# Security (CS4028)

## Lecture 6. Asymmetric Cryptography II

Chunyan Mu

chunyan.mu@abdn.ac.uk

The School of Natural and Computing Sciences

# Schedule

| | Week | Lecture 1 | Lecture 2 | Tutorial |
|---|---|---|---|---|
| | 1 | Intro to course & security | Intro to Crypto | - |
| | 2 | Symmetric Crypto | Hash | Math for crypto |
| ⇒ | 3 | Asymmetric Crypto-1 | Asymmetric Crypto-2 | Symmetric Crypto |
| | 4 | Signatures | Zero Knowledge Proof | Asymmetric Crypto |
| | 5 | Certificates | Authentication | Signature & certificates |
| | 6 | Access Control | AC models | Authentication |
| | 7 | Information flow control | Information flow control | Access control |
| | 8 | Management | Protocols | Concepts & management |
| | 9 | Network security | Network security | Protocols and communications |
| | 10 | Advanced topics | Advanced topics | Network |
| | 11 | Revision | | |

# Outline

# Outline

Review: Symmetric-key Cryptography

Basic mathematical background (optional)

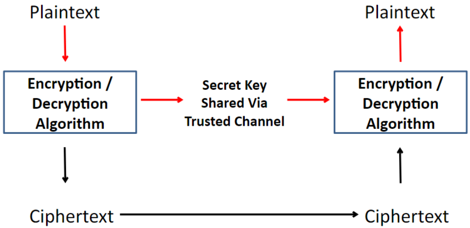
Diffie-Hellman Key Exchange
- Diffie-Hellman key exchange: general overview
- The protocol
- The man-in-the-middle attack

ElGamal
- key generation
- signature
- encryption/decryption

Summary

# Review: Symmetric-key Encryption

## Secret Key Encryption




- Same key is used by sender and receiver, has to be share via some trusted channel.
- What do you do when there is no secure (trusted) channel?

# Review: Symmetric-key Encryption

## Key Distribution

- With any symmetric algorithm, the key must be agreed upon by sender and receiver in a secure way
- Before 1976, key exchange was one of the biggest problems in secure communications.
- Various people and groups arrived at “asymmetric” algorithm solutions around the same period. Some public, some not.

# Review: Symmetric-key Encryption

## Key Distribution

Possible Strategies:

- A key could be selected by A and physically delivered to B
- A third party could select the key and physically deliver it to A and B
- If A and B have previously used a key, one party could transmit the new key by encrypting it with the old key
- If both A and B have an encrypted connection with a third party C, C could deliver a key on the encrypted links to A and B

# Review: Symmetric-key Encryption

## Key Establishment/agreement

Possible Strategies:

- Modern internet imposes new requirements.
- Need something that scales up to deal with huge numbers of communicating peers.
- Pairs of peers may be new to each other.
- May need to minimise dependence on trusted-third parties to deliver keys to peers.
  - Some intelligence agencies would like to roll this back a bit.

# Outline

# The order of an integer

## Definition: order

The order of an integer $m$ modulo a (natural) number $n$ is defined to be the smallest positive integer power $r$ such that:

$$m^r = 1 \bmod n$$

# The order of an integer

## Definition: order

The order of an integer $m$ modulo a (natural) number $n$ is defined to be the smallest positive integer power $r$ such that:

$$m^r = 1 \bmod n$$

## Example: What's the order of 3 modulo 13?

$$3^1 \equiv 3 \pmod{13}$$
$$3^2 \equiv 9 \pmod{13}$$
$$3^3 \equiv 1 \pmod{13}$$

So the order of 3 modulo 13 is 3.

# The order of an integer

## Remark

- The order $r$ of $m$ modulo $n$ is denoted by $ord_n(m)$
- $ord_n(m)$ does not always exists, e.g.,
  - $ord_{24}(3)$: any even power of 3 yields 9 modulo 24, and any odd power of 3 is 3 modulo 24

$$3^3 = 3^1 \pmod{24},\ 3^2 = 3^4 = 9 \pmod{24}$$

  - $ord_{24}(12)$: positive power vanishes

$$12^2 = 144 \pmod{24} = 0 \pmod{24}$$

- For such numbers there does not exist a positive finite power to yield its order, and the order is then defined as infinite.

# Generators

## Notation: $\mathbb{Z}_n^*$

- The symbol $\mathbb{Z}_n$ denotes the complete set of residues modulo $n$, i.e., $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$.
- The symbol $\mathbb{Z}_n^*$ denotes the reduced set of residues modulo $n$, i.e., $\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n : gcd(x, n) = 1\}$.
- Recall that 1 is relatively prime to all the other numbers.

## Definition: generator

An element $g \in \mathbb{Z}_n^*$ is a generator $\bmod n$ (or a generator of the set $\mathbb{Z}_n^*$) if for each $a \in \mathbb{Z}_n^*$ there exists some $x$ where:

$$g^x \equiv a \bmod n$$

# Generators

### Definition: generator

An element $g \in \mathbb{Z}_n^*$ is a generator $\bmod n$ (or a generator of the set $\mathbb{Z}_n^*$) if for each $a \in \mathbb{Z}_n^*$ there exists some $x$ where:

$$g^x \equiv a \bmod n$$

Example:Find all the generators of $\mathbb{Z}_7^*$

| m | $m^1$ | $m^2$ | $m^3$ | $m^4$ | $m^5$ | $m^6$ | $ord_7(m)$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 4 | 1 | 2 | 4 | 1 | 3 |
| **3** | **3** | **2** | **6** | **4** | **5** | **1** | **6** |
| 4 | 4 | 2 | 1 | 4 | 2 | 1 | 3 |
| **5** | **5** | **4** | **6** | **2** | **3** | **1** | **6** |
| 6 | 6 | 1 | 6 | 1 | 6 | 1 | 2 |

# The Discrete Logarithm Problem (DLP)

Evaluating the expression $a^x \bmod n$ (modular exponentiation) is easy. The inverse problem of *modular exponentiation* is that of **finding the discrete logarithm of a number**. This is a hard problem:

### The Discrete Logarithm Problem (DLP)

Find $x$ such that $a^x \equiv b \pmod{n}$.

### Example

If $3^x \equiv 15 \pmod{17}$, then $x = 6$.

Not all discrete logarithm have solutions, e.g., :

$$3^x \equiv 7 \pmod{13}$$

# Outline

# Key agreement: Diffie-Hellman

## General overview: Diffie-Hellman key exchange

- A method of securely exchanging cryptographic keys over a public channel
- One of the earliest practical examples of public key exchange implemented within the field of cryptography.
- Published in 1976 by Diffie and Hellman, the earliest publicly known work that proposed the idea of a private key and a corresponding public key
- Allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel.

# Key agreement: Diffie-Hellman

## General overview: Diffie-Hellman key exchange

An analogy illustrates the concept of public key exchange:

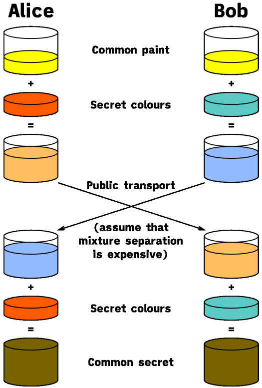



Image taken from: `https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange`

# Diffie-Hellman for two parties

## Protocol based on discrete logarithms

0. Alice and Bob agree on a large prime number $n$ and an integer $g$, such that $g$ is a generator mod $n$
1. Alice chooses a large random integer $x$ and Bob chooses a large random integer $y$
2. Alice computes and sends Bob $X = g^x \mod n$,
   while Bob computes and sends Alice $Y = g^y \mod n$
3. Alice computes $k_B = Y^x \mod n = g^{yx} \mod n$
4. Bob computes $k_A = X^y \mod n = g^{xy} \mod n$
   $k_A = k_B$ is used as the secret key Alice and Bob will share

# Diffie-Hellman for two parties

## An example

0. Alice and Bob agree on $g = 7$, while $n = 23$.
1. Alice chooses $x = 5$ and Bob chooses $y = 8$.
2. Alice computes $X = 7^5 \mod 23 = 17$,
   Bob computes $Y = 7^8 \mod 23 = 12$,
   Alice sends 17 to Bob and Bob sends 12 to Alice.
3. Alice computes $k_B = 12^5 \mod 23 = 18$
4. Bob computes $k_A = 17^8 \mod 23 = 18$

$k_A = k_B = 18$ is the encryption key for the session.

# Diffie-Hellman for four parties

## Protocol (based on discrete logarithms)

0. Alice, Bob, Carol and Dave agree on a large prime number $n$ and an integer $g$, such that $g$ is a generator $\mod n$
1. Choices:
   - Alice chooses a random large integer $w$
   - Bob chooses a random large integer $x$
   - Carol chooses a random large integer $y$
   - Dave chooses a random large integer $z$

# Diffie-Hellman for four parties

## Protocol (based on discrete logarithms)

2. Sent messages (round 1):
   - Alice sends Bob $W = g^w \mod n$
   - Bob sends Carol $X = g^x \mod n$
   - Carol sends Dave $Y = g^y \mod n$
   - Dave sends Alice $Z = g^z \mod n$
3. Sent messages (round 2):
   - Alice sends Bob $Z' = Z^w \mod n = g^{zw} \mod n$
   - Bob sends Carol $W' = W^x \mod n = g^{wx} \mod n$
   - Carol sends Dave $X' = X^y \mod n = g^{xy} \mod n$
   - Dave sends Alice $Y' = Y^z \mod n = g^{yz} \mod n$

# Diffie-Hellman for four parties

## Protocol (based on discrete logarithms)

4. Sent messages (round 3):
   - Alice sends Bob $Y'' = Y'^w \mod n = g^{yzw} \mod n$
   - Bob sends Carol $Z'' = Z'^x \mod n = g^{zwx} \mod n$
   - Carol sends Dave $W'' = W'^y \mod n = g^{wxy} \mod n$
   - Dave sends Alice $X'' = X'^z \mod n = g^{xyz} \mod n$
5. Computation:
   - Alice computes $k_1 = X''^w \mod n = g^{xyzw} \mod n$
   - Bob computes $k_2 = Y''^x \mod n = g^{yzwx} \mod n$
   - Carol computes $k_3 = Z''^y \mod n = g^{zwxy} \mod n$
   - Dave computes $k_4 = W''^z \mod n = g^{wxyz} \mod n$

$k_1 = k_2 = k_3 = k_4$ the secret key Alice, Bob, Carol and Dave will share

# The man-in-the-middle attack

Main drawback of Diffie-Hellman protocol:

- vulnerable to man-in-the-middle attack

Imagine that Mallory can intercept communications...

- Consider the following man-in-the-middle attack:
  1. Alice sends Bob her public key $PK_A$
  2. Mallory intercepts $PK_A$ and sends Bob his public key $PK_M$
  3. Bob replies by sending his public key $PK_B$
  4. Mallory intercepts $PK_B$ and sends Alice $PK_M$

# The man-in-the-middle attack

Main drawback of Diffie-Hellman protocol:

- vulnerable to man-in-the-middle attack

Imagine that Mallory can intercept communications...

- Now, any time Alice sends a message to Bob, since she is using $PK_M$ instead of $PK_B$, Mallory will:
  1. intercept it
  2. decrypt it using $SK_M$ (eventually modify it)
  3. encrypt it using $PK_B$
  4. send it to Bob

Man-In-The-Middle (MITM) attack works because Diffie Hellman does not authenticate participants

# Outline

# ElGamal: overview

## Overview

1. asymmetric key encryption algorithm for public-key cryptography, which is based on Diffie-Hellman key exchange
2. described by Taher ElGamal in 1984

## The algorithm

1. key generation
2. Signature and verification
3. encryption
4. decryption

# ElGamal: key generation

### Initial generation of keys

Alice wants to send a message to Bob.

1. Bob choose a prime number $p$
2. Bob choose two random numbers $g$ (a generator of $\mathbb{Z}_p^*$) and $x$ less than $p$
3. Bob compute $y = g^x \bmod p$

$p, g$ and $y$ are <u>PUBLIC</u>, $x$ is <u>PRIVATE</u>

# ElGamal: signature

## Signature

1. Consider a given message $M$ Bob wants to sign, he:
   1.1 chooses a SECRET random number $k$ relatively prime to $p-1$
   1.2 computes $s_1 = g^k \bmod p$
   1.3 computes $s_2$, such that $M = (xs_1 + ks_2) \bmod (p-1)$
2. Signature $S = (s_1, s_2)$
3. Note that:
   - $s_2$ depends on $x$ which is PRIVATE
   - $s_1$ depends on $k$ which is SECRET
4. Signature verification: Alice verifies
   $y^{s_1} {s_1}^{s_2} \bmod p = g^M \bmod p$

# ElGamal: signature

## Why it works?

$$
\begin{aligned}
& y^{s_1} \cdot s_1^{s_2} \mod p \\
= \; & g^{xs_1} \cdot g^{ks_2} \mod p \quad \text{(since: } y = g^x \bmod p, s_1 = g^k \bmod p\text{)} \\
= \; & g^{xs_1 + ks_2} \mod p \\
= \; & g^{(p-1)j+M} \mod p \quad \text{(for some } j\text{, since:} xs_1 + ks_2 = M \bmod p-1\text{)} \\
= \; & g^M \cdot (g^{(p-1)})^j \mod p \\
= \; & g^M \cdot 1^j \bmod p \qquad \text{(since: } g^{p-1} = 1 \bmod p\text{)} \\
= \; & g^M \mod p
\end{aligned}
$$

# ElGamal

Example: Consider the message $M = 5$ Bob wants to sign

1. consider $p = 11$ (PUBLIC), $g = 2$ (PUBLIC)
2. choose private key $x = 8$ (PRIVATE)
3. calculate $y = g^x \bmod p = 2^8 \bmod 11 = 3$ (PUBLIC)
4. choose a random $k = 9$ (SECRET) such that $gcd(k, p-1) = gcd(9, 10) = 1$
5. compute $s_1 = g^k \bmod p = 2^9 \bmod 11 = 6$

# ElGamal

Example: Consider the message $M = 5$ Bob wants to sign

6. compute $s_2$ such that $M = (xs_1 + ks_2) \bmod p - 1$ that is $M = (8 \times 6 + 9 \times s_2) \bmod 10 \rightarrow s_2 = 3$
7. generate signature $(s_1, s_2) = (6, 3)$
8. Alice verify the signature:

$$\begin{aligned} y^{s_1} \times {s_1}^{s_2} \bmod p &= 3^6 \times 6^3 \bmod 11 \\ &= 729 \times 216 \bmod 11 = 10 \\ g^M \bmod p &= 2^5 \bmod 11 = 10 \end{aligned}$$

# ElGamal: encryption

## Encryption

- Alice wants to encrypt a given message $M$ to Bob under his public key $(p, g, y)$, she:
  1. chooses a *SECRET* random number $k$ relatively prime to $p-1$
  2. computes $c_1 = g^k \bmod p$
  3. computes $c_2 = y^k \times M \bmod p$
- Ciphertext $C = (c_1, c_2)$
- Note that:
  - $p, g, y$ are PUBLIC, $k$ is SECRET
  - ciphertext $C$ is twice the size of the plaintext $M$

# ElGamal encryption

### Example: Consider the plaintext $M = 5$

Consider Alice wants to send plaintext $M = 5$ to Bob under his public key $(p, g, y) = (11, 2, 3)$.

1. Alice chooses at random $k = 9$ (SECRET) such that $gcd(k, p - 1) = 1$, that is $gcd(9, 10) = 1$
2. Alice computes $c_1 = g^k \bmod p = 2^9 \bmod 11 = 6$
3. Alice computes $c_2 = y^k \times M \bmod p = 3^9 \times 5 \bmod 11 = 9$
4. Alice generates ciphertext $C = (c_1, c_2) = (6, 9)$

# ElGamal: decryption

## Decryption

Consider Bob wants to decrypt a given ciphertext $C = (c_1, c_2)$ using his private key $x$, he:

- calculates $s = c_1^x \bmod p$
- then computes $M = c_2 \cdot s^{-1} \bmod p$ to obtain the plaintext
- the decryption algorithm produces the intended message, since

$$
\begin{aligned}
s &= c_1^x = g^{kx} \bmod p \\
s^{-1} &= (g^{kx})^{-1} \\
c_2 \cdot s^{-1} &= (y^k \times M) \times (g^{kx})^{-1} \bmod p \\
&= (g^{kx} \times M) \times (g^{kx})^{-1} \bmod p \\
&= M \bmod p
\end{aligned}
$$

# ElGamal decryption

Example: Consider the ciphertext $C = (6, 9)$

Bob wants to decrypt ciphertext $C = (6, 9)$ using his private key $x = 8$:

1. $s = c_1^x \bmod p = 6^8 \bmod 11 = 4$
2. $s^{-1} = 3 \bmod 11$
3. $M = c_2 \cdot s^{-1} \bmod p = 9 \times 3 \bmod 11 = 5$

# Outline

# Summary

## This lecture

- RSA: key generation, encryption and decryption
- Diffie-Hellman key exchange
- ElGamal: key generation, signature, encryption and decryption

## Next lecture

- Signatures