Key Exchange

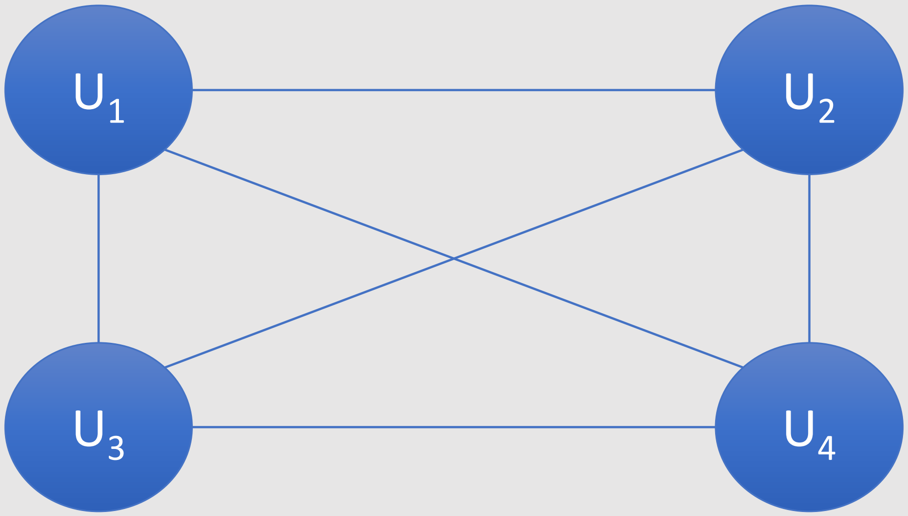
Outline

- Trusted Third Parties
- Merkle Puzzles
- The Diffie-Hellman Protocol

Trusted Third Parties

Key Management

Problem: **n** users. Storing mutual secret keys is difficult

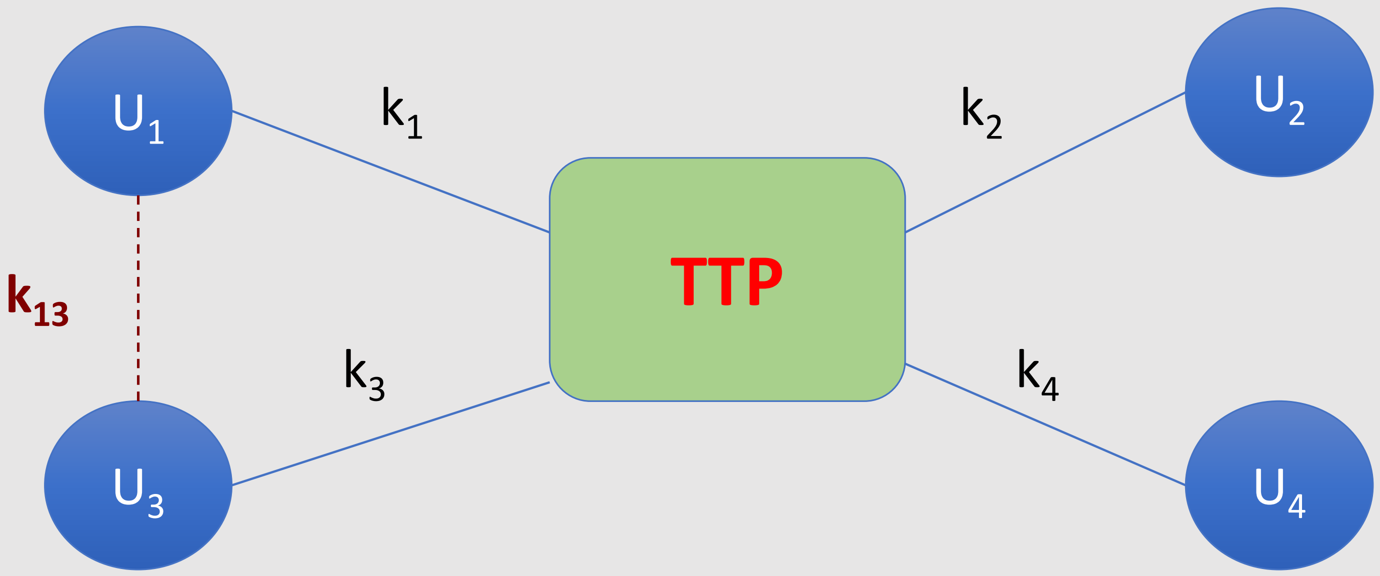


O(n) keys per user

O(n^2) keys in total

A Better Solution

Online **Trusted Third Party** (TTP)



Every user only remembers **ONE key**

Generating keys: A toy protocol

Alice wants a shared key with Bob. **Eavesdropping** security **only**.

<u>Bob</u> **(k_B)** | **<u>Alice</u>** **(k_A)** | **<u>TTP</u>**

"Alice wants key with Bob"

choose random k_{AB}

$E(k_A,$ "Alice, Bob" $\| \ k_{AB})$;

ticket = $E(k_B,$ "Alice, Bob" $\| \ k_{AB})$

ticket

$\mathbf{k_{AB}}$ $\mathbf{k_{AB}}$

$\boxed{\text{(E,D) a CPA-secure cipher}}$

Generating keys: A toy protocol

Alice wants a shared key with Bob. Eavesdropping security only.

Eavesdropper sees: $E(k_A,\ \text{“A, B”} \parallel k_{AB})$; $E(k_B,\ \text{“A, B”} \parallel k_{AB})$

(E,D) is CPA-secure $\Rightarrow$ eavesdropper learns nothing about k_{AB}

Note: **TTP needed for every key exchange, knows all session keys (k_{AB}).**

(basis of Kerberos system)

Key Question

Can we generate shared keys **without** an **online TTP**?

Answer: **yes!**

Starting point of **public-key cryptography**:

- Merkle (1974),
- Diffie-Hellman (1976),
- RSA (1977)
- …

Merkle Puzzles

Key exchange without an online TTP?

- Goal: Alice and Bob want a shared key, unknown to eavesdropper
- Security against **eavesdropping only** (**no tampering, no message injection**)

- **Can this be done using generic symmetric crypto?**

Merkle Puzzles (1974)

Answer: yes, but **very inefficient**

Main tool: **"puzzles"**

- Puzzles: Problems that can be solved with "some effort"
- Example:
  - $E(k,m)$ a symmetric cipher with $k \in \{0,1\}^{128}$
  - **puzzle = E(P, "message")** where **$P = 0^{96} \,||\, b_1 \ldots b_{32}$**
  - To "solve" a puzzle, find **P** by trying all $\mathbf{2^{32}}$ possibilities

Merkle Puzzles

Alice:

- Prepare **2^{32} puzzles:**
  - For $i = 1, \ldots, 2^{32}$ choose random $\mathbf{P_i} \in \mathbf{\{0,1\}^{32}}$ and random $\mathbf{x_i}, \mathbf{k_i} \in \mathbf{\{0,1\}^{128}}$ $\quad \mathbf{x_i \neq x_j}$

    Set $\mathbf{puzzle_i} \longleftarrow \mathbf{E(\ 0^{96} \parallel P_i,\ \text{"Puzzle \# "} \parallel x_i \parallel k_i\ )}$
  - Send $\mathbf{puzzle_1, \ldots, puzzle_{2^\wedge 32}}$ to Bob.

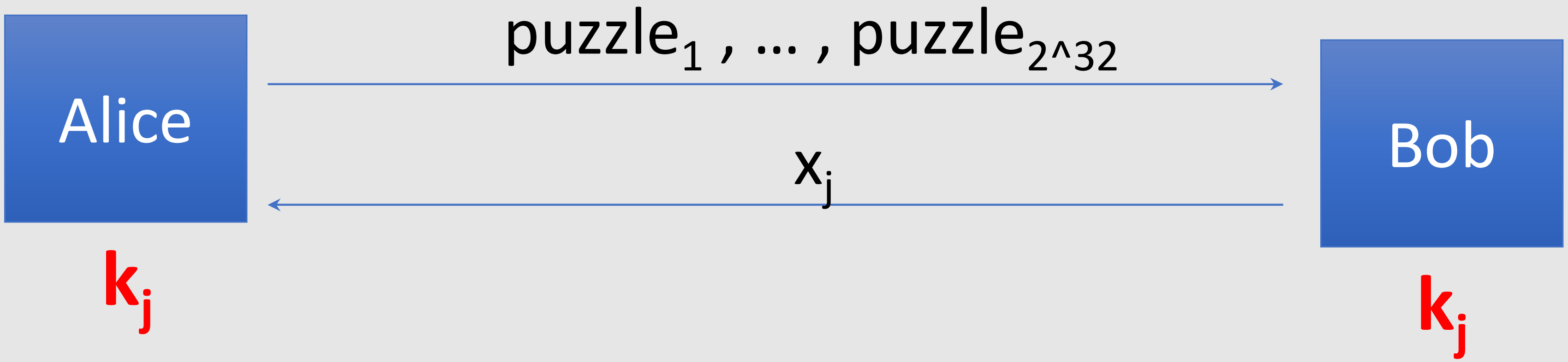
Bob:

- Choose a **random** $\mathbf{puzzle_j}$ and **solve** it by *brute-force*.
- Obtain $\mathbf{(x_j, k_j)}$ and use $\mathbf{k_j}$ as shared secret.
- Send $\mathbf{x_j}$ to Alice.

Alice:

- Lookup puzzle with number $\mathbf{x_j}$.
- Use $\mathbf{k_j}$ as shared secret.

In a figure



Alice's work: **$O(2^{32})$** (prepare 2^{32} puzzles) in general **$O(n)$**

Bob's work: **$O(2^{32})$** (solve **one** puzzle) in general **$O(n)$**

Eavesdropper's work: **$O(2^{64})$** (solve **2^{32}** puzzles) in general **$O(n^2)$**

The eavesdropper didn't know which puzzle has been chosen by Bob

Impossibility Result

Can we achieve a better gap using a general symmetric cipher?

Answer: **unknown**

The Diffie-Hellman Protocol

Key exchange without an online TTP?

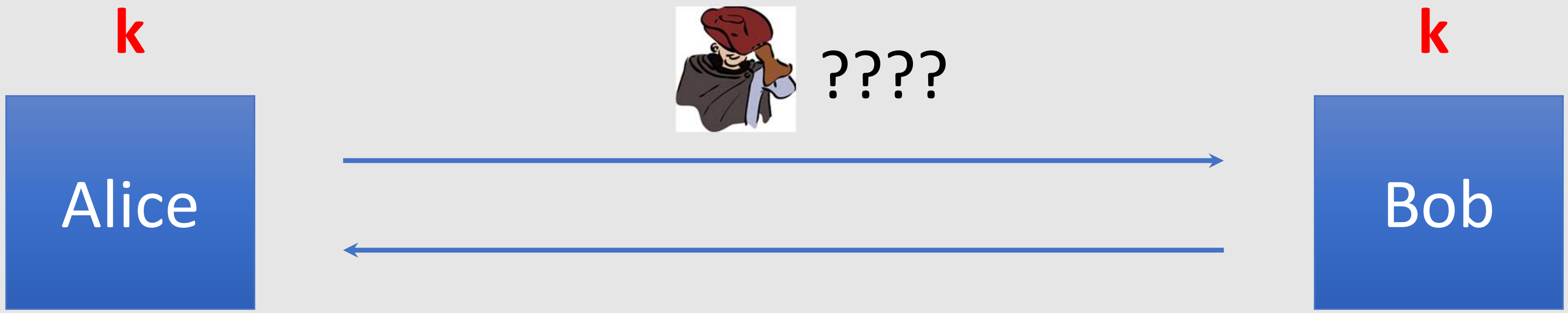
- Goal: Alice and Bob want a shared key, unknown to eavesdropper
- Security against **eavesdropping only** (**no tampering**)

- **Can this be done with an exponential gap?**

The Diffie-Hellman Protocol

High-level idea:

- Alice and Bob **do NOT share any secret information beforehand**
- Alice and Bob exchange messages
- After that, Alice and Bob have agreed on a shared secret key **k**
- **k** unknown to eavesdropper



The Diffie-Hellman Protocol

(Security) Based on the **Discrete Logarithm** Problem:

Given

- **g**
- **p**
- **g^k mod p**

Find k

The Diffie-Hellman Protocol

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

Fix an integer **g** in {2, …, p-2}

Alice

Bob

Choose random **a** in {1,…,p-2}

$\mathbf{g^a \bmod p}$ →

Choose random **b** in {1,…,p-2}

← $\mathbf{g^b \bmod p}$

Alice computes $\mathbf{(g^b)^a \bmod p}$ = $\mathbf{g^{ab} \bmod p}$ = Bob computes $\mathbf{(g^a)^b \bmod p}$

Alice and Bob now share **SECRET KEY** $\mathbf{g^{ab} \bmod p}$

Security

Eavesdropper sees: **p**, **g**, **g^a mod p**, and **g^b mod p**

Can she compute **g^{ab} mod p** ??

How hard is the DH function mod p?

Suppose prime **p** is **n** bits long.

Best known algorithm (*General number field sieve: GNFS*): run time is exponential in **n**: exp($\tilde{O}(\sqrt[3]{n})$)

Insecure against Man-in-The-Middle (MiTM)

- As described, the protocol is insecure against **active** attacks

Alice	**MiTM**	**Bob**
$A = g^a \bmod p$ →	a'	$A' = g^{a'} \bmod p$ →
← $B' = g^{b'} \bmod p$	b'	← $B = g^b \bmod p$
$g^{ab'} \bmod p$	$g^{ab'}, g^{a'b} \bmod p$	$g^{a'b} \bmod p$

- Attacker relays traffic from Alice to Bob and reads it in clear