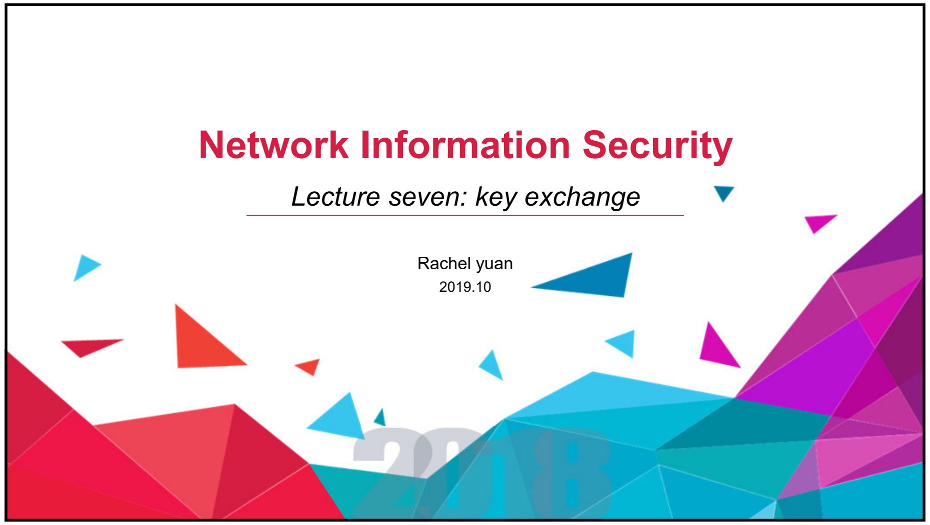


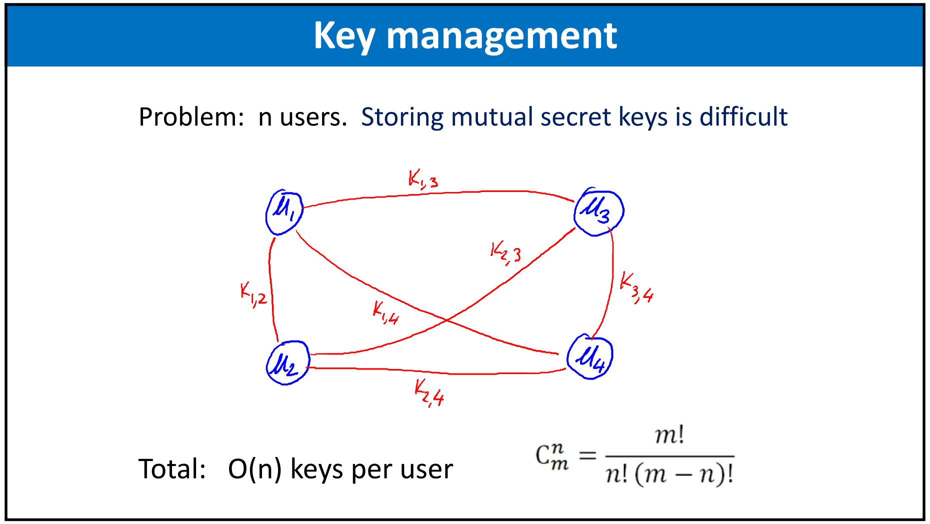
Network Information Security

Lecture seven: key exchange

Rachel yuan

2019.10

1

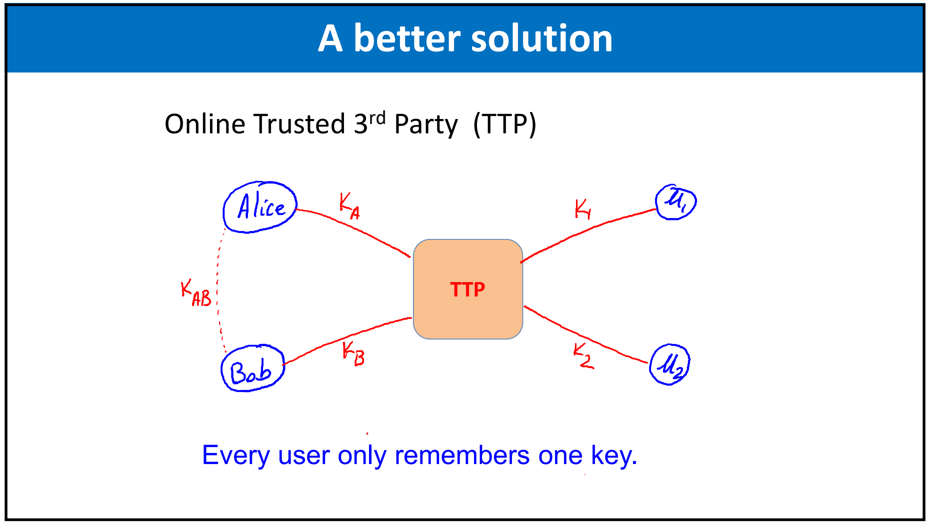


Key management

Problem: n users. Storing mutual secret keys is difficult

Total: O(n) keys per user $C_m^n = \frac{m!}{n!\,(m-n)!}$

2

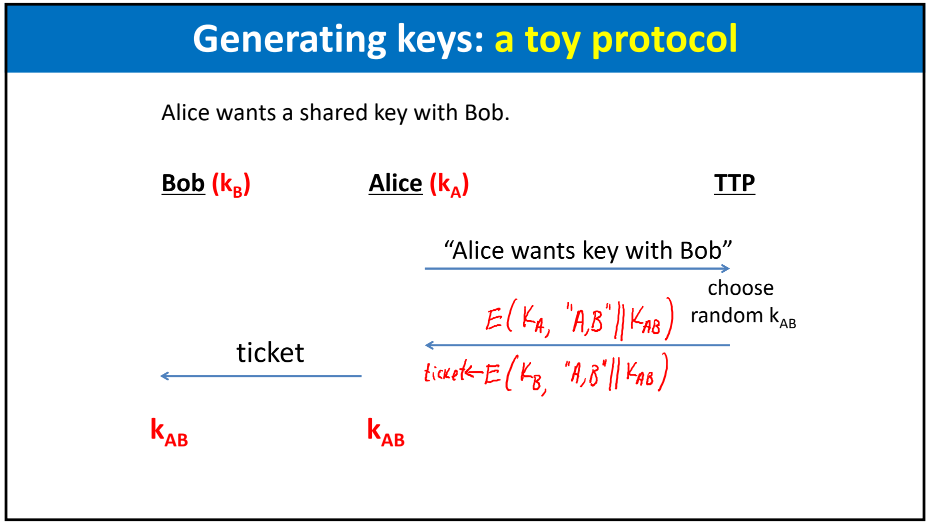


A better solution

Online Trusted 3rd Party (TTP)

Every user only remembers one key.

3



Generating keys: a toy protocol

Alice wants a shared key with Bob.

Bob (k_B) **Alice** (k_A) **TTP**

"Alice wants key with Bob" →

choose random k_{AB}

← $E(K_A, \text{"A,B"} \| K_{AB})$

$\text{ticket} \leftarrow E(K_B, \text{"A,B"} \| K_{AB})$

← ticket

k_{AB} k_{AB}

4

Generating keys: a toy protocol

Alice wants a shared key with Bob.
Eavesdropping(窃听) security only.

Eavesdropper sees: E(k_A, “A, B” ll k_{AB}) ; E(k_B, “A, B” ll k_{AB})

Eavesdropper learns nothing about k_{AB}

Note: TTP needed for every key exchange, knows all session keys.

(basis of Kerberos system)

5

Toy protocol: insecure against active attacks

Example: insecure against replay attacks (重放攻击)

Attacker records session between Alice and merchant Bob

- For example a book order

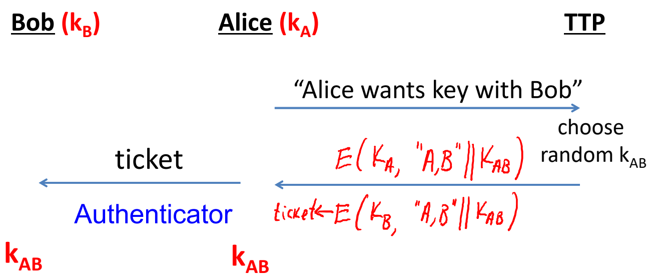
Attacker replays session to Bob

- Bob thinks Alice is ordering another copy of book

6

A bit modification to the toy protocol

Alice wants a shared key with Bob.



Bob (k_B) **Alice** (k_A) **TTP**

“Alice wants key with Bob”

choose random k_{AB}

$E(K_A, \text{"A,B"} \| K_{AB})$

ticket ← $E(K_B, \text{"A,B"} \| K_{AB})$

ticket

Authenticator

k_{AB} k_{AB}

E(K_{AB}, “alice”||timestamp)

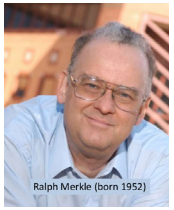
This modification is used in real protocol, such as Kerberos.

7

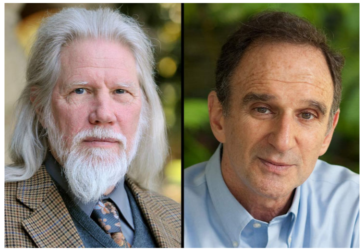
Can we generate shared keys without an **online** trusted 3rd party?

8

Merkle	D-H	RSA
1974	1976 Stanford	1977 MIT



Ralph Merkle (born 1952)

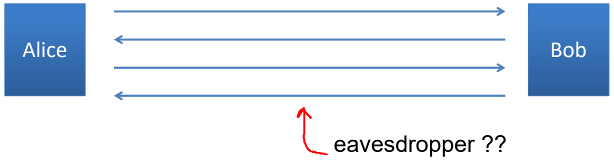


9

Key exchange without an online TTP?

Goal: Alice and Bob want shared key, unknown to eavesdropper

- For now: security against eavesdropping only (no tampering)



Can this be done using generic symmetric crypto?

10

The Diffie-hellman protocol

11

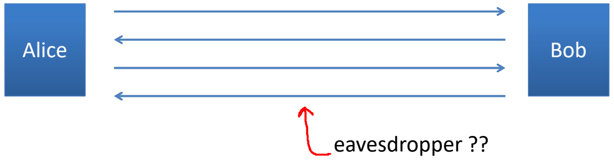
- In 2002, Hellman suggested the algorithm be called **Diffie–Hellman–Merkle key exchange**
- The system...has since become known as Diffie–Hellman key exchange. While that system was first described in a paper by Diffie and me, it is a public key distribution system, a concept developed by Merkle, and hence should be called 'Diffie–Hellman–Merkle key exchange' if names are to be associated with it. I hope this small pulpit might help in that endeavor to recognize Merkle's equal contribution to the invention of public key cryptography

12

Key exchange without an online TTP?

Goal: Alice and Bob want shared secret, unknown to eavesdropper

- For now: security against eavesdropping only (no tampering)



13

Wrap up

- Primitive root （原根）

For a prime p, exist a number g (1<=g<=p), if g mod p, g^2 mod p, …, g^(p-1) mod p, are a permutation of 1 to p-1, then g is a primitive root of prime p.

- Discrete logarithm (离散对数)

a = g^i mod p (0 <= i <= p-1) , i is called the index or discrete logarithm of a to the base g modulo p

- One way function

y = f(x), x → y is easy, and y → x is very hard.

14

The number 3 is a primitive root modulo 7[1] because

$$\begin{array}{rcrcrcrcrcl}
3^1 & = & 3 & = & 3^0 \times 3 & \equiv & 1 \times 3 & = & 3 & \equiv & 3 \pmod 7 \\
3^2 & = & 9 & = & 3^1 \times 3 & \equiv & 3 \times 3 & = & 9 & \equiv & 2 \pmod 7 \\
3^3 & = & 27 & = & 3^2 \times 3 & \equiv & 2 \times 3 & = & 6 & \equiv & 6 \pmod 7 \\
3^4 & = & 81 & = & 3^3 \times 3 & \equiv & 6 \times 3 & = & 18 & \equiv & 4 \pmod 7 \\
3^5 & = & 243 & = & 3^4 \times 3 & \equiv & 4 \times 3 & = & 12 & \equiv & 5 \pmod 7 \\
3^6 & = & 729 & = & 3^5 \times 3 & \equiv & 5 \times 3 & = & 15 & \equiv & 1 \pmod 7 \\
3^7 & = & 2187 & = & 3^6 \times 3 & \equiv & 1 \times 3 & = & 3 & \equiv & 3 \pmod 7
\end{array}$$

15

The Diffie-Hellman protocol

Fix a large prime p (e.g. 600 digits)

Fix an integer g in {1, …, p}

Alice **Bob**

choose random **a** in {1,…,p-1} choose random **b** in {1,…,p-1}

"Alice", $A \leftarrow g^a \pmod p$ →

← "Bob", $B \leftarrow g^b \pmod p$

$\mathbf{B^a}$ (mod p) = $(g^b)^a$ = $\mathbf{k_{AB} = g^{ab}}$ (mod p) = $(g^a)^b$ = $\mathbf{A^b}$ (mod p)

16

Security

Eavesdropper sees: p, g, A=g^a (mod p), and B=g^b (mod p)

Can she compute g^{ab} (mod p) ??

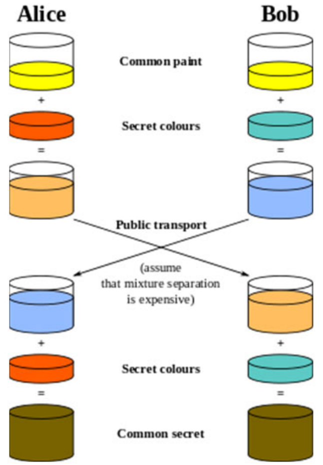
$\exp(\ \tilde{O}(\sqrt[3]{n})\ )$

More generally, if there is an ***exponential gap*** between users and attacker, the algorithm is secure.

17

if *p* is a prime of at least 600 digits, then even the fastest modern computers cannot find *a* given only *g*, *p* and g^a mod *p*. Such a problem is called the **discrete logarithm problem**

18



19

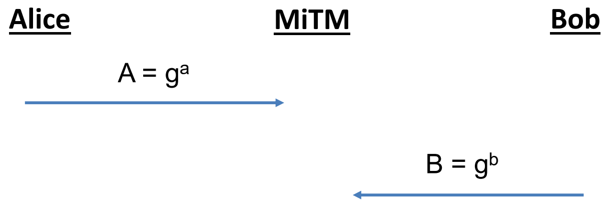
D-H example

- Alice and Bob agree to use a modulus *p* = 23 and base *g* = 5 (which is a primitive root modulo 23).
- Alice chooses a secret integer ***a* = 6**, then sends Bob *A* = g^a mod *p*
  - *A* = 5^6 mod 23 = 8
- Bob chooses a secret integer ***b* = 15**, then sends Alice *B* = g^b mod *p*
  - *B* = 5^{15} mod 23 = 19
- Alice computes ***s*** = B^a mod *p*
  - ***s*** = 19^6 mod 23 = **2**
- Bob computes ***s*** = A^b mod *p*
  - ***s*** = 8^{15} mod 23 = **2**
- Alice and Bob now share a secret (the number **2**).

20

Insecure against man-in-the-middle

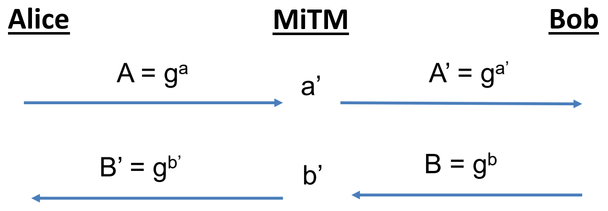
the protocol is insecure against **active** attacks



21

Insecure against man-in-the-middle

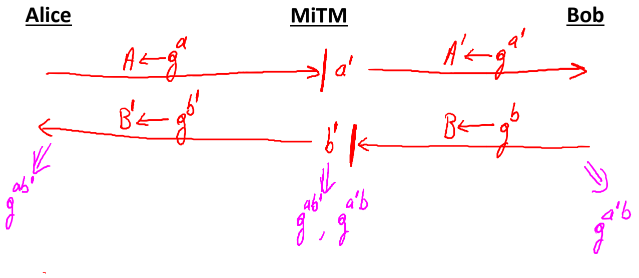
the protocol is insecure against **active** attacks



22

Insecure against man-in-the-middle

the protocol is insecure against **active** attacks



23