

Mobile and Wireless Security

CSCE 496/896

Lecture # 3

Basics of cryptography and security



Instructor: Nirnimesh Ghose
Computer Science and Engineering

Classification of Security

Unconditionally secure : Unlimited power of adversary, perfect (*eg.* : one-time pad).

Provably secure : Under the assumption of well known hard mathematical problem.

Computationally secure : Amount of computational effort by the best-known methods (*Practical Secure*).

Provable Security

This refers to **mathematical proofs**, which are common in cryptography.

The aim of the proof is to show that the attacker (attacker model is defined) must **solve** the underlying **hard problem** in order to break the security of the modelled system.

Such a proof generally does not consider side-channel attacks or other implementation-specific attacks.

Computational Security

Against Brute-force attack and currently best-known cryptanalysis approach

The **cost** of **breaking** the **cipher exceeds** the **value** of the encrypted information

The time required to break the cipher exceeds the useful lifetime of the information

Lecture Set Overview

Asymmetric key cryptography

- Public key cryptography

- Digital signature

Key management

- Key exchange

- Key distribution centers

- Certifications authorities

- Types of keys

Public Key Encryption



Everybody know Bob's **public key**

- How is this achieved in practice?

Only Bob knows the corresponding **private key**

Goal:

Alice wants to send a secret message to Bob

Bob wants to authenticate

Application of PKC

Encryption to achieve confidentiality

Anyone can encrypt with public key.

Only someone with private key can decrypt.

Digital signature for authentication

“Sign” a message with private key.

Verified by public key.

Session key establishment

Exchange messages to establish a secret session key

Switch to symmetric cryptography with session key ([why?](#))

Requirements

Key generation: Computationally easy to generate a pair (public PK and private SK) key.

Computationally infeasible to determine SK from PK .

Encryption: Given plaintext (P) and PK , easy to compute ciphertext $C = E_{PK}(P)$.

Decryption: Given ciphertext (C) and SK , easy to compute plaintext $P = D_{SK}(C)$.

Infeasible to compute P from C without access to SK .

RSA

Named after Rivest, Shamir, and Adleman.

Public key (PK) → Encryption

Private key (SK) → Decryption

Key length is variable: 512, 1024, 2048 bits.

Input is variable (< key length).

Output is of key length.

Used for: Short message exchange (eg: secret key), digital signature.

RSA Algorithm

Choose two large prime numbers p and q ,

compute $n = p \times q$ and $\varphi(n) = (p - 1) \times (q - 1)$.

Choose e , which is **relatively prime** to $\varphi(n)$.

Relatively prime numbers have a greatest common factor (gcf) of 1.

Compute **multiplicative inverse** of $e \bmod \varphi(n)$ as d (Euclid's algorithm)

$e \times d = 1 \bmod \varphi(n)$.

Now,

Public key (PK) $\rightarrow \langle e, n \rangle$.

Private key (SK) $\rightarrow \langle d, n \rangle$.

Encryption and Decryption

Public key (PK) $\rightarrow \langle e, n \rangle$.

Private key (SK) $\rightarrow \langle d, n \rangle$.

Encryption:

$$c = m^e \bmod n, \text{ where } m \in [0, n - 1].$$

Decryption:

$$m = c^d \bmod n.$$

Correctness:

$$\begin{aligned} c^d \bmod n &= (m^e \bmod n)^d \bmod n \\ &= (m^e)^d \bmod n \\ &= (m^e)^d \bmod n \\ &= m \bmod n = m \end{aligned}$$

Remember: $e \times d = 1 \bmod \varphi(n)$

Why is RSA Secure?

RSA problem: Given c , $n = pq$ and e such that $\gcd(e, (p-1)(q-1)) = 1$

Find m such that $c = m^e \pmod n$

Which means recover m from ciphertext c and public key (n, e)

For sufficiently large prime numbers, there is no known efficient algorithm for solving this.

Factoring problem: Given a positive integer n ,

find primes p_1, \dots, p_k such that $n = (p_1)^{e_1} (p_2)^{e_2} \dots (p_k)^{e_k}$.

If factoring is easy, then RSA problem is easy, but there is no known reduction from factoring to RSA.

In other words, it may be possible to break RSA (i.e., take e^{th} root of c) without factoring n .

Integrity in RSA Encryption

Plain RSA does not provide integrity

Given encryptions of m_1 and m_2 , attacker can create encryption of $m_1 \cdot m_2$

$$(m_1^e) \cdot (m_2^e) \bmod n = (m_1 \cdot m_2)^e \bmod n.$$

Attacker can convert m into m^k without decrypting

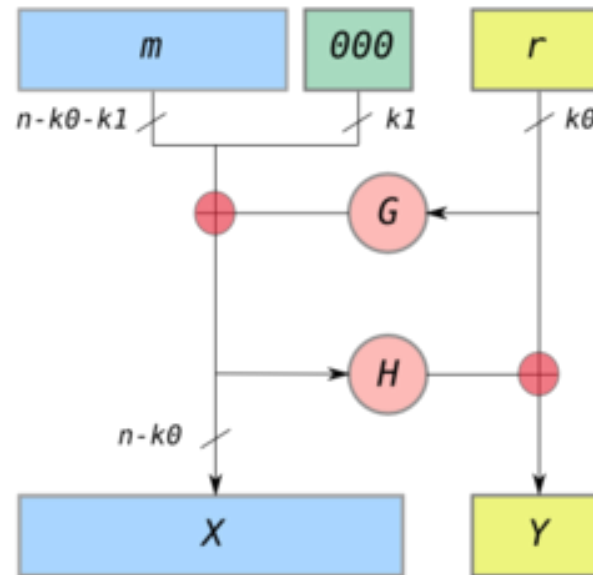
$$(m^e)^k \bmod n = (m^k)^e \bmod n$$

Optimal Asymmetric Encryption Padding (OAEP)

In practice, OAEP is used

Resulting encryption is plaintext-aware: infeasible to compute a valid encryption without knowing plaintext

If hash functions are “good” and RSA problem is hard



Digital Signature: Basic Idea



Everybody know Bob's **public key**.

Only Bob knows the corresponding **private key**.

Goal:

Bob sends a "digitally signed" message.

To compute a signature, must know the **private key**.

To verify a signature, enough to know the **public key**.

Examples of Digital Signature

RSA signature

ElGamal Signature

Chaum's blind signature

RSA Signature

Public key (PK) $\rightarrow \langle e, n \rangle$.

Private key (SK) $\rightarrow \langle d, n \rangle$.

Signature:

$s = m^d \bmod n$, where $m \in [0, n - 1]$.

Signing and decryption are the same operation in RSA.

It's infeasible to compute s on m if the entity doesn't know d .

m, s are transmitted.

Verification:

$m \stackrel{?}{=} s^e \bmod n$.

Just like encryption

Anyone who knows n and e (public key) can verify

In practice, also need to use padding & hashing

Advantages of PKC

Achieves **confidentiality without shared secrets**

Useful in open environment (e.g.: Internet)

No “chicken-and-egg” key establishment problem.

With symmetric crypto, two parties must share a secret before communicating .

Achieves **authentication without shared secrets**.

Use digital signature for message authentication

Disadvantage of PKC

Computation is 2-3 orders of magnitude slower.

Modular exponentiation is an expensive computation.

Typical usage: use public-key crypto to establish a shared secret (session key), to switch to symmetric cryptography.

Keys are longer.

RSA requires 1024 bits key compared to 128 bits required for AES.

Relies on unproven number-theoretic assumption.

PKC will fail if someone discovers an easy way for factoring.

Key Management

Trusted Intermediaries

Symmetric key problem:

How do two entities establish shared secret key over network?

Solution: Trusted key distribution center (KDC) acting as intermediary between entities.

Public key problem:

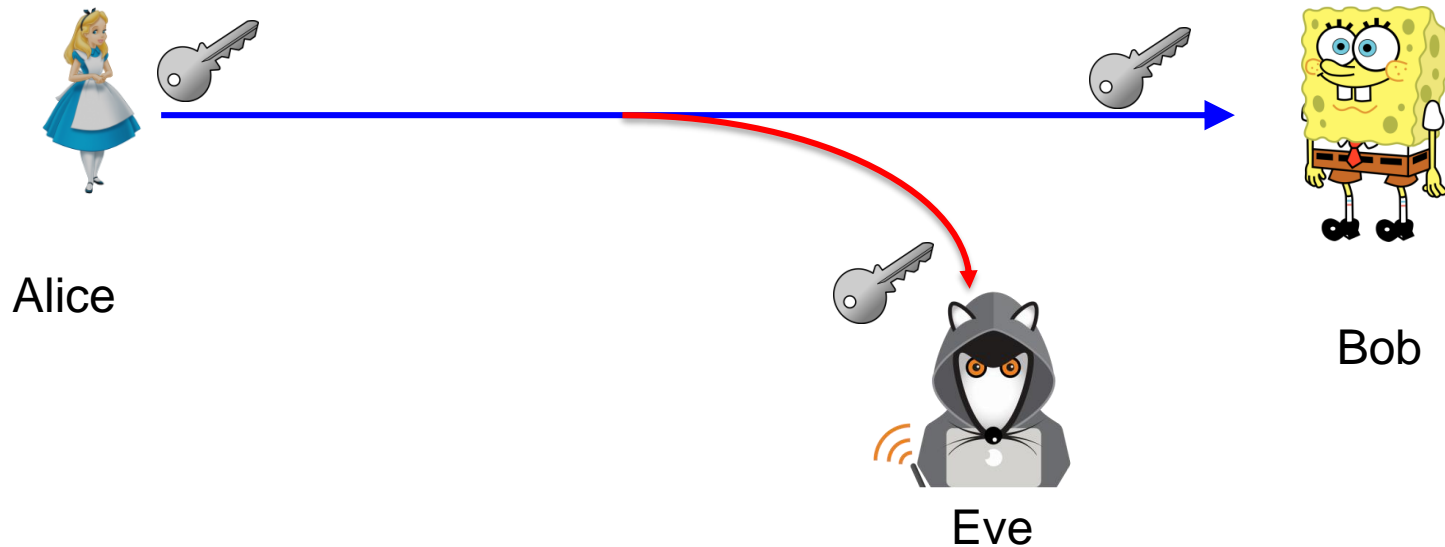
When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Mallory's?

Solution: Trusted certification authority (CA)

Symmetric Key-Exchange Protocols

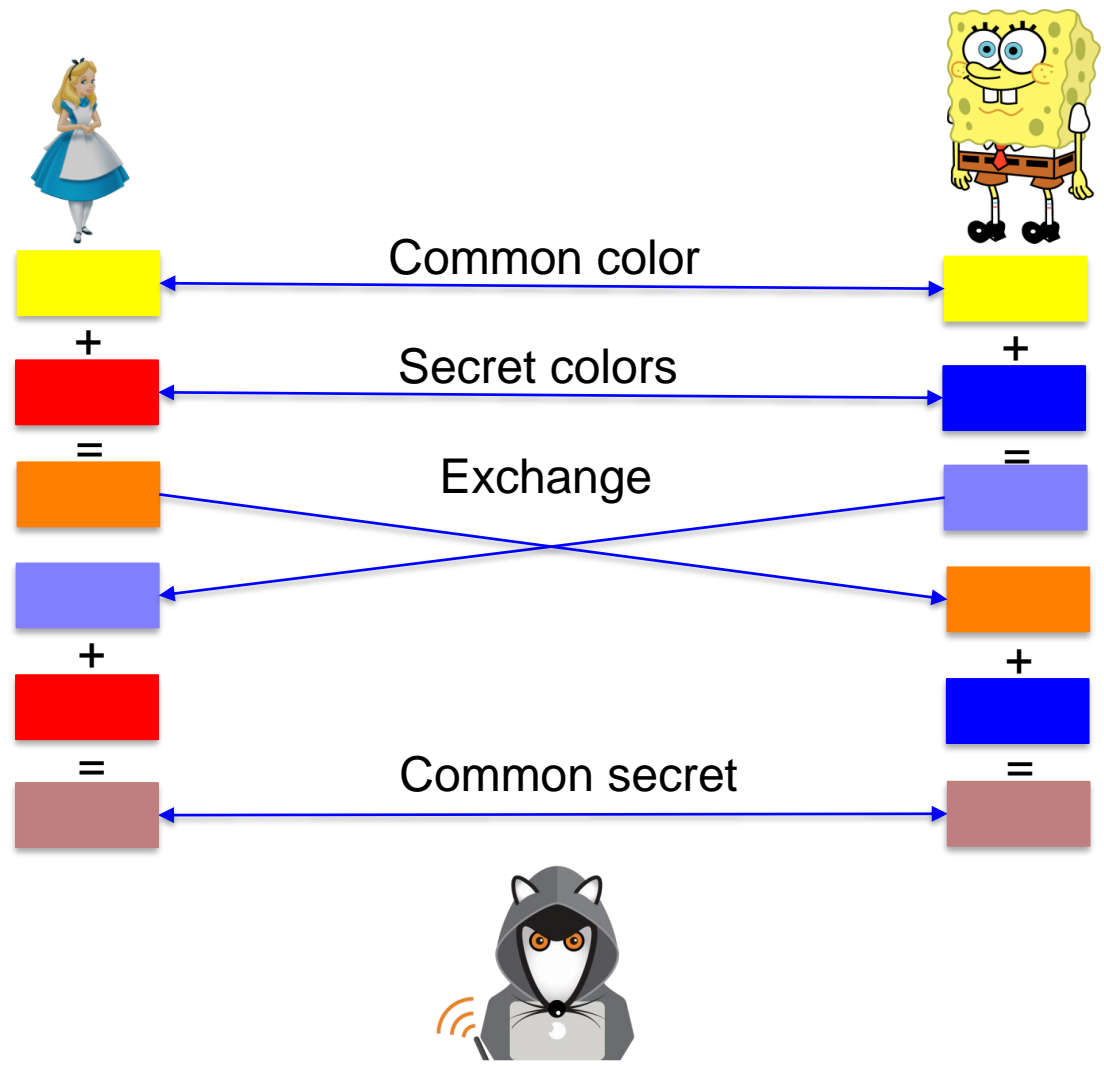
Alice and Bob can exchange symmetric key over the public channel.

Problem: Any eavesdropper can listen and extract the key.

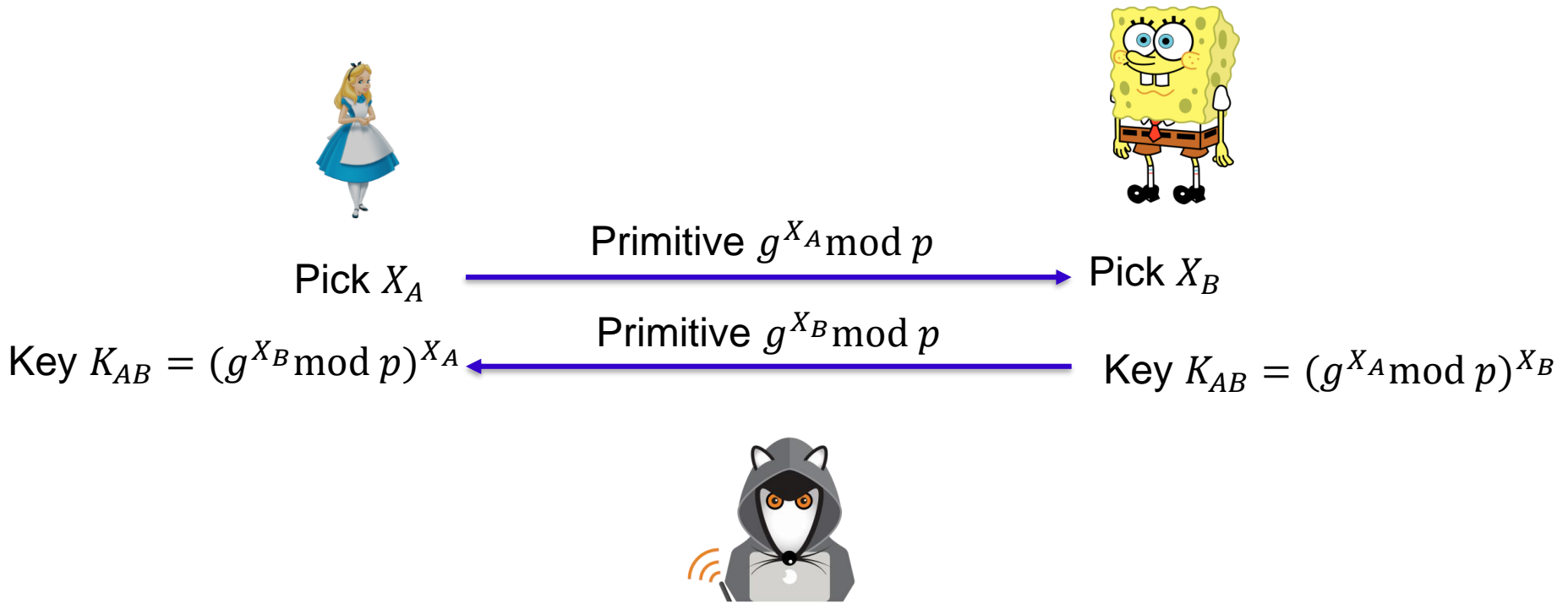


Solution: Diffie-Hellman key exchange

Diffie-Hellman Protocol (1976)



Diffie-Hellman Protocol (1976)



Why is Diffie-Hellman Secure?

Discrete Logarithm (DL) problem:

Given $\rightarrow g^{X_A} \bmod p$, g and p ; it is hard to extract X_A .

There is no efficient algorithm to perform the operation.

This is not enough for DH to be secure.

Computational Diffie-Hellman (CDH) problem:

Given $\rightarrow g^{X_A} \bmod p$ and $g^{X_B} \bmod p$; it is hard to compute $g^{X_A X_B} \bmod p$.

It is easy when at least X_A or X_B is known.

Decisional Diffie-Hellman (DDH) problem:

Given $\rightarrow g^{X_A} \bmod p$ and $g^{X_B} \bmod p$;

it is hard to differentiate between

$g^{X_A X_B} \bmod p$ and $g^r \bmod p$ for any r random number.

Diffie-Hellman unsecure against MitM Attack



Pick X_A $\xrightarrow{g^{X_A} \bmod p}$

Pick X_M

$\xrightarrow{g^{X_M} \bmod p}$ Pick X_B

$$K_{AM} = (g^{X_M} \bmod p)^{X_A} \xleftarrow{g^{X_M} \bmod p}$$

$$K_{AM} = (g^{X_A} \bmod p)^{X_M} \xleftarrow{g^{X_B} \bmod p}$$

$$K_{MB} = (g^{X_B} \bmod p)^{X_M}$$

$$K_{MB} = (g^{X_M} \bmod p)^{X_B}$$