ECEN 685-885 - Machine Learning in Cyber-security

Dr. Mahmoud Nabil

Dr. Mahmoud Nabil
mnmahmoud@ncat.edu

North Carolina A & T State University

March 11, 2020

Talk Overview

1. Introduction
2. Mathematical Background
   - Finite Groups
3. Basic Cryptographic Primitives
   - Symmetric key Cryptography
   - Asymmetric key Cryptography
   - Key Exchange Protocols
   - Cryptographic Hash Functions
   - Elliptic Curve Cryptography
   - Bilinear Pairing
   - Secure Multi-Party Computation
     - Garbeled Circuits
     - Arithemtic Circuits
4. Cryptographic Libraries

Outline

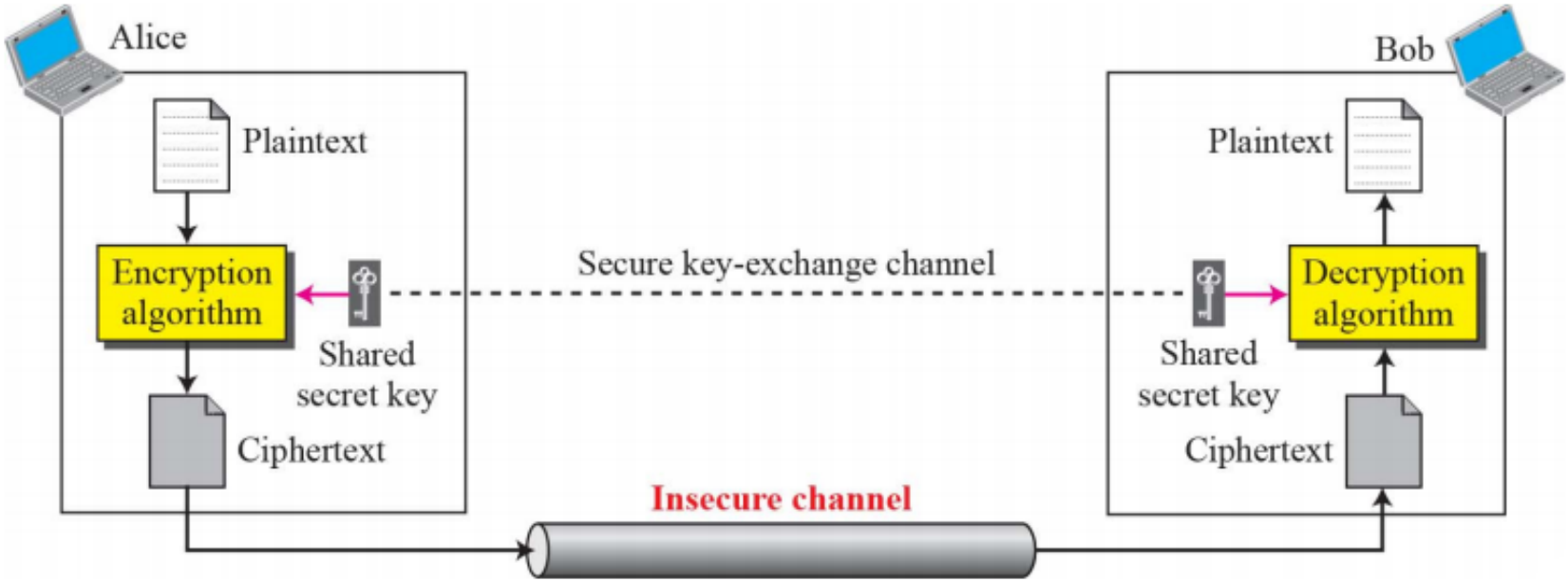
What is Cryptology?

- Cryptology
  - Cryptography
  - Cryptanalysis
- Cryptography, a word with Greek origins, means **secret writing**.
- However, we use the term to refer to the science and art of transforming messages to make them secure and immune to attacks.
- A cipher is a function which transforms a plaintext message into a ciphertext by a process called encryption.
- Plaintext is recovered from the ciphertext by a process called decryption.

Encryption / Decryption

“attack at midnight” - *plaintext*

“buubdl bu njeojhiu” - *ciphertext*



Cryptanalysis

- The science and study of breaking ciphers, i.e., the process of determining the plaintext message from the ciphertext
- Objective to recover key not just message
- A common approach is brute-force attack - try all possible cases

Overview of Security Requirements

1. Confidentiality
2. Entity Authentication
3. Data Integrity
4. Non-Repudiation
5. Access Control
6. Availability

Note

Most of these security properties can be achieved through cryptographic functions.

Confidentiality

Confidentiality

Ability to keep information communicated between (among) authorized parties private.

- Observer should not be able to recover information
- In a stronger sense, an observer cannot determine the parties involved or whether a communication session occurred
- Usually achieved by encryption
- Snooping refers to unauthorized access to or interception of data.
- Traffic analysis refers to obtaining some other type of information by monitoring online traffic.

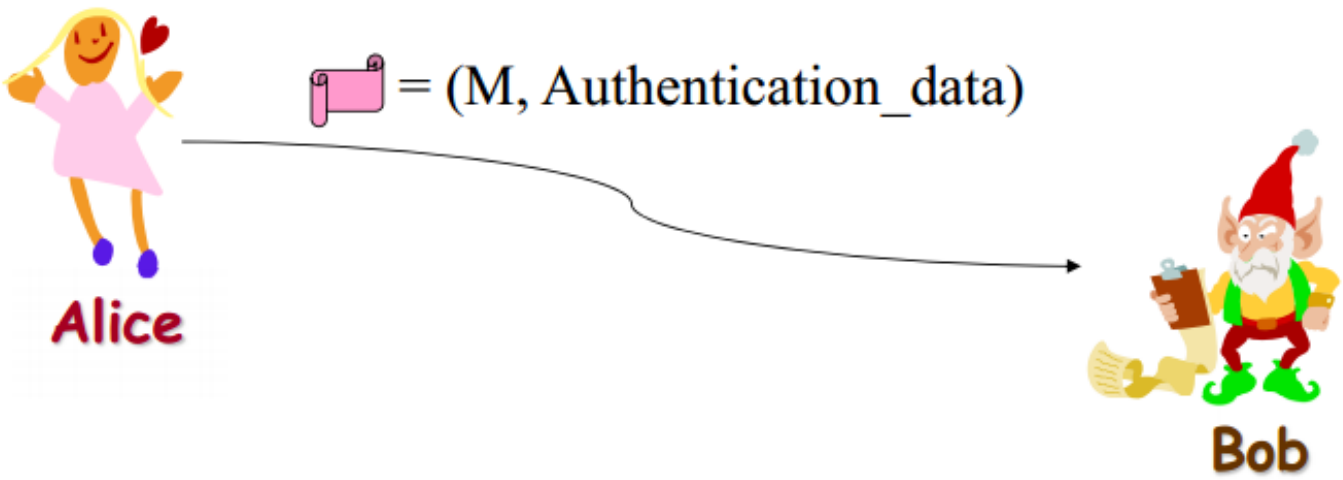
Entity Authentication (1/2)

Entity Authentication

Ability of the authorized parties in a communication session to ascertain the identity of other authorized parties

- Mutual or one-way authentication
- An entity authentication may be a user authentication or a device authentication
- The party to be authenticated must provide some verifiable evidence to prove it is the entity identified by the identity.
- The verification is to check whether the specific authentication data is valid, which can be generated only by the party with the claimed identity.
- Authentication can thwart **masquerading or spoofing** attacks which happen when attackers impersonate somebody else.

Entity Authentication (2/2)



- Alice sends a msg M to Bob
- Bob wants to be sure M is really from Alice
- Alice attaches with the message an authentication data such as signature or message authentication code (MAC).
- No one can generate this data except Alice (the owner of a secret key).
- Bob is able to verify the authentication data

Data Integrity

Data Integrity

Ability to ascertain that information exchanged has not been subject to additions, deletions, modifications or undue delay.

Attacks threatening integrity

- **Modification:** attackers intercept a message and change it. Modification can be thwarted by using signature or message authentication code.
- **Replay:** attackers obtain a copy of a message sent by a user and later tries to replay it. Usually timestamps are used to thwart replaying.

Integrity vs Authentication

- Integrity is to prevent from altering the message content, while authenticity is to prevent from altering the message source.
- These two are inseparable. It is often to use a single cryptographic functionto provide both.
- Usually, signature and hash functions or MAC can provide integrity and authentication

Non-Repudiation

Non-Repudiation

Ability to prevent an authorized party from denying the existence or contents of a communication session.

- Repudiation means that sender of a message might later deny that she has sent the message; the receiver of the message might later deny that he has received the message.
- Ability to prove to an independent third party at a later date the author and contents of a message.
- Digital signature is used to achieve this property.

Access control

Access control

Aims to prevent unauthorized access to resources.

- Permission to access a resource is called authorization.
- Usually the same piece of knowledge used for authentication is used to grant the authorization to the resource.

Availability

Availability

The information created and stored by an organization needs to be available to authorized entities.

- Network/system should be available all the time.
- **Denial of service (DoS)** is a very common attack that targets availability . It may slow down or totally interrupt the service of a system.

So what is security?

Security

Security is about how to prevent attacks, or – if prevention is not possible – how to detect attacks and recover from them.

- **Passive Attacks**
  - **Eavesdropping:** The attacker simply listens and tries to interpret the data being exchanged - if the data is in-the- clear, they succeed
  - **Traffic Analysis:** the attacker gains information by determining how much activity is there, where access points are located,
  - Difficult to detect. Should be prevented.
- **Active Attacks**
  - Attempts to alter system resources or affect their operation, examples: masquerade (spoofing), replay, modification (substitution, insertion, destruction), denial of service.
  - Difficult to prevent. Should be detected.

Categorization of Passive and Active attacks

Attacks	Passive/Active	Threateing
Snooping Traffic Analysis	Passive	Confidentiality
Modification Masquerading/Spoofing Replaying Repudiation	Active	Integrity Authentication Access Control Non Repudiation
Denial of Service	Active	Avaliablity

Kerckhoff's Principle

Generally assumed that the attacker knows everything about the cryptosystem except the key.

Note

If, for security, the system requires that details of the system be kept secret, it is not considered secure.

Trust Model

Trust Model

A trust model is to define trust relations among different parties.

- A trust relation can be defined for any two parties with a mutually equal trust or a asymmetric trust.
- A trust model may include assumptions on the physical environment. For example, we may trust a server located in a companys building more than a wireless access point installed in a rest area of highway.
- A trust model is crucial in establishing a secure communication system. It can go wrong in many ways and result in security holes.

Adversary and Threat Models

Threat Model

A Threat model is to define parties that may attack the system and the attackers' objectives - what they want to do?.

- Also we define how attackers will attack the system?
- What is the attackers' capability, e.g., regarding computational power?
- Is the attacker a single party or colluding with others?

Outline

Groups

A set of objects, along with a binary operation (meaning an operation that is applied to two objects at a time) on the elements of the set, must satisfy/has the following four properties.

1. **Closure** with respect to the operation. Closure means that if a and b are in the set, then the element $a \circ b = c$ is also in the set. The symbol $\circ$ denotes the operator for the desired operation.
2. **Associativity** with respect to the operation. Associativity means that $(a \circ b) \circ c = a \circ (b \circ c)$
3. **A unique identity element** with regard to the operation $\circ$. An element i would be called an identity element if for every a in the set, we have $a \circ i = a$.
4. **An Inverse element** for each element with regard to the operation. That is, for every a in the set, the set must also contain an element b such that $a \circ b = i$. assuming that i is the identity element.

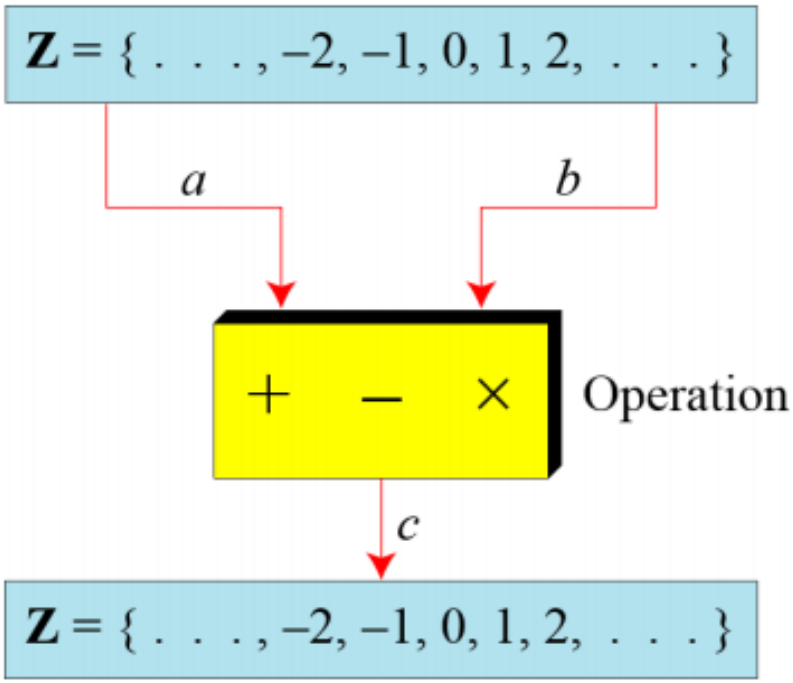
Groups

- In general, a group is denoted by $\{G, \circ\}$ where G is the set of objects and $\circ$ is the operator.
- Infinite groups, meaning groups based on sets of infinite size.
- A finite group contains finite number of elements. The number of elements in G is called the group order and is denoted as $|G|$

Ex.

- The set of all integers: positive, negative, and zero - along with the operation of arithmetic addition constitutes a group
- The set of all even integers : positive, negative, and zero under the operation of arithmetic addition is a group If the operation on the set elements is commutative, the group is called an abelian group. An operation $\circ$ is commutative if $a \circ b = b \circ a$.

Group Example



Add:	$5 + 9 = 14$	$(-5) + 9 = 4$	$5 + (-9) = -4$	$(-5) + (-9) = -14$
Subtract:	$5 - 9 = -4$	$(-5) - 9 = -14$	$5 - (-9) = 14$	$(-5) - (-9) = +4$
Multiply:	$5 \times 9 = 45$	$(-5) \times 9 = -45$	$5 \times (-9) = -45$	$(-5) \times (-9) = 45$

Formal Summary

Defination

A group is a set G together with a binary operation $\circ$ on G that the following properties hold.

Associativity, that is for any $a, b, c \in G, (a \circ b) \circ c = a \circ (b \circ c)$

There is an identity or unity $e \in G$ such that for all $a \in G$, $a \circ e = e \circ a = a$

For each $a \in G$ there exist an Inverse element $a^{-1} \in G$ such that $a \circ a^{-1} = e$

Sometimes, we denote the group as triple $(G, \circ, e)$ if the group also satisfy

$$\text{for all a, b} \in G\text{: a} \circ \text{b} = \text{b} \circ \text{a}$$

Then the group is called abelian or commutative group.

Example

- $\mathbf{Z}$, the set consisting of all integers.
- $\mathbf{Q}$, the set consisting of all rational numbers.
- $+$ and $\times$ are ordinary addition and multiplication.

Then

- $(\mathbf{Z}, +, 0)$, $(\mathbf{Q}, +, 0)$, $(\mathbf{Q^*}, \times, 1)$ are all groups where $\mathbf{Q^*}$ is the set of all nonzero rational numbers.
- Furthermore, the are abelian.

How about $(\mathbf{Z^*}, \times, 1)$?

Notes on Groups

If the group operation is addition, the group also allows for subtraction. Similarly, multiplicative groups allow division.

- A group is guaranteed to have a special element called the identity element. The identity element of a group is frequently denoted by the symbol 0.
- As you now know, for every element a, the group must contain its inverse element b such that $a + b = 0$, where the operator $+$ is the group operator.
- So if we maintain the illusion that we want to refer to the group operation as addition, we can think of b in the above equation as the additive inverse of a and even denote it by -a. We can therefore write $a + (-a) = 0$ or more compactly as $a - a = 0$.
- In general $a - b = a + (-b)$ where -b is the additive inverse of b with respect to the group operator $+$. We may now refer to an expression of the sort a - b as representing subtraction

Outline

Set of Residues

One of the most important structures in crypt: Residues modulo n

Let n be a positive integer $n > 1$ and Z_n represent the set of reminder of all integers on division n, then

$$Z_n = \{0, 1, 2, \ldots, n-1\}$$

We define $a + b$ and $a \times b$ the ordinary sum and product of a and b reduced by modulo n respectively. Let

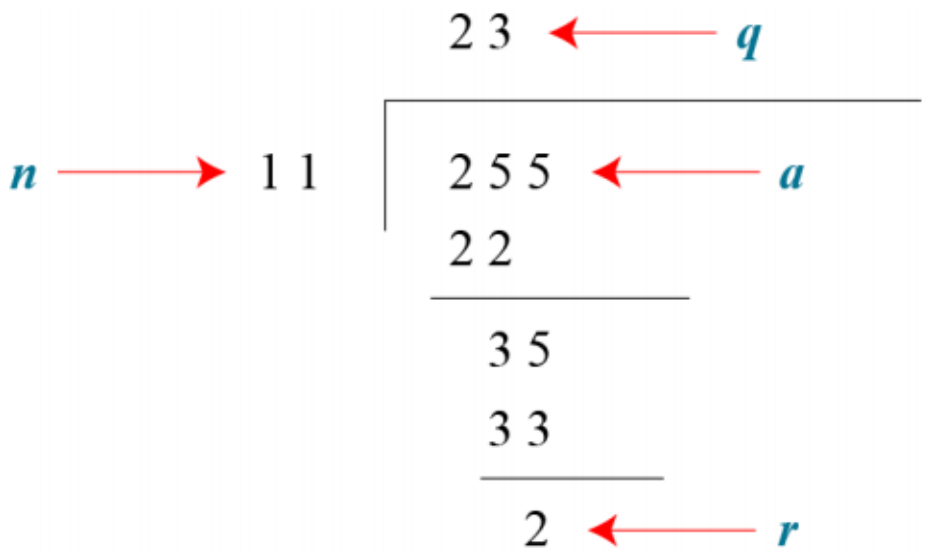
$$Z_n^* = \{a \in Z_n | a \neq 0\}$$

Then

- $(Z_n, +, 0)$ forms a group residue n.
- $(Z_p^*, \times, 1)$ forms a group residue prime p.

Example

255 mod 11 = 2



27 mod 5 = 2
36 mod 12 = 0

First Property:	$(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$
Second Property:	$(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$
Third Property:	$(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$

(a+n) mod n = a

Set of Residues

The modulo operation creates a set, which in modular arithmetic is referred to as the set of least residues modulo n, or Z_n.

$$Z_n = \{0, 1, 2, \ldots, n-1\}$$

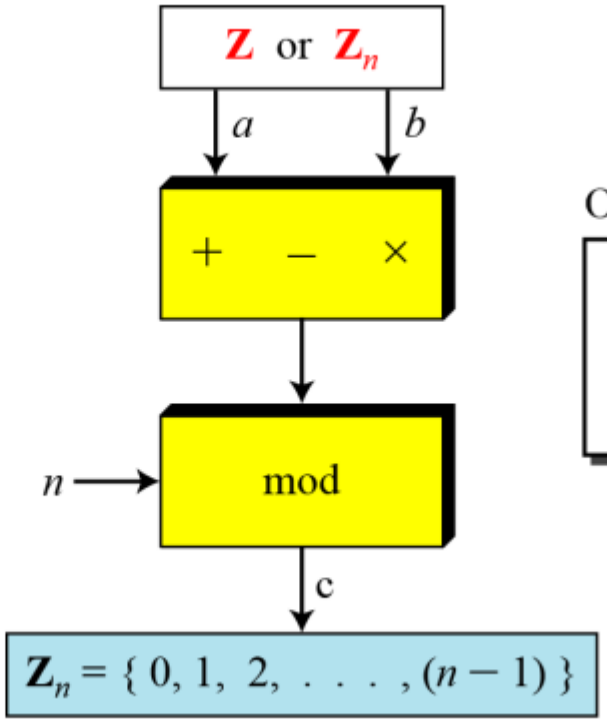
Ex.

- $Z_2 = \{0, 1, 2\}$ prime residue
- $Z_6 = \{0, 1, 2, 3, 4, 5, 6\}$
- $Z_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ prime residue

Operation in Z_n

The three binary operations that we discussed for the set Z can also be defined for the set Z_n. The result may need to be mapped to Z_n using the mod operator.

Binary operations in Z_n



Operations

$(a + b) \bmod n = c$

$(a - b) \bmod n = c$

$(a \times b) \bmod n = c$

Example

Perform the following operations (the inputs come from Zn):

1. Add 7 to 14 in $\mathbf{Z}_{15}$.
2. Subtract 11 from 7 in $\mathbf{Z}_{13}$.
3. Multiply 11 by 7 in $\mathbf{Z}_{20}$.

Sol.

1. (14+7) mod 15 $\rightarrow$ 21 mod 15 = 6
2. (7-11) mod 13 $\rightarrow$ (-4) mod 13 = 9
3. (7×11) mod 20 $\rightarrow$ 77 mod 20 = 17

Inverses

When we are working in modular arithmetic, we often need to find the inverse of a number relative to an operation.

- An additive inverse (relative to an addition operation)
- A multiplicative inverse (relative to a multiplication operation).

Additive Inverse

In Z_n, two numbers a and b are additive inverses of each other if

$$a + b \equiv 0 \mod n$$

In modular arithmetic, each integer has an additive inverse. The sum of an integer and its additive inverse is congruent to 0 modulo n.

Ex.
Find all additive inverse pairs in Z_{10}
Sol.

The six pairs of additive inverses are (0, 0), (1, 9), (2, 8), (3, 7), (4, 6), and (5, 5).

Multiplicative Inverse

In Z_n, two numbers a and b are the multiplicative inverse of each other if

$$a \times b \equiv 1 \bmod n$$

In modular arithmetic, an integer may or may not have a multiplicative inverse. When it does, the product of the integer and its multiplicative inverse is congruent to 1 modulo n.

Ex.
Find all multiplicative inverse pairs in Z_{11}
Sol.

The seven pairs of multiplicative inverses are: (1, 1), (2, 6), (3, 4), (5, 9), (7, 8), (9, 5), and (10, 10).

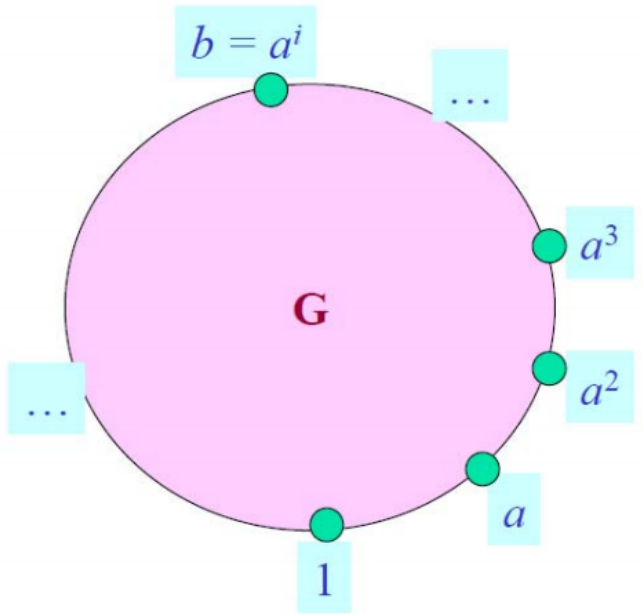
Cryptography often uses these two sets: Z_p and Z_p^*. The modulus in these two sets is a prime number.

Cyclic Group

Cyclic Group

A multiplicative group is said to be cyclic if there is an element $a \in G$ such that for any $b \in G$ there is some integer i with $b = a^i$ Such an element a is called the group generator of the cyclic group, and we write $G =< a >$

Ex.



Examples

- $(Z_3^*, \times, 1)$, cyclic group with generator 2.

$$Z_3^* = \{1,2\} = <2> = \{2^0 = 1, 2^1 = 2\},\ 2^2 = 1 \bmod 3$$

- $(Z_7^*, \times, 1)$, cyclic group with generator 3.

$$Z_7^* = \{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6\}$$

$$Z_7^* = <3> = \{3^0 = 1,\ 3^2 = 2,\ 3^1 = 3,\ 3^4 = 4,\ 3^5 = 5,\ 3^3 = 6,\ 3^6 = 1\}$$

- $(Z_5^*, \times, 1)$, cyclic group with generator 3.

$$Z_5^* = \{0,\ 1,\ 2,\ 3,\ 4\}$$

$$Z_5^* = <2> = \{2^0 = 1,\ 2^1 = 2,\ 2^3 = 3\ ,2^2 = 4\}$$

Rings

- If we can define **one more operation** on an abelian group, we have a ring, provided the elements of the set satisfy some properties with respect to this new operation also.
- A ring is typically denoted $\{R,+,\}$ where R denotes the set of objects, $+$ the operator with respect to which R is an abelian group, the the additional operator needed for R to form a ring.

Properties of the Elements with Respect to the Ring Operator

1. R must be closed with respect to the additional operator .
2. R must exhibit associativity with respect to the additional operator .
3. The additional operator (that is, the multiplication operator) must distribute over the group addition operator. That is
   $a \ (b + c) = a \ b + a \ c$
   $(a + b) \ c = a \ c + b \ c$

The multiplication operation is frequently shown by just concatenation in such equations:
$a(b + c) = ab + ac$
$(a + b)c = ac + bc$

Examples of Rings

- The set of all even integers, positive, negative, and zero, under the operations arithmetic addition and multiplication is a ring.
- The set of all integers under the operations of arithmetic addition and multiplication is a ring.
- The set of all real numbers under the operations of arithmetic addition and multiplication is a ring.

Commutative Rings

- A ring is commutative if the multiplication operation is commutative for all elements in the ring. That is, if all a and b in R satisfy the property $ab = ba$
- The previous three examples are for commutative rings.

Rings Formal Summary

Ring

A ring (R, +,) is a set R, together with two binary operations, denoted by +, such that:

- R is abelian group with respect to +
- is associative, that is,

$$(a \quad b) \quad c = a \quad (b \quad c) \text{ for all } a, b, c \in R$$

- The distributive law hold for ; that is, for all a, b, c $\in$ R we have

$$a \quad (b + c) = a \quad b + a \quad c$$

Ex.

1. (Z, +,), (Q, +,) infinite rings
2. (Z_n, +,) finite ring (residue class ring modulo n)

Integral Domain

An integral domain R,+, is a commutative ring that obeys the following two additional properties:

1. **ADDITIONAL PROPERTY 1:** The set R must include an identity element for the multiplicative operation. That is, it should be possible to symbolically designate an element of the set R as 1 so that for every element a of the set we can say a 1 = 1 a = a
2. **ADDITIONAL PROPERTY 2:** Let 0 denote the identity element for the addition operation. If a multiplication of any two elements a and b of R results in 0, that is if ab = 0 then either a or b must be 0.

Examples

- The set of all integers under the operations of arithmetic addition and multiplication.

Fields

Field

A field, denoted $\{F, +, \}$, is an integral domain whose elements satisfy the following **additional property**:

- For every element a in F, except the element designated 0 (the identity element for the '+' operator), there must also exist in F its multiplicative inverse.

That is, if $a \in F$ and $a + b = 0$, then there must exist an element $b \in F$ such that $a \ b = b \ a = 1$ where '1' symbolically denotes the element which serves as the identity element for the multiplication operation.

For a given a, the multiplicative inverse is designated as a^{-1}.

Note That

A field has a multiplicative inverse for every element except the element that serves as the identity element for the group operator.

Examples of Fields

1. The set of all real numbers under the operations of arithmetic addition and multiplication
   - Is a field.
2. The set of all rational numbers under the operations of arithmetic addition and multiplication
   - Is a field.
3. The set of all even integers, positive, negative, and zero, under the operations arithmetic addition and multiplication.
   - **NOT** a field.
4. The set of all integers under the operations of arithmetic addition and multiplication
   - **NOT** a field.

Field Summary

A field is a set of elements with addition and multiplication operations satisfying these rules:

1. **Two operations** (addition and multiplication) are defined on every element in the field.
2. **Closure** The addition/multiplication of two elements in the field gives an element in the field.
3. **Associativity** $a+(b+c) = (a+b)+c$ and $a(bc) = (ab)c$, where a, b and c are elements in the field.
4. **Commutativity** $a+b = b+a$ and $ab = ba$
5. **Additive identity** Theres a single element denoted 0 such that $a + 0 = a$ for all elements in the field.
6. **Multiplicative identity** There is an element denoted 1 that $a1 = a$ for every element in the field.
7. **Multiplicative inverse** For a given a, where $a \neq 0$ the multiplicative inverse is designated as a^{-1}

Z_p Facts

- We are dealing with primes p on the order of 300 digits long, (1024 bits).
- For a prime p let $Z_p = \{0, 1, 2, ..., p-1\}$.
  - Elements of Z_p can be added and multiplied modulo p.
  - The inverse of $x \in Z_p$ is an element a satisfying a * x = 1 mod p
  - Fermats theorem: for any g ≠ 0 mod p we have: $g^{p-1} = 1$ mod p.

    Example: 3^4 mod 5 = 81 mod 5 = 1
  - All elements $x \in Z_p$ except for x = 0 are invertible.

    Simple inversion algorithm: $x^{-1} = x^{p-2}$ mod p.
- we can deal with $(Z_p, +, \times)$ as field.

Z_p^* Facts

- Z_p^* is a cyclic group. In other words, there exists $g \in Z_p^*$ such that $Z_p^* = \{1, g, g^2, g^3, ..., g^{p-2}\}$. g is called a generator of Z_p^*.

  Example: Z_7^*: $< 3 > = \{1, 3, 3^2, 3^3, 3^4, 3^5, 3^6\}$
  $= \{1, 3, 2, 6, 4, 5\}\ (\text{mod} 7) = Z_7^*$.

- Not every element of Z_p^* can be a generator.

Problems that are believed to be hard in Z_p^*

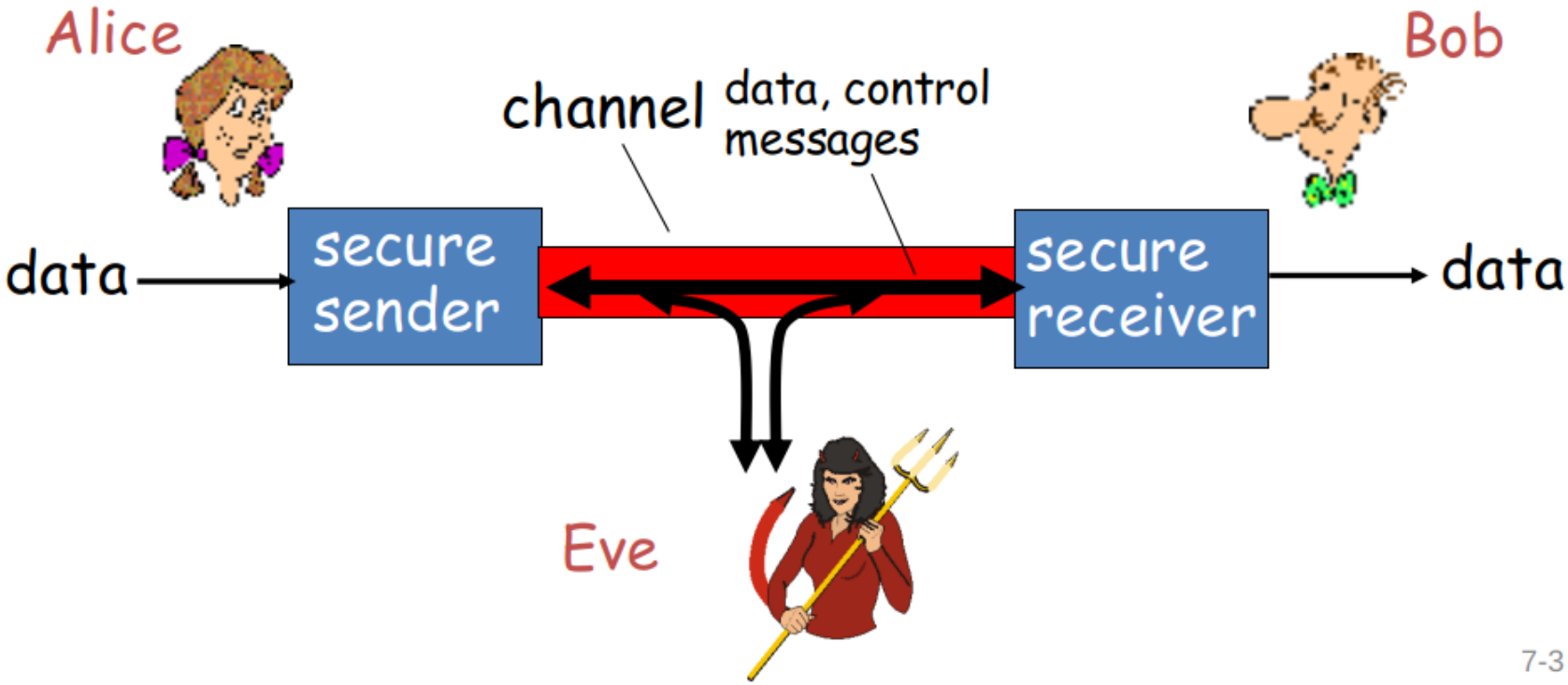
- Let g be a generator of Z_p^*. Given x $\in Z_p^*$ find an r such that x $= g^r$ mod p. This is known as the **discrete log problem**.
- Let g be a generator of Z_p^*. Given x, y $\in Z_p^*$ where x $= g^{r_1}$ and y $= g^{r_2}$. Find z $= g^{r_1 r_2}$. This is known as the **Diffie-Hellman problem**.

Outline

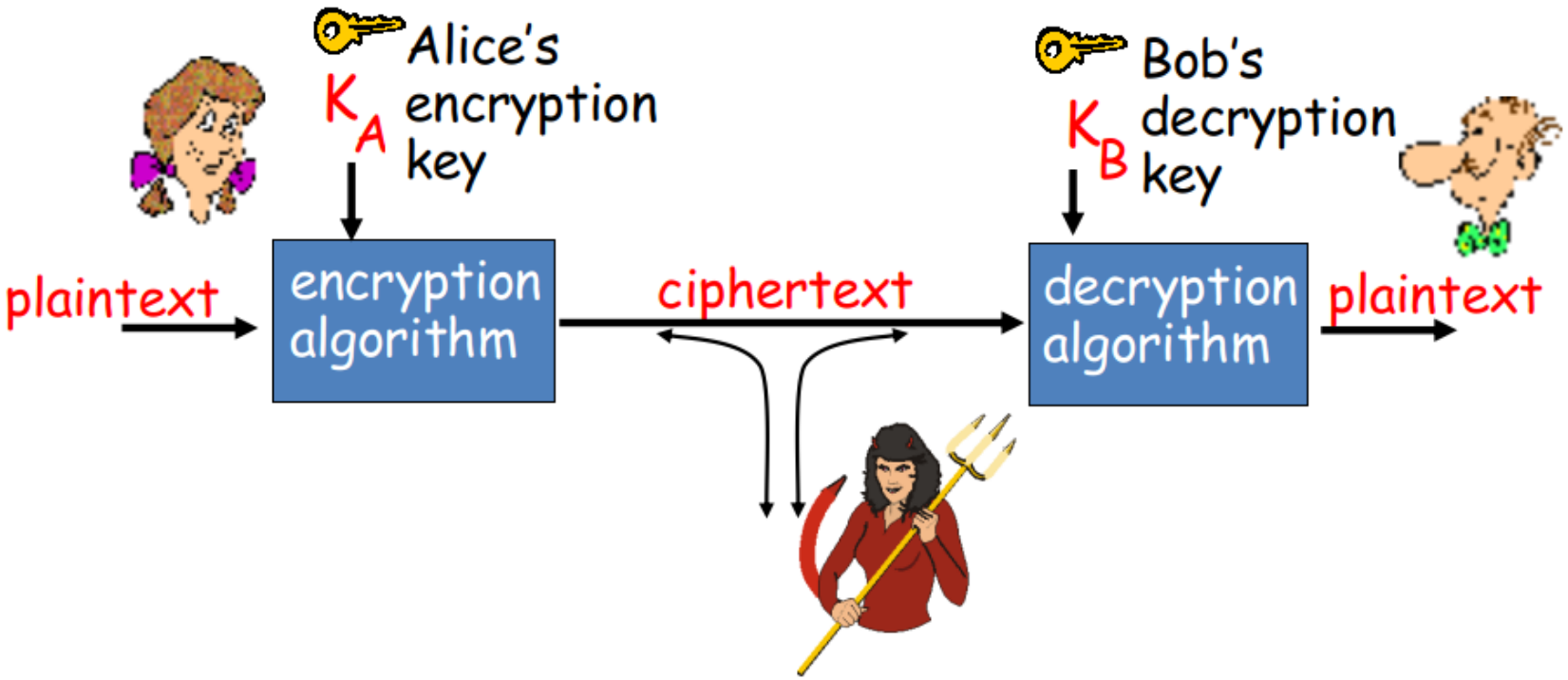
Outline

Friends and Enemies

- Bob, Alice (lovers!) want to communicate securely
- Eve (or Trudy, intruder) may intercept, delete, add messages



The Language of Cryptography

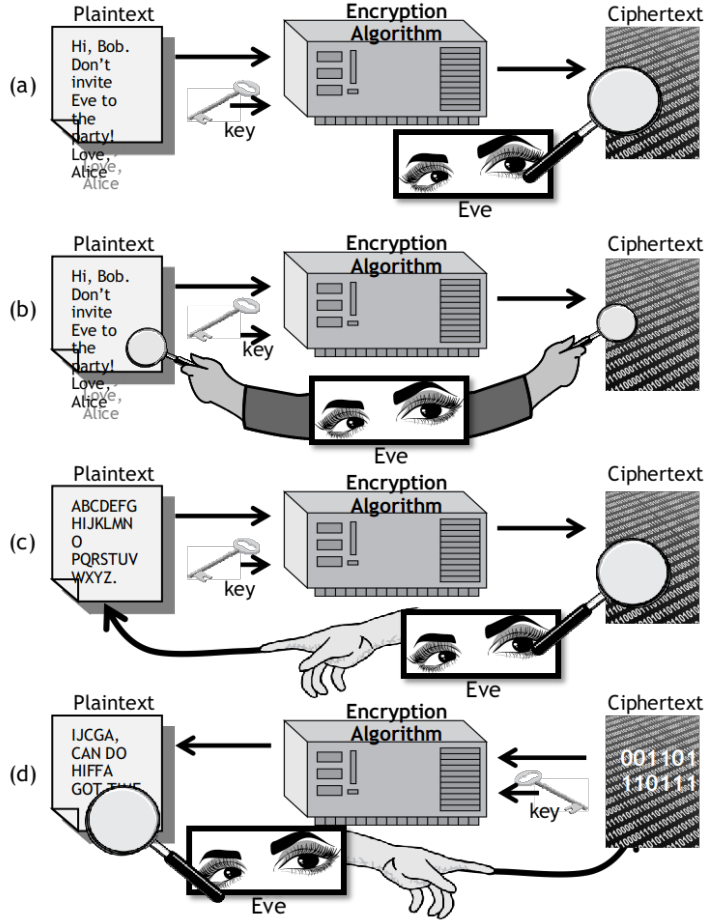


- Symmetric key crypto: sender, receiver keys identical
- Public-key crypto: encryption key (public), decryption key secret (private)

Attacks

The following are cryptanalysis methods to measure the security of encryption schemes. (ordered from weakest to strongest)

- **Attacker may have**
  1. Collection of ciphertexts (known ciphertext attack)
  2. collection of plaintext/ciphertext pairs (known plaintext attack)
  3. collection of plaintext/ciphertext pairs for plaintexts selected by the attacker (chosen plaintext attack)
  4. collection of plaintext/ciphertext pairs for ciphertexts selected by the attacker (chosen ciphertext attack)



Brute Force Attack

- Try all possible keys K and determine if $D_K(C)$ is a likely plaintext
  - Requires some knowledge of the structure of the plaintext (e.g., PDF file or email message)
- Key should be a sufficiently long random value to make exhaustive search attacks unfeasible

Symmetric Cryptosystem

- **Scenario**
  - Alice wants to send a message (plaintext P) to Bob.
  - The communication channel is insecure and can be eavesdropped
  - If Alice and Bob have previously agreed on a symmetric encryption scheme and a secret key K, the message can be sent encrypted (ciphertext C)
- **Issues**
  - What is a good symmetric encryption scheme?
  - What is the complexity of encrypting/decrypting?
  - What is the size of the ciphertext, relative to the plaintext?

Symmetric Key Cryptography Basics

- **Notation**
  - Secret key K
  - Encryption function $E_K(P)$
  - Decryption function $D_K(C)$
  - Plaintext length typically the same as ciphertext length
  - Encryption and decryption are one-to-one mapping functions on the set of all n-bit arrays
- **Efficiency**
  - functions E_K and D_K should have efficient algorithms
- **Consistency**
  - Decrypting the ciphertext yields the plaintext
  - $D_K(E_K(P)) = P$

Classical Cryptography

- Transposition Cipher
- Substitution Cipher
  - Simple substitution cipher (Caesar cipher)
  - Vigenere cipher
  - One-time pad

Transposition Cipher: rail fence

- Write plaintext in two or more rows
- Generate ciphertext in column order

Example:
"HELLOWORLD"

HLOOL
ELWRD

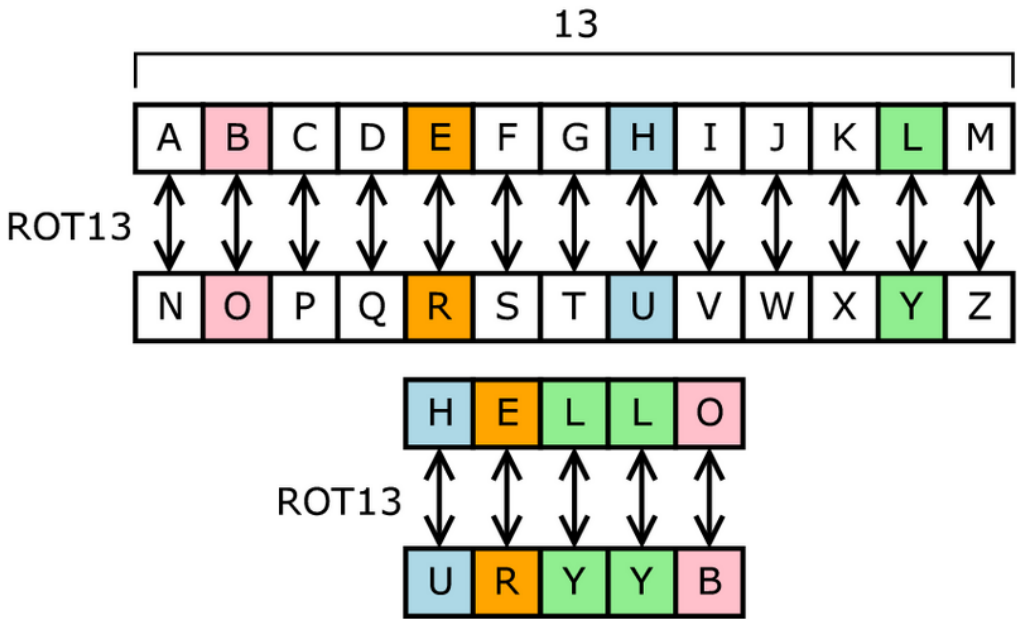
ciphertext: HLOOLELWRD

Note

Problem: does not affect the frequency of individual symbols

Substitution Ciphers

- Each letter is uniquely replaced by another.
- There are 26! possible substitution ciphers for English language.
- Also know as **Caesar Cipher**
- One popular substitution cipher for some Internet posts is ROT13.

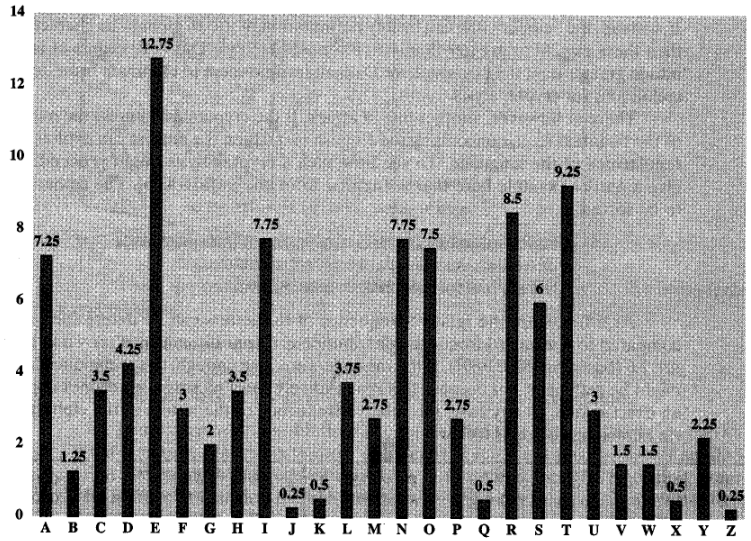


Frequency Analysis

- Letters in a natural language, like English, are not uniformly distributed.
- Knowledge of letter frequencies, including pairs and triples can be used in cryptologic attacks against substitution ciphers.

a:	8.05%	b:	1.67%	c:	2.23%	d:	5.10%
e:	12.22%	f:	2.14%	g:	2.30%	h:	6.62%
i:	6.28%	j:	0.19%	k:	0.95%	l:	4.08%
m:	2.33%	n:	6.95%	o:	7.63%	p:	1.66%
q:	0.06%	r:	5.29%	s:	6.02%	t:	9.67%
u:	2.92%	v:	0.82%	w:	2.60%	x:	0.11%
y:	2.04%	z:	0.06%				

8.1: Letter frequencies in the book *The Adventures of Tom Sawyer*, by Twain.



Vigenere Cipher

- Idea: Uses Caesar's cipher with various different shifts, in order to hide the distribution of the letters.
- A key defines the shift used in each letter in the text
- A key is repeated as many times as required to become the same length of the plaintext.

Example.

Plain text: l a t t a c k
Key: 2 3 4 2 3 4 2 (key is "234")
Cipher text: K d x v d g m

Problem of Vigenere Cipher

- Vigenere is easy to break (Kasiski, 1863):
- Assume we know the length of the key. We can organize the ciphertext in rows with the same length of the key. Then, every column can be seen as encrypted using Caesar's cipher.
- Length of the key can be induced using several techniques.

One-Time Pads

- Extended from Vigenere cipher
- There is one type of substitution cipher that is absolutely unbreakable.
- The one-time pad was invented in 1917 by Joseph Mauborgne and Gilbert Vernam We use a block of shift keys, $(k_1, k_2, ..., k_n)$, to encrypt a plaintext, M, of length n, with each shift key being chosen uniformly at random.
- Since each shift is random, every ciphertext is equally likely for any plaintext.

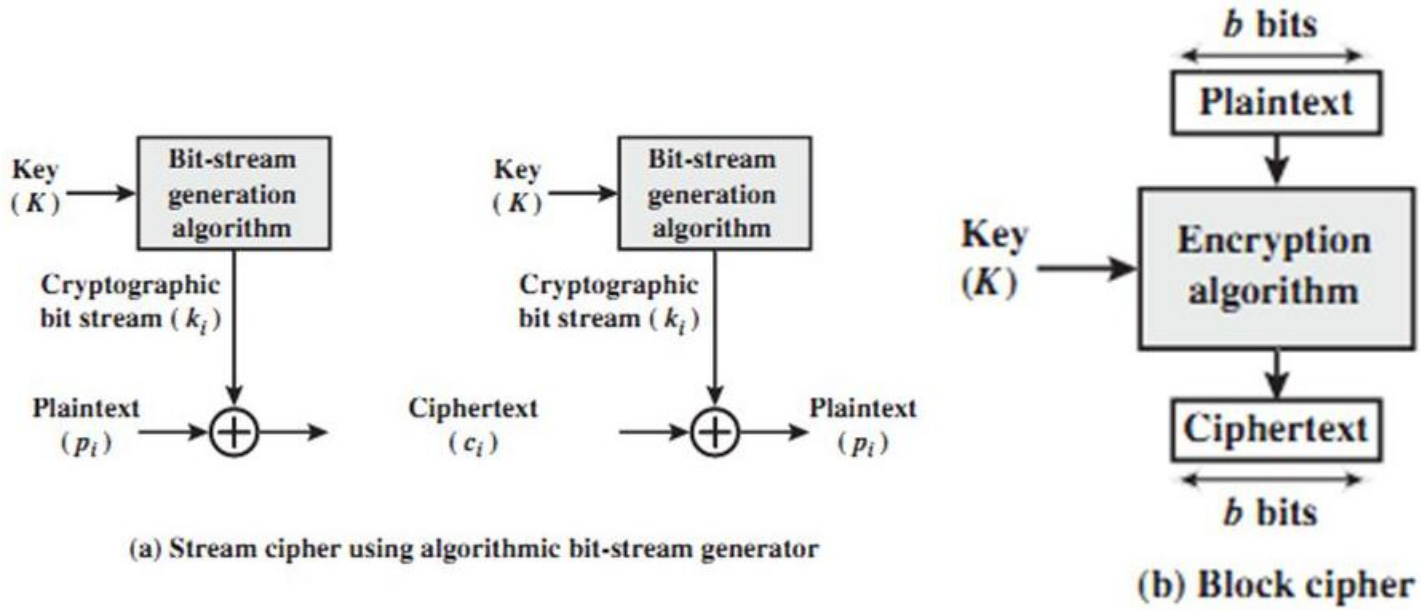
Note that

One time pad is not practical to implement. Why?

- The key has to be as long as the plaintext

Practical Symmetric Key Encryption

- Two Types of symmetric key encryption
  - Block Ciphers.
  - Stream Ciphers.



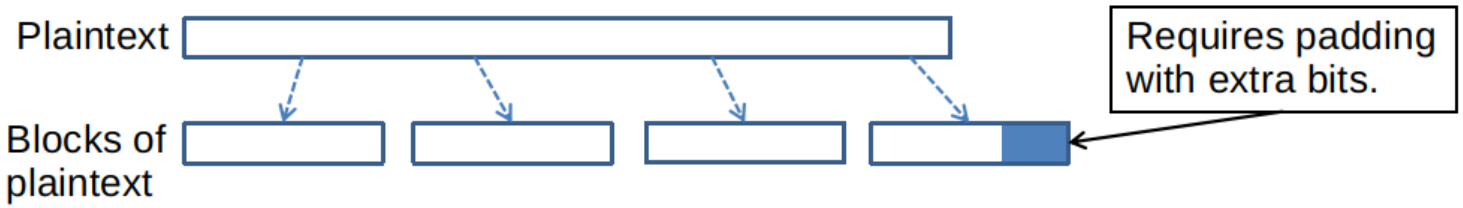
(a) Stream cipher using algorithmic bit-stream generator

(b) Block cipher

Block Cipher

- In **a block cipher**:
  - Plaintext and ciphertext have fixed length b (e.g., 128 bits)
  - A plaintext of length n is partitioned into a sequence of m blocks.

    P[0], . . . , P[m-1].
- Each message is divided into a sequence of blocks and encrypted or decrypted in terms of its blocks.



Padding

- Block ciphers require the length of the plaintext to be a multiple of the block size b
- Padding the last block needs to be unambiguous (cannot just add zeroes)
- When the block size and plaintext length are a multiple of 8, a common padding method (PKCS5) is a sequence of identical bytes, each indicating the length (in bytes) of the padding

Example for b = 128 (16 bytes)
Plaintext: "Roberto" (7 bytes)
Padded plaintext: "Roberto999999999" (16 bytes), where 9 denotes the number and not the character

Desirable Properties of Ciphers

Security

1. **Diffusion** Process of spreading effect of plaintext or key as widely as possible over ciphertext

Avalanche effect

Approximately half of the ciphertext bits change (at random) in response to a one bit change in the plaintext or the key.

2. **Confusion** The relationship between key and ciphertext bits should be complicated. Ciphertext and plaintext should appear to be statistically independent

Efficiency

1. High encryption and decryption rate.
2. Simplicity (easier to implement and analyze).
3. Suitability for hardware or software.
4. Key size should be small, but large enough to preclude exhaustive key search.

Block Ciphers in Practice

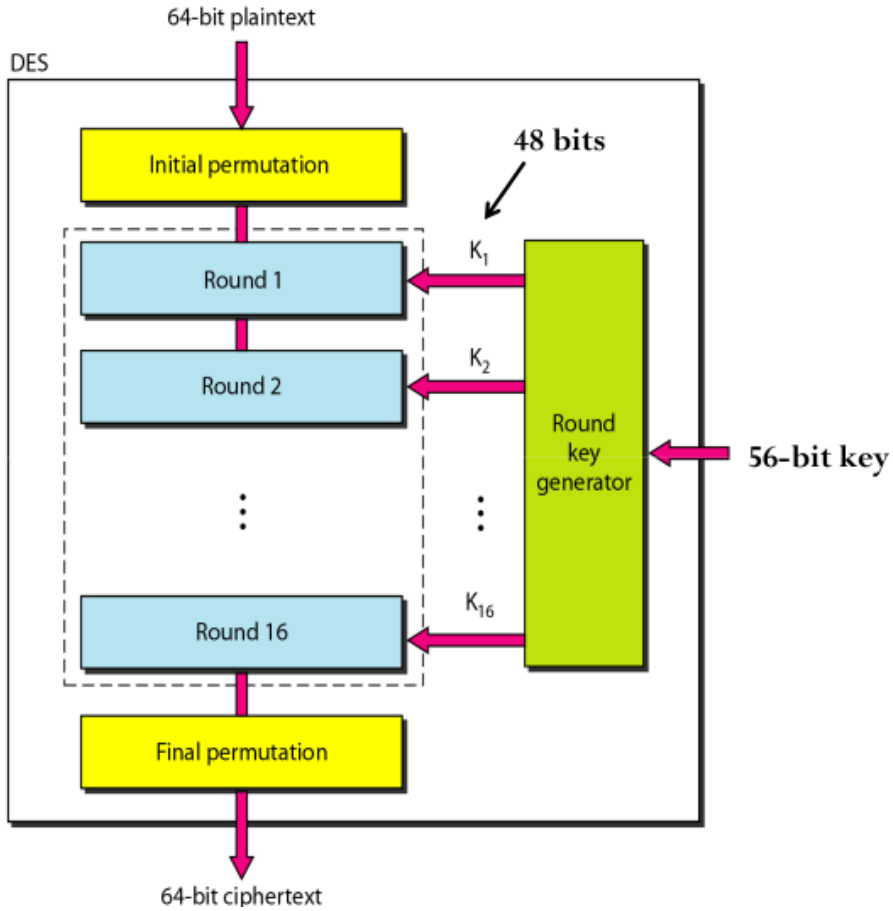
- **Data Encryption Standard (DES)**
  - Developed by IBM and adopted by NIST in 1977 64-bit blocks and 56-bit keys
  - Small key space makes exhaustive search attack feasible since late 90s
- **Triple DES (3DES)**
  - Nested application of DES with three different keys K_A, K_B, and K_C
  - Effective key length is 168 bits, making exhaustive search attacks unfeasible
  - $C = E_{K_C}(D_{K_B}(E_{K_A}(P)))$; $P = D_{K_A}(E_{K_B}(D_{K_C}(C)))$
  - Equivalent to DES when $K_A = K_B = K_C$ (backward compatible)
- **Advanced Encryption Standard (AES)**
  - Selected by NIST in 2001 through open international competition and public discussion
  - 128-bit blocks and several possible key lengths: 128, 192 and 256 bits
  - Exhaustive search attack not currently possible
  - AES-256 is the symmetric encryption algorithm of choice

Data Encryption Standard (DES)

Underling principle: Take something simple and use it several times; hope that the result is complicated.

- Easy to implement. The code of one round can be repeated.

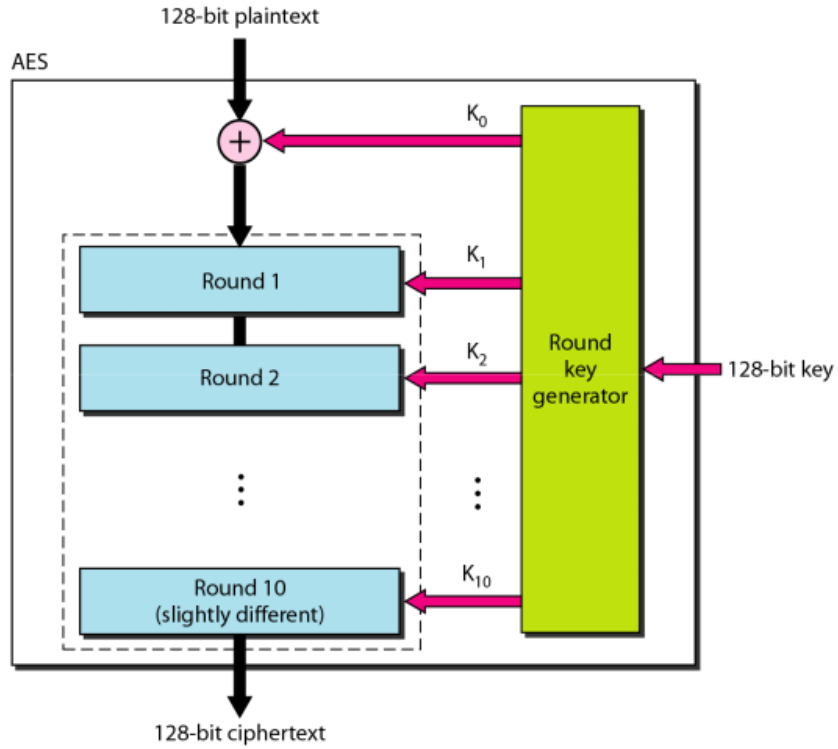
It was observed that alternating rounds of simple substitutions and transpositions could produce a strong cipher (even though individual operations are not strong).



Each round is simply some substitution and permutation operation

Advanced Encryption Standard (AES)

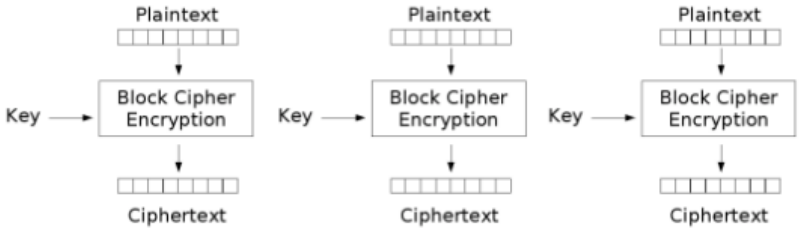
- Same underlying principle
- AES operates on 128-bit blocks. It is designed to be used with keys that are 128, 192, or 256 bits long, yielding ciphers known as AES-128, AES-192, and AES-256.



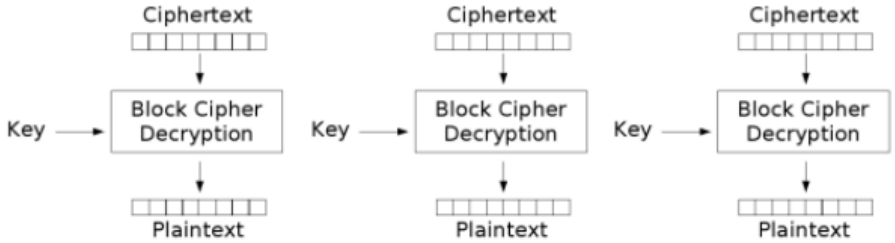
Each round is simply some substitution and permutation operation

Block Cipher Modes

- A block cipher mode describes the way a block cipher encrypts and decrypts a sequence of message blocks.
- Electronic Code Book (ECB) Mode (is the simplest):
  - Block P[i] encrypted into ciphertext block C[i] = $E_K(P[i])$



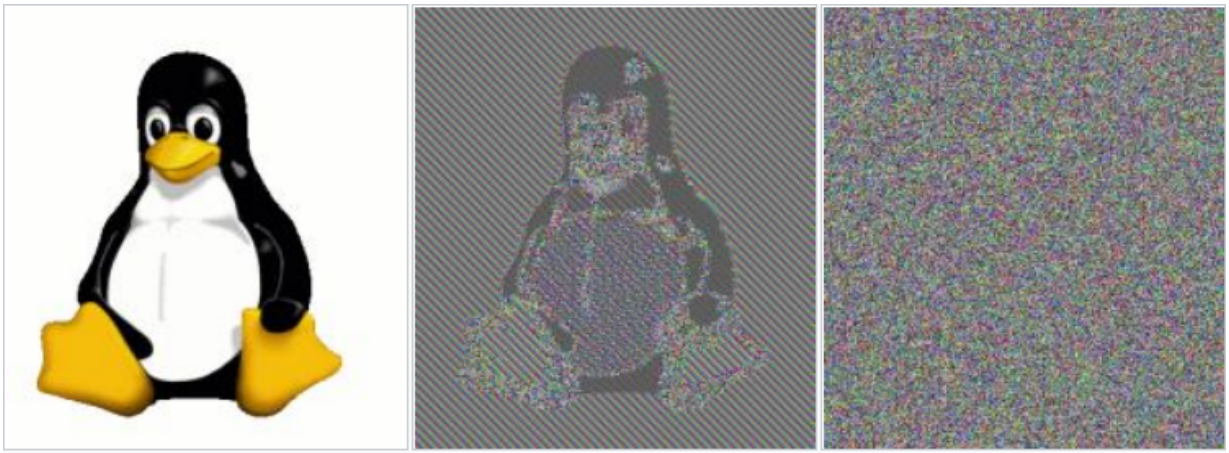
Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

Block Ciphers Modes (ECB)

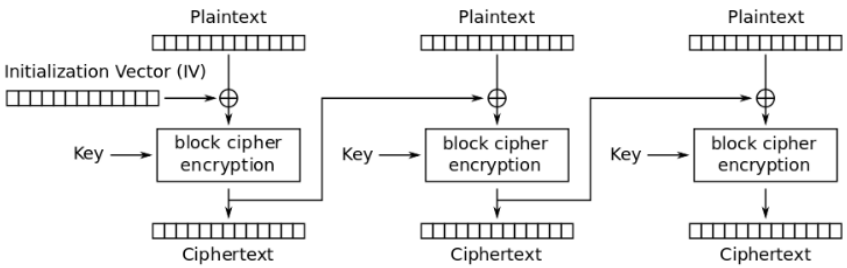
- **Strengths**
  - Is very simple
  - Allows for parallel encryptions of the blocks of a plaintext
  - Can tolerate the loss or damage of a block
- **Weakness**
  - Documents and images are not suitable for ECB encryption since patters in the plaintext are repeated in the ciphertext



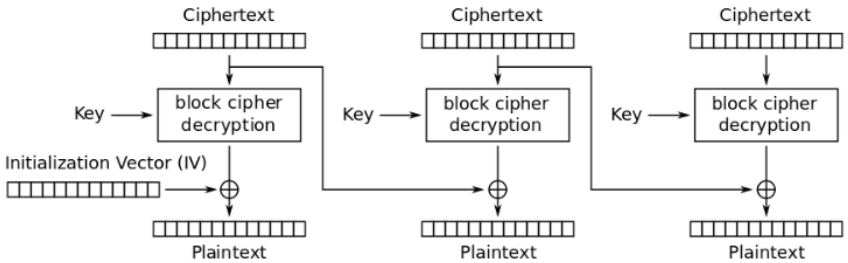
Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

Cipher Block Chaining (CBC) Mode

- In Cipher Block Chaining (CBC) Mode
  - The previous ciphertext block is combined with the current plaintext block
  - $C[i] = E_K(C[i-1] \oplus P[i])$
  - $C[-1] = V$, a random block separately transmitted encrypted (known as the initialization vector)
  - Decryption: $P[i] = C[i-1] \oplus D_K(C[i])$



Cipher Block Chaining (CBC) mode encryption

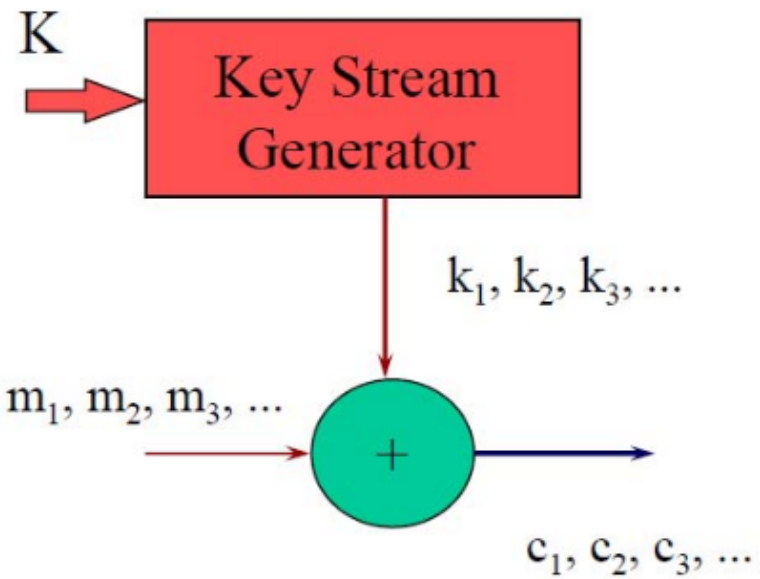


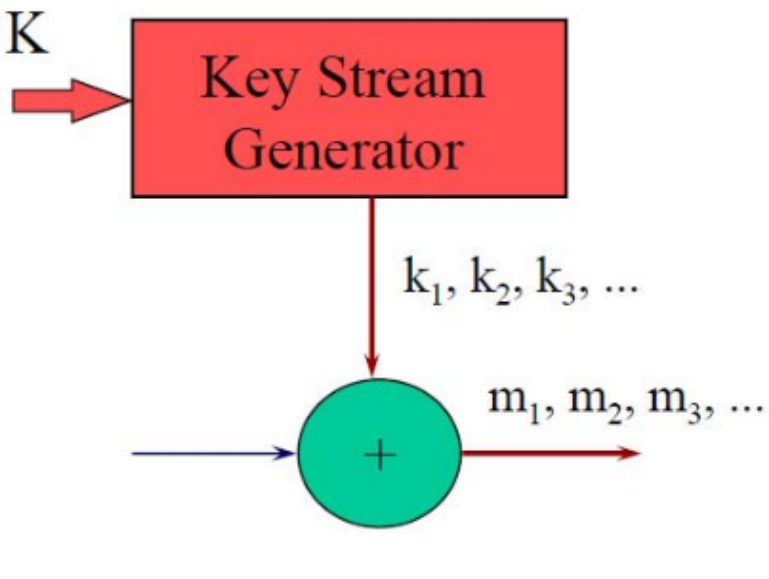
Cipher Block Chaining (CBC) mode decryption

Block Ciphers Modes (CBC)

- **Strengths**
  - Doesnt show patterns in the plaintext
  - Is the most common mode
  - Is fast and relatively simple
- **Weakness**
  - CBC requires the reliable transmission of all the blocks sequentially
  - CBC is not suitable for applications that allow packet losses (e.g., music and video streaming)
  - Error propagation: One bit error in a ciphertext block C_j has an effect on the j-th and the following plaintexts.

Stream Ciphers

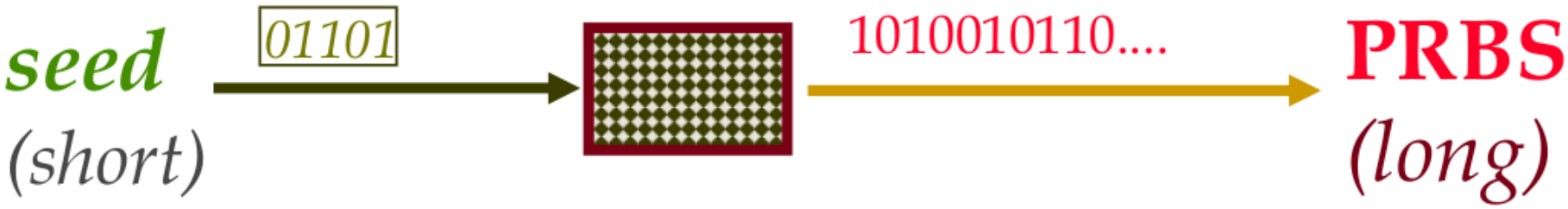




Stream Ciphers

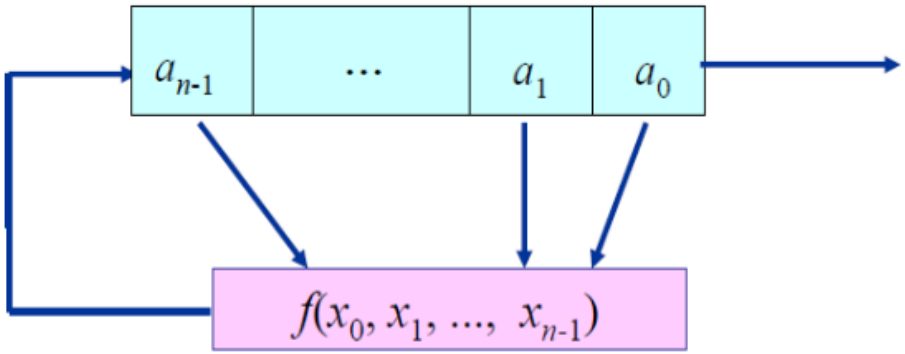
- Suitable when data is transmitted in serial form (one bit at a time).
- A pseudorandom generator is used to generate Key-Stream which is combined with Message-Stream to produce Cipher- Stream
- No error propagation a ciphertext character that is modified during transmission affects only the decryption of that character. May be advantageous when transmission errors are probable such as in wireless transmissions.
- Shannons Result (1948): One-time-pad is unbreakable assuming the key stream is random and cannot be reconstructed.
- One-time-pad means that different messages are encrypted by different key streams.
- The key stream is generated independently at the sender and receiver

Pseudorandom generator



- A random number is a number that cannot be predicted by an observer before it is generated.
- A cryptographic pseudo-random number generator (PRNG) is a mechanism that takes as input a (random and secret) seed, and outputs a longer pseudorandom sequence called the keystream.
- if designed, implemented, and used properly, then even an adversary with enormous computational power should not be able to distinguish the PRNG output from a real random sequence

Pseudorandom generator: Feedback Shift Registers (FSR)



For example, the next state of Fig. 1 is

where

$$a_n = f(a_0, a_1, \ldots, a_{n-1})$$

and the device outputs one bit a_0.

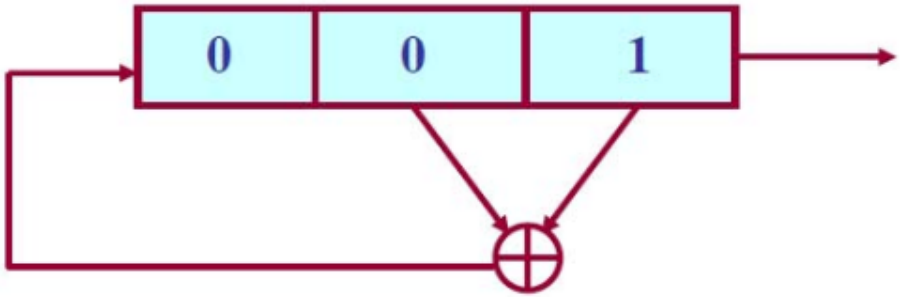
At each clock pulse: the state of each memory stage is shifted to the next stage in line, i.e., there is a transition from one state to next

Pseudorandom generator: Feedback Shift Registers (FSR)

Example A 3-stage LFSR with a feedback function

$$f(x_0, x_1, x_2) = x_0 + x_1$$

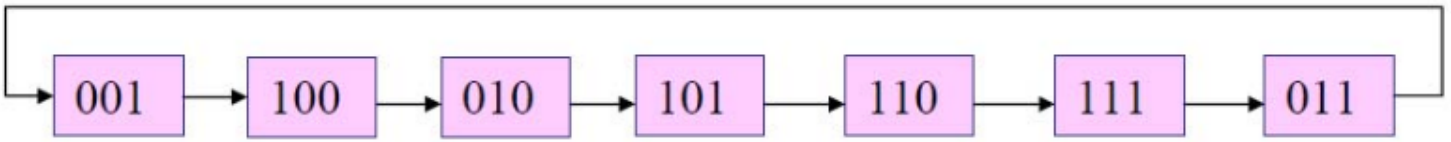
(1) Implementation



(2) Recursive relation:

$$a_{3+k} = a_{1+k} + a_k, k = 0, 1, \cdots$$

(3) State Diagram



Desirable properties of PRNGs

- The adversary cannot compute the internal state of the PRNG, even if she has observed many outputs of the PRNG
- The adversary cannot compute the next output of the PRNG, even if she has observed many previous outputs of the PRNG
- It should be difficult to detect the seed from the output stream.
- Needs synchronization between the sender and the receiver. If a character is inserted into or deleted from the ciphertext stream then synchronization is lost and the plaintext cannot be recovered. Additional techniques must be used to recover from loss of synch.
- The keystream should be indistinguishable from a random sequence.

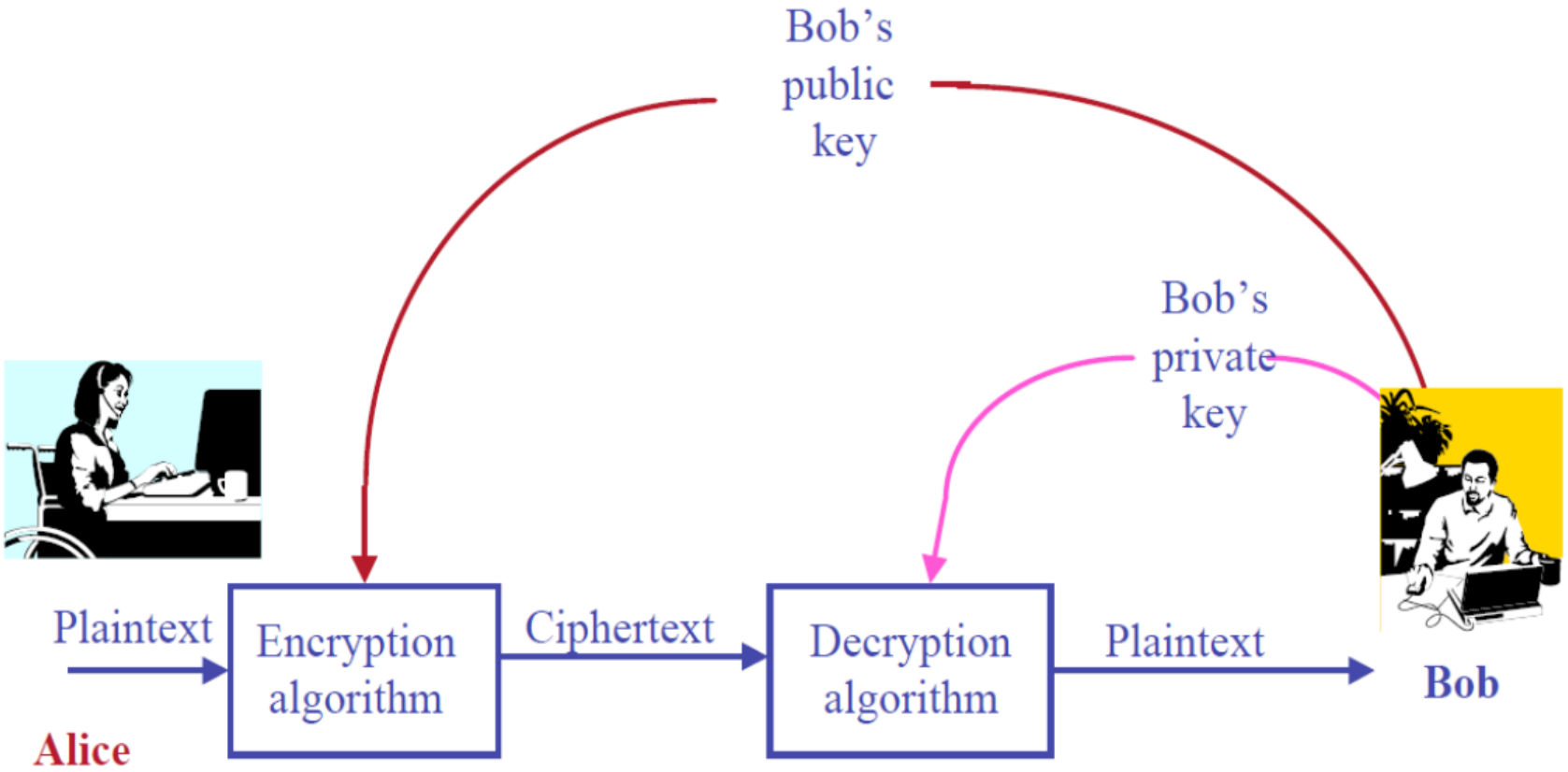
Dangers of Keystream Reuse

What if we reused the same keystream for different ciphertexts?

$$\overbrace{(M_1 \oplus K)}^{C1} \oplus \overbrace{(M_2 \oplus K)}^{C2} = M_1 \oplus M_2$$

- Analysis on $M_1 \oplus M_2$ can reveal one or both plaintexts.
- The same seed should not be reused to avoid generating the same key stream.

Outline



- Symmetric-key and asymmetric-key ciphers are complements of each other; the advantages of one can compensate for the disadvantages of the other.
- Asymmetric-key ciphers are sometimes called public-key ciphers.

Public Key Cryptosystms Security

Public Key cryptosystems are base on the difficulty of the following computational problems.

1. **Factoring Large Integers** (Will be covered)
   - It is easy to compute n $= p \times q$ given two large primes p and q, but it is hard to find p and q given n. Used in RSA
2. **Finite Discrete Logarithms** (Will be covered)
   - Given g and (g^a mod P) are two points in a finite field, it is infeasible to calculate a, where P is a large prime number.
3. **Elliptic Curve Discrete Logarithms** (Will be covered)
   - Given P and aP are two points in an elliptic curve, it is infeasible to calculate a.
4. **Lattice-based Cryptography** (Will not be covered)
   - Resistant to attack by both classical and quantum computers.
   - Is based on shortest vector problem (SVP).

RSA

- RSA is one of the first public-key cryptosystems. First published 1977.
- Depends on the difficulty of factorizing large prime numbers.

Choosing Keys:

1. Choose two large prime numbers p, q. (e.g., 1024 bits each)
2. Compute n = pq, z = (p-1)(q-1)
3. Choose e (with e<n) that has no common factors with z. (e, z are relatively prime).
4. Choose d such that ed mod z = 1 . (i.e., e and d are multiplicative inverses with respect to modulo z)
5. Public key is (n,e). Private key is (n,d)

RSA: Encryption, decryption

1. Given (n, e) and (n, d) as computed above
2. To encrypt bit pattern, m, compute $c = m^e \bmod n$ (i.e., remainder when m is divided by n)
3. To decrypt received bit pattern, c, compute $m = c^d \bmod n$ (i.e., remainder when c is divided by n)

Magic Happens

$$m = \underbrace{(m^e \bmod n)^d}_{c} \bmod n$$

RSA example

Bob chooses p=5, q=7. Then n=35, z=24.

e=5 (so e, z are relatively prime).
d=29 (so ed = 1 mod z).

- **Encrypt**
  - Let m =12
  - $m^e = 12^5 = 1524832$
  - m^e mod n = 17
- **Decrypt**
  - Let c =17
  - $c^d = 17^{29} = 481968572106750915091411825223071697$
  - c^d mod n = 12

Computationally expensive!

Why it works?

$$m = \underbrace{(m^e \bmod n)^d}_{c} \bmod n$$

Useful number theory result: If p,q prime and n = pq, then:

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

$$\begin{aligned}(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{ed \bmod (p-1)(q-1)} \bmod n \\ &= m^1 \bmod n \\ &= m\end{aligned}$$

remember ed are multiplicative inverses for modulo (p-1)(q-1)

RSA First Property

$$E_e(D_d(m)) = m = E_d(D_e(m))$$

1. use public key first, followed by private key
2. use private key first, followed by public key

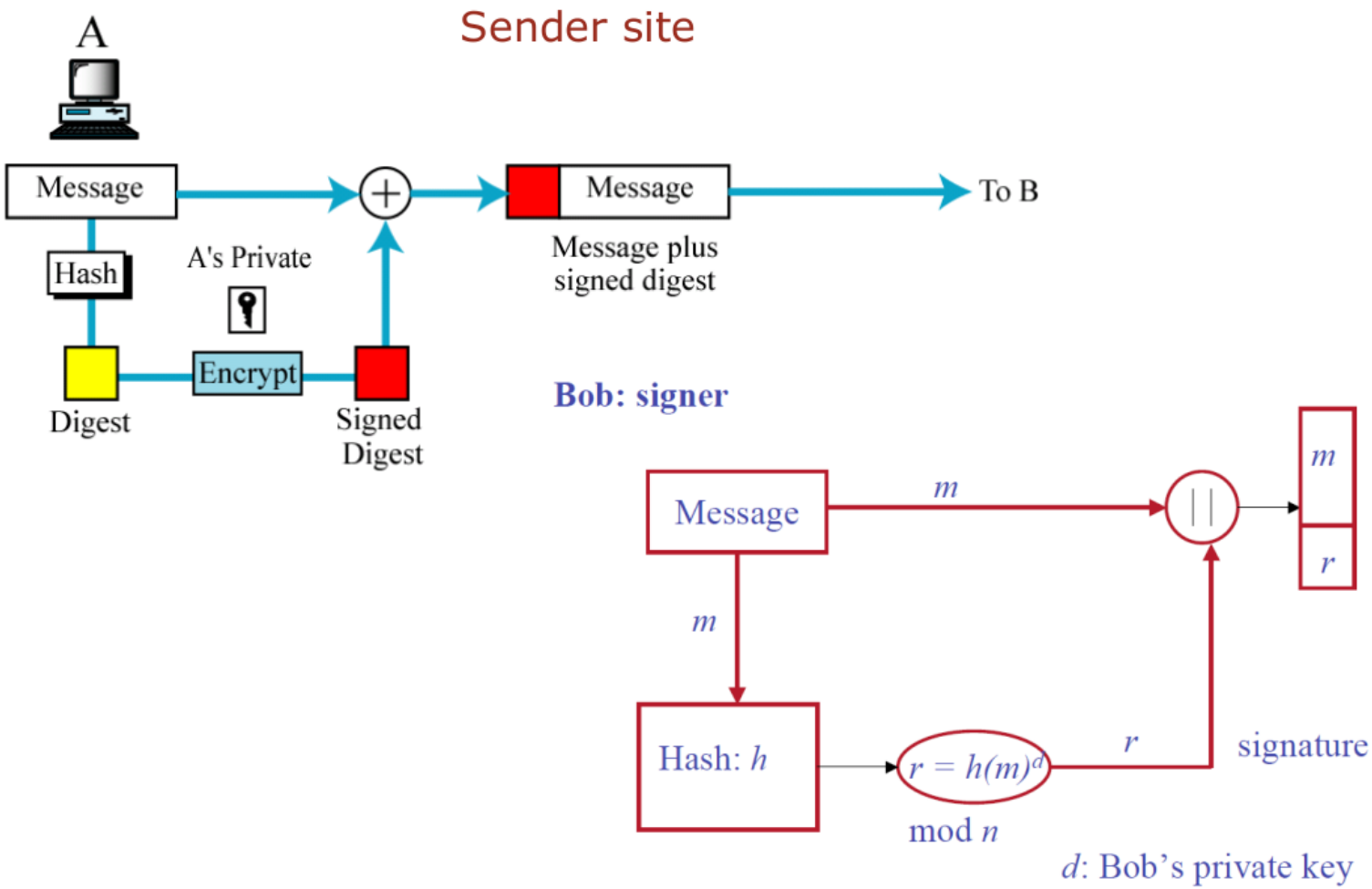
Result is the same in both cases

- Encryption with private key is used for signing messages
  - In digital signature, only one entity can sign a message **(i.e., private key holder)**, and any one can verify the signature **(i.e., public key holder(s))**.

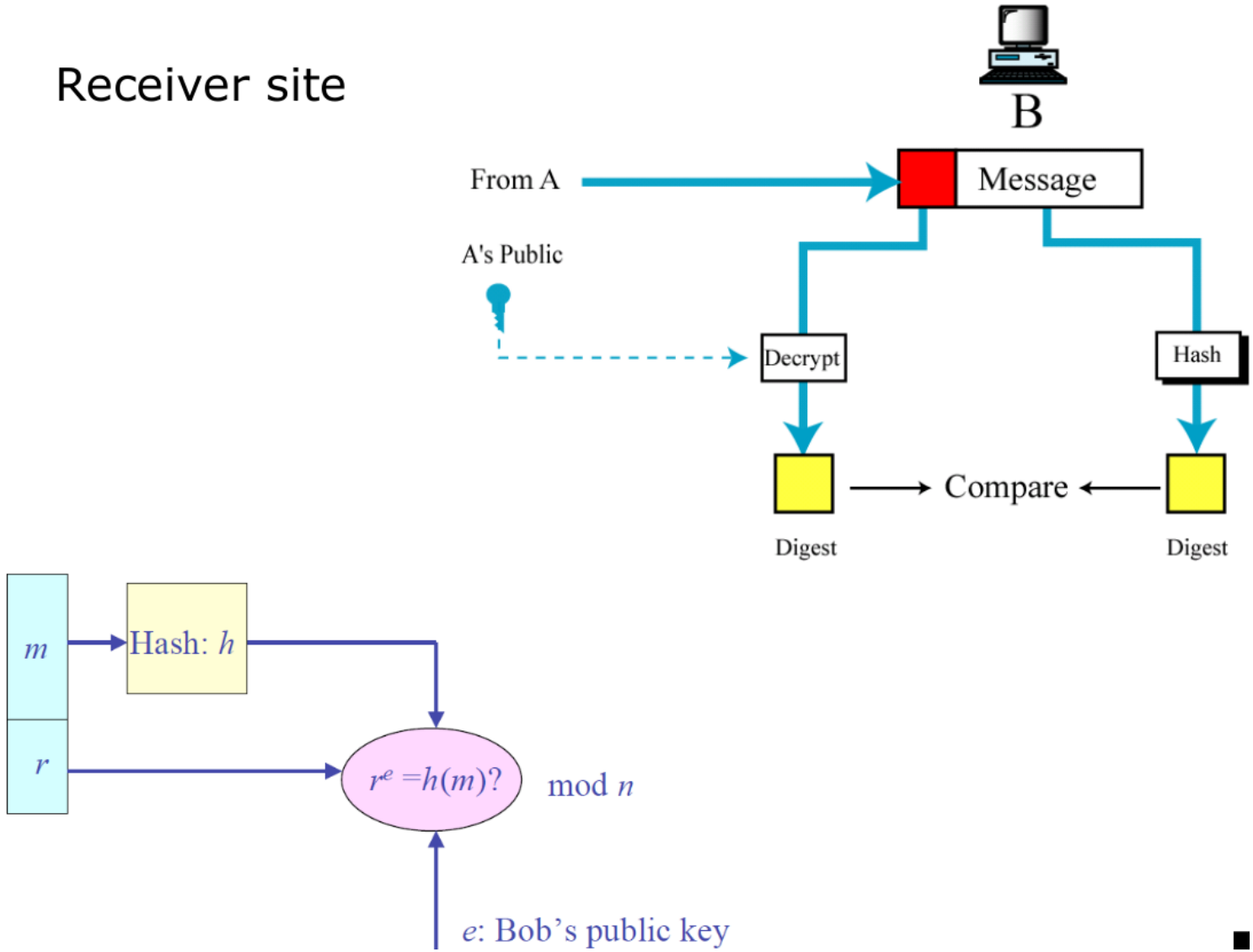
Ex.

- Alice send (m, Enc_e(m))
- Bob check $Dec_d(Enc_e(m)) \stackrel{?}{=}$ m
- Entity authentication and data interity are achieved but not the confidentiality.

RSA signature scheme



RSA signature scheme



RSA Second Property

$$\begin{aligned} E(m_1).E(m_2) &= m_1^e.m_2^e \bmod n \\ &= (m_1.m_2)^e \bmod n \\ &= E(m_1.m_2) \end{aligned}$$

RSA is Partially homomorphic cryptosystem
(Allow multiplication on encrypted data)

RSA Security

- Security of RSA based on difficulty of factoring of n=pq
  - Widely believed
  - Best known algorithm takes exponential time

RSA in practice:

- Time consuming process
- DES is 100 times (in sw) and 10000 times (in hw) faster than RSA.
- Usually exponentiation operations are time consuming.
- Key length should be at least 1024 or 2048

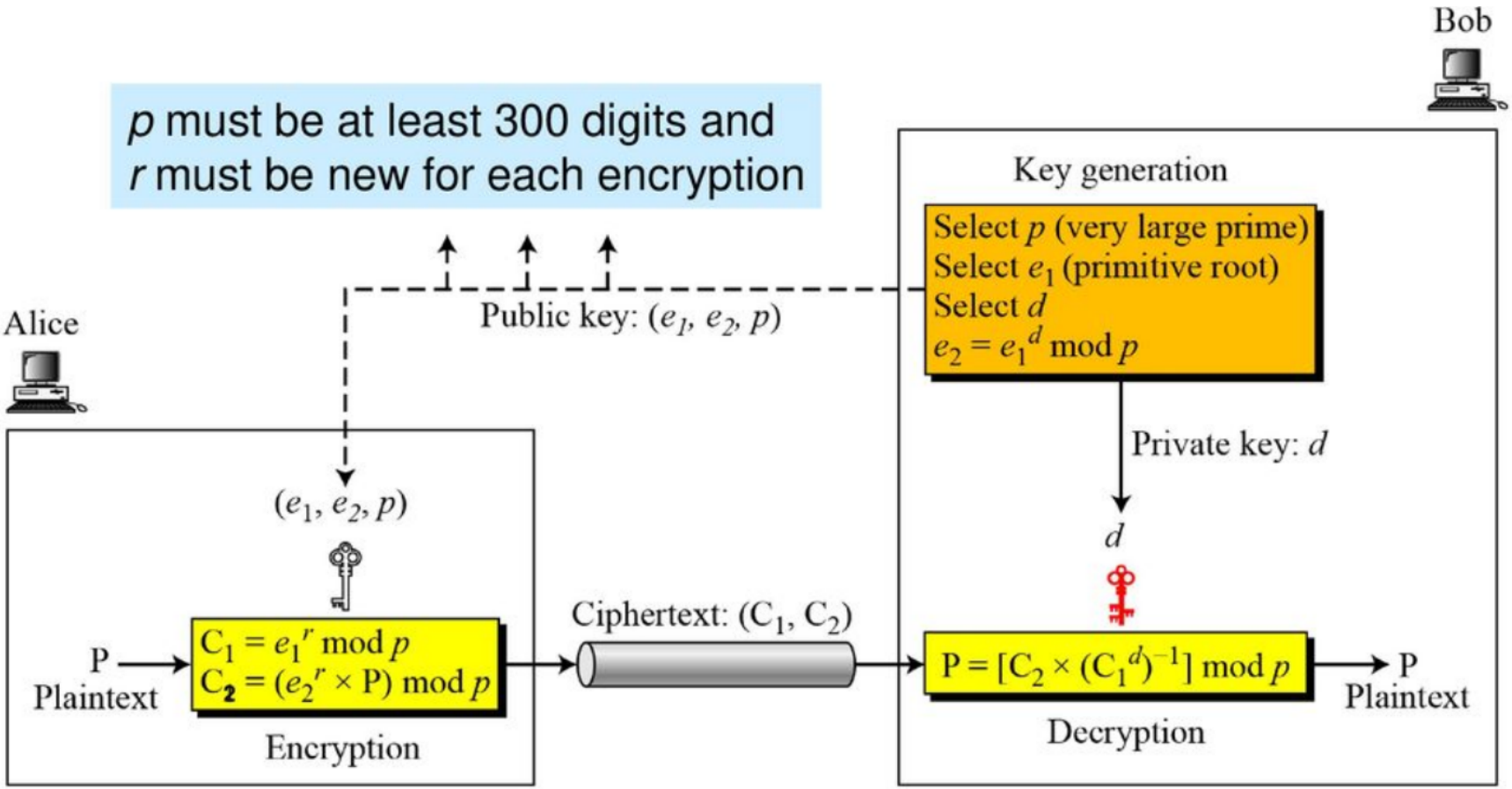
Elgamal Encryption

- Depends on the difficulty of computing discrete logarithms (Discrete logarithm problem)
- It was described by Taher Elgamal (Egyptian Cryptograper) in 1985.

Discrete logarithm problem

Given g a generator (primitive root) of finite field Z_p^* and g^a it is difficult to get back a.

Elgamal Encryption



Bob Public key: e_1, e_1, p

Bob Private key: d

Elgamal example

- Bob chooses p = 11 and $e_1 = 2$, and d = 3 then $e_2 = e_1^d = 8$
- Public key is (2, 8, 11) and the private key is 3

- **Encryption:** Alice chooses r = 4 and calculates C1 and C2 for the plaintext 7

Plaintext: (7)
C1 = e_1^r mod 11 = 16 mod 11 = 5 mod 11
C2 = $P \times e_2^r = 7 \times 4096$ mod 11 = 6 mod 11
Cipher: (5,6)

- **Decryption:** Bob receives the ciphertexts (5 and 6) and calculates the plaintext Plaintext:

$$C2 \times (C1^d)^{-1} \text{ mod } 11 = 6 \times ((5^3)^{-1}) \text{ mod } 11 = 6 \times 3 \text{ mod } 11 = 7$$

Elgamal properties

- It has the advantage that the same plaintext gives a different ciphertext (with near certainty) each time it is encrypted. If an attacker knows a plaintext-ciphertext pair, he cannot know the plaintext when it is encrypted several times.
- - The random number r should not be reused. Why?
  - Assume P_1 and P_2 are encrypted using the same r.
  - The ciphertext of message M_1 is $C_1 = (e_1^r, e_2^r P_1)$
  - The ciphertext of message P_2 is $C_2 = (e_1^r, e_2^r P_2)$
  - $C_1 \times C_2^{-1} = (1, P_1/P_2)$ If the attacker knows one message he can get the other

Elgamal Homomorphic Property

$$\begin{aligned} E(m_1).E(m_2) &= (e_1^{r_1}, e_2^{r_1} P_1) * (e_1^{r_2}, e_2^{r_2} P_2) \\ &= (e_1^{r_1+r_2}, P_1.P_2 e_2^{r_1+r_2}) \\ &= E(m_1.m_2) \end{aligned}$$

RSA is Partially homomorphic cryptosystem
(Allow multiplication on encrypted data)

Other Homomorphic Schemes

- Paillier Scheme: Allow homomorphic addition.

$$E(m_1).E(m_2) = E(m_1 + m_2)$$

$$E(m_1)^k = E(k.m_1)$$

- Fully Homomorphic Encryption:
  - Allow homomorphic addition and homomorphic.
  - Based on lattice based cryptography.
  - Crypto Nets [1] uses homomorphic encryption to do encrypted prediction for neural networks.

[1] https://arxiv.org/pdf/1412.6181.pdf

Comparison between Symmetric and Asymmetric key Cryptography

Advantages of Symmetric Key Cryptography

- Symmetric-key algorithms are generally fast, but Key management is a problem.
- However, asymmetric key algorithms are slow:
  - In practice, asymmetric key algorithms are typically hundreds to thousands times slower than symmetric key algorithms.

Drawbacks of Symmetric Key:

- Key Establishment problem
- Key management problem How?
- Non-repudation is hard to achieve. Why?

That does not mean symmetric key cryptography is useless. It is used with asymmetric key cryptography as will be explained later.

Comparison between Symmetric and Asymmetric key Cryptography

Advantages of Asymmetric Key Cryptography

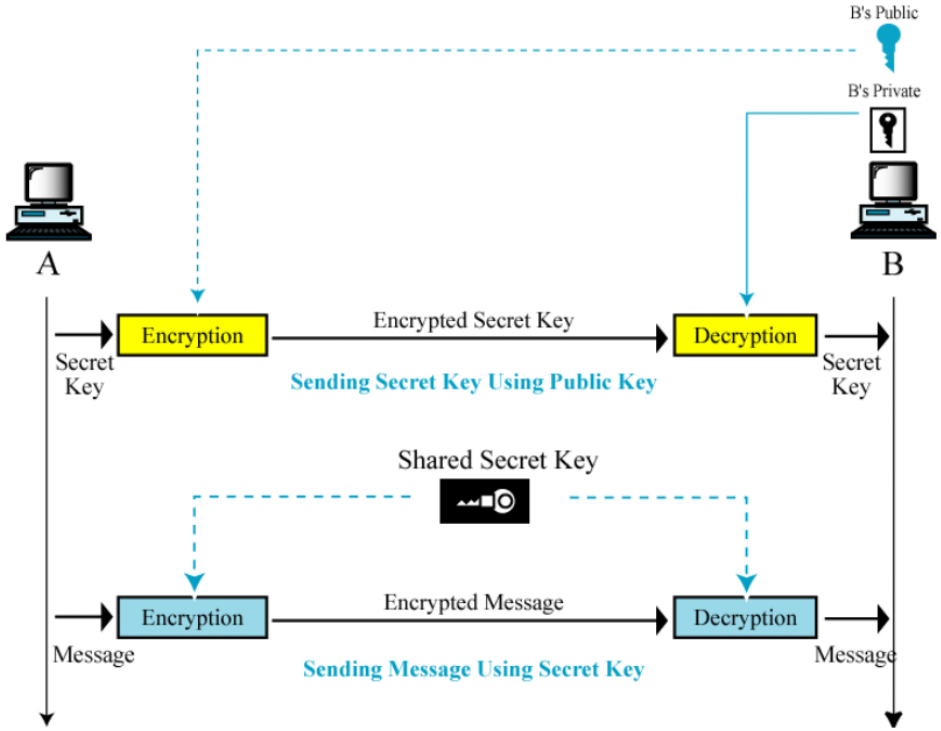
- No requirements for a secret channel for key transfer.
- Each user has only one key pair which simplifies the key management
- Facilaites the provision of non-repudation
- Allow some homomorphic operations on the encrypted data.

Drawbacks of Asymmetric Key:

- Asymmetric keys are typically larger than symmetric keys.
- Asymmetric Key schemes are slower than symmetric key counterparts.

Hybrid Schemes

Public key cryptography is used to share a symmetric key and symmetric key cryptography is used for data encryption because it is more efficient.



- DES (or AES) for encrypting actual data
- RSA for encrypting corresponding DES session key

Outline

Session Keys Establishment Protocols

Why Session Keys?

- Limit available ciphertext for cryptanalysis
- Limit exposure caused by the compromise of a session key
- Usually Dyanamic: To avoid long-term storage of a large number of secret keys (keys are created on-demand when actually required)

How to generate?

- A shared secret is derived by the parties as a function of information contributed by each, such that no party can predetermine the resulting value. Ex. Diffie Hellman protocol.
- One party cannot control the value of the established keys

What is a good protocol for sharing a session key

1. Symmetry of users' input
2. Session key freshness - assurance that key is new and not being reused
3. Efficiency of establishment and verification: total number of bits transmitted (i.e., bandwidth used), number of messages exchanged, complexity of computations by each party (speed), possibility of precomputations to reduce on-line computational complexity.
4. No requirement of third party.
5. Non-repudiation of key
6. Perfect Forward Secrecy disclosure of a session key does not disclose the session keys that were previously used.
7. Perfect Backward Secrecy: compromise of a session keys does not compromise session keys that will be used
8. Key confirmation: One party is assured that a second party actually possesses the session key. Possession of a key can be demonstrated by producing a one-way hash value of the key or encryption of known data with the key

Derive the session key from a long term key

- A and B exchange a long term Key Ksess that is then used to derive a unique key for each message passed during the day

$$K_1 = E_{K_{Sess}}[X]$$

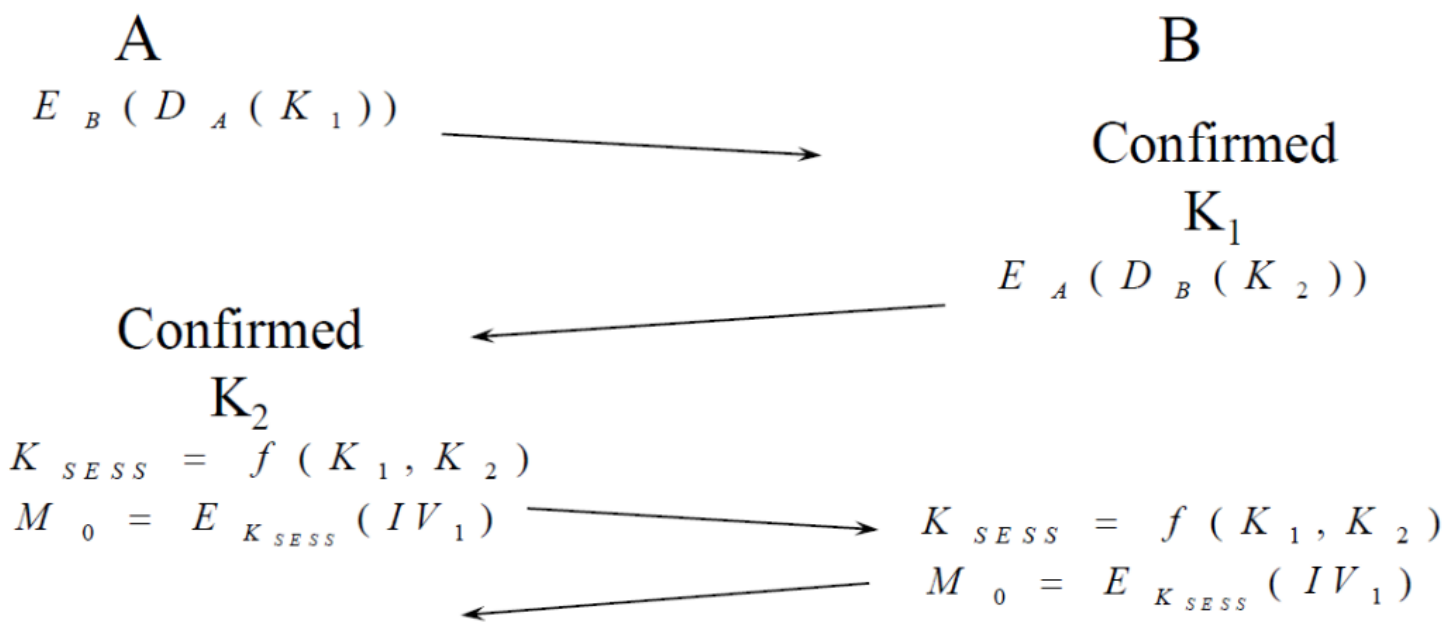
$$K_2 = E_{K_1}[X]$$

$$K_n = E_{K_{n-1}}[X]$$

If an attacker can recover any message key

- Attacker can recover all messages after the recovered key (complete forward break)
- If the encryption system is secure, then the attacker cannot break previous keys (break-backwards protection)

Key Confirmation Example



E: encryption by the public key

D: encryption with the private key

- Key Confirmation: achieved by encrypting a known value
- Mutual authentication
- Key is fresh if K1 or K2 has not been used before

Diffie-Hellman Key Exchange Protocol

Public System Parameters

p: is a large prime number
g: a generator of Z_p^*

Alice

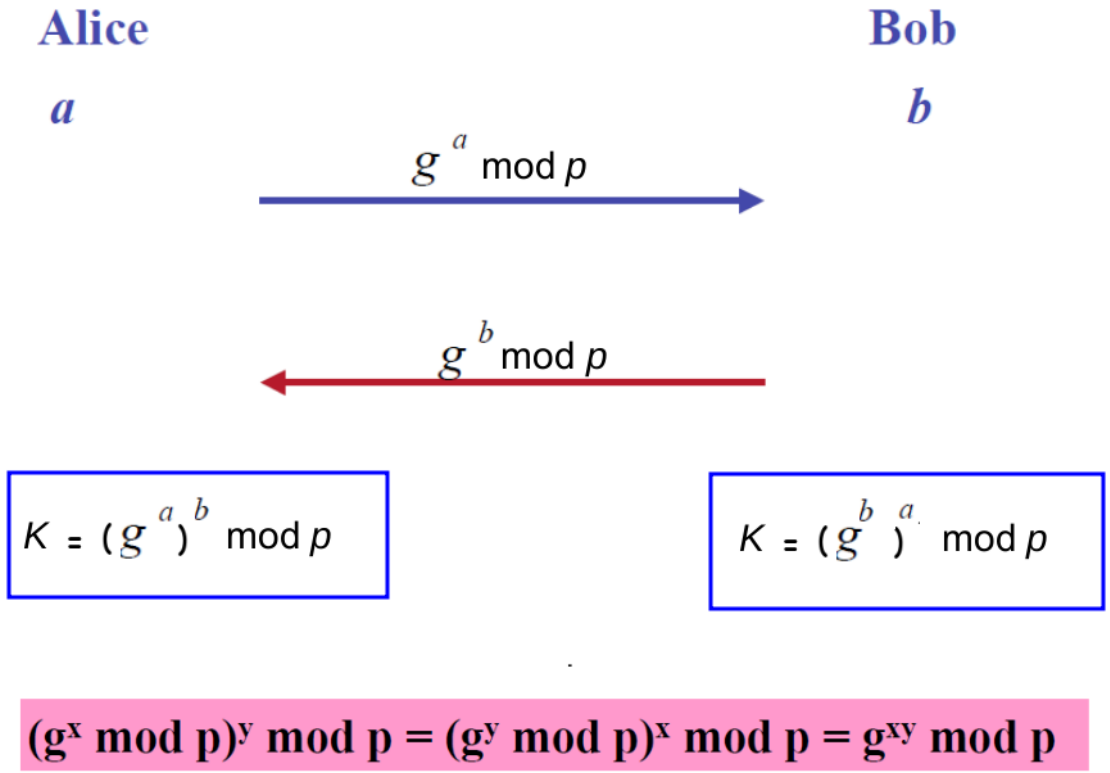
Private key: a, 0 < a < p and GCD(a, p-1) = 1
Public key: g^a

Bob

Private key: b, 0 < b < p and GCD(b, p-1) = 1
Public key: g^b

a and b are secret and should be large
p and g are public

Diffie-Hellman Key Exchange Protocol

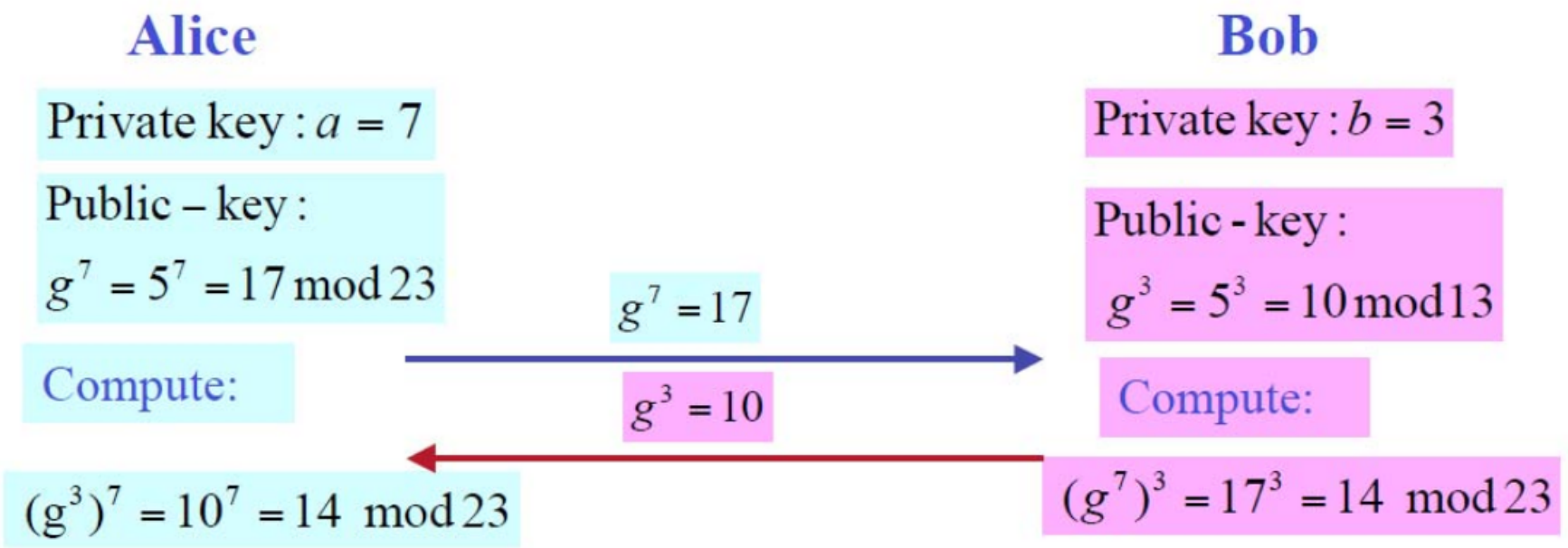


$$(g^x \bmod p)^y \bmod p = (g^y \bmod p)^x \bmod p = g^{xy} \bmod p$$

- K is the session symmetric key.
- a and b are not sent in clear because they are secret.
- Sending g^a and g^b is secure because it is not feasible to know a given g and g^a

Diffie-Hellman Example

Example Let $p = 23$. Then $g = 5$ is a primitive element of GF(p).



The secret information shared by Alice and Bob is 14.

Attacker: known $\left.\begin{matrix} g^7 = 17 \\ g^3 = 10 \end{matrix}\right\} \Rightarrow g^{21} = 14?$

Diffie-Hellman Summary

Diffe-Hellman Problem:

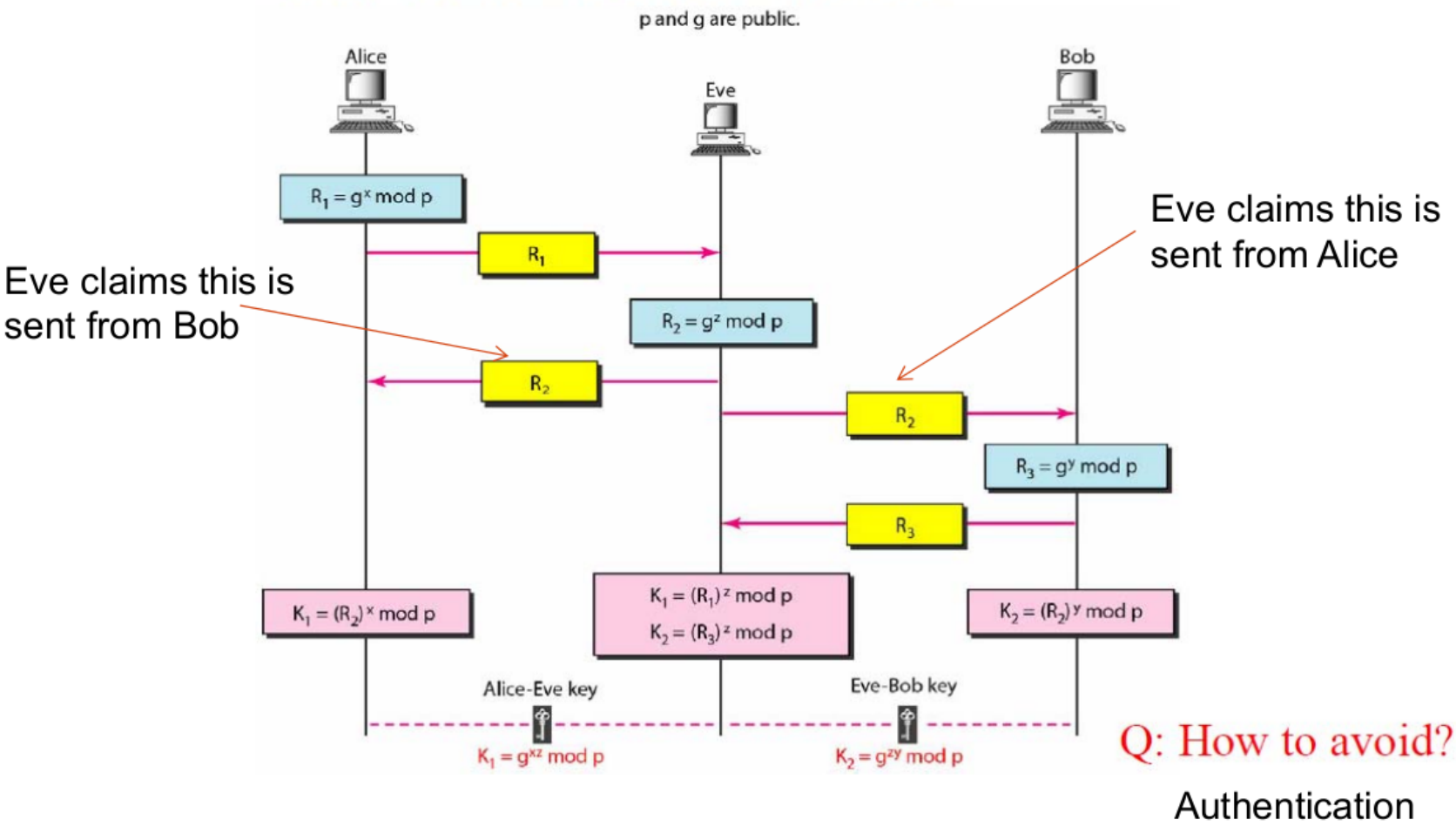
Given g^a and g^b, compute g^{ab}.

Thus the Diffe-Hellman key exchange scheme is secure if the DH problem is computationally infeasible.

The DH problem is computational feasible if the solving discrete logarithm in $GF(p)$ is computationally feasible.

Thus, we may say that the security of the DH key exchange scheme is based on the difficulty of solving discrete logarithm in the finite field GF(p).

Man-In-Middle Attack



Alice thinks that she exchanged a key with Bob

Eve can act as Alice and Bob

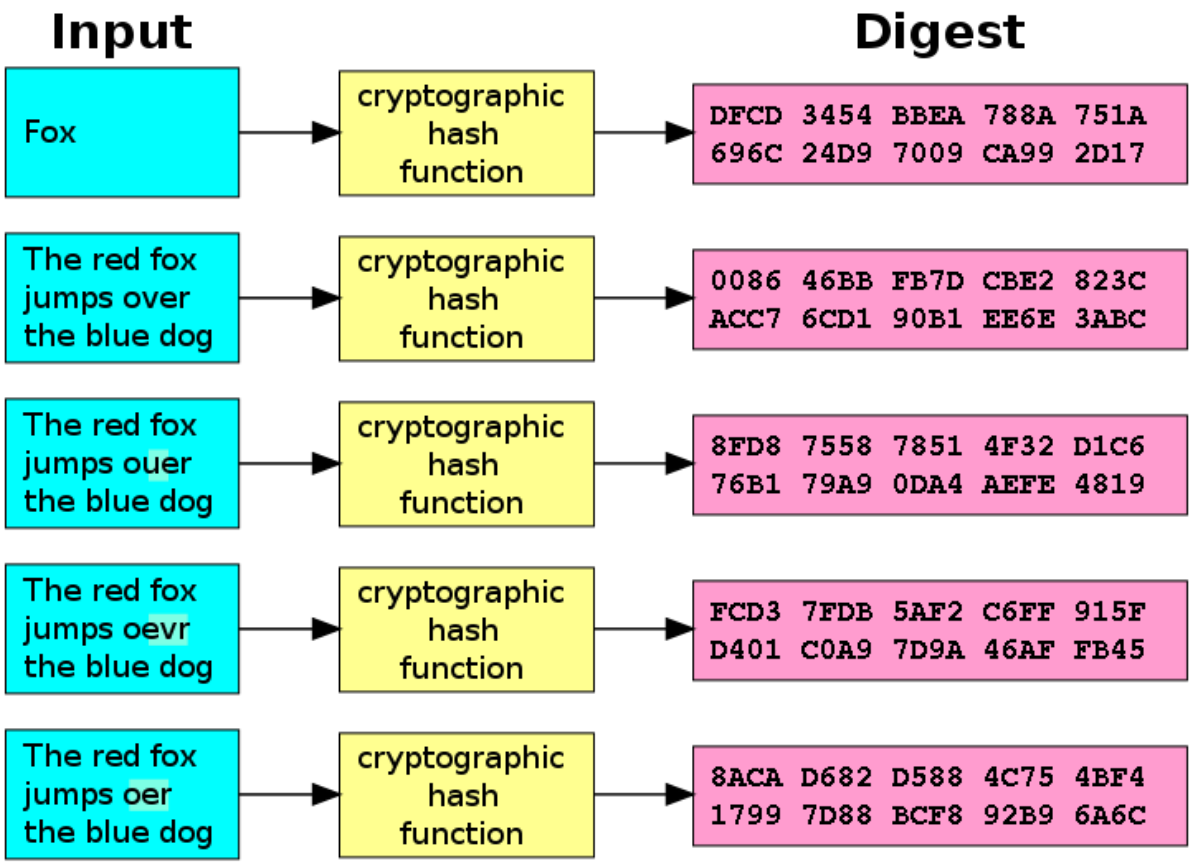
Summary Applications for Asymmetric Key Cryptography

- **Encryption/Decryption:** provides confidentiality
  The sender encrypts a message with the receivers public key.
  The receiver decrypts the ciphertext with his/her private key.
- **Digital signature:** provides authentication
  The sender signs a message with its private key.
  The receiver decrypts the ciphertext with the senders public key
- **Key exchange** Two parties cooperate to exchange a session key using their public keys (Diffie Hellman)

Outline

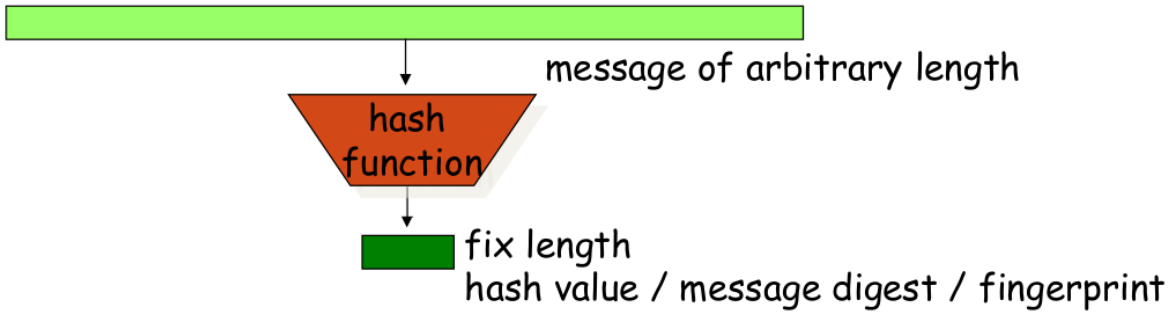
Cryptographic hash function

A Cryptographic hash function is a hash function which takes an input (or 'message') of any size and returns a fixed-size string of bytes.



The output is called 'hash value', 'message digest', 'digital fingerprint', 'digest' or 'checksum'.

Cryptographic Hash Functions



What are the applications?

- Message integrity checks
- Digital signatures
- Authentication.
- Storing Password.
- Core of the Blockchain technology.

Hash Function Requirements

1. Ease of computation: given m, it is easy to compute H(m).
2. H() is a public function any one can calculate H(m) from m because a key is not needed.
3. Efficient: does not need too much energy or computations, i.e., computational time is too short.
4. One-way property (preimage resistance): given a hash value Y= H(m), it is computationally infeasible to find m given Y.
5. Randomness: Avalanche Effect

```
Message:          "A hungry brown fox jumped over a lazy dog"
SHA1 hash code:   a8e7038cf5042232ce4a2f582640f2aa5caf12d2

Message:          "A hungry  brown fox jumped over a lazy dog"
SHA1 hash code:   d617ba80a8bc883c1c3870af12a516c4a30f8fda
```

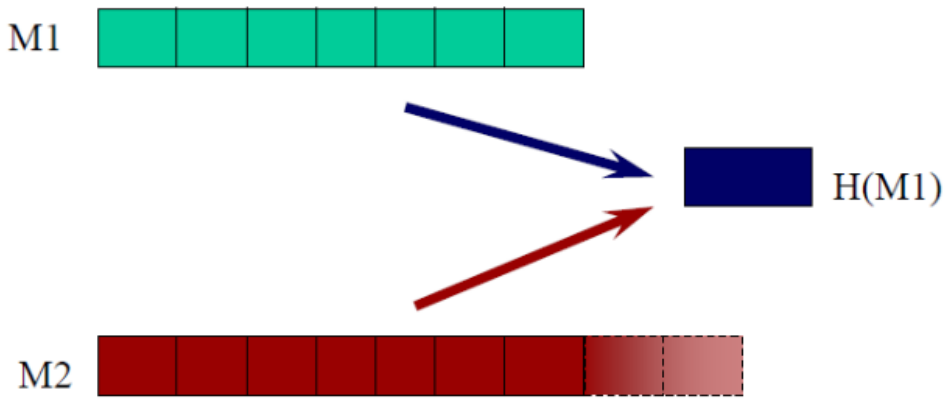
The only difference between the two messages is the extra space

Hash Collision

Collision

Collision means two different messages generate the same hash value.

- Since we have Many-to-one mapping collisions are unavoidable however, finding collisions are difficult
- The hash value of a message can serve as a compact representative image of the message (similar to fingerprints).



It should not be possible to deterministically find M2 (of any length) that creates the same Hash value - Collision

Hash Collision Resistance

Weak collision resistance (2nd preimage resistance)

Given an input m1, it is computationally infeasible to find a second input m2 such that h(m2) = h(m1) m2 is called the preimage of m1.

Strong collision resistance (collision resistance)

It is computationally infeasible to find any two distinct inputs m1 and m2 such that h(m1) = h(m2).

Bithday Paradox

- One of the classical paradoxes that is used to determine the number of hashes needed to get a collision.

Question: What are the chances that at least two people share a birthday in a group of 23?

We can start with the chance that every possible pair has different birthday.

- With 23 people we have 253 pairs:
$$C(23, 2) = \frac{23 \cdot 22}{2} = 253$$
- The chance of 2 people having different birthdays is:
$$1 - \frac{1}{365} = \frac{364}{365} = .997260$$
- The chance all pairs are different:
$$\frac{364}{365}^{253} = 0.4995$$
- The chance of at least two pairs are the same:
$$1 - \frac{364}{365} = 0.5005$$

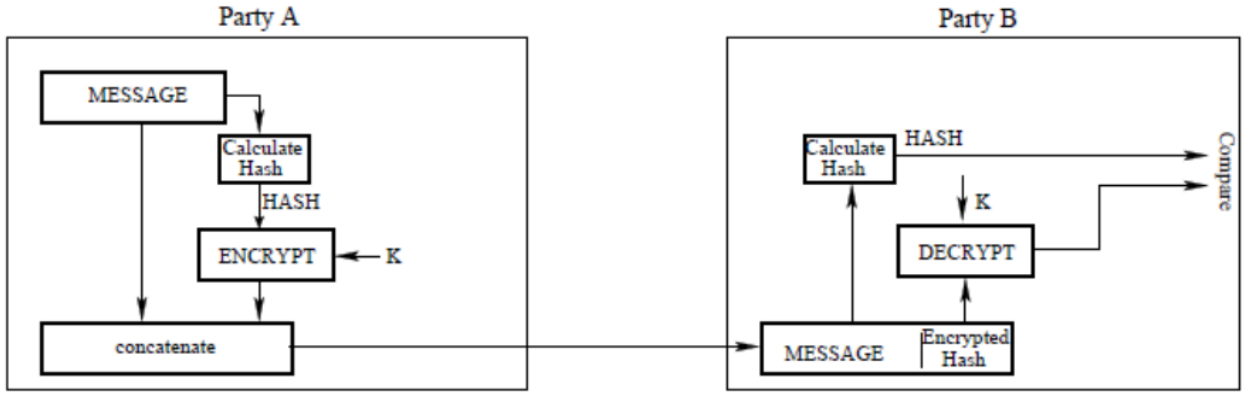
Birthday Attack

- $\sqrt{n}$ is roughly the number you need to have a 0.5% chance of a match with your space size is n.
- It gets more accurate with large n
  - $\sqrt{365}$ is about 20. This comes into play in cryptography for the birthday attack
- For n bits hash, $\sqrt{2^n} = 2^{\frac{n}{2}}$ randomly chosen messages, with high probability, there will be a collision pair.
- If n = 128 bits, expected number of steps = 2^64 (barely feasible)
- Finding collisions a preimage needs 2^n (infeasible for large n)

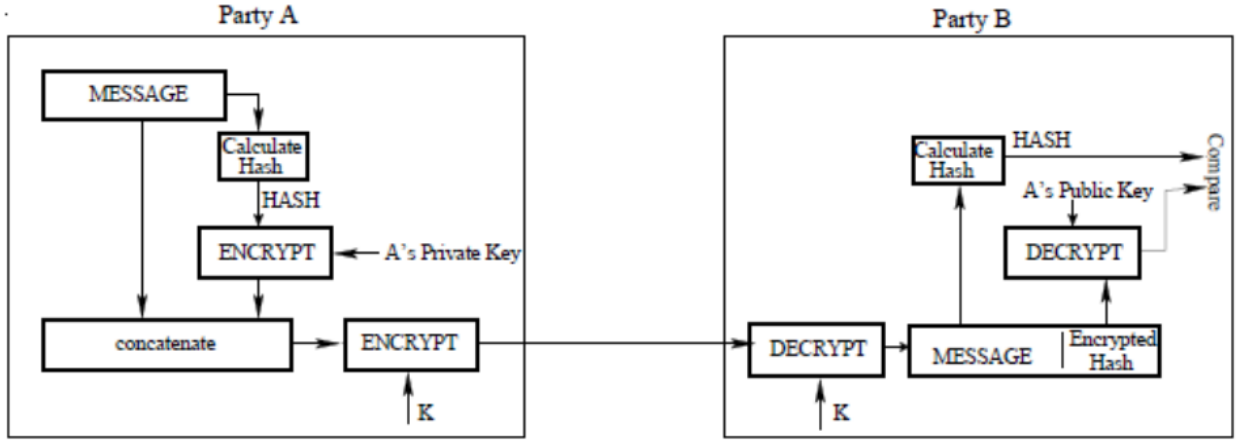
Digital Signature and Hash Functions

- **Signatures:** Signing the message digest rather than the message often improves the efficiency because the message digest is usually much smaller in size than the message.
  - Any change to the message in transit will result in a different message digest, and the signature will fail to verify.
  - Suppose that Alice can find two messages x1 and x2 , with $x1 \neq x2$ and $H(x1) = H(x2)$. Alice can sign x1 and later claim to have signed x2.

Some Applications of Hash Functions



Origin authentication, non repudation and data integrity.



Confidentiality, Origin authentication, non repudation, and data integrity.

Two very common hash functions

MD5 (Message Digest Algorithm):

- MD5 produces as output a 128-bit "fingerprint" or "message digest" of the input.
- MD5 is much faster than SHA.
- Wang and Yu (2004) found collisions for (full) MD5 in 2^{39} steps (about 15 minutes).
- MD5 should not be used if collision resistance is required, but is probably okay as a one-way hash function.
- MD5 is still used today

SHA (Secure Hash Algorithm):

- SHA-1 produces a 160-bit output called a message digest.
- Collisions found for SHA1 in 2005
- In 2001, NSA (National Security Agency) proposed variable output-length versions of SHA-1. Output lengths are 256 bits (SHA-256), 384 bits (SHA-384) and 512 bits (SHA-512).
- SHA-256 used in blockchain.

How hash are designed?

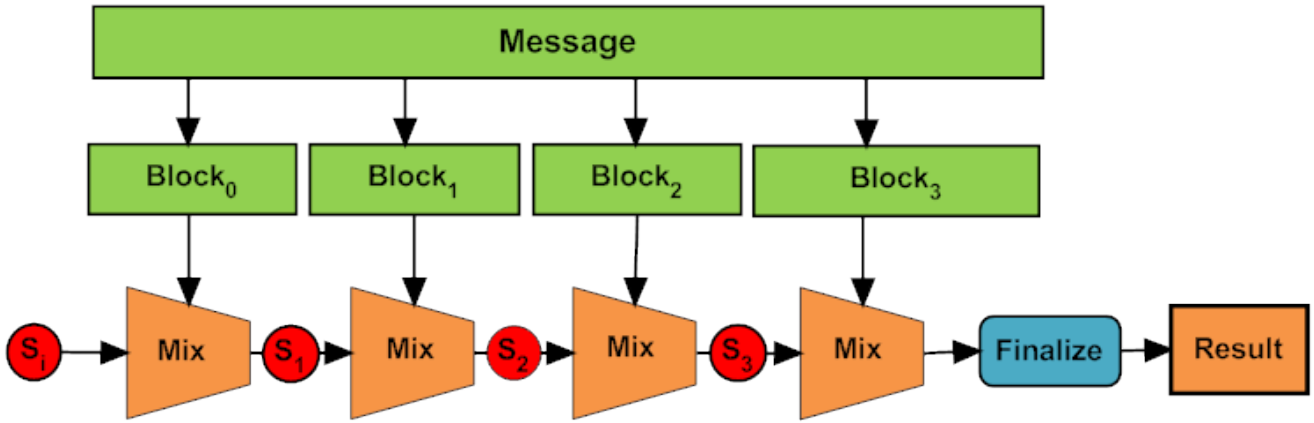
The following is a list of the commonly used operations:

- **Bitwise Operations:** Not (!), Or (|), And (&), Xor ($\wedge$), Shift-Left/Right (<<, >>), Rotate-Left/Right (<<<, >>>)
- **Mathematical Operations:** Addition (+), Multiplication (*)
- **Lookup Tables:** List of prime numbers, List of magic numbers, S-Box, P-Box

A common method for constructing collision resistant cryptographic hash functions is known as the MerkleDamgard construction.

How hash are designed?

MerkleDamgard construction



1. Initialise an internal state
2. Consume the message N-bits at a time (aka block size typically 32-bits, 64-bits, 128-bits etc..)
3. Perform a mixing operation with the current block and the internal state
4. Update the internal state with the the result of [3]
5. If there are any remaining bytes in the message proceed to [2]
6. Finalize the internal state and return the hash value

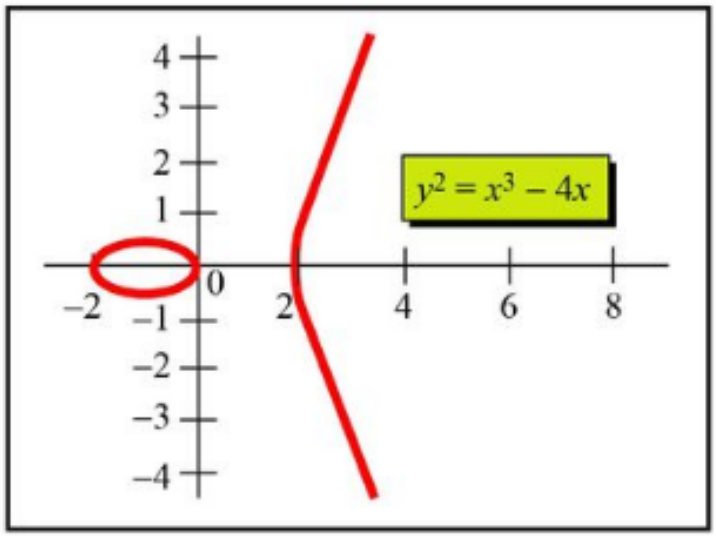
Outline

Elliptic Curve Cryptography (ECC)

- ECC gives the same level of security with smaller key sizes than RSA or Elgamal
- General elliptic curve equation

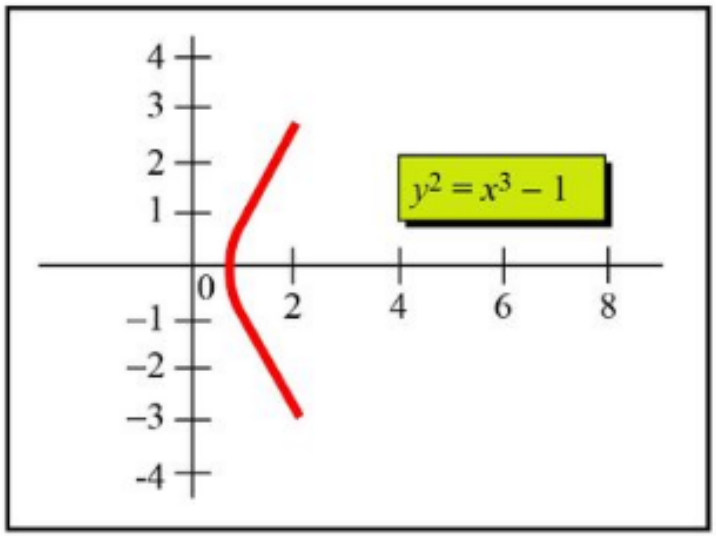
$$y^2 + b_1xy + b_2y = x^3 + a_1x^2 + a_2x + a3$$

- Example of Elliptic Curves



a. Three real roots

$y^2 = x^3 - 4x$

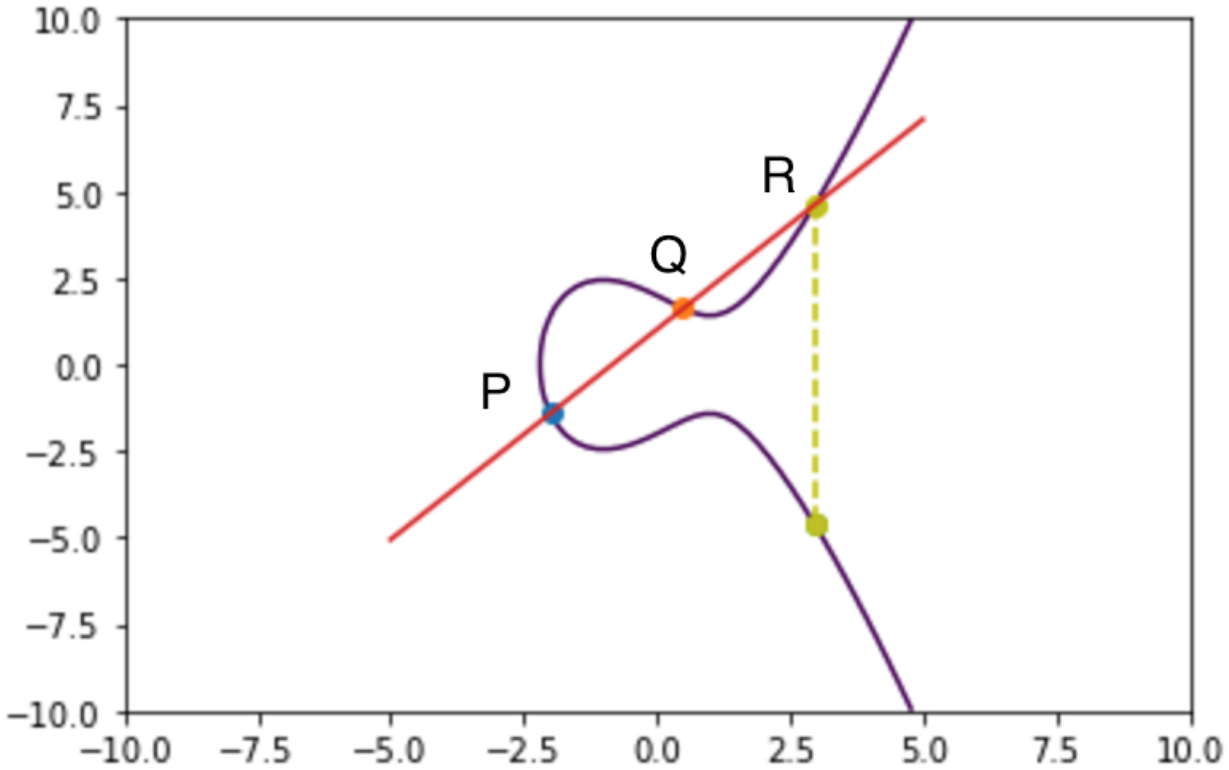


b. One real and two imaginary roots

$y^2 = x^3 - 1$

Elliptic Curve Properties

- Symmetry over the y-axis
- if a line intersects two points in the curve it will always intersect a third.



Closure Property

Elliptic Curve Cryptosystem

Cryptographic Elliptic curves

- y^2 mod p = $(x^3 + ax + b)$ mod p
- p is prime
- a and b: non negative integers less than p and satisfy $(4a^3+27b^2)$ mod p $\neq 0$

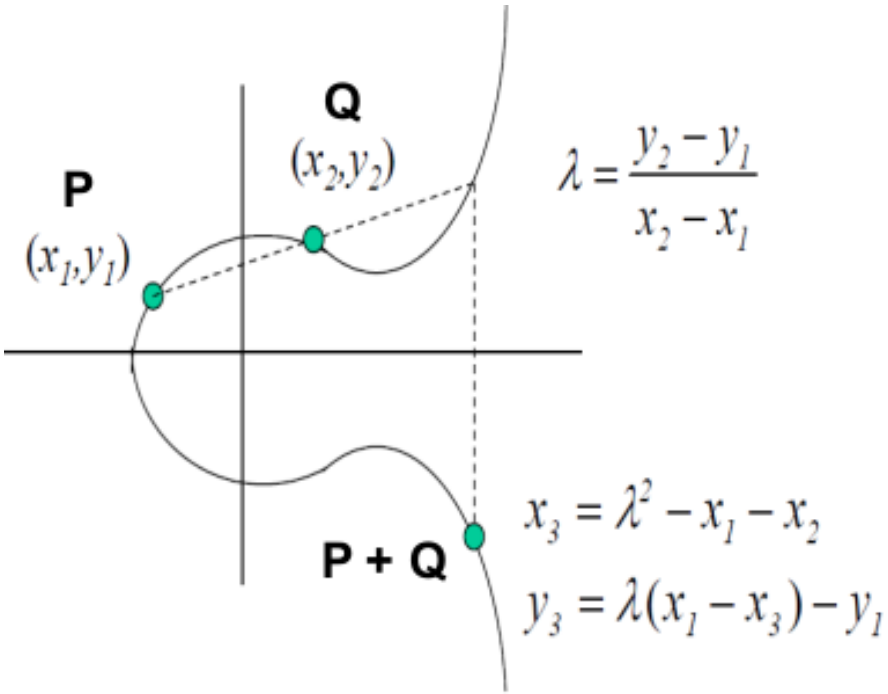
Elliptic Curve Cryptosystem

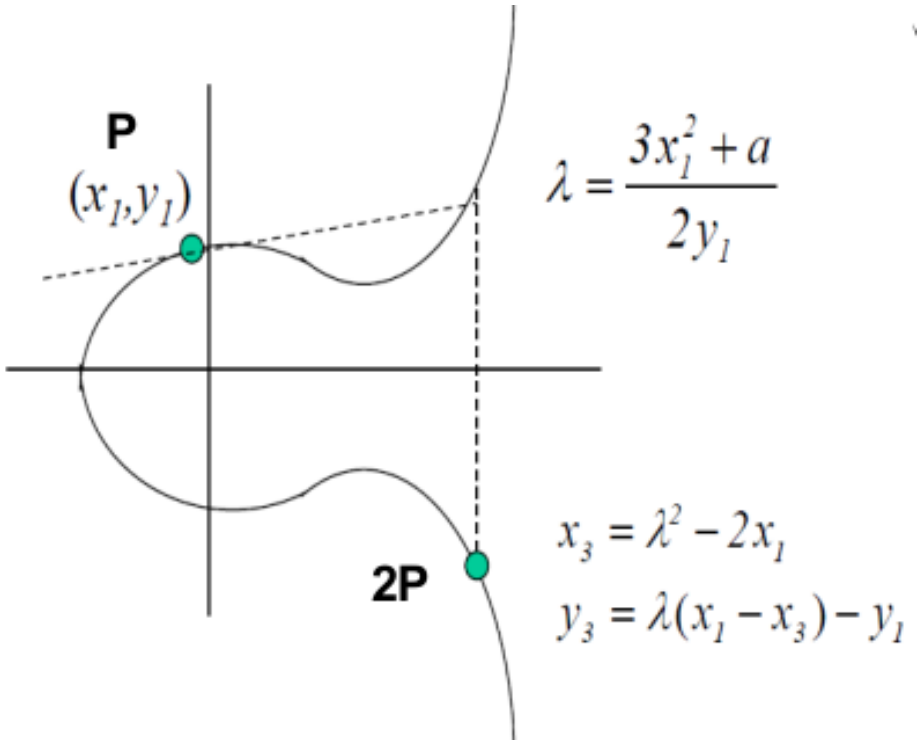
- If p is a large prime number
- Z_p is a finite field modulo q
- Then an elliptic curve $E_p(a, b)$ over Z_p is the set of points (x, y) with x, y $\in Z_p$ satisfy the following equation together with special point O, called as point at infinity.

$$y^2 \bmod p = (x^3 + ax + b) \bmod p$$

where x, y, a, b $\in Z_q$

Elliptic Curve Cryptosystem





P+O = O+P = P

P + (-P) = O

- ECC addition is analog of modulo multiply
- ECC repeated addition is analog of modulo exponentiation

Elliptic Curve Cryptosystem

Elliptic Curve Laws

For $P = (x_1, y_1) \in E_p$ and $Q = (x_2, y_2) \in E_p$ Then

1. $-P = (x_1, -y_1)$
2. $P + Q = R = (x_3, y_3)$ where the coordinates of the point R are defined by

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y1$$

where

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases}$$

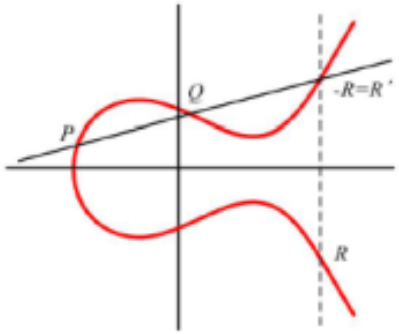


Figure: Point Addition

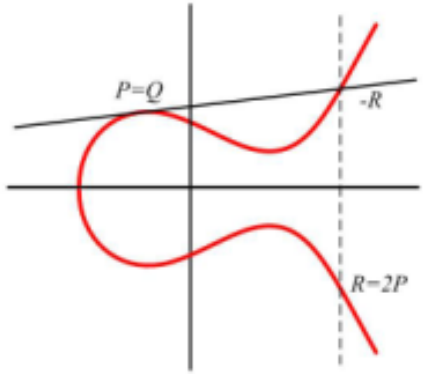


Figure: Point Doubling

How group points of Elliptic Curve can be computed?

Consider an elliptic curve

$$E : y^2 = x^3 + x + 6 \text{ over } GF(11)$$

To find all points (x, y) of E for each $x \in GF(11)$, compute $z = x^3 + x + 6 \mod 11$ and determine whether z is a **quadratic residue** (QR).
If so, solve $y^2 = z$ in $GF(11)$. We can find there are totally **12** points on this curve.

x	$x^3 + x + 6$	QR?	y
0	6	no	–
1	8	no	–
2	5	yes	4, 7
3	3	yes	5, 6
4	8	no	–
5	4	yes	2, 9
6	8	no	–
7	4	yes	2, 9
8	9	yes	3, 8
9	7	no	–
10	4	yes	2, 9

An integer q is called a quadratic residue modulo n if there exists an integer y such that:

$$y^2 \equiv q \pmod{n}.$$

How group points of Elliptic Curve can be computed?

Example (continued)

Let $P = (2, 7)$ is generator. **We know this**

We can compute $2P = (x_2, y_2)$ as follows.

$$\begin{aligned}
\lambda &= \frac{3x_1^2 + a}{2y_1} = \frac{3 \cdot 2^2 + 1}{2 \cdot 7} = \frac{13}{14} = 2 \cdot 3^{-1} = 2 \cdot 4 = 8 \pmod{11} \\
x_2 &= \lambda^2 - 2x_1 = 8^2 - 2 \cdot 2 = 5 \pmod{11} \\
y_2 &= (x_1 - x_2)\lambda - y_1 = (2 - 5) \cdot 8 - 7 = 2 \pmod{11}
\end{aligned}$$

Therefore, we obtain $2P = (5, 2)$.

How group points of Elliptic Curve can be computed?

Example (continued)

Let $3P = P + 2P = (x_3, y_3)$. Then we can compute $3P$ as follows.

$$\begin{aligned} \lambda &= \frac{y_2 - y_1}{x_2 - x_1} = \frac{2-7}{5-2} = 2 \pmod{11} \\ x_3 &= \lambda^2 - x_1 - x_2 = 2^2 - 2 - 5 = 8 \pmod{11} \\ y_3 &= (x_1 - x_3)\lambda - y_1 = (2-8)\cdot 2 - 7 = 3 \pmod{11} \end{aligned}$$

Hence, we obtain $3P = (8, 3)$.

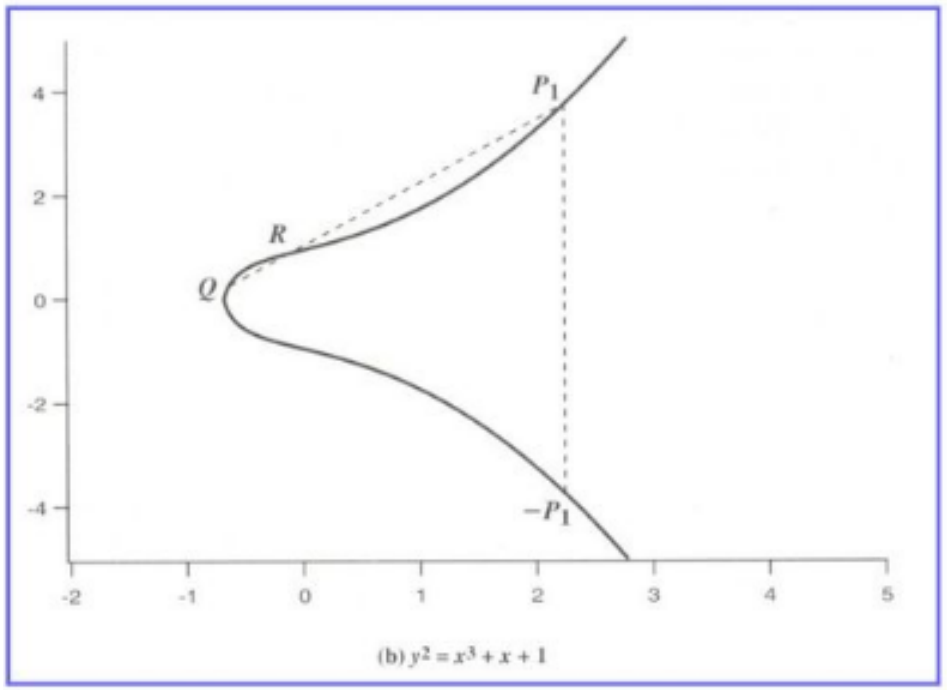
Similarly, we can also compute the cyclic group generated by P.

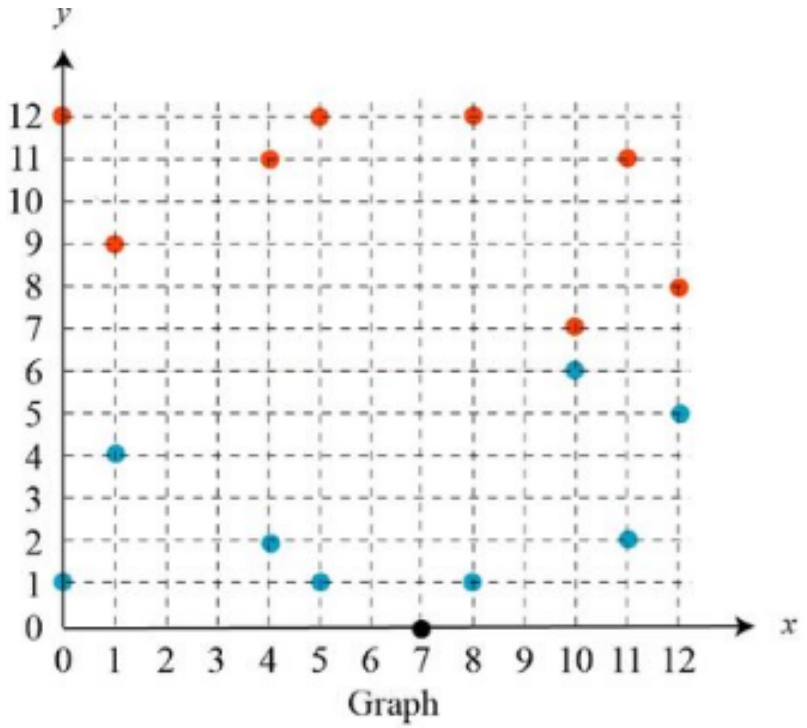
$$\begin{array}{cccc} P = (2,7) & 2P = (5,2) & 3P = (8,3) & 4P = (10,2) \\ 5P = (3,6) & 6P = (7,9) & 7P = (7,2) & 8P = (3,5) \\ 9P = (10,9) & 10P = (8,8) & 11P = (5,9) & 12P = (2,4) \end{array}$$

$$13P = P + 12P = 2P + 11P = 3P + 10P = \cdots = \mathcal{O}$$

Elliptic Curve Summary

- The equation of the elliptic curve is modulo prime p
- x, y, a, b $\in Z_p$
- The points (x,y) on the Elliptic curve form an additive group with an identity O (point at infinity) i.e., (x,y) $\in E_p(a, b)$





Elliptic Curve cryptography (ECC) Security

- Given P and aP, there is no way to compute a (Elliptic curve discrete logarithm problem), but given P and a, it is easy to compute aP.
- This is more difficult than factorization → can use much smaller key sizes than with RSA for the same security level.
- Shorter Key means less operations for encryption
- With similar level of security, ECC offers significant computational reduction

Elliptic Curve Encryption/Decryption (Elgamal analogy)

1. Find an Elliptic Curve $E_p(a, b)$ and a generator P
2. Alice and Bob share $E_p(a, b)$ and P (i.e., public parameters)
3. Bob select $n_B \in Zp$ as Private Key and compute $P_B = n_B P$ as Public Key

Encryption

Alice selects a random k and Encrypt a message P_m as follows

$$C_m = (kP, P_m + kP_B)$$

Decryption

Bob use his private key n_B to decrypt back P_m as the following

$$P_m + kP_B - n_B kP = P_m + kn_B P - n_B kP = P_m$$

Note

To recover the message, the attacker know P and kP, recovering k can break the encryption (ECDLP).

Key Exchange using ECC (Diffie Hellman Analog)

- Find an Elliptic Curve $E_p(a, b)$ and a generator P

Key Exchange between A and B

- Bob selects $n_B \in Zp$ as Private Key and compute $P_B = n_B P$ as Public Key
- Alice selects $n_A \in Zp$ as Private Key and compute $P_A = n_A P$ as Public Key
- Alice generates a session key as $K_{AB} = n_A \times P_B = n_A \times n_B \times P$
- Bob generates a session key as $K_{BA} = n_B \times P_A = n_B \times n_A \times P = K_{AB}$

Comparable Key Sizes for Equivalent Security

Symmetric scheme (or security level) (key size in bits)	ECC-based scheme (size of *n* in bits)	RSA/DSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

- As with elliptic-curve cryptography in general, the bit size of the public key believed to be needed for ECDSA is about twice the size of the security level in bits.
- A security level of 80 bits means that an attacker requires the equivalent of about 2^{80} operations to find the private key

Outline

Bilinear Pairing

- One way function but with some "Homomorphic Properties"
- Built on the elliptic curve crypto system.
- Establish relationship between cryptographic groups.

Defination

A bilinear map takes an input from group G_1 with generator g_1 and an input from group G_2 with generator g_2 and maps the result to G_t

$$e : G_1 \times G_2 \rightarrow G_t$$

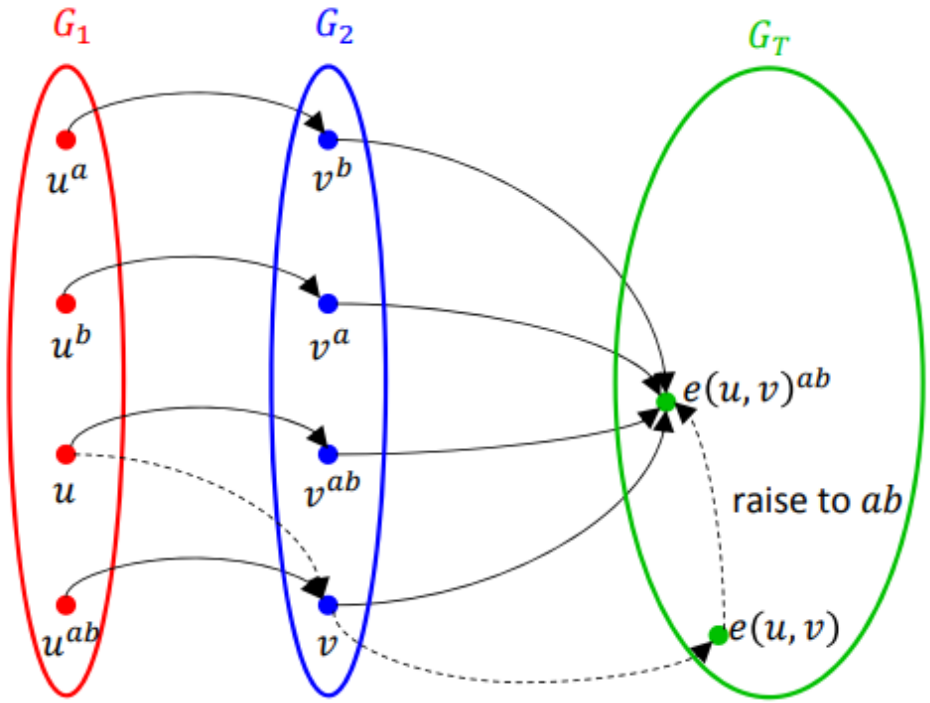
Bilinear maps are called pairings because they associate pairs of elements from G_1 and G_2 with elements in G_t.

Note: G_1 and G_2 are additive groups while G_t is multiplicative group

Bilinear Pairing Properties

$$e(u^a, v^b) = e(u^b, v^a) = e(u^{ab}, v) = e(u, v^{ab}) = e(u, v)^{ab}$$

$\forall\ u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}$



Other Notations

- Sometimes G_1 and G_2 is written additively
  - In this case P, Q normal names for elements of G_1 and $G2$
  - Bilinear property expressed as $P \in G_1$, $Q \in G_2$, and $\forall$ a, b $\in$ Z,

    $$e(aP, bQ) = e(bP, aQ) = e(P, abQ) = e(abP, Q) = abe(P,Q)$$
- Sometimes G_1 and G_2 are considered to be the same group G. In this case, the pairing is assumed to be symmetric.
- In litreature usually this notation of both G_1 and G_2 are written multiplicatively is used.
- Existing algorithms to compute bilinear pairing Weil and Tate Pairing Algorithm. Very Complex !

Most Common New Problems

Some new problems have been defined and assumed hard in the new bilinear context. (Out of the Course Scope)

Bilinear Diffie-Hellman Given g, g^a, g^b, g^c, compute $e(g,g)^{abc}$ (something like a "three-way" CDH but across the two groups)

Decisional Bilinear Diffie-Hellman Distinguish $g, g^a, g^b, g^c, e(g,g)^{abc}$ from $g, g^a, g^b, g^c, e(g,g)^z$

k-Bilinear Diffie-Hellman Inversion Given $g, g^y, g^{y^2}, \ldots g^{y^k}$, compute $e(g,g)^{\frac{1}{y}}$

k-Decisional Bilinear Diffie-Hellman Inversion Distinguish $g, g^y, g^{y^2}, \ldots g^{y^k}, e(g,g)^{\frac{1}{y}}$ from $g, g^y, g^{y^2}, \ldots g^{y^k}, e(g,g)^z$

Application of Bilinear Pairing

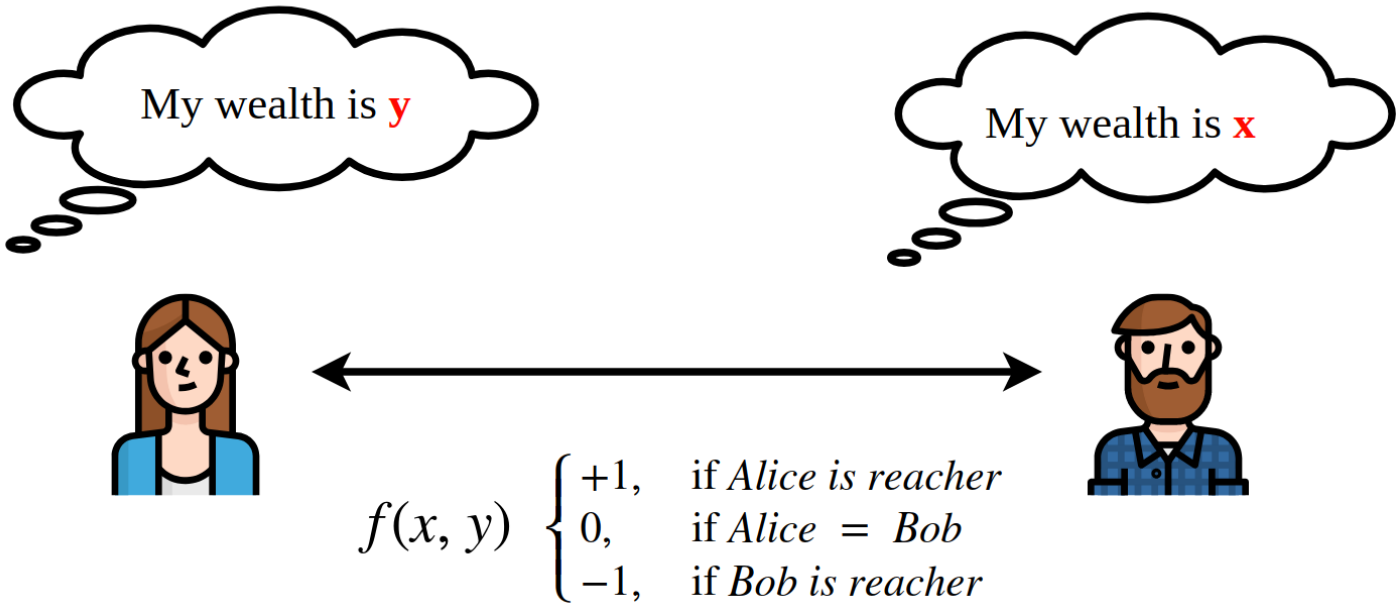
- Secure dot product. Given two encrypted vectors **a** and **b** calculate the encryption of their dot product. On Board
- One level Homomorphic Addition and Multiplication on encrypted data.
- Dot product is the basic operation for evaluating a deep learning model.
- Bilinear pairing is used for privacy preserving inference on encrypted data here[2]. Application on MNIST dataset with 97.5% accuracy.

[2] https://eprint.iacr.org/2018/206.pdf

Outline

Secure Multi-Party Computation

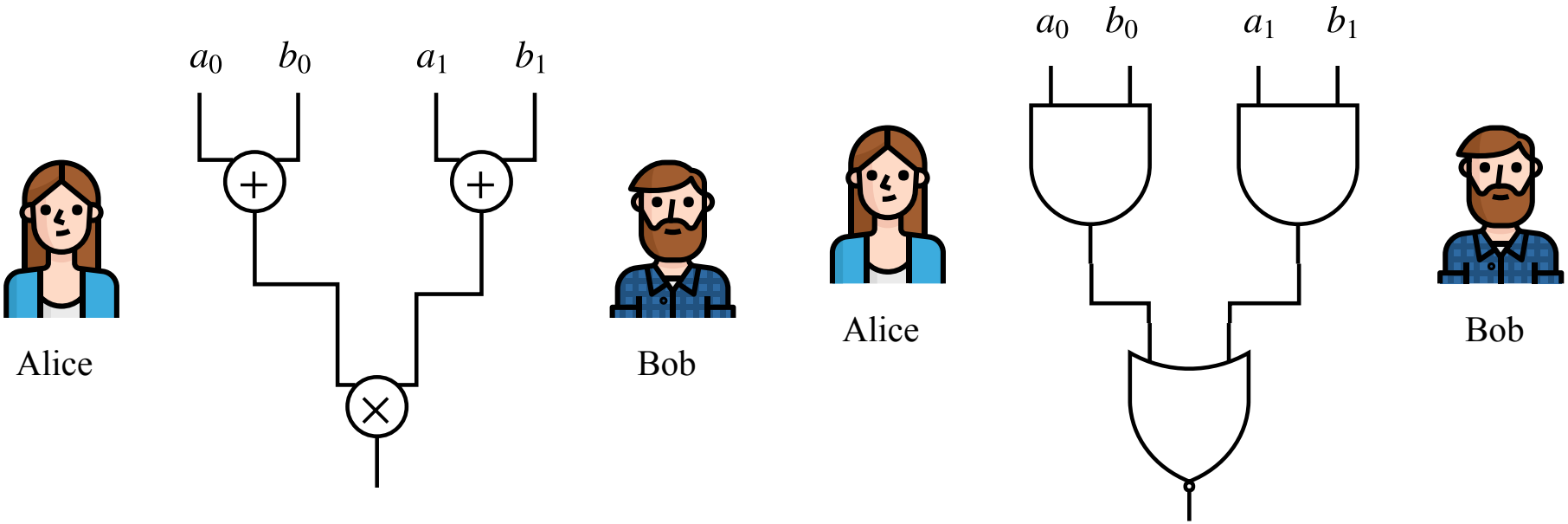
It is a subfield of cryptography with the goal of creating methods for parties to jointly compute a function over their inputs while keeping those inputs private.



Note that:

Yao's Millionaires' problem 1982. Can be solved with Arithematic or Garbeled Circuits.

Yao's Millionaires' Problem Solution

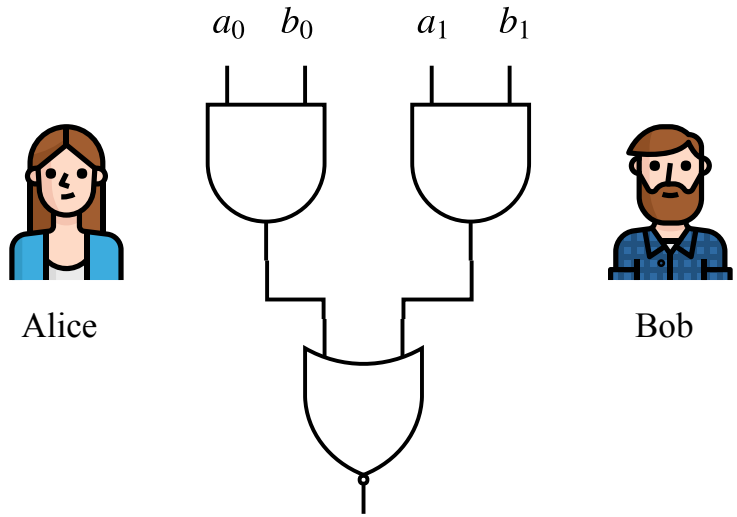


1. Binary (Garbeled) Circuits Protocol.
2. Arithematic Circuits Protocol.

Garbeled Circuits

Alice and Bob wants to evaluate a circuit on their private inputs wihout revealing them. e.g., who is richer?

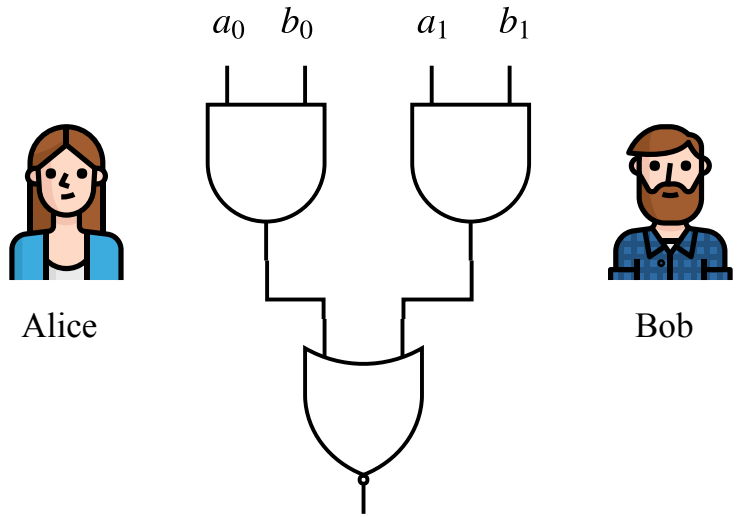
1. Assume a_0, a_1 are the inputs for alice while b_0, b_1 are the inputs for Bob.
2. Alice will be called the Generator.
3. Bob will be called the Evaluator.
4. By executing the following protocole, both alice and bob will evalute the circuit in a secure manner.



Garbeled Circuits

Alice and Bob wants to evaluate a circuit on their private inputs wihout revealing them. e.g., who is richer?

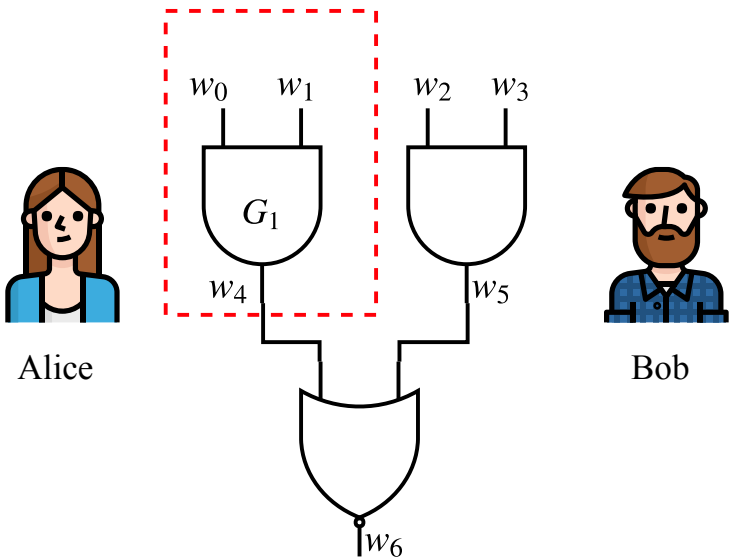
1. Assume a_0, a_1 are the inputs for alice while b_0, b_1 are the inputs for Bob.
2. Alice will be called the Generator.
3. Bob will be called the Evaluator.
4. By executing the following protocole, both alice and bob will evalute the circuit in a secure manner.



Garbeled Circuits

Lets consider the secure evaluation for only one gate G_1. Alice generate a garbeled table for G_1 as follows.

1. All wires are assigned lables that corresponds to random nounces.
2. w_0, w_1, w_4 are the labels for G_1.
3. Each wire label can have two values. e.g., w_0^0, w_0^1.



Garbeled Circuits

Alice generate garbeled table for G_1 as follows

w_0	w_1	w_4
0	0	0
0	1	0
1	0	0
1	1	1

$\Rightarrow$

Garbeled Table
$Enc_{w_0^0, w_1^0}(pad \| w_4^0)$
$Enc_{w_0^0, w_1^1}(pad \| w_4^0)$
$Enc_{w_0^1, w_1^0}(pad \| w_4^0)$
$Enc_{w_0^1, w_1^1}(pad \| w_4^1)$

.. } Permuted

Table: AND gate logic table.

Note that:

The garbeled table contains encrypted values, meaningless to anyone except Alice.

Garbeled Circuits

- Evaluator Bob will get the garbeled table for G_1 and the input for Alice at wire w_0 (assume it is w_0^0). Note that w_0^0 is meaningless to Bob.
- Assume Bob input is w_1^1, Bob needs to get this value without letting Alice know it. How?.

Using Oblivious Transfer

Oblivious Transfer [3]

Alice Generator (e,d)

Choose two random values x_0, x_1

Send x_0, x_1 →

Bob Evaluator

Bob should choose x_1

Calculate $v = (x_1 + r^e) mod\ n$

← Send v

$k_0 = (v - x_0)^d mod\ n = garbage$

$k_1 = (v - x_1)^d mod\ n = r$

$\hat{w}_1^0 = w_1^0 + k_0$

$\hat{w}_1^1 = w_1^1 + k_1$

Send $\hat{w}_1^0, \hat{w}_1^1$ →

Calculate $w_1^1 = \hat{w}_1^1 - r$

[3] https://arxiv.org/ftp/arxiv/papers/1705/1705.08963.pdf

Arithemtic Circuit

- Two or more parties can evaluate any function on their private inputs using secret sharing.
- SDPZ is an Multi-Party-Computation MPC protocol allowing joint computation of arithmetic circuits.
- Known as SPDZ ('speedz')
- SPDZ is used for privacy-preserving deep-learning library called tensorflow encrypted[4].

[4] https://github.com/tf-encrypted/tf-encrypted

What is secret sharing?

- Suppose a field element $a \in Z_p$.
- Split a up at random (uniformly) into two pieces, $a = a_1 + a_2$,
- Give party P_1 the value a_1 and P_2 the value a_2.
- Neither party knows the value a, but together they can recover it.
- We will write $< a >$ to mean that the value a is secret-shared between all parties (i.e. for each i, party P_i has a_i, where $\sum a_i = a$)
- We call this method as additive secret sharing.

Overview of secret-sharing MPC

1. **Providing Inputs**: The parties first secret-share their inputs; *i*.e. input x^i is shared so that $\sum_j x_j^i = x^i$ and party P_j holds x_j^i (and P_i which provides input is included in this sharing, even though it knows the sum).
2. **Circuit Evaluation**: The parties perform additions and multiplications on these secret values by local computations and communication of certain values. By construction, the result of performing an operation is automatically shared amongst the parties (i.e. with no further communication or computation).
3. **Opening the Result**: Finally, the parties 'open' the result of the circuit evaluation. This last step involves each party sending their 'final' share to every other party (and also performing a check that no errors were introduced by the adversary along the way).

Providing Inputs

1. P_i with x^i
2. Split x^i up at random (uniformly) into n pieces $\sum_j x_j^i = x^i$.
3. Send x_j^i to P_j and keep x_i^i
4. Thus we have turned x^i into $< x^i >$

Circuit Evaluation

- Suppose we want to compute some arithmetic circuit on our data; that is, some series of additions and multiplications.
- Our goal is to apply the addition and the multiplications on the shared x^i (i.e., $< x^i >$)

Addition

Suppose we want to compute $< a + b >$ given some $< a >$ and $< b >$

Party P_i has a_i and b_i, each party can locally compute $a_i + b_i$, and hence, since $\sum_i a_i + \sum_i b_i = \sum_i (a_i + b_i)$, they obtain a secret sharing $< a + b >$ of the value a+b.

Multiplication by Constant

Suppose we want to compute $< \alpha a >$ given some public value α and $< a >$

Each party multiplies its share by the public value α, then since $\sum_i \alpha a_i = \alpha \sum_i a_i = \alpha a$ the parties obtain a secret sharing $< \alpha a >$ of the value αa.

Addition and multiplication by public constant can be done locally by each party

Multiplication

Suppose we want to compute $< xy >$ given some $< x >$ and $< y >$:

In 1991, Donald Beaver observed that if we already have three secret-shared values, called a triple, $< a >$, $< b >$ and $< c >$ such that $c = ab$. Then we can compute $< xy >$. How?

1. Each party broadcasts $x_i - a_i$ and $y_i - b_i$, then each party P_i can compute x-a and y-b (so these values are publicly known)
2. Each party compute
$$z_i = c_i + (x - a)b_i + (y - b)a_i$$
3. Additionally, one party (chosen arbitrarily) adds on the public value (x-a)(y-b) to their share.

Summing all the shares up, the parties get

$$\sum_i z_i = c + (x - a)b + (y - b)a + (x - a)(y - b) = xy$$

Notes on the multiplication triplets

- We need lots of triples as much as the number of multiplication we have in our circuit.
- They are completely independent of the circuit to be evaluated.
- We can generate these triples at any point prior to evaluating the circuit.
- The values of a, b and c are not known by any parties when generated - each party only knows its share.
- A trusted party can supply these triplets OR some cryptographic protocols can be used to generate them.

MACs

- MACs are used to protect the integrity of the shared values against modification of any party
- Before any preprocessing computation takes place, the parties agree on some MAC key α with which they MAC all their data and which they secret-share amongst themselves so that no individual party knows the MAC key.
- This MAC key is used to MAC all the data in the circuit. For each secret-shared value, there is also a secret-shared MAC on that value.
- After the circuit has been evaluated, the parties open the secret corresponding to the circuit output and also the MAC on it, and check that the MAC is correct. If an actively corrupt party cheats at any point in the circuit evaluation, the final MAC check reveals this has happened. Note that this means the parties could be computing on invalid data.

Outline

Common Cryptographic Libraries

- Oblivious C https://oblivc.org/main/
- Tensorflow Encrypted https://github.com/tf-encrypted/tf-encrypted
- OpenSSL http://www.openssl.org/
- Cryptlib http://www.cs.auckland.ac.nz/ pgut001/cryptlib/
- Crypto++ http://www.cryptopp.com/
- BSAFE http://www.rsa.com/node.aspx?id=1204
- MIRACL(Multiprecision Integer and Rational Arithmetic C/C++ Library) http://www.shamus.ie
- Pycharm

Questions