# 网络安全技术

刘振

上海交通大学 计算机科学与工程系

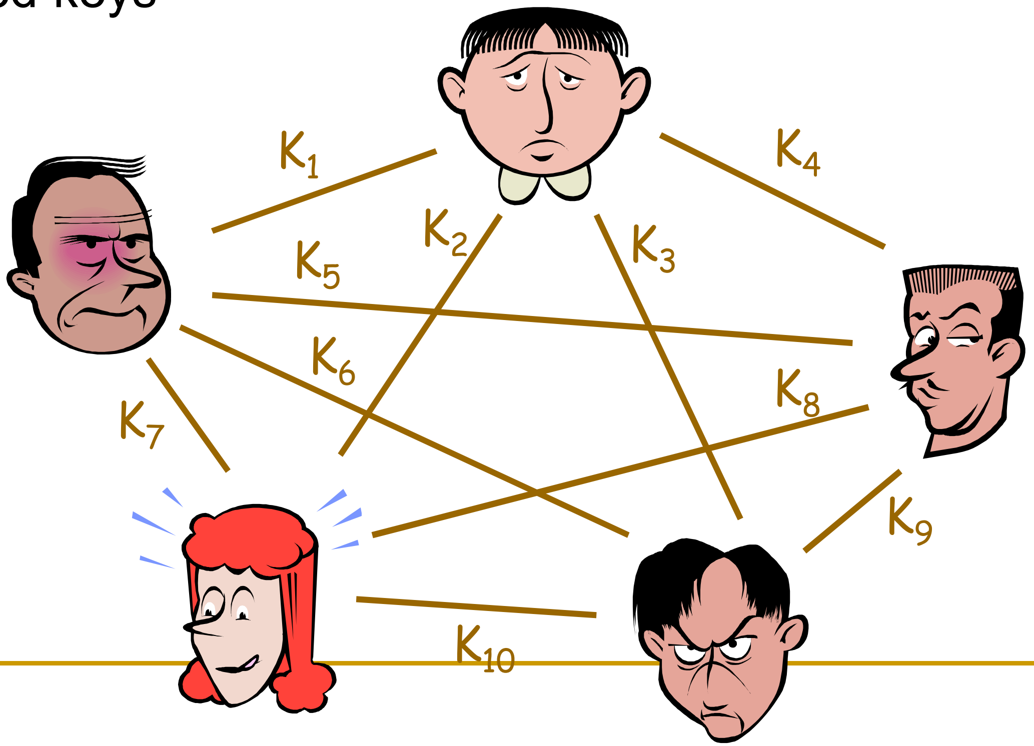
电信群楼3-509

liuzhen@sjtu.edu.cn

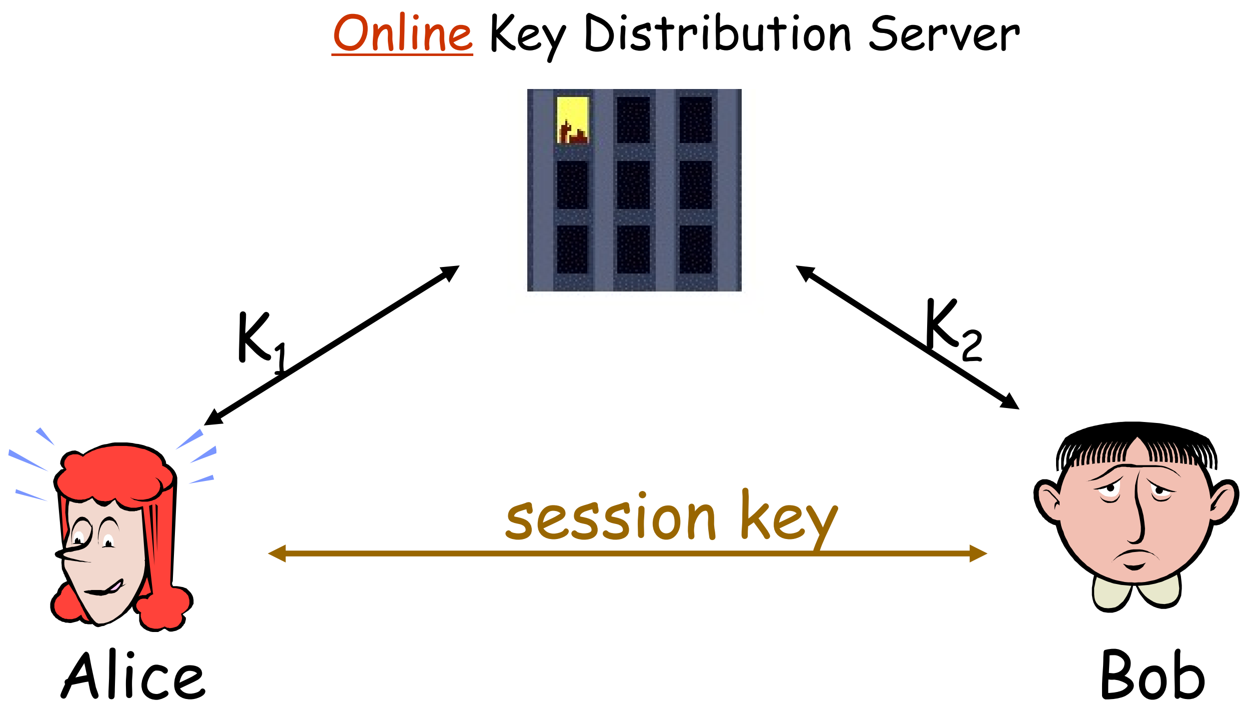
# Public Key Cryptography: Encryption

# Symmetric Key Management

- Each pair of communicating entities needs a shared key
- For an *n*-party system, there are $n(n-1)/2$ distinct keys in the system and each party needs to maintain $n$-1 distinct keys.
- How to reduce the number of shared keys in the system
  1. Centralized key management
  2. Public keys
- How to set up shared keys



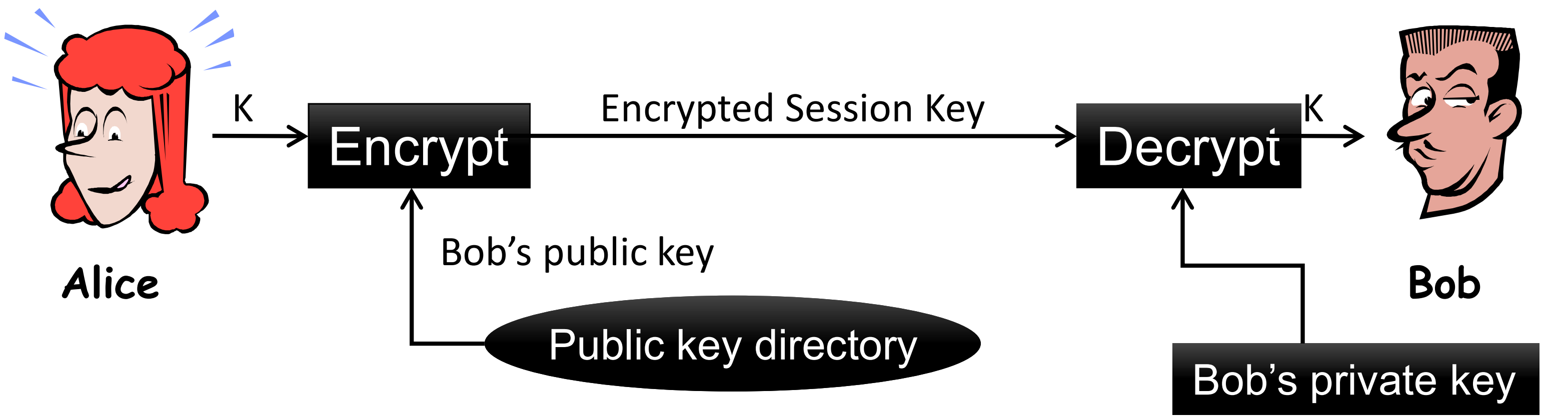
# Centralized Key Management




- Only *n* long-term secret keys, instead of *n*(*n*-1)/2 in the system.
- Each user shares one long-term secret key with the Server.
- The Server may become the **single-point-of-failure** and the performance bottleneck.
- Secret keys are used only for the secure delivery of session keys.
- Real data are encrypted under session keys.

# Public key Encryption

- Receiver Bob has a key pair: public and private
  - **publish** the public key such that the key is publicly known
  - Bob keeps the private key secret
- Other people use Bob's public key to encrypt messages for Bob
- Bob uses his private key to decrypt

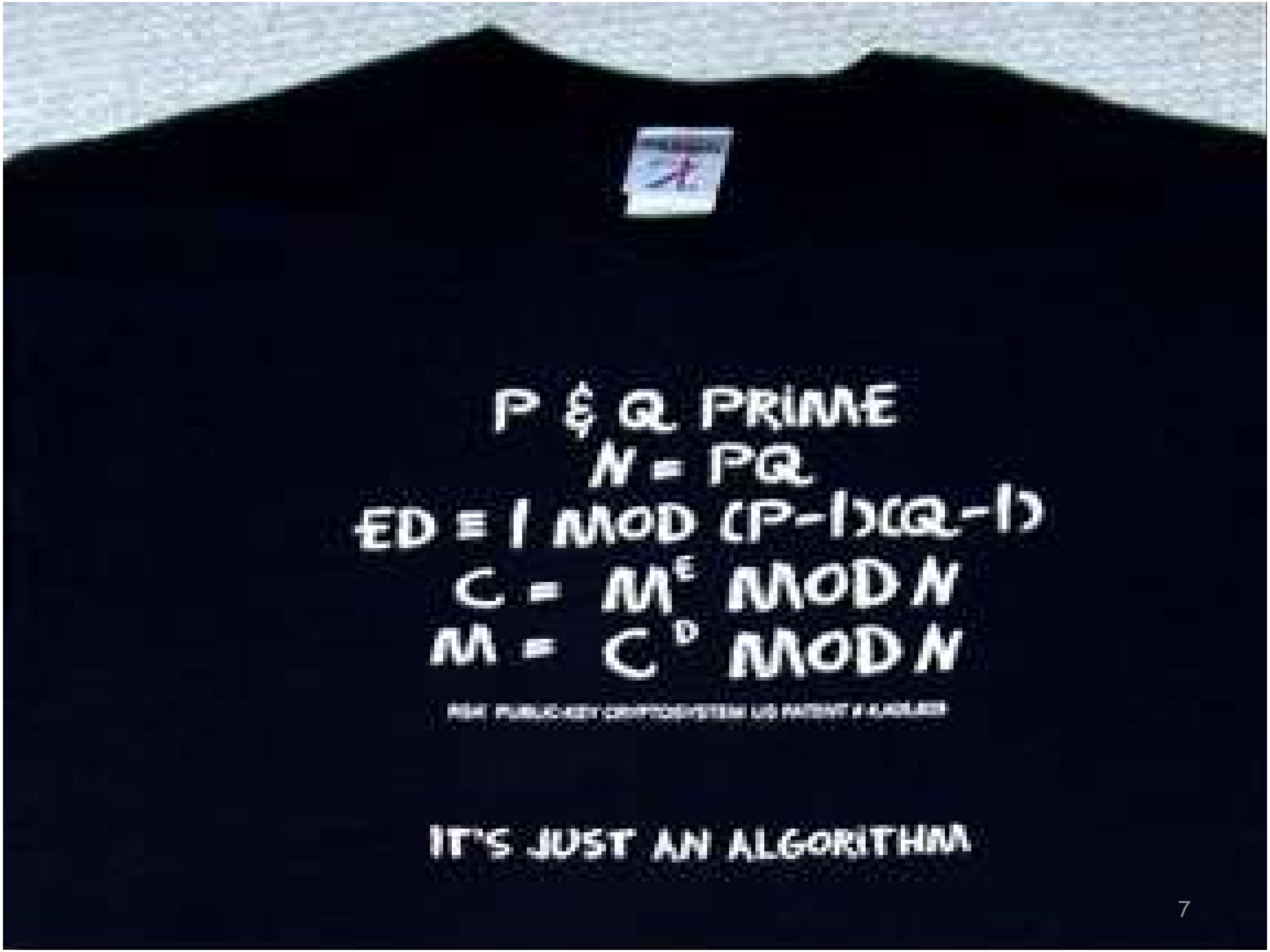



- Security requirement 1: difficult to find private key or plaintext from ciphertext
- Security requirement 2: difficult to find private key from public key

# Motivation of Public Key Cryptography (Summary)

- One problem with symmetric key algorithms is that the sender needs a secure method for telling the receiver about the encryption key.
- Plus, you need a separate key for everyone you might communicate with (scalability issue).

- Public key algorithms use a public-key and private-key pair to tackle the two problems
  - Each receiver has a (public key, private key) pair.
  - The public key is publicly known (published).
  - A sender uses the receiver's public key to encrypt a message.
  - Only the receiver can decrypt it with the corresponding private key.


P & Q PRIME
N = PQ
ED ≡ 1 MOD (P-1)(Q-1)
C = M^E MOD N
M = C^D MOD N
IT'S JUST AN ALGORITHM

# Rivest, Shamir, and Adleman (RSA)

- Randomly choose two large and roughly equal-length prime numbers, $p$ and $q$.
  - E.g. $|p| = |q| = 512$ bits
- Sets $n = pq$ ($n$ is called the public modulus)
- Randomly choose $e$ such that $\gcd(e, \phi(n)) = 1$.
  - $e$ is called the public exponent.
  - $\phi(n) = \phi(pq) = (p-1)(q-1)$
- Compute $d$ such that $de \equiv 1 \pmod{\phi(n)}$.
  - In other words, $d$ is the modular inverse of $e$ modular $\phi(n)$.
  - d is called the private exponent.

- **Public Key: $PK = (n, e)$, Private Key: $SK = d$**

- **Encryption: $C = M^e \bmod n$**
- **Decryption: $M = C^d \bmod n$**

Given a RSA public key (n,e), can we encrypt any message $M \in Z_n$?

# An Example of RSA Encryption and Decryption

- Choose two primes $p$=47 and $q$=71 ⇨ $n = pq = 3337$.
- Choose $e$ such that it is relatively prime to $\phi(n) = 46\text{x}70 = 3220$.
  - e.g. $e = 79$.
- Compute $d = e^{-1} \bmod \phi(n)$ using extended Euclidean algorithm.
  - $d \equiv 79^{-1} \pmod{3220} = 1019$
- Public key PK = $(n, e) = (3337,79)$
- Private key SK = $d = 1019$

- Encrypt $M = 688$ ⇨ $688^{79} \bmod 3337 = 1570$
- Decrypt $C = 1570$ ⇨ $1570^{1019} \bmod 3337 = 688$

# Security of RSA

- **RSA Problem (RSAP)** : Given
  - a positive integer $n$ that is a product of two distinct equal-length primes $p$ and $q$,
  - a positive integer $e$ such that $\gcd(e, (p-1)(q-1)) = 1$, and
  - an integer $c$ chosen randomly from $Z_n^*$

  find an integer $m$ such that $m^e \equiv c \pmod{n}$. Note: $p$ and $q$ are not given.
- The intractability of the RSAP forms the basis for the security of the RSA public-key cryptosystem.
  - **RSAP** is closely related to the Factorization Problem but not known to be equivalent.
- **Factorization Problem (FACTORING)** : **Given a positive integer $n$, find its prime factorization; that is, write $n = p_1^{e1} p_2^{e2} \ldots p_k^{ek}$ where the $p_i$ are primes and each $e_i \geq 1$.**
  - E.g. $72 = 2^3 \cdot 3^2$
- The value of the RSA public exponent $e$ can be small, say 16 bits long, but the value of $d$ should be large, say at least 1000 bits long.

When e is too small, it is insecure.

# RSAP and FACTORING

- RSAP $\leq_P$ FACTORING : The RSA problem can efficiently be reduced to the factorization problem.
- If one can solve FACTORING, then one can solve RSAP.
- **Open Problem** : Is FACTORING $\leq_P$ RSAP?
  - It is widely believed that it is true, although no proof of this is known.

# More about RSA Security Strength

- The strength of the RSA algorithm depends on the difficulty of doing prime factorization of large numbers:
  - Knowing the public key <$e$, $n$>, if the cryptanalyst could factor $n = pq$, then $\phi(n)$ (= $(p - 1)(q - 1)$) is obtained
  - Knowing $e$ and $\phi(n)$, $d$ can be obtained with a known algorithm (Euclid's algorithm) for finding multiplicative inverse ($de = 1 \bmod \phi(n)$)
- To break an RSA encryption (i.e., finding the decryption key) by brute force (i.e., by trying all possible keys) is not feasible given the relative large size of the keys
  - A better approach is to solve the prime factorization problem.
  - The best known factorization algorithms seem to indicate that the number of operations to factorize a number $n$ is estimated by

$$\exp\left((\ln n)^{1/3}(\ln \ln n)\right)$$

# RSA: Key Length vs. Security Strength

- RSA is inefficient – it gains strength slowly
  - RSA-1024 is equivalent to an 80-bit symmetric key
  - RSA-2048 is equivalent to a 112-bit key (3DES)
  - RSA-3072 is equivalent to 128-bit key (AES)
  - RSA-7680 is equivalent to an 192-bit AES key
  - RSA-15,380 is required to equal an AES-256 key!

  - the performance of large size RSA is terrible
- The computation time required for larger keys increases rapidly
  - The time required for signing is proportional to the cube of the key length
  - RSA-2048 operations require 8 times as long as RSA-1024
    - Example – 60ms for RSA-1024 sign. 600ms for RSA-2048
    - RSA-15,360 would take 3375 times RSA-1024, or 200 seconds!

# ElGamal Encryption Scheme

- Let p be a large prime.
- Let $Z_p^* = \{ 1, 2, 3, \ldots, p-1 \}$
- Let $Z_{p-1} = \{ 0, 1, 2, \ldots, p-2 \}$
- $a \in_R S$ means that a is randomly chosen from the set S
- Let $g \in Z_p^*$ such that none of $g^1 \bmod p$, $g^2 \bmod p$, …, $g^{p-2} \bmod p$ is equal to 1.

**Public Key Pair:**

- Private key: $x \in_R Z_{p-1}$
- Public key: $Y = g^x \bmod p$

**Encryption:**

1. $r \in_R Z_{p-1}$
2. $A = g^r \bmod p$
3. $B = MY^r \bmod p$ where $M \in Z_p^*$ is the message.

Ciphertext C = (A, B).

**Decryption:**

1. $K = A^x \bmod p$
2. $M = B\, K^{-1} \bmod p$

# An Example of ElGamal Encryption and Decryption

- Let $p = 2357$

  $g = 2$

  Private key: $x = 1751$

  Public key: $Y = g^x = 2^{1751} = 1185 \pmod{2357}$
- Encryption:
  - say $M = 2035$
  1. Pick a random number $r = 1520$
  2. Computes

     $A = g^r \equiv 2^{1520} \equiv 1430 \pmod{2357}$

     $B = MY^r \equiv 2035 \times 1185^{1520} \equiv 697 \pmod{2357}$
  - The ciphertext $C = (A, B) = (1430, 697)$
- Decryption:
  1. Computes $K \equiv A^x \equiv 1430^{1751} \equiv 2084 \pmod{2357}$
  2. $M \equiv B K^{-1} \equiv 697 \times 2084^{-1} \equiv 2035 \pmod{2357}$

# Security of ElGamal Encryption Scheme

**Encryption:**

1. $r \in_R Z_{p-1}$
2. $A = g^r \bmod p$
3. $B = MY^r \bmod p$ where $M \in Z_p^*$ is the message.

Ciphertext $C = (A, B)$.

- Given $C = (A, B)$ and public key $Y = g^x \bmod p$, find M without knowing x.

1. If adversary can get r from $A=g^r \bmod p$, then the scheme is broken.
2. If adversary can get x from $Y=g^x \bmod p$, then the scheme is broken.
3. From $A=g^r \bmod p$ and $Y=g^x \bmod p$, if adversary can compute $g^{rx} \bmod p$, then the scheme is broken.

- First two correspond to DLP (Discrete Logarithm Problem)
- The last one corresponds to Diffie-Hellman Problem

## Deterministic Encryption vs. Probabilistic Encryptoin

- Deterministic Encryption
  - Encrypting same messages will generate same ciphertexts
- Probabilistic Encryption
  - Encrypting same messages will generate different ciphertexts

# Discrete Logarithm Problem (DLP)

- Let p be a prime number. Given two integers: g, y
  - g and y are integers chosen randomly in $Z_p^*$.
- Find a such that $g^a \bmod p = y$
- a is called the discrete log of y to the base g mod p.

**DLP (Discrete Log Problem)**

- Given a, g and p, compute $y \equiv g^a \bmod p$ is EASY
- However, given y, g and p, compute a is HARD

**Factoring (revisit)**

- Given p and q, compute n = pq is EASY
- However, given n, compute the prime factors p and q is HARD

DLP Example:

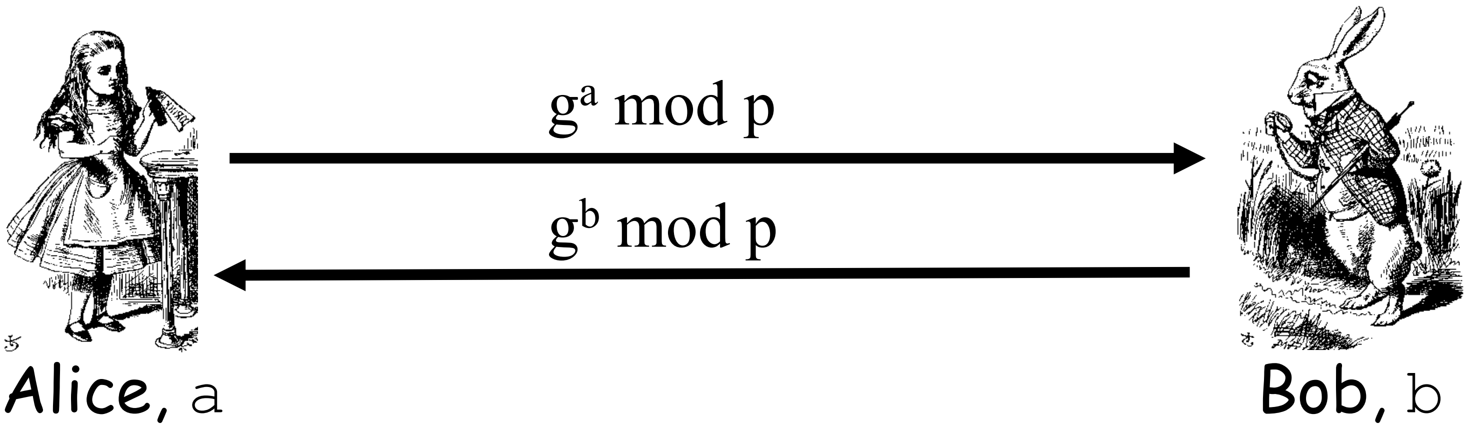
- For p=97, g = 5 and y= 35, compute a such that $g^a \bmod p = 35$.
  - We need to try all possibilities (from 1 to 96) to obtain such a

- When p is large, DLP is hard
- In practice, p should at least be 1024 bits long.
- Practical problems (not to be discussed in this course): How to generate and verify such a large prime number p? How to generate g?

# Diffie-Hellman Problem

- Given $A=g^x \bmod p$ and $B=g^y \bmod p$, find $C=g^{xy} \bmod p$.
- If DLP can be solved, then Diffie-Hellman Problem can be solved.
- Open Problem: If Diffie-Hellman Problem can be solved, can DLP be solved?

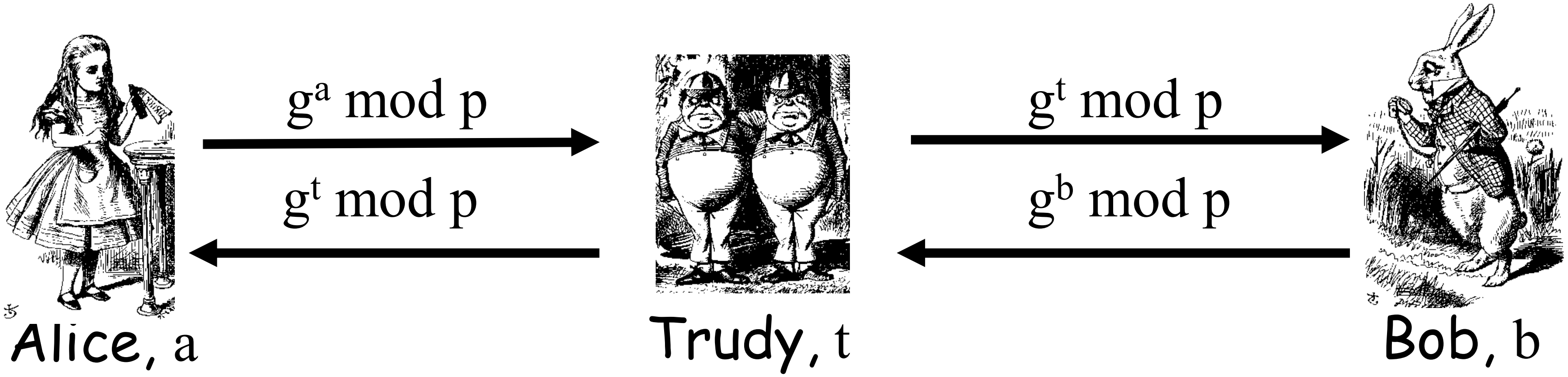
# Diffie-Hellman Key Exchange




- Alice computes $(g^b)^a = g^{ba} = g^{ab} \bmod p$
- Bob computes $(g^a)^b = g^{ab} \bmod p$
- Could use $K = g^{ab} \bmod p$ as symmetric key

- This key exchange scheme is secure against eavesdroppers if Diffie-Hellman Problem is assumed to be hard to solve.
- However, it is insecure if the attacker in the network is active: man-in-the-middle attack. "Active" means that the attacker can intercept, modify, remove or insert messages into the network.

# Man-in-the-Middle Attack (MITM)




- Trudy shares secret $g^{at}$ mod p with Alice
- Trudy shares secret $g^{bt}$ mod p with Bob
- Alice and Bob don't know Trudy exists!

# Public key vs. Symmetric key

| Symmetric key | Public key |
|---|---|
| Two parties MUST trust each other | Two parties DO NOT need to trust each other |
| Both share the same key (or one key is computable from the other) | Two separate keys: a public and a private key |
| Attack approach: bruteforce | Attack approach: solving mathematical problems (e.g. factorization, discrete log problem) |
| Faster | Slower (100-1000 times slower) |
| Smaller key size | Larger key size |
| Examples: DES, 3DES, DESX, RC6, AES, … | Examples: RSA, ElGamal, ECC,… |

# Summary

- PKE Concept
- RSA Encryption
  - RSA Assumption
  - Factoring Assumption
- ElGamal Encryption
  - DL Assumption
  - DH Assumption
- DH Key Exchange
  - MITM Attack