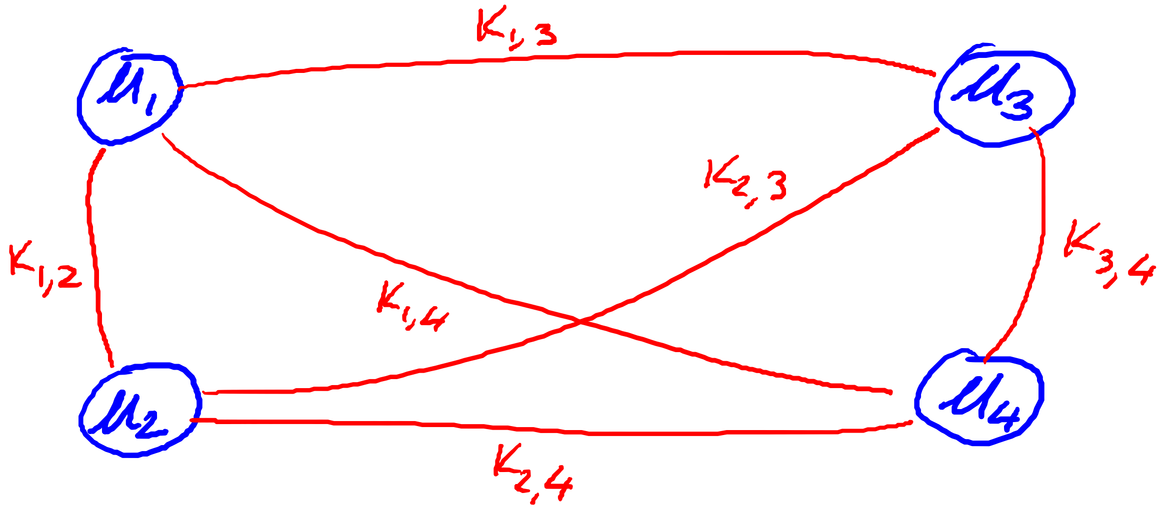
Basic key exchange

Basic key exchange

Trusted 3rd parties

Key management

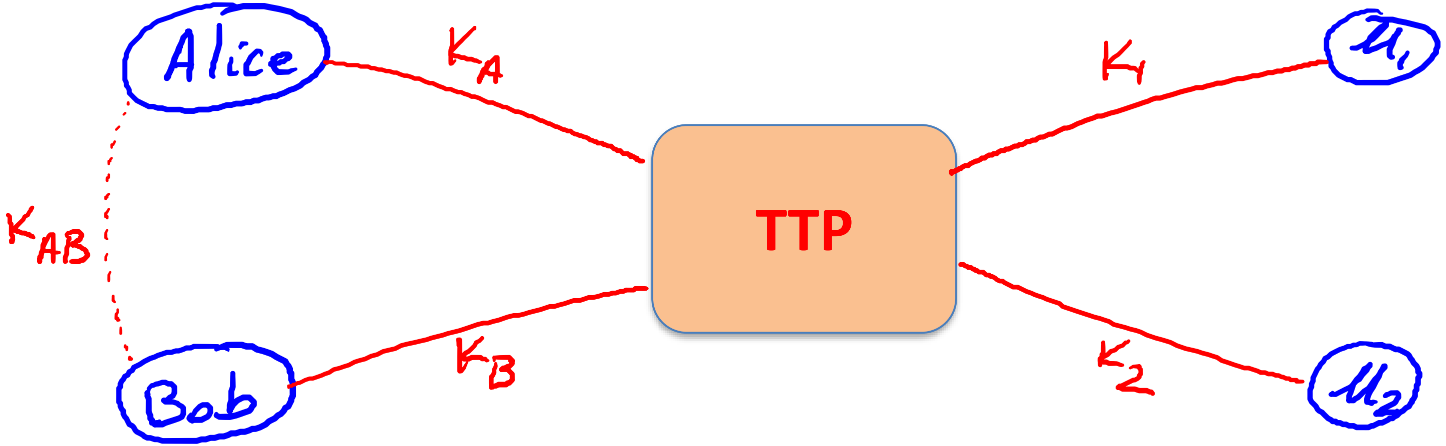
Problem: n users. Storing mutual secret keys is difficult



Total: O(n) keys per user

A better solution

Online Trusted 3rd Party (TTP)



Every user only remembers one key.

Generating keys: a toy protocol

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

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 $\leftarrow E(k_B, \text{"A,B"} \| k_{AB})$

ticket

k_{AB} k_{AB}

(E,D) a CPA-secure cipher

Generating keys: a toy protocol

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

Eavesdropper sees: $E(k_A,$ "A, B" $\| \; k_{AB})$; $E(k_B,$ "A, B" $\| \; k_{AB})$

(E,D) is CPA-secure $\Rightarrow$

eavesdropper learns nothing about k_{AB}

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

(basis of Kerberos system)

Toy protocol: insecure against active attacks

Example: insecure against replay attacks

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

Key question

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

Answer: yes!

Starting point of public-key cryptography:

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

- More recently: ID-based enc. (BF 2001), Functional enc. (BSW 2011)

End of Segment

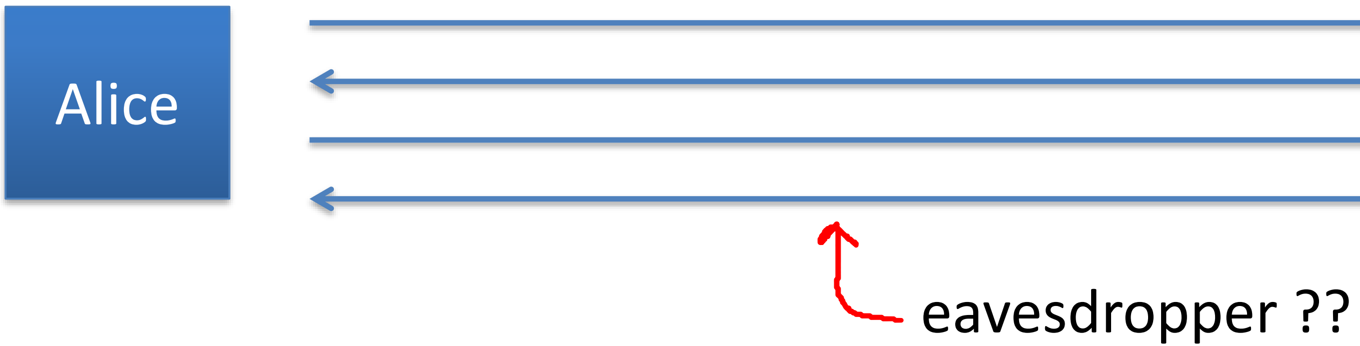
Basic key exchange

Merkle Puzzles

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?

Merkle Puzzles (1974)

Answer: yes, but very inefficient

Main tool: puzzles

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

Merkle puzzles

Alice: prepare 2^{32} puzzles

- For $i=1, \ldots, 2^{32}$ choose random $\mathbf{P_i} \in \{0,1\}^{32}$ and $\mathbf{x_i}, \mathbf{k_i} \in \{0,1\}^{128}$

  set puzzle$_i$ $\longleftarrow$ $E(\,0^{96} \,\|\, \mathbf{P_i}$, **"Puzzle # $\mathbf{x_i}$" $\|$ $\mathbf{k_i}$** $)$

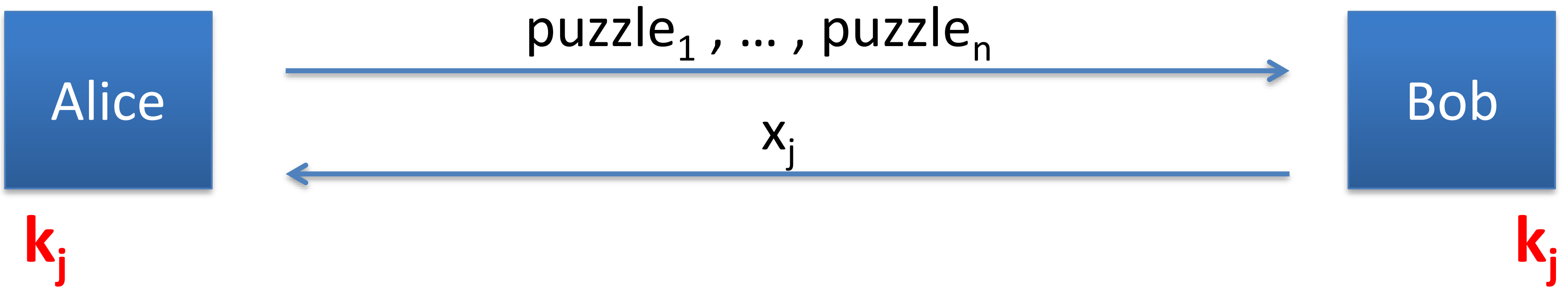
- Send $\text{puzzle}_1, \ldots, \text{puzzle}_{2^{32}}$ to Bob

Bob: choose a random puzzle$_j$ and solve it. Obtain $(\,x_j, k_j\,)$.

- Send x_j to Alice

Alice: lookup puzzle with number x_j. Use k_j as shared secret

In a figure



Alice's work: O(n) (prepare n puzzles)

Bob's work: O(n) (solve one puzzle)

Eavesdropper's work: $O(n^2)$ (e.g. 2^{64} time)

Impossibility Result

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

Answer: unknown

But: roughly speaking,

quadratic gap is best possible if we treat cipher as a black box oracle [IR'89, BM'09]

End of Segment

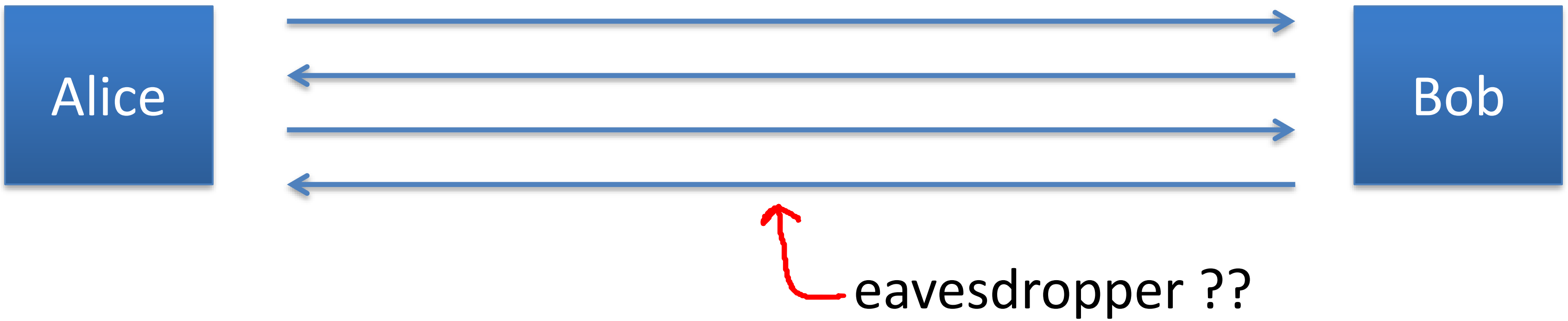
Basic key exchange

The Diffie-Hellman protocol

Key exchange without an online TTP?

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

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



Can this be done with an exponential gap?

The Diffie-Hellman protocol (informally)

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

Fix an integer g in $\{1, \ldots, p\}$

Alice **Bob**

choose random **a** in $\{1,\ldots,p-1\}$ choose random **b** in $\{1,\ldots,p-1\}$

"Alice", $A \leftarrow g^a \pmod{p}$ →

← "Bob", $B \leftarrow g^b \pmod{p}$

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

Security (much more on this later)

Eavesdropper sees: p, g, $A=g^a \pmod p$, and $B=g^b \pmod p$

Can she compute $g^{ab} \pmod p$??

More generally: define $DH_g(g^a, g^b) = g^{ab} \pmod p$

How hard is the DH function mod p?

How hard is the DH function mod p?

Suppose prime p is n bits long.

Best known algorithm (GNFS): run time exp($\tilde{O}(\sqrt[3]{n})$)

cipher key size	modulus size	Elliptic Curve size
80 bits	1024 bits	160 bits
128 bits	3072 bits	256 bits
256 bits (AES)	**15360** bits	512 bits

As a result: slow transition away from (mod p) to elliptic curves

Elliptic curve
Diffie-Hellman

Insecure against man-in-the-middle

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

Alice | **MiTM** | **Bob**

$A \leftarrow g^{a}$ $\longrightarrow$ | a' | $A' \leftarrow g^{a'}$ $\longrightarrow$

$\longleftarrow$ $B' \leftarrow g^{b'}$ | b' | $\longleftarrow$ $B \leftarrow g^{b}$

$g^{ab'}$ | $g^{ab'}, g^{a'b}$ | $g^{a'b}$

attacker relays traffic from Alice to Bob and reads it in the clear

Another look at DH

Facebook

g^a g^b g^c g^d

<u>Alice</u> **<u>Bob</u>** **<u>Charlie</u>** **<u>David</u>** …

a b c d

$K_{AC}=g^{ac}$ $K_{AC}=g^{ac}$

An open problem

$n=2$: DH
$n=3$: Known (Joux)
$n \geq 4$: open

Facebook

g^a g^b g^c g^d

Alice	**Bob**	**Charlie**	**David**	…
a	b	c	d	
K_{ABCD}	K_{ABCD}	K_{ABCD}	K_{ABCD}	

End of Segment