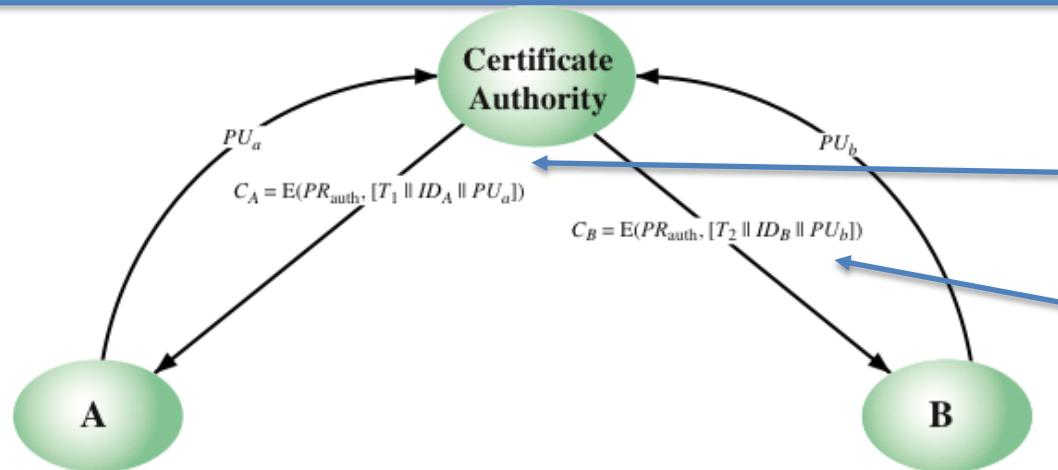
Using Public-key certificates

- This is the most sophisticated method, first suggested by Kohnfelder. Users get real access to keys.
- Here, key authority need not be online all the time, at least theoretically.
- What is a certificate?
- A form which binds identity of a users with its public key.
- The method allows others to verify the validity of the certificates.
- A certificate should have minimum of this form:
- Alice : < IDA=Alice, PU_A, Signature of PU_A by the Authority, PU_{auth} >.

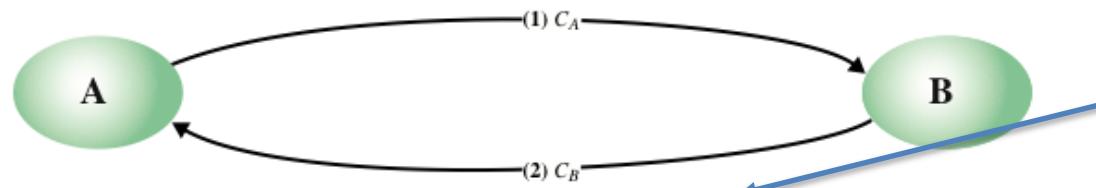
- **Name:** Alice
- **Public key:** 0x187fe35...
- **Certificate expiration:** 2021-12-31-23:59:59 UTC
- **Usage restrictions:** ...



Exchange of Certificates



(a) Obtaining certificates from CA



(b) Exchanging certificates

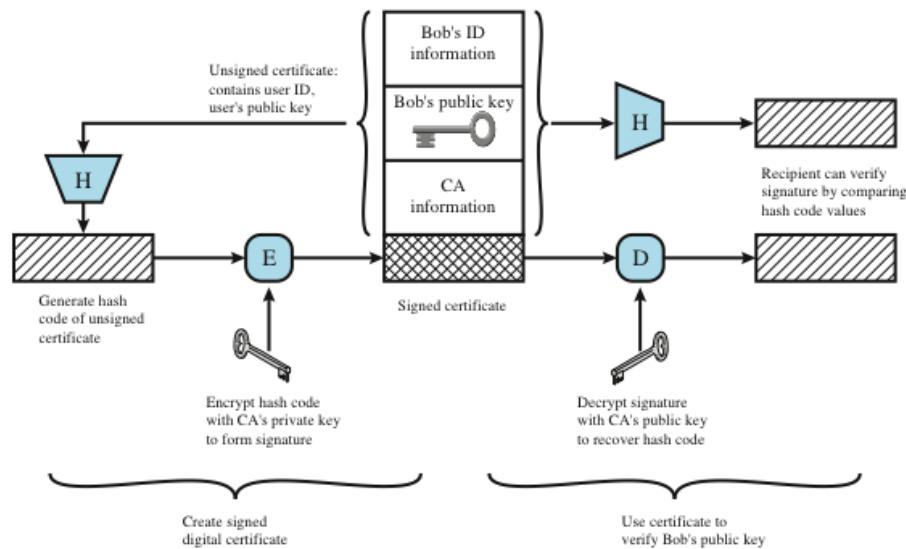
Look at the structure of the certificate,
we follow this method in
workshops and exam

Since A and B already
have a trusted relation
with the authority, the public
Keys are implicitly authenticated
After the exchange.

From the textbook Fig 14.13

Figure 14.13 Exchange of Public-Key Certificates

X.509 Certificates



Read Section 14.4 for the details of X.509 certificates

From the textbook Fig 14.14

Figure 14.14 Public-Key Certificate Use

X.509 Structure

- The standard notation for a certificate of:

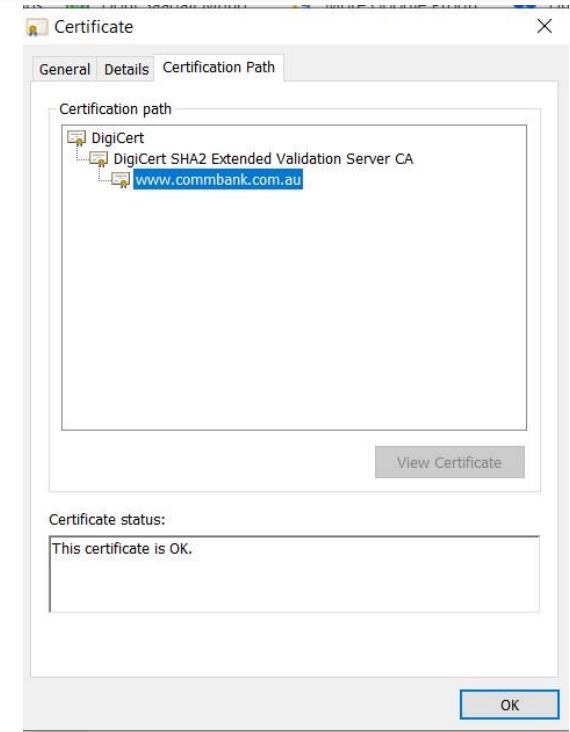
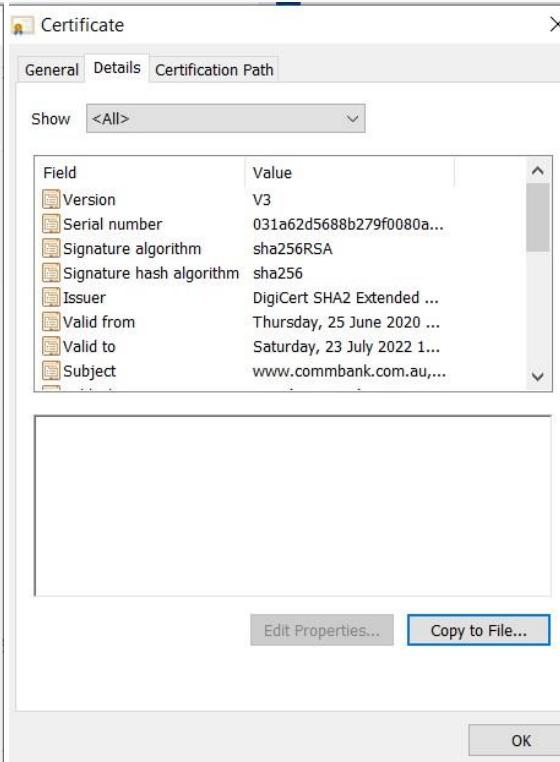
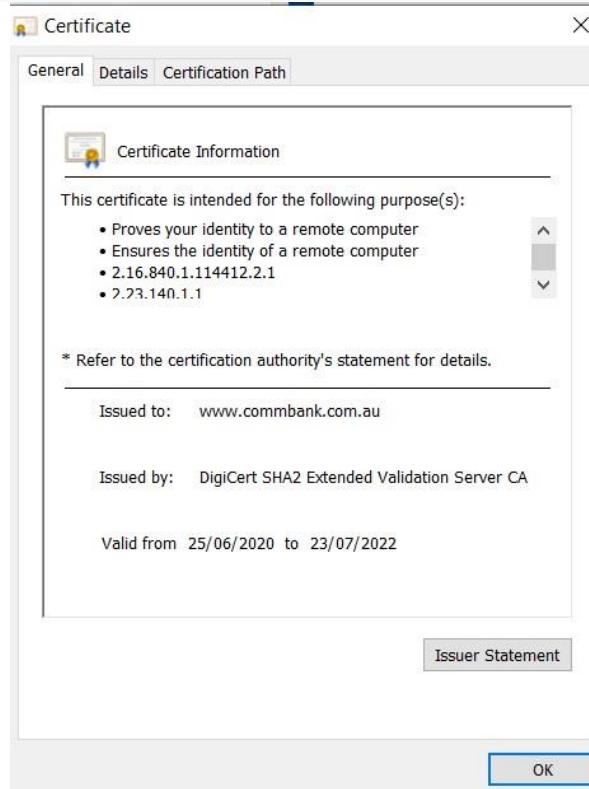
CA<<A>>

= CA {V, SN, AI, CA, UCA, A, UA, Ap, TA}.

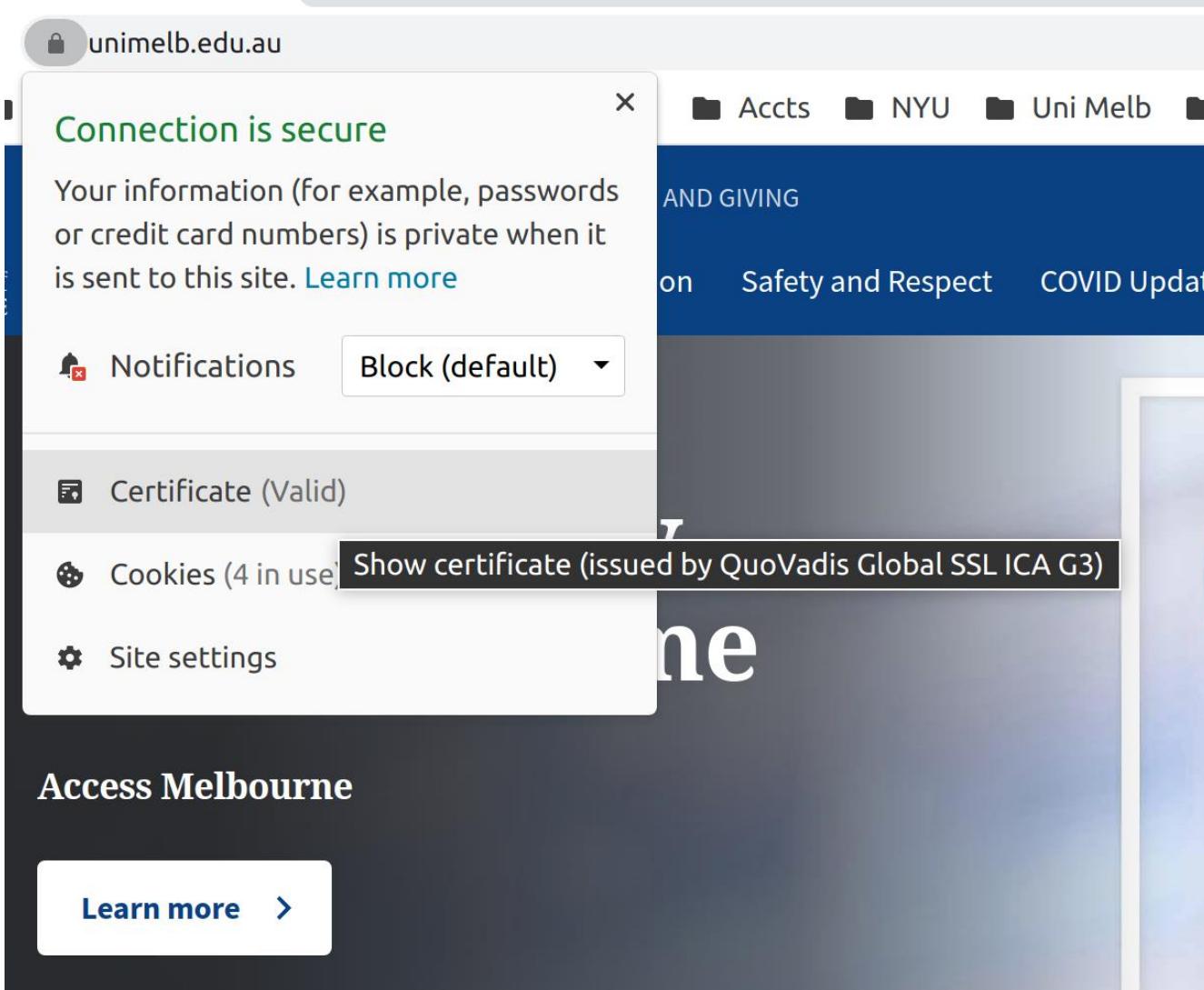
- with the meaning CA signs the certificate for user A with its private key.
- Please refer to a small document that I uploaded for the X.509 and the PKI Infrastructure,

- ❑ Version
- ❑ Serial number
- ❑ Signature algorithm identifier
- ❑ Issuer name
- ❑ Period of validity
- ❑ Subject name
- ❑ Subject's public-key information
- ❑ Issuer unique identifier
- ❑ Subject unique identifier
- ❑ Extensions
- ❑ Signature

Example



Practical example: web certificates



The screenshot shows a web browser window for unimelb.edu.au. A tooltip from the browser's security bar is displayed, stating "Connection is secure" and explaining that information sent to the site is private. It includes links to "Learn more" and "Show certificate". The tooltip also lists "Notifications", "Certificate (Valid)", "Cookies (4 in use)", and "Site settings". In the background, the University of Melbourne's "Access Melbourne" page is visible.

Connection is secure

Your information (for example, passwords or credit card numbers) is private when it is sent to this site. [Learn more](#)

Notifications Block (default)

Certificate (Valid)

Cookies (4 in use) Show certificate (issued by QuoVadis Global SSL ICA G3)

Site settings

Access Melbourne

Learn more >

Practical example: web certificates

Certificate Viewer: *.unimelb.edu.au

General Details

This certificate has been verified for the following usages:

SSL Server Certificate

Issued To

Common Name (CN) *.unimelb.edu.au
Organization (O) University of Melbourne
Organizational Unit (OU) <Not Part Of Certificate>

Issued By

Common Name (CN) QuoVadis Global SSL ICA G3
Organization (O) QuoVadis Limited
Organizational Unit (OU) <Not Part Of Certificate>

Validity Period

Issued On Tuesday, April 27, 2021 at 11:43:02 AM
Expires On Wednesday, April 27, 2022 at 11:53:00 AM

Fingerprints

SHA-256 Fingerprint D3 39 C2 A9 F7 87 13 B6 D4 B5 AA 6F 33 E9 F1 3F
5D C0 43 96 A6 A0 80 37 AA 8A B7 8C 51 9A 18 BE
SHA-1 Fingerprint 84 17 37 47 DD 2C 36 FE 1C 0A 6C BE EA 2D F0 F5
15 F4 D7 05

Practical example: web certificates

Certificate Viewer: *.unimelb.edu.au

[General](#)[Details](#)

Certificate Hierarchy

- ▼ Builtin Object Token:QuoVadis Root CA 2 G3
 - ▼ QuoVadis Global SSL ICA G3

Certificate Fields

Not Before

Not After

Subject

- ▼ Subject Public Key Info

Subject Public Key Algorithm

Subject's Public Key

- ▼ Extensions

Certificate Basic Constraints

Certification Authority Key ID

Field Value

Modulus (2048 bits):

B0 B7 5C 9B 35 80 B8 1E D3 BC DD AA 51 A1 A4 85
2B E7 C4 71 C2 78 FB AE 4D 22 2C 09 D3 4D 22 7A
FE B9 7C 7E 9F CE 12 23 D9 A2 76 08 5D D4 86 0B
25 D0 00 D0 75 05 05 3C 10 1D 11 00 63 0B 7C

U. Parimiappan & J. Bonneau

Certificate Advantages

- When Bob receives a certificate from Alice, how does he know that is authentic?
- $C_A : < IDA=Alice, PU_A, \text{Signature of } PU_A \text{ by the Authority}, PU_{\text{auth}} >$.
- In the absence of any other information, he is still in the dark as he was when Alice have him public key directly.
- However, now he can also obtain a certificate from the authority for his public key:
- $C_B : < IDA=Bob, PU_B, \text{Signature of } PU_B \text{ by the Authority}, PU_{\text{auth}} >$.
- Now, when he received C_A , he can verify that the authority is same as in his certificate (C_B).
- Then he can verify the signature of (PU_A) by using the public key of PU_{auth} found on his certificate, thus clearly establishing the authenticity of Alice's public key.

Ah problem again!

- Are we now solved the problem of authentication of public key?
 - As long as Alice's certificate has not changed or compromised he would be fine.
 - But situations could change at Alice's side: certificate gets expired, compromised or Alice wants to change the public key.
 - In such situations, you are now back to square one.
-
- **How does this problem can be solved?**
 - A simplest way for Bob to determine somehow that the certificate he received is still valid or not.
 - This is achieved by what is known as “revocation list” maintained by the authority.
 - Hang on, you wanted to solve the problem of authority not being a bottle neck but now authority still need to back again online.
 - However, this service is only required for a fraction of users not everyone.

Revocation

- Revocation of certificates is a very important practical problem that Industry is faced with.
- This has resulted in a large business to maintain public infrastructure.
- Certificate maintenance and verification is an important topic. We will look at the issues in one of the workshop problems.
- Also, not all users share a same CA, then we need to solve the problem of verifying certificates issued by different CA's.
- These are achieved through an organization of CA's in hierarchical fashion and each CA certify other CA's.
- Stallings explanation is given in the next slide. More details can be obtained from Additional material I have placed on LMS.

Revocation remains a difficult problem

- Revocation is required when a private key is compromised.
- What are the measures in place?
 - Option 1: Have an expiry date for the certificate
 - This could take potentially months or years
 - Option 2: Incorporate online validity checks (OCSP)
 - Active attacker can block OCSP queries
 - "Like a seat belt that works great, unless you're in an accident"
 - Option 3: Use Certificate Revocation Lists (CRLs)
 - Preferred option today. Browsers regularly download a list of expired certificates

CA Hierarchy Use

Forward certificates: Certificates of X generated by other CAs, and

Reverse certificates

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

A acquires B certificate using chain:

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

B acquires A certificate using chain:

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

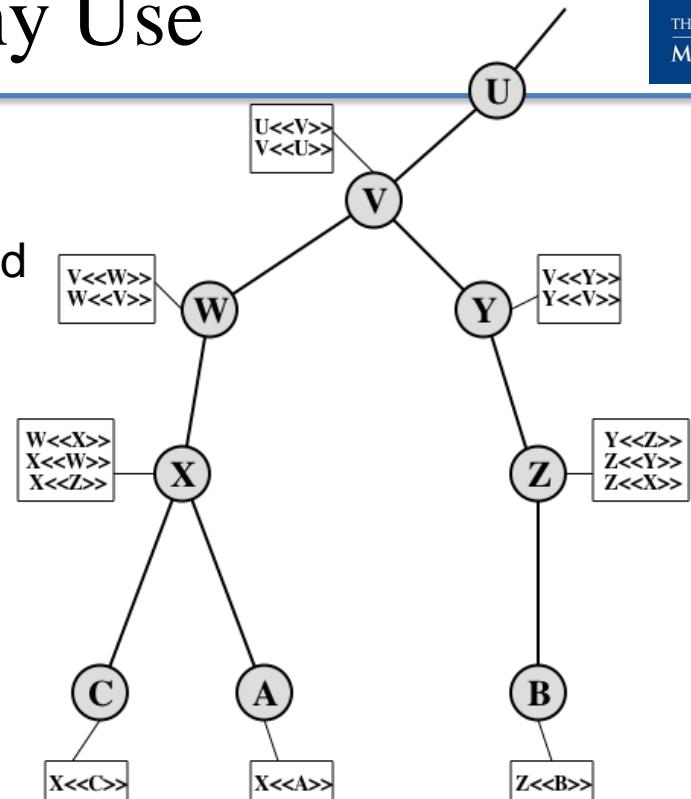
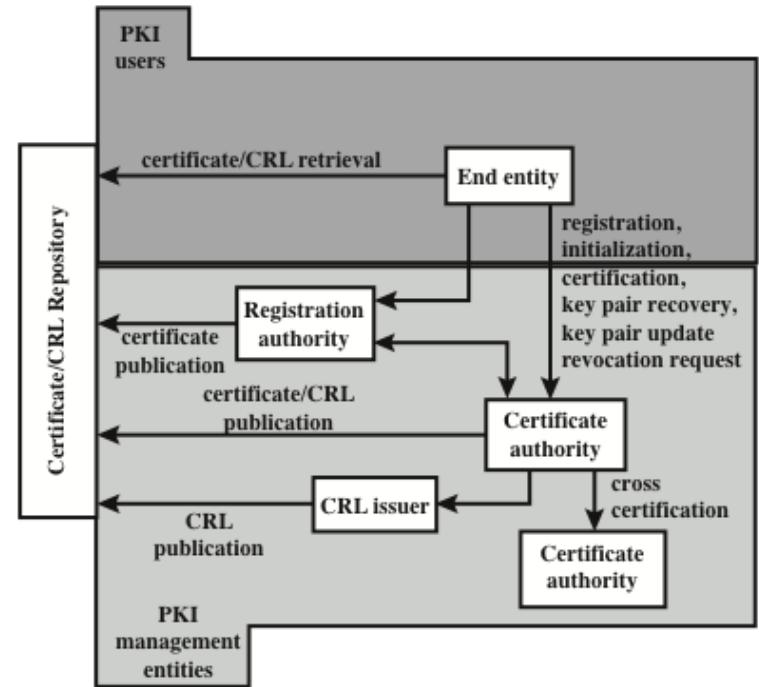


Figure 14.16 X.509 CA Hierarchy: a Hypothetical Example

From the textbook Fig 14.16

Public Key Infrastructure

- The textbook discusses the issues in detail, please refer to Section 14.5



From the textbook Fig 14.17

Figure 14.17 PKIX Architectural Model

Modern Methods: Mobile Computing

- How does applications validate updates?
- When an application downloads an update which come with a signature by the trusted entity. If the verification passes, then the application updates the data on its file.
- This validation is possible because the app certificate is correctly stored in the mobile at the time of app installation.
- E.g. Android Application Signing:
<https://source.android.com/docs/security/apksigning>

Summary

- How can Crypto help you to build “Security Mechanisms”?
- Symmetric cryptography:
 - Can provide a new private channel given a private channel in the past.
 - Can provide a large private channel, given a small private channel
 - Can provide mutual private channels given private channels with a trusted party
- Asymmetric cryptography:
 - Can provide a private channel given a public channel with integrity
 - Can provide a private channel given a trusted directory service

Week 8



Lecture 1

Key Management (Public Key)

Lecture 2

Finite Fields and ElGamal Encryption

Workshop 8: Workshop based on Lectures in Week 7

Quiz 8

Week 9



Lecture 1

Signatures from Discrete Log Assumptions

Lecture 2

Public Holiday, no lecture.

Workshop 9: Workshop based on Lectures in Week 8

Quiz 9

Signatures from Discrete Log Assumptions

COMP90043

Lecture 1

Outline



1.1 Signatures from Discrete Log assumptions

- Main ideas
- Direct Signature Scheme
- Ideas behind ElGamal signature
- Schnorr
- DSA
- BLS

Goal: Signatures from DLP

- In RSA, signature and encryption functions are inverses
 - $\text{Encrypt}(m) = m^e \pmod{N}$ $\text{Decrypt}(c) = c^d \pmod{N}$
 - $\text{Sign}(m) = m^d \pmod{N}$ $\text{Verify}(s) = s^e \pmod{N}$
- RSA security is based on the difficulty of factoring
- We've seen public-key encryption based on the discrete log problem
 - Diffie-Hellman Key Exchange (earlier)
 - ElGamal Encryption (last week)
- Can we produce signatures that rely on discrete log assumptions?
 - Yes, though not as straightforward as with RSA

Recap: Digital Signatures

- Digital signature scheme API:
 - **Keygen()** -> (public key, private key)
 - **Sign(private key, message)** -> signature
 - **Verify(message, signature, public key)** -> yes/no
- Requirements
 - Correctness: valid signatures should always verify
 - Security: nobody should be able to *forge* a signature without the signing key

Building a signature scheme

- As before, each user (eg. Alice) generates their key:
 - choose a secret key (number): $1 < x_A < p-1$
 - compute her **public key**: $y_A = g^{x_A} \text{ mod } p$
- Requirements:
 - Signature generation must require the secret x_A and **the message**
 - And possibly, some randomness
 - Verification function must only require public parameters
- Assume all parties know the group parameters (e.g. p, g)
 - Could also be another group, such as an elliptic curve group

Recap: properties from group theory

- Consider p a prime number
- There are $p-1$ integers less p which are relatively prime to p
- For all x , $x^{p-1} = 1 \bmod p$
- Similarly, $x^{t(p-1)} = 1 \bmod p$ for any integer t
- In other words, $x^m = 1 \bmod p$ if $m = 0 \bmod (p-1)$

Taking it one step further

- Consider p (a prime) as before
- For any integers i, k we have: $x^{i+k(p-1)} = x^i \pmod{p}$
- Why? Because $x^{i+k(p-1)} = x^i x^{k(p-1)} = x^i (x^{p-1})^k = x^i 1^k = x^i \pmod{p}$
- This gives us a stronger result:
- If $i = j \pmod{p-1}$ then $x^i = x^j \pmod{p}$

Essential idea: verifying linear equations

- Signer can provide values a, b, g^{x_1}, g^{x_2} (but not x_1 and x_2)
- Anybody can verify if $y = (a \cdot x_1 + b \cdot x_2) \bmod (p-1)$ by checking if:

$$g^y = (g^{x_1})^a \cdot (g^{x_2})^b \bmod p$$

- Only a person who knows x_1, x_2 could have set this up
- This notion leads to the original ElGamal Signature scheme

ElGamal Signature Scheme Concept

- Alice has public key $y_A = g^{x_A} \text{ mod } p$ and private key $1 < x_A < p - 1$
- Let $m = H(M)$, the hash of the full message M
- We would like to set up a linear equation where m is the *result*:

$$m = S_1 \cdot x_A + S_2 \cdot k$$

- Choose a random k
 - Let $S_1 = g^k \text{ mod } p$
 - Let $k \cdot S_2 + x_A \cdot S_1 = m \pmod{p-1}$
- (Given S_1 & S_2 this a equation with two unknowns)
- Verifier knows S_1, S_2, g^{x_A}
 - $(S_1)^{S_2} (y_A)^{S_1} = g^m \text{ mod } p$
- ← Signing Equation
- ← Verification Equation

ElGamal Digital Signatures (textbook)

- Each user (eg. A) generates their key
 - chooses a secret key (number): $1 < x_A < q-1$
 - compute her **public key**: $y_A = a^{x_A} \text{ mod } q$
- Alice signs a message M to Bob by computing:
- the hash $m = H(M)$, $0 \leq m \leq (q-1)$ and $\text{gcd}(k, q-1) = 1$
- compute "temporary key": $S_1 = a^k \text{ mod } q$
- compute k^{-1} , the inverse of $k \text{ mod } (q-1)$
- compute the value: $S_2 = k^{-1}(m - x_A S_1) \text{ mod } (q-1)$
- signature is: (S_1, S_2)
- any user B can verify the signature by computing:
$$V_1 = a^m \text{ mod } q ; V_2 = y_A^{S_1} (S_1)^{S_2} \text{ mod } q$$
- Signature is valid if $V_1 = V_2$

Example



- Consider GF(19), $p = 19$ and $g = 10$.
- Alice chooses $x_A = 16$ and hence $y_A = 10^{16} = 4$.
- Show the calculations for $m = 14$.

Example

- Consider GF(19), $p = 19$ and $g = 10$.
- Alice chooses $x_A = 16$ and hence $y_A = 10^{16} = 4$.
- Show the calculations for $m = 14$, with $k = 7$
- $S_1 = g^k = 10^7 = 15 \pmod{19}$
- $k^{-1} \pmod{p-1} = k^{p-2} \pmod{p-1} = 7^{17} \pmod{18} = 13$
- $S_2 = k^{-1}(m - x_A S_1) \pmod{p-1} = 13(14 - 16 \cdot 15) \pmod{18} = 14$
- Verify: $(S_1)^{S_2} (y_A)^{S_1} = g^m \pmod{p} = 15^{14} \cdot 4^{15} = 10^{14} \pmod{19}$
 $16 = 16 \pmod{19}$

Recap: Requirements of Digital Signatures

- A user's public key is related to their secret key using a one-way function
- A signature or tag should be able to be created using an efficient algorithm
- A verification algorithm should be efficient for anyone with the public key
- Signatures should be *unforgeable* by anyone
- This leads to *non-repudiation*: signer can't later deny creating a signature

Why are ElGamal signatures unforgeable?

- Attacker gets g^{xA} but not x_A
- Attacker can pick any k , $S_1 = g^k$
- Need to find values S_2 , m' such that $g^{m'} = (g^k)^{S_2} \cdot (g^{xA})^{S_1}$
- Solving this is as hard as computing discrete logarithms mod p
- Why? Because if attacker knows k , then they can solve for x_A :

$$x_A = (m' - k \cdot S_2) S_1^{-1} \pmod{p-1}$$

Major security risk: Revealing the secret k for any signature leaks private key

More security warnings for El Gamal

- Revealing the secret k for any signature reveals x_A
- Even re-using k for two signatures (without revealing it) leaks x_A !
 - Suppose attacker is given m, m' with signatures (S_1, S_2) and (S_1, S'_2)
 - Can solve for x_A !
 - Left as an exercise for tutorial this week...
 - More complex attacks exist if related values k_1, k_2 are used
 - Bottom line: **randomness is critical to El Gamal and related signatures**

Variations of ElGamal

- The signing equation is a function of k and x_A , where k is random, x_A is the secret corresponding to the public key $y_A = g^{x_A} \bmod p$
- ElGamal uses the signing equation: $kS_2 + x_A S_1 = m \bmod (p - 1)$
- The corresponding verification is $(S_1)^{S_2} (y_A)^{S_1} = (g^m) \bmod p$
- Other signing equations are possible. Consider:
- $x_A S_2 + k S_1 = m \bmod (p - 1) \Rightarrow S_2 = (x_A)^{-1} (m - k S_1) \bmod (p-1)$
- Verification is $(y_A)^{S_2} (S_1)^{S_1} = (g^m) \bmod p$
- Do you see the advantage of this formulation?
- In the exam you may be given a different signing equation and you need come up with a verification and argue if it is secure or not

Recap: subgroups and generators

- Consider p a prime number
- All values x generate a *cyclic subgroup* $\langle x \rangle = \{x, x^2, x^3, \dots\}$
- It is easy to find a generator g such that $\langle g \rangle$ has size $p-1$
- We say that g is a *generator* of the multiplicative group \mathbf{Z}_p^*
- Typically, we pick a generator of a subgroup with prime order q

More details

- Consider p (a prime) and a generator g for a subgroup of prime order q as before: $g^q \equiv 1 \pmod{p}$
- For any integers i, k we have: $g^{i+kq} \equiv g^i \pmod{p}$
- Why? Because $g^{i+kq} = g^i g^{kq} \equiv g^i (g^q)^k \equiv g^i 1^k \equiv g^i \pmod{p}$
- This gives us a stronger result:
 $g^i \equiv g^j \pmod{p}$ if and only if $i \equiv j \pmod{q}$

Schnorr Signature Scheme

- Choose a generator g of a subgroup of order q (working mod p)
 - Alice's public key $y_A = g^{x_A} \text{ mod } p$ and private key $1 < x_A < p - 1$
 - Choose a random k
 - Let $S_1 = H(g^k \text{ mod } p \parallel M) \text{ mod } q$
 - Let $S_2 + x_A S_1 = k \pmod{q}$
- ← Signing Equation
-
- Verifier knows S_1, S_2, g^{x_A}
 - $H[g^{S_2} (y_A)^{S_1} \text{ mod } p \parallel M] = S_1$
- ← Verification Equation
-
- Major efficiency gains: S_1, S_2 of size q , not p . Arithmetic also simpler
 - In practice, p is usually 1024 bits, but q is only 160 or 256 bits

Evolution of DL signature algorithms

- ElGamal was the original (introduced 1985) but is rarely used
- Schnorr (1990) is more efficient and still commonly used
 - Unfortunately, it was patented until 2008, barring universal adoption
- DSA/DSS were standardized in 1991 to avoid Schnorr patents
 - Combines ideas from ElGamal, Schnorr with similar sizes to Schnorr
 - Worse in all respects than Schnorr and more complex to implement/understand
 - Details provided in supplemental material
- Boneh-Lynn-Schaham/BLS (2004) is the smallest scheme known
 - Signatures have only one component (size 256 bits)
 - Relies on a special elliptic curve group with a *pairing*
 - Gaining increased use today

DSA Signature Scheme

- Choose a generator g of a subgroup of order q (working mod p)
 - Alice's public key $y_A = g^{x_A} \text{ mod } p$ and private key $1 < x_A < p - 1$
 - Choose a random k
 - Let $S_1 = g^k \text{ mod } p$
 - Let $S_2 + x_A S_1 = m \cdot k^{-1} \pmod{q}$
- ← Signing Equation
-
- Verifiers knows S_1, S_2, g^{x_A}
 - $g^{S_2} (y_A)^{S_1} = (S_1^{-1})^m \text{ mod } p$
- ← Verification Equation
-
- Signature size is similar to Schnorr. Arithmetic is more complex
 - *Threshold signing* is not easy, unlike Schnorr

Threshold signing

- Private key is split into n *key shares* $x_1, x_2, \dots x_n$
- Share-holders can compute a *partial signature* $S_i = \text{Sign}(m, x_i)$
- Compute a signature from t partial signatures $S = \text{Combine} (S_1, S_2, \dots S_t)$
 - t -out-of- n threshold scheme
- Application: control of a bank account (3-of-5 managers must authorize)
- Application: 2-factor authentication (2-of-2)

BLS Signature Scheme

- Work in an elliptic curve group with a pairing function e
- Alice's public key is $y_A = g^{x_A}$ and private key $1 < x_A < q$
- $S = H(m)^{x_A}$ ← **Signing Equation**
- Verifiers knows $g^{x_A}, H(m)^{x_A}$
- Check if $e(y_A, H(m)) = e(g, S)$ ← **Verification Equation**
- Relies on *bilinear* property of pairing operator e
- Signature is just one group element (though pairing is relatively slow)
- *Threshold signing* is easy. Other benefits like *aggregation*
- No randomness required to sign

Comparison

Scheme	Year	Sig. size	Threshold?	Assumption
RSA	1978	2048	yes	factoring
ElGamal	1985	2048	yes	discrete log
Schnorr	1990	512	yes	discrete log
DSA	1991	512	no	other
BLS	2004	256	yes	CDH, pairing

Week 9



Lecture 1

Signatures from Discrete Log Assumptions

Lecture 2

Public Holiday, no lecture.

Workshop 9: Workshop based on Lectures in Week 8

Quiz 9

Week 11



Lecture 1

Transport Layer Security

Additional Material SSL from the textbook

Lecture 2

A guest Lecture from Industry

Workshop 10: Workshop based on Lectures in Week 9

Quiz 9

Transport Layer Security

COMP90043

Lecture 1

Acronyms for web encryption

- **SSL** = Secure Sockets Layer
 - Netscape-developed standards
 - v1 (never) v2 (1995) v3 (1997)
 - All are obsolete now and should never be used
- **TLS** = Transport Layer Security
 - IETF-developed standards
 - v1.0 (1999) v1.1 (2006) v1.2 (2008) v1.3 (2018)
 - Before 1.2 is considered obsolete now
 - Modifies TCP (also layers with QUIC, etc.)
- **HTTPS** = HTTP+TLS/SSL
 - Includes certificate/domain name bindings
 - Includes other web-specific considerations

Encryption at multiple network layers

Layer	Protocols
Application	HTTPS (web traffic), PGP/SMIME (email) Signal/MLS (encrypted chat)
Transport	TLS/SSL
Internet	IPSec/Wireguard etc. (VPNs)
Link	WPA/WEP (WiFi networks), GSM (mobile networks)

TLS design goals

- Drop-in replacement for TCP
- Confidentiality:
 - Session data encrypted (some metadata exposed)
- Integrity:
 - MACs/AEAD used
- Authentication:
 - Can be one-sided (common on web) or mutual
 - Uses certificates to verify public keys
- Performance
 - Users should not notice increased latency
- Extensibility
 - Future upgrades supported as new crypto attacks discovered
- Verifiability (TLS 1.3)
 - Formal methods used to analyse protocol + implementations

TLS uses hybrid encryption

- Consider downloading a large file (1 GB)
- Assume RSA-2048 public key encryption
- # of encryptions required = $2^{30}/2^8 = 4,194,304$
- Assume encryption about 1000 ops/second
- Encryption time = 4194 seconds = 66 minutes!

Symmetric encryption is very fast

- The same 1 GB file splits into 67,108,864 AES blocks
- Assuming AES encryption speed of 10M blocks/s
- Total encryption time is about 7 seconds
 - As fast as *line speed* for a 1 Gbit/sec network connection
- Hence hybrid encryption
 - Public key operations to exchange symmetric keys
 - Symmetric keys to encrypt data

Steps to a TLS connection

- Protocol negotiation
 - Client/server share supported ciphers, pick best
 - Server sends public key
- Key exchange
 - Use public key crypto to agree on shared secret
- Key derivation (key schedule)
 - Derive client/server encryption/MAC keys from MS
- Record encryption
 - Encrypt each "record" using symmetric crypto
- Session resumption
 - Optionally, re-use MS for future connections

Protocol negotiation

- Client/server share supported *cipher suites*

Example for TLS 1.2:

```
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,  
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,  
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,  
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,  
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,  
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,  
TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_GCM_SHA384,  
TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,  
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_3DES_EDE_CBC_SHA
```

Protocol negotiation

- Goal: find best mutually-supported algorithms
- Enable gradual deployment/retirement
- Support locally-mandated ciphers
 - Previously, controversial "export grade" ciphers
 - "null" testing ciphers

Protocol downgrade attacks

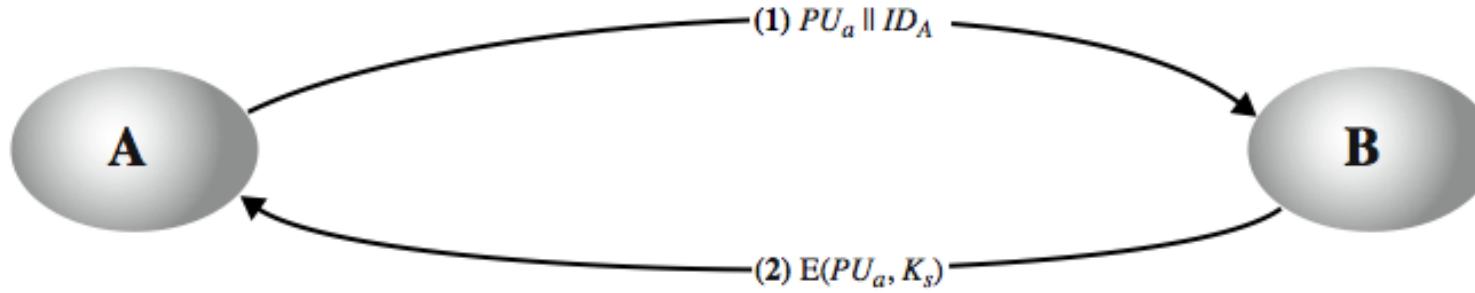
- Supported ciphersuites exchanged in the clear
- What if a network attacker changes them?
 - Trick client/server into picking weaker algorithms
- TLS 1.2 fix: share hash of negotiation transcript
- TLS 1.3 fix: only 5 ciphersuites, all strong

Two families of key exchange

- "RSA Style"
 - Client chooses master secret, encrypts for server
 - Includes randomness from Server Hello
 - Efficient, but no forward secrecy
- "Diffie Hellman style"
 - Both sides send DH shares, combine for master secret
 - One or both DH shares are digitally signed for integrity
 - Ensures forward secrecy
 - Only option as of TLS 1.3

Recap from Week 7: RSA Key Distribution

1. A generates a public/private key pair $[PU_a, Pra]$ and transmits a message to B consisting of PUa and an identifier of $[A, ID_A]$
2. B generates a secret key, K_s , and transmits it to A, encrypted with A's public key.
3. A computes $D(PRa, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s .
4. A discards PUa and PRa and B discards PUa.



From the textbook a version of Fig 14.7

Recap from Week 8: Diffie-Hellman exchange

Public Parameters: g: generator, order of the cyclic group: (p-1), prime: p

Alice

Choose $N_a = 2$

$$g^{N_a} = 2^2 = 4 = M_a$$

Compute

$$K_{ab} = M_b^{N_a} = 9^2 = 4$$

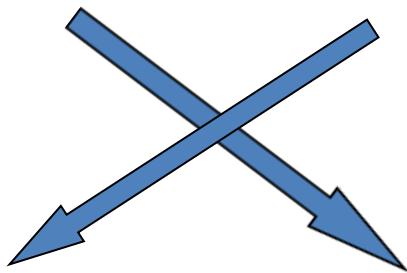
Bob

Choose $N_b = 6$

$$g^{N_b} = 2^6 = 9 = M_b$$

Compute

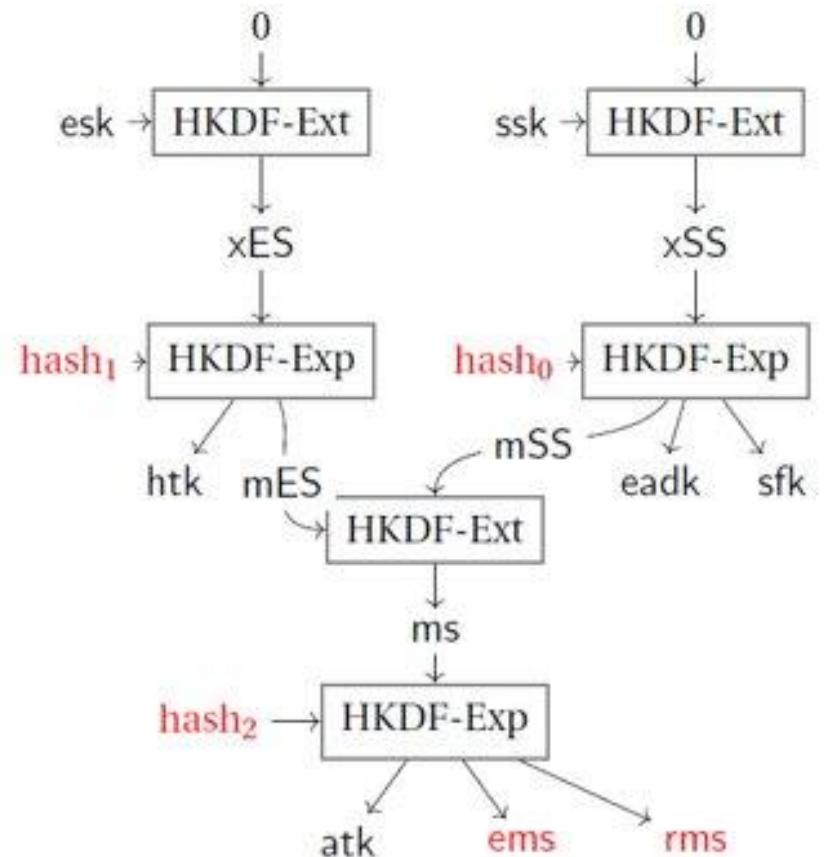
$$K_{ba} = M_a^{N_b} = 4^6 = 4$$



$$K_{ab} = K_{ba} = 4$$

Key derivation

- From master secret:
 - Client encrypt key
 - Client MAC key
 - Server encrypt key
 - Server MAC key



Keys make up "connection state"

Server and client random

- Byte sequences that are chosen by the server and client for each connection

Server write MAC secret

- The secret key used in MAC operations on data sent by the server

Client write MAC secret

- The secret key used in MAC operations on data sent by the client

Server write key

- The secret encryption key for data encrypted by the server and decrypted by the client

Client write key

7/10/2022

- The symmetric encryption key for data encrypted by the client and decrypted by the server

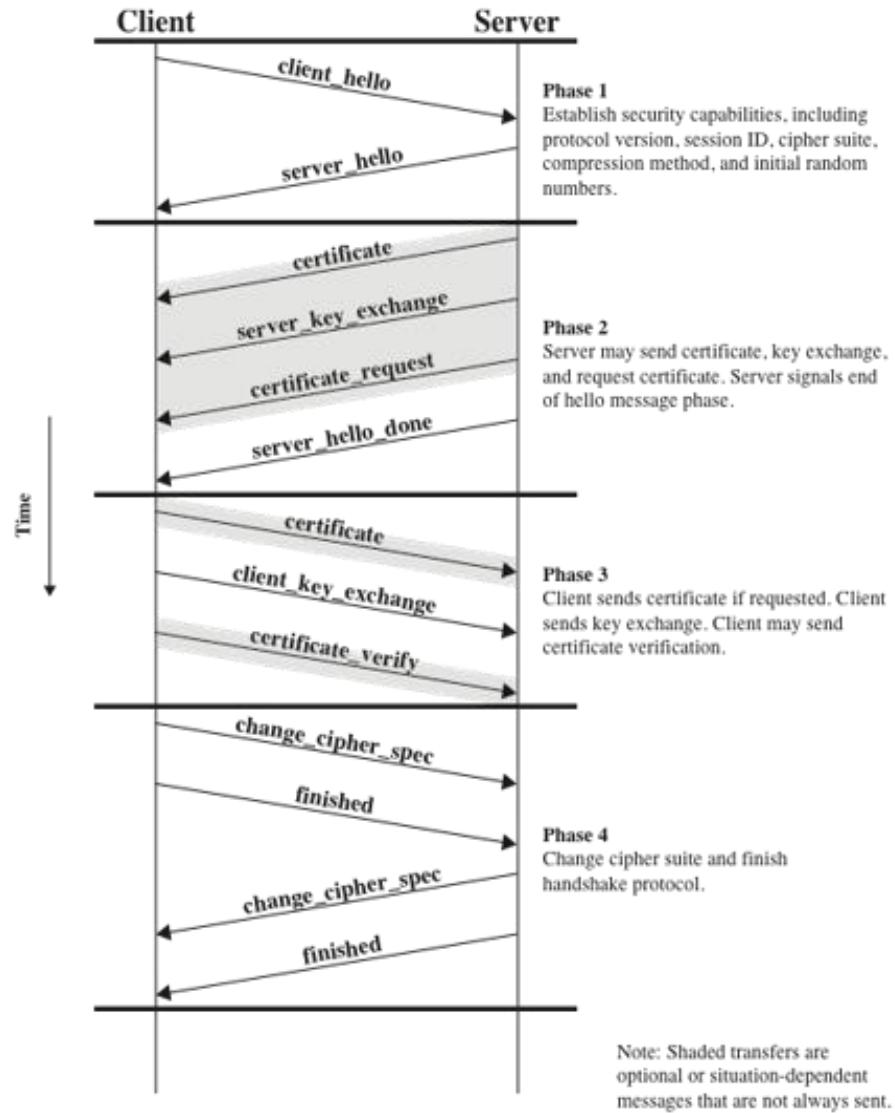
Initialization vectors

- When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key
- This field is first initialized by the SSL Handshake Protocol
- The final ciphertext block from each record is preserved for use as the IV with the following record

Sequence numbers

- Each party maintains separate sequence numbers for transmitted and received messages for each connection
- When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero
- Sequence numbers may not exceed $2^{64} - 1$

Full TLS handshake



Record encryption

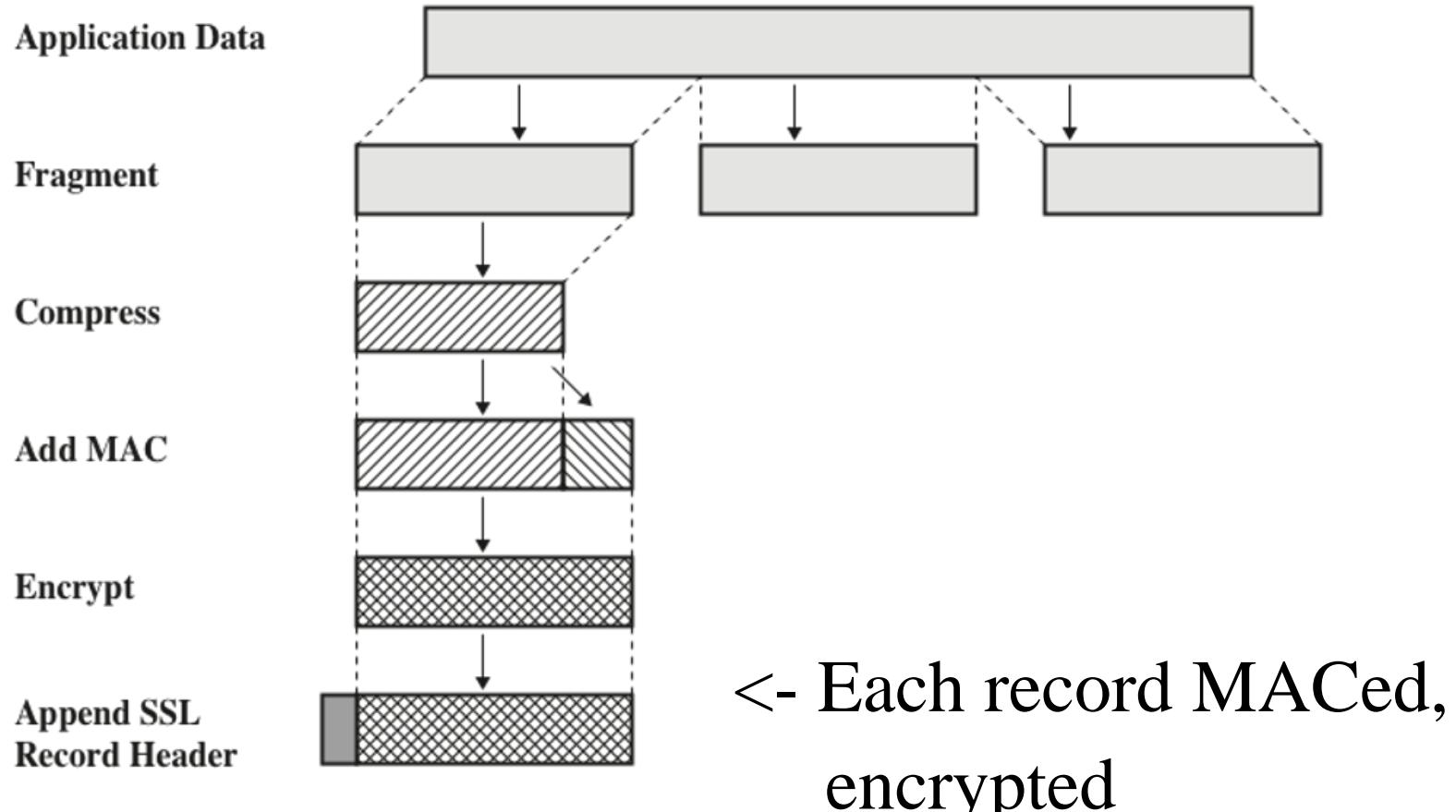


Figure 17.3 SSL Record Protocol Operation

Session resumption

- New TLS handshake required per TCP connection
- Loading a web page can require hundreds
- To save time, re-use master secret
- **Session IDs**
 - Client server, each store master secret
 - ID is opaque handle (can be a hash of MS)
- **Session tickets**
 - Only clients store tickets
 - Ticket is encrypted MS using long-term ticket key on server

Security Issues



- TLS is very complicated
- Many corner-case protocol bugs
 - BREACH, CRIME, POODLE, DROWN, Logjam
- Many subtle implementation bugs
 - Heartbleed, Lucky13, FREAK
- Very slow to upgrade servers
 - TLS 1.1 still running, 15 years later

TLS 1.3 improvements

- Reduced complexity
 - DH only, only 5 ciphersuites
- Forward secrecy
- Reduced latency
 - 1-RTT and 0-RTT modes
- Verifiability
 - Protocol verified by automated tools

Useful tools

- The Illustrated TLS connection
 - <https://tls.ulfheim.net/>, <https://tls13.ulfheim.net/>
- Qualys SSL report card
 - <https://www.ssllabs.com/ssltest/analyze.html>
- Qualys SSL Pulse
 - <https://www.ssllabs.com/ssl-pulse/>

Week 11



Lecture 1

Transport Layer Security

Additional Material SSL from the textbook

Lecture 2

A guest Lecture from Industry

Workshop 10: Workshop based on Lectures in Week 9

Quiz 9

Week 11



Lecture 1

Authentication

Additional Material on Kerberos from the textbook

Lecture 2

Review

Workshop 11: Workshop based on Lectures in Week 11

No Quiz this week.

Authentication

COMP90043

Lecture 1

Authentication

Lecture 1

1.1 Human Authentication

- Remote User-Authentication principles
- Means of Authentication
- Password storage

1.2 Cryptographic Authentication

- Replay Attacks.
- Protocols Remote User Authentication
 - Needham-Schroeder (NS) Protocol
 - Denning's modification
 - Neuman's modifications

Authentication principles

The process of verifying an identity claimed by or for a system entity - RFC 4949 (Internet Security Glossary)

Why authenticate?

- Personalization
- Access to private information
- Access control (e.g. subscription to an online service)
- Attribution of user actions
- Auditing misbehaviour

Remote User Authentication

- Stallings presents two important steps:
- **Identification:** User claims an identifier
 - Email address
 - Username
 - Real-world name
- **Verification:** Users presents authenticating information to corroborate claimed identity
 - Password
 - Stored login cookies
 - Many more...

The NIST Model

Please read NIST 800-63-2 for more details;
Stallings 15.1 gives a brief summary

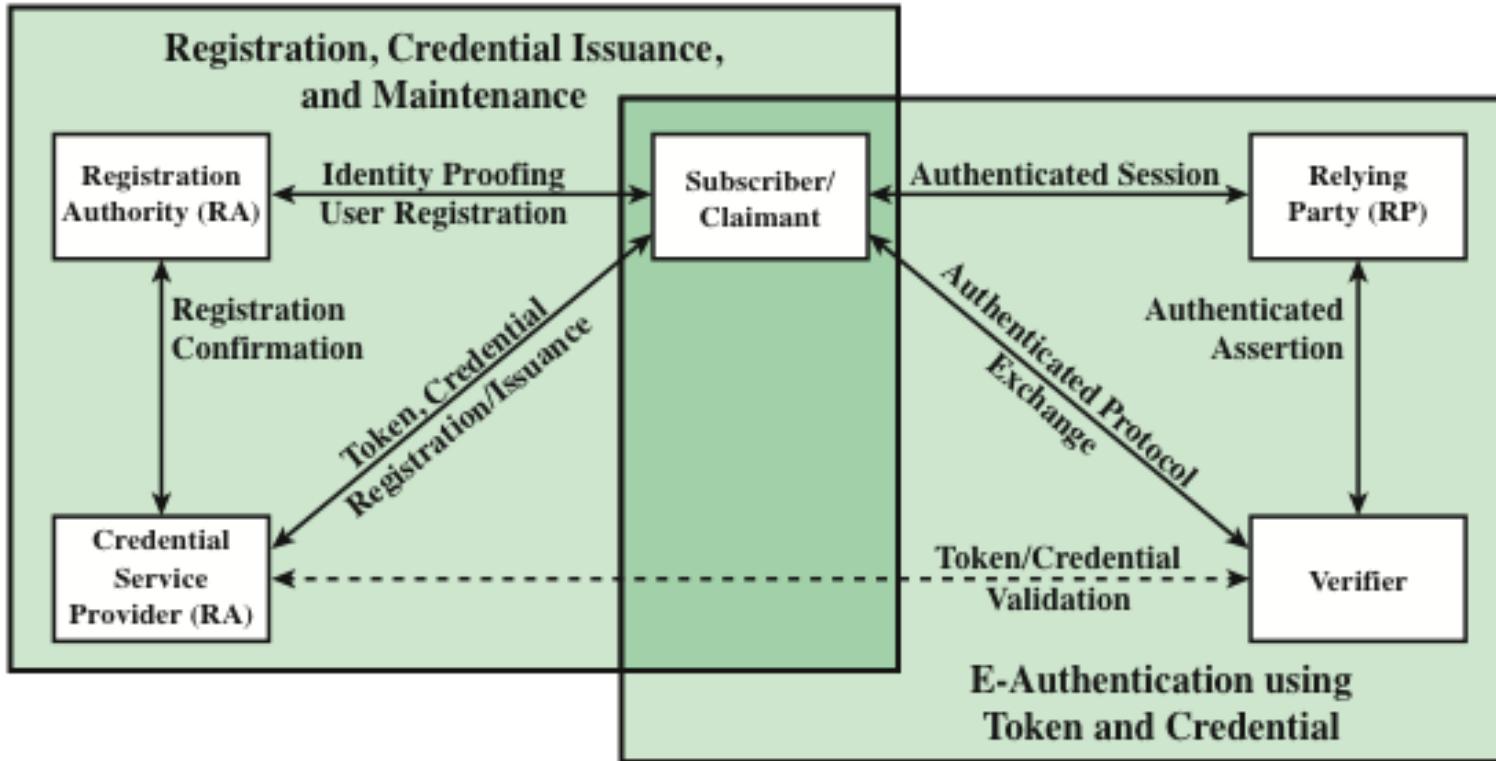


Figure 15.1 The NIST SP 800-63-2 E-Authentication Architectural Model

Means of human authentication

- **Something you know:**
 - Explicit: password, personal identification number (PIN)
 - Implicit: answers to personal questions, account history
- **Something you have:**
 - Authentication dongle/ FIDO token
 - Smart phone
 - Physical keys
- **Something you are** (static biometrics):
 - Fingerprint, iris patterns, facial recognition
- **Something you do** (dynamic biometrics):
 - Voice, handwriting, typing rhythm

Individually, any factor can fail

- **Something you know:**
 - Something you've forgotten
- **Something you have:**
 - Something you've lost
- **Something you are** (static biometrics):
 - Something you were but aren't anymore
- **Something you do** (dynamic biometrics):
 - Something you can't do right now

Many ways to steal a password

- Guess it
- Observe in transit
 - Passwords are static
- Compromise server database
- Compromise another site where it's reused
- Phishing-trick user into entering in wrong place
- Shoulder surfing-observe user typing it

Password storage to minimize data breaches

- Version 1: store password in cleartext
- Immediately vulnerable if database breached

USERNAME	PASSWORD
Alice	WildCAT789!
Bob	iLoveAlice?
Carol	WildCAT789!

Password storage to minimize data breaches

- Version 2: store Hash(password) only
- Problem: *password cracking*
 - Attacker tests the hash of many common passwords

USERNAME	PASSWORD_HASH
Alice	0x7215b2b4a8043d20a67bf3cb27d7a475
Bob	0xc1674a0c39a172aea52b94c28dec803
Carol	0x7215b2b4a8043d20a67bf3cb27d7a475

Password storage to minimize data breaches

- Version 3: Hash(salt || password)
- Cracker must target each user separately
 - Modern hardware might test billions of guesses/s

USERNAME	SALT	SALTED_HASH
Alice	a1630d	0x621bafdc496f89
Bob	1674a0	0xea880ec937dc5e
Carol	621baf	0x7215b2b4a8043d

Password storage to minimize data breaches



- Version 4: SlowHash(salt || password)
- Iterate SHA-256 10,000 times instead of once
 - Server must repeat this to check passwords

USERNAME	SALT	SLOW_SALTED_HASH
Alice	a1630d	0x621bafdc496f89
Bob	1674a0	0xea880ec937dc5e
Carol	621baf	0x7215b2b4a8043d

Password storage to minimize data breaches

- More advanced versions are now used
- Memory-hard hash functions (argon2, balloon)
 - Require lots of memory to compute
 - More difficult to build custom cracking chips
- Use a MAC with the key kept in an HSM
 - Attacker cannot export the key

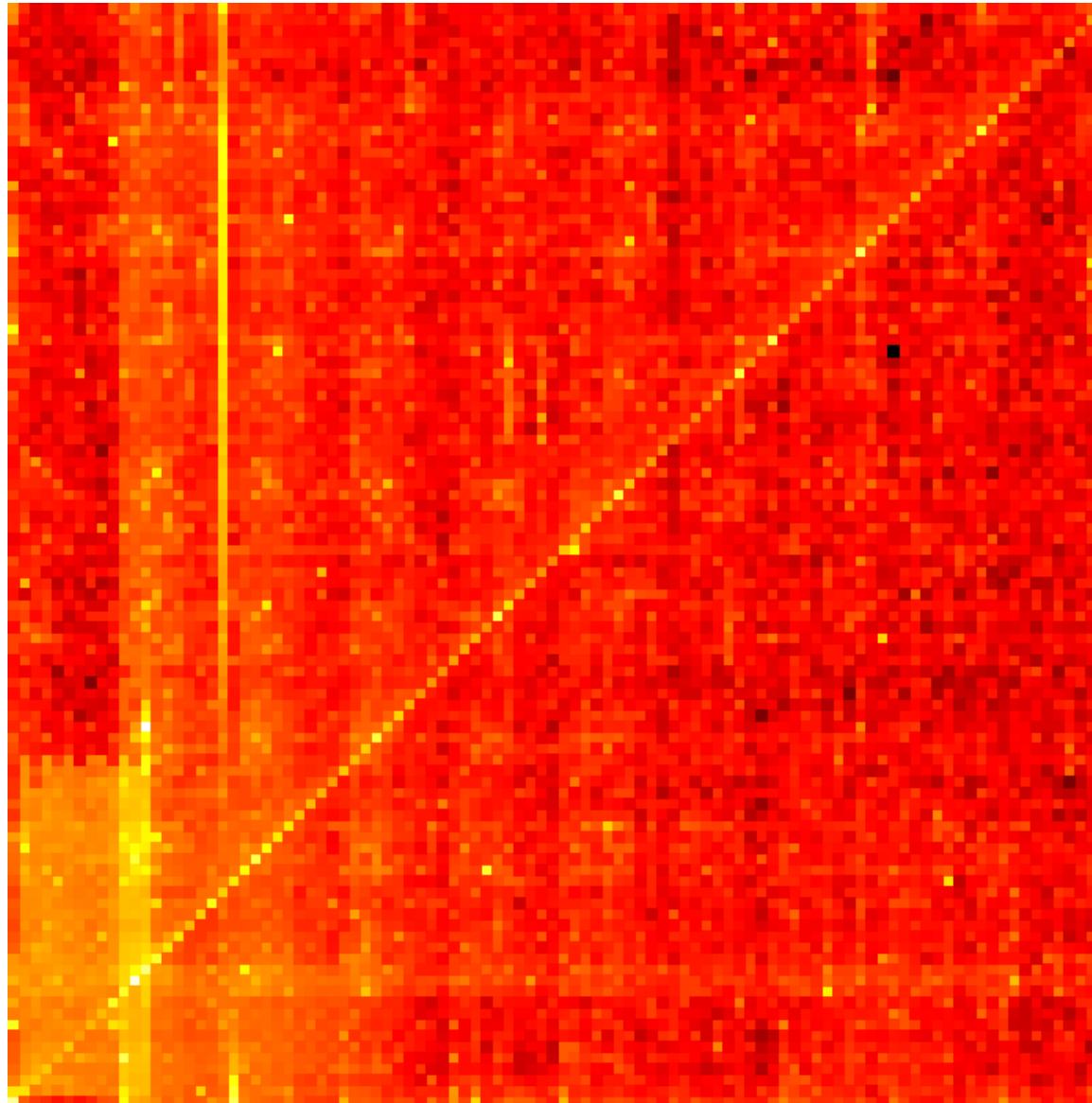
Password guessing scenarios

- Offline guessing (cracking)
 - Attacker has obtained a hash of user's password
 - Attack limited only by computation budget
 - 50-95% of passwords typically vulnerable
 - Strongly diminishing returns
- Online guessing
 - Attacker must check with the actual service
 - Service uses *rate limiting*
 - e.g. lock account after 10 incorrect guesses
 - *Non-targeted attackers* might try 1 guess for many accounts

Measures of guessing difficulty

- Number of possible values (H_0)
 - Infinite for passwords! Finite for PINs, phone numbers
 - Only meaningful if randomly assigned
- Min-entropy H_∞
 - $\log(p(\text{most common item}))$
 - Worst-case security
- Guesswork
 - How many guesses does it take to compromise 50% of users?
- Shannon entropy
 - Often used, but no direct relation to guessing difficulty
- Cracking difficulty
 - Measured empirically using real cracking software

PIN statistics



PIN statistics

rank	PIN	frequency
#1	1234	10.713%
#2	1111	6.016%
#3	0000	1.881%
#4	1212	1.197%
#5	7777	0.745%
#6	1004	0.616%
#7	2000	0.613%
#8	4444	0.526%
#9	2222	0.516%
#10	6969	0.512%

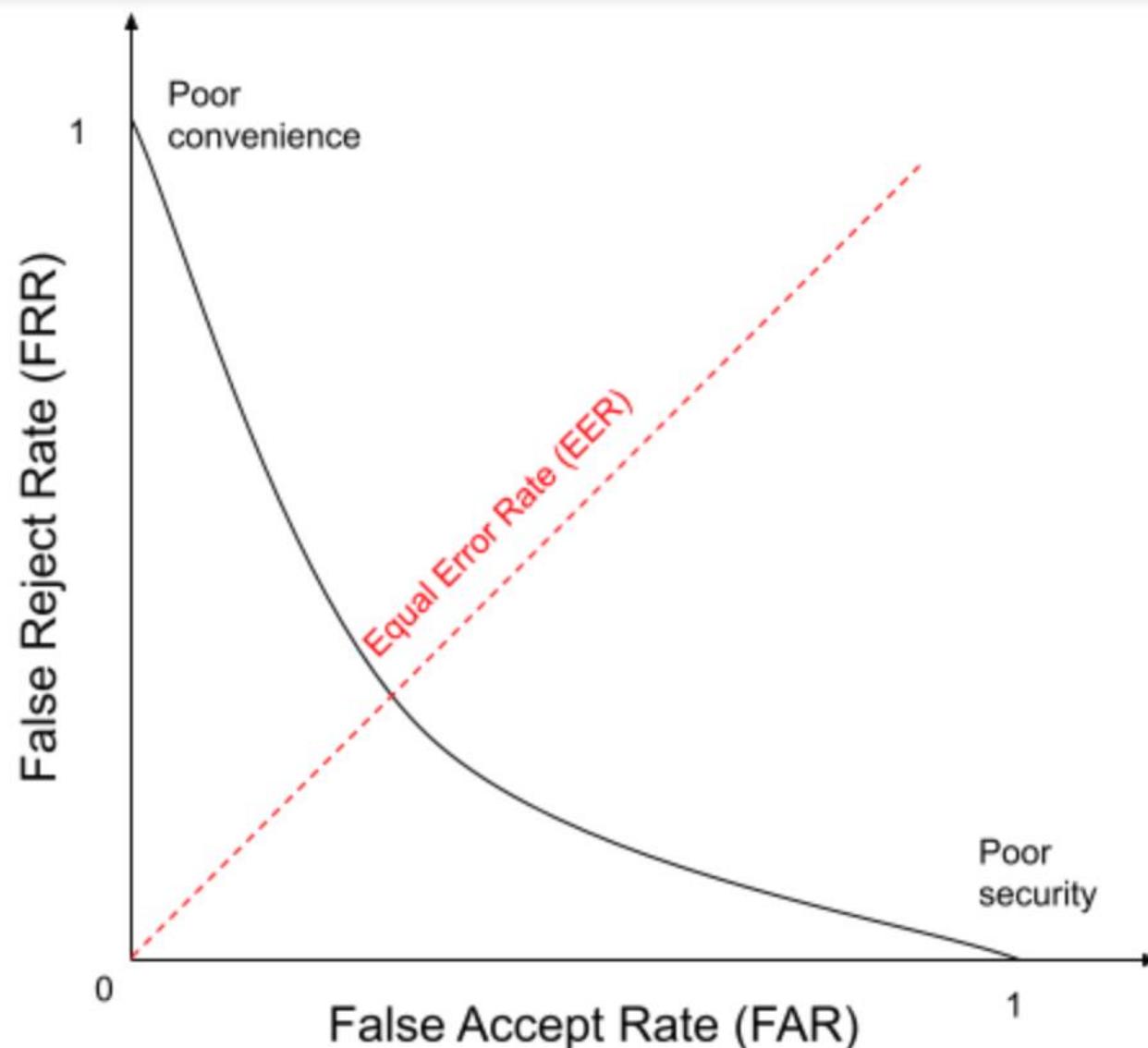
Additional problems with implicit questions

- Accessibility
 - Not everybody knows mother's maiden name
- Memorability/consistency
 - Your favorite color might change
- Social attacks
 - Friends and family know, and they can be adversaries
- Public records
 - Birth/death/marriage, school attendance, cars
- Cultural
 - Mother's maiden name may still be in common use

Biometrics

- All biometrics are inherently noisy
 - Need to capture analog signal, e.g. image of iris
 - The human body is constantly changing
- Biometrics requires a *classifier*
 - Recognition: is this Alice's iris?
 - Matching: whose iris is this?
 - Need to balance accuracy vs. reliability

Receive Operating Characteristics (ROC)



Many other challenges to biometrics

- Population diversity
 - Between 1-5% of strangers "share" fingerprints
 - Iris recognition is much stronger theoretically
- Live-tissue detection
 - Is a face, or a picture of a face?
 - Difficult/inconvenient in unsupervised settings
- Mimicry
 - Particularly a problem for voice

Many other challenges to biometrics

- Accessibility
 - Some humans don't have fingers, eyes, etc.
 - Some biometrics change over many years (e.g. voice)
- Cultural
 - Capturing faces, eyes can feel invasive
 - In some cases, users have religious objections
- Cost
 - Can require specific hardware to capture

Authentication

Lecture 1

1.1 Human Authentication

- Remote User-Authentication principles
- Means of Authentication
- Password storage

1.2 Cryptographic Authentication

- Replay Attacks.
- Protocols Remote User Authentication
 - Needham-Schroeder (NS) Protocol
 - Denning's modification
 - Neuman's modifications

Cryptographic authentication

- Assume both parties know cryptographic-strength key material
 - Shared symmetric keys
 - Public key of other party
- Guessing attacks should be eliminated
- Generally, not possible to achieve with human brains
 - Protocols exist (Hopper-Blum) but are too hard for mass adoption

Mutual Authentication Protocols

- Both communicating parties verify each other's identity and exchange session keys for subsequent interactions.
- Sometimes authentication is only one-sided
 - E.g. HTTPS in most cases
- Two important challenges:
 - 1. **Confidentiality:** the exchanged session keys are kept confidential
 - 2. **Freshness:** Ensure that the exchange is current and prevent replay

Mutual Authentication

Central to the problem of authenticated key exchange are two issues:

Timeliness

- Important because of the threat of message replays
- Such replays could allow an opponent to:
 - compromise a session key
 - successfully impersonate another party
 - disrupt operations by presenting parties with messages that appear genuine but are not

Confidentiality

- Essential identification and session-key information must be communicated in encrypted form
- This requires the prior existence of secret or public keys that can be used for this purpose

Approaches to ensure Freshness

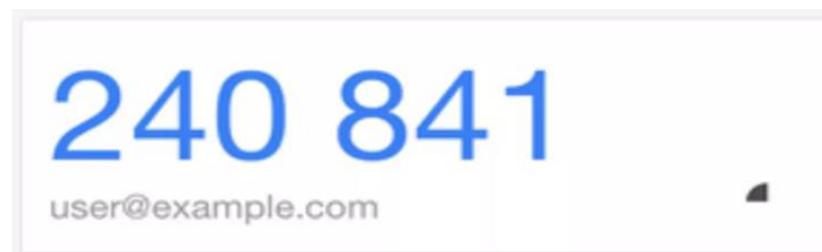
- Sequence numbers
 - Messages accepted only if sequence number is in the proper order
 - Requires mutual state and synchronization, easy to screw up
- Timestamps
 - Messages accepted only if they include a recent timestamp
 - Requires that clocks among the various participants be synchronized
 - Replay attacks are still possible within a narrow time window
- Challenge/response
 - Party A, expecting a fresh message from B, first sends B a nonce (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value
 - Requires interaction

Sequence numbers

- For authentication i , prover sends $H(i \parallel k)$
 - Equivalently, can use a MAC, PRF, block cipher
- Can repeat in reverse for mutual authentication
- Challenge: adversary can capture, replay
- Challenge: synchronization

Timestamps

- To authenticate at time $t \pm \Delta$, prover sends:
 - $H(t \parallel k)$
 - Equivalently, can use a MAC, PRF, block cipher
 - Verifier accepts within Δ of t
- Can repeat in reverse for mutual authentication



- Challenge: synchronization

Challenge-response

- Verifier sends random challenge (nonce) n
- Prover responds with $H(n \parallel k)$
 - Equivalently, can use a MAC, PRF, block cipher
- Challenge: requires two-way communication
- Challenge: nonce should never repeat

Needham-Schroeder

- Let us look at the protocol using symmetric key systems for use over distributed environments.
- The protocol is similar to the Needham-Schroeder protocol we considered earlier.
- A two-level hierarchy of symmetric encryption keys are employed.
- We need a trusted key distribution centre (KDC) and each user shares a master key with the KDC. The KDC generates session keys required for the authenticated sessions.

Applications

- Kerberos: Part of Project Athena-Authentication service over open distributed environment-variants are used everyday- See additional notes for study.
 - This is similar to windows login, when you login to your system-you will login once which will help you to get access to various other services in the system, file system, print system etc.
- Federated Identity Management: Relatively new concept: a common identity management scheme across multiple enterprises works in a large scale with millions of uses –See additional notes.
- We will discuss some protocol issues in the workshop.

Needham-Schroeder (NS) Protocol

- As described by Stallings:
 1. A->KDC: $ID_A \parallel ID_B \parallel N_1$
 2. KDC -> A: $E(K_a, [K_s \parallel ID_B \parallel N_1 \parallel E(K_b, [K_s \parallel ID_A])])$
 3. A -> B: $E(K_b, [K_s \parallel ID_A])$
 4. B -> A: $E(K_s, [N_2])$
 5. A -> B: $E(K_s, [f(N_2)])$

Issues with NS protocol

- Main goal: secure distribution of session keys between communicating parties.
- It is vulnerable to a replay attack if an old session key is compromised.
- How do you address this situation?
- Dennings modification using timestamps.
- Neuman approach of using nonces.

NS Dennings Modification.

- Stallings discusses the following modification by Denning, 1981.
 1. A → KDC: $ID_A \parallel ID_B$
 2. KDC → A: $E(K_a, [K_s \parallel ID_B \parallel T \parallel E(K_b, [K_s \parallel ID_A \parallel T])])$
 3. A → B: $E(K_b, [K_s \parallel ID_A \parallel T])$
 4. B → A: $E(K_s, N_1)$
 5. A → B: $E(K_s, f(N_1))$
- T: Time stamp, assures fresh keys for A and B;
- Verification: that $|Clock - T| < \text{delta } t1 + \text{delta } t2$
- delta t1: mean discrepancy between KDC's and local clocks
- delta t2: mean network delay time

Suppress-Replay Attacks

- The NS –Dennings uses global synchronized clocks-expensive to maintain.
- What is a Suppress-Replay attack?
- An adversary stores a past message and he replays later when the timestamp in the message becomes current at the recipient's machine.
- How to avoid?
- Parties regularly synchronize their clocks with that of KDC.
- Use handshaking protocols using nonces

NS Protocol

- Denning 81 Modification

1. A → KDC: $ID_A \parallel ID_B$
2. KDC → A: $E(K_a, [K_s \parallel ID_B \parallel T \parallel E(K_b, [K_s \parallel ID_A \parallel T])])$
3. A → B: $E(K_b, [K_s \parallel ID_A \parallel T])$
4. B → A: $E(K_s, N_1)$
5. A → B: $E(K_s, f(N_1))$

- Neuman 93 Modification

1. A → B: $ID_A \parallel N_a$
2. B → KDC: $ID_B \parallel N_b \parallel E(K_b, [ID_A \parallel N_a \parallel T_b])$
3. KDC → A: $E(K_a, [ID_B \parallel N_a \parallel K_s \parallel T_b]) \parallel E(K_b, [ID_A \parallel K_s \parallel T_b]) \parallel N_b$
4. A → B: $E(K_b, [ID_A \parallel K_s \parallel T_b]) \parallel E(K_s, N_b)$

Think why the later is an interesting solution to resist SR attacks?

Authentication using Public Keys

- We've seen three examples already
- TLS-RSA/Merkle exchange
 - Client encrypts session key using server's public key
 - Server responds to prove correct decryption
 - No forward secrecy
- TLS-Diffie-Hellman
 - Diffie-Hellman exchange with one or two signatures on key shares
 - Forward secrecy, not deniable
- Signal: Triple Diffie-Hellman
 - Combine long-term keys with ephemeral keys
 - Forward secrecy, deniable

Week 11



Lecture 1

Authentication

Additional Material on Kerberos from the textbook

Lecture 2

Review

Workshop 11: Workshop based on Lectures in Week 11

No Quiz this week.