CIS 5560

Cryptography
Lecture 13

Course website:

pratyushmishra.com/classes/cis-5560-s24/

Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Announcements

- **Final Exam May 10, 2024, 9-11AM, DRLB A2**
- **HW6 is out, due 3/12 at 1PM**
- **Midterm coming up: 3/14 in class**
  - 80 minutes long, starts at 1:47PM
  - We will provide a cheat sheet with all the information (definitions, proof strategies, etc) you will need
  - 3/12 will be a review session in class.

Recap of Last Lecture

- Number Theory refresher
  - Arithmetic modulo primes
  - Fermat's Little Theorem
  - Cyclic groups
  - Discrete Logarithms

The Multiplicative Group $\mathbb{Z}_p^*$

$\mathbb{Z}_p^*$: $(\{1,\ldots,\mathrm{p}-1\},$ group operation: • mod p)

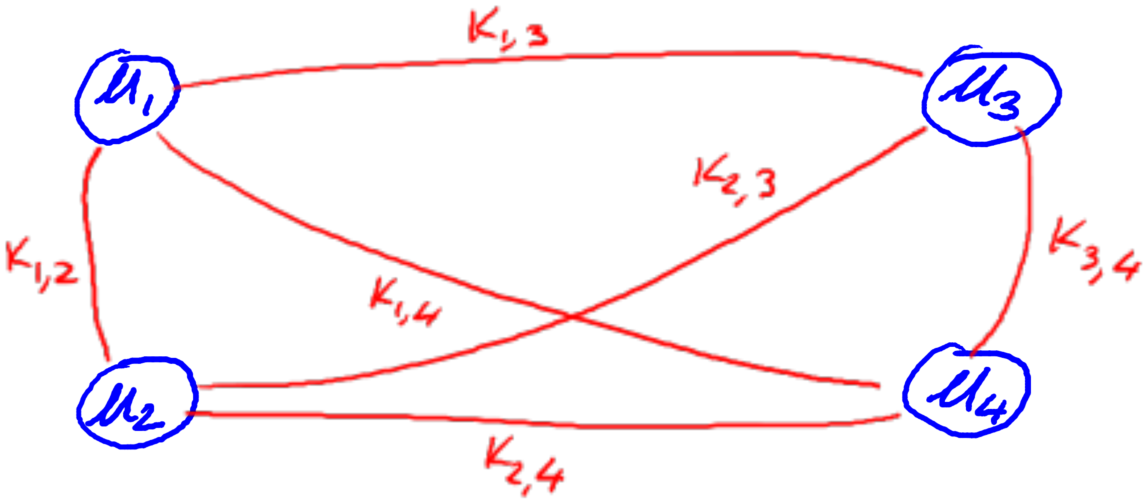
- Computing the group operation is easy.
- Computing inverses is easy: Extended Euclid.
- Exponentiation (given $g \in \mathbb{Z}_p^*$ and $x \in \mathbb{Z}_{p-1}$, find g^x mod p) is easy: **Repeated Squaring Algorithm.**
-
- The discrete logarithm problem (given a generator g and $h \in \mathbb{Z}_p^*$, find $x \in \mathbb{Z}_{p-1}$ s.t. $h = g^x$ mod p) is **hard**, to the best of our knowledge!

Today's Lecture

- Key Exchange
  - Merkle puzzles
  - Diffie—Hellman
    - Computational Diffie—Hellman Problem

Key management

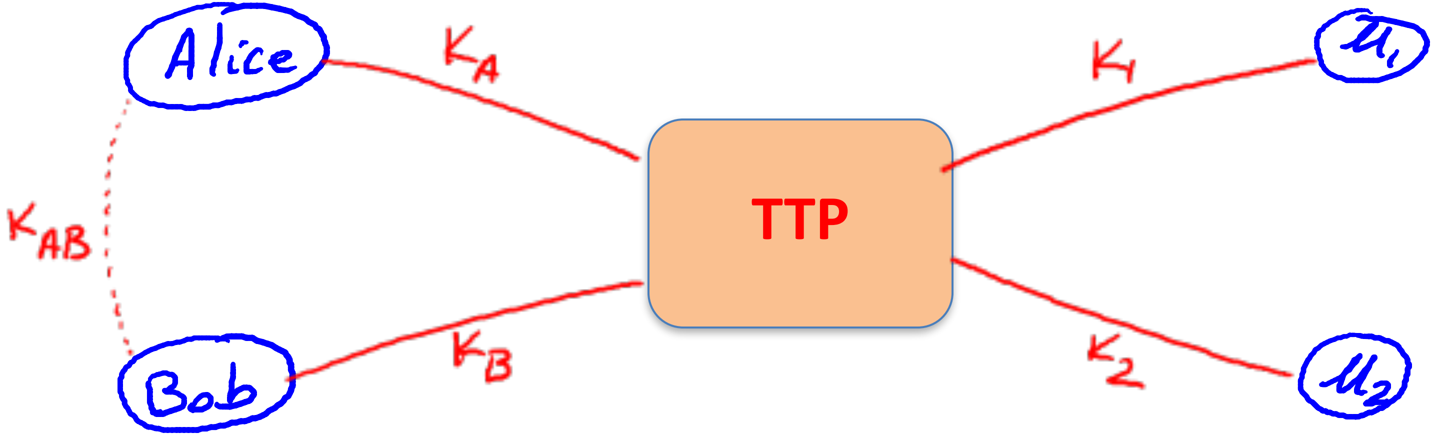
Problem: n users. Storing mutual secret keys is difficult



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 → TTP: "Alice wants key with Bob"

TTP: choose random k_{AB}

TTP → Alice: $E(k_A, \text{"A,B"} \| k_{AB})$

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

Alice → Bob: 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" ll $k_{AB})$; $E(k_B,$ "A, B" ll $k_{AB})$

(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 3rd party?

Answer: yes!

Starting point of public-key cryptography:

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

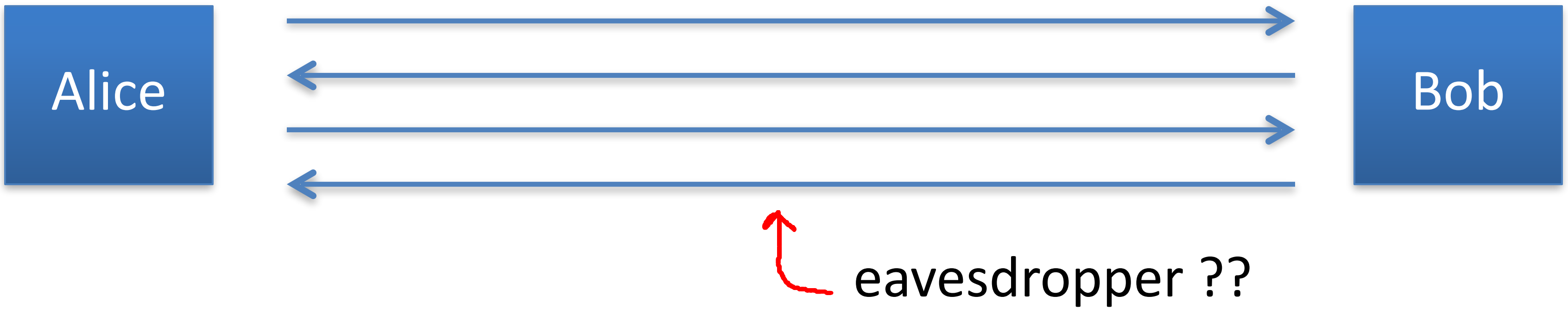
- More recently: ID-based enc. (BF 2001), Functional enc. (BSW 2011)

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 $\mathbf{P_i} \in \mathbf{\{0,1\}^{32}}$ and $\mathbf{x_i, k_i} \in \mathbf{\{0,1\}^{128}}$

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

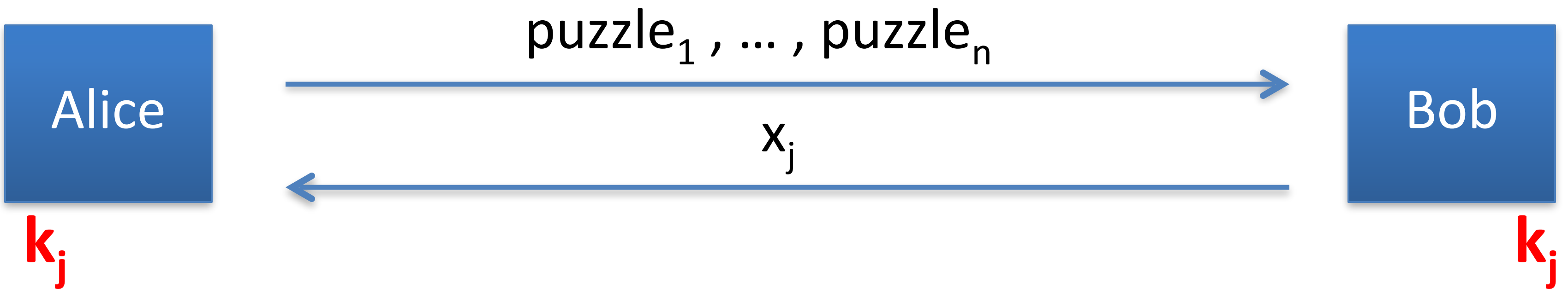
- Send $\text{puzzle}_1, \ldots, \text{puzzle}_{2^{32}}$ to Bob

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]

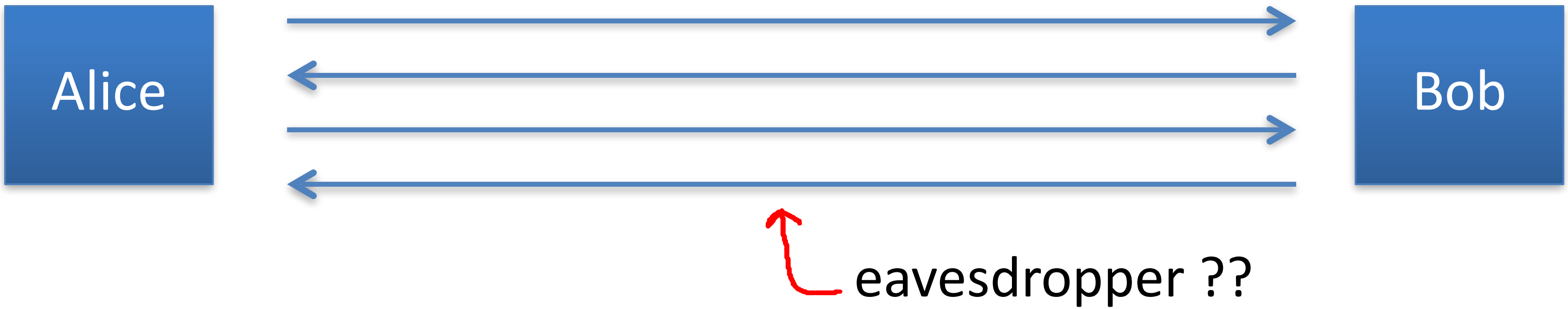
Better key exchange:

Diffie—Hellman

Key exchange without an online TTP?

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

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



eavesdropper ??

Can this be done with an exponential gap?

The Diffie-Hellman protocol (informally)

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

Fix generator g of $\mathbb{Z}_p^*$

Alice **Bob**

choose random **a** in {1,…,p-1}

choose random **b** in {1,…,p-1}

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

$\longrightarrow$

"Bob", $B \leftarrow g^b \pmod p$

$\longleftarrow$

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

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

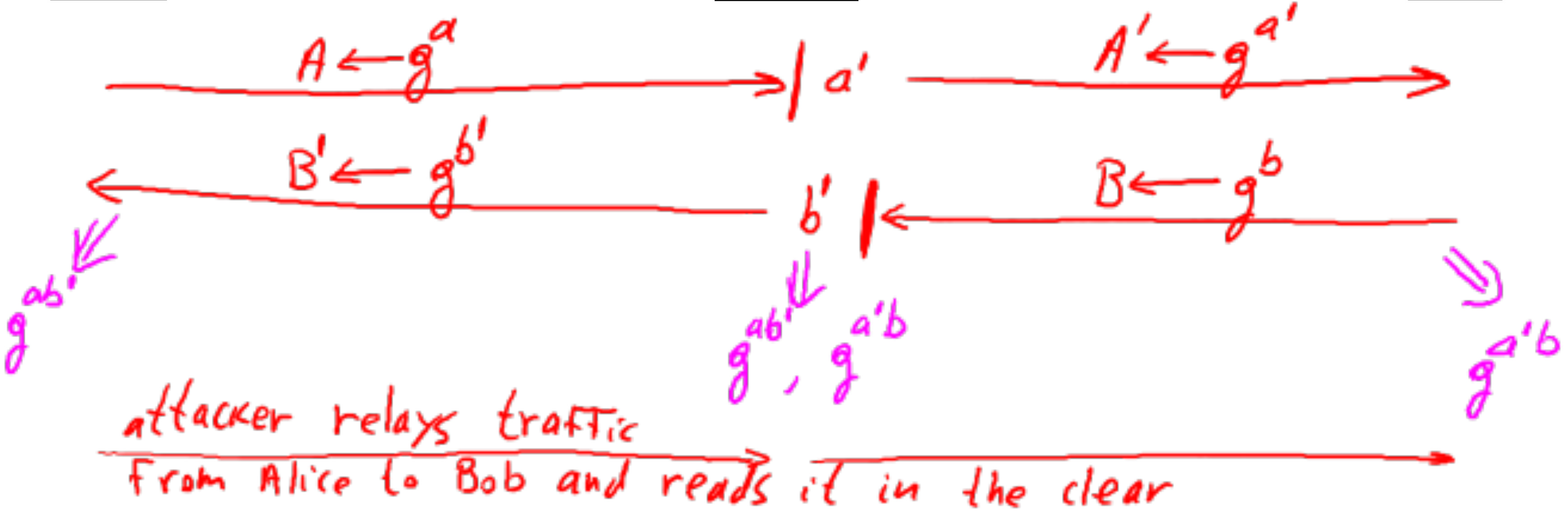
As a result: slow transition away from (mod p) to elliptic curves

Elliptic curve
Diffie-Hellman

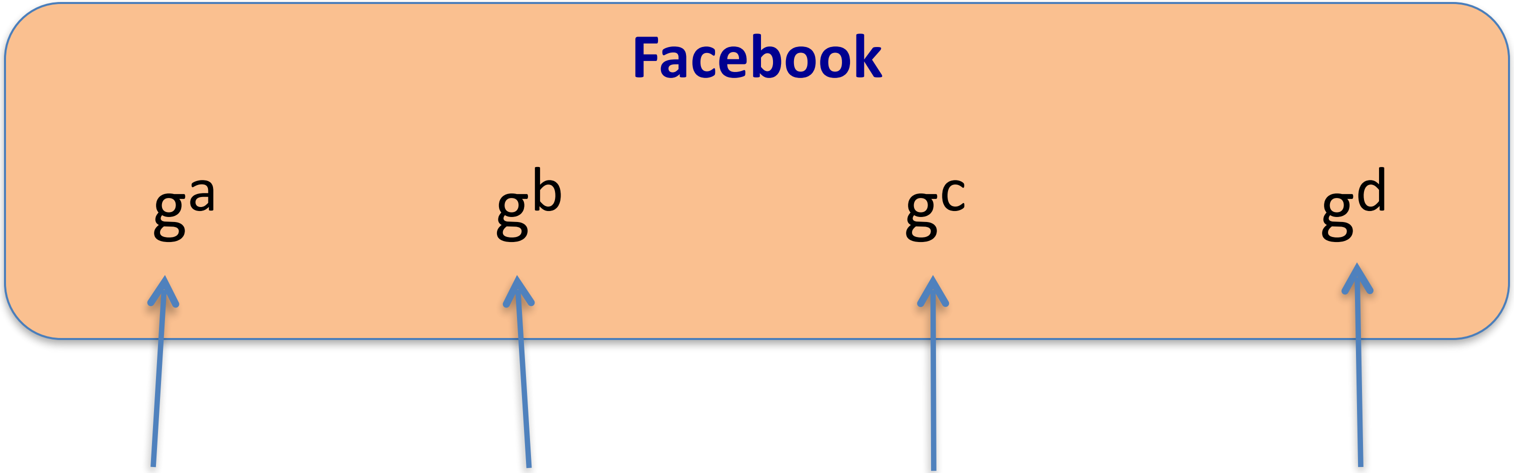
Security against man-in-the-middle?

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

Alice **MiTM** **Bob**



Another look at DH



Alice	**Bob**	**Charlie**	**David**	
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	
$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	
Alice	**Bob**	**Charlie**	**David**	
a	b	c	d	• • •
K_{ABCD}	K_{ABCD}	K_{ABCD}	K_{ABCD}	

Computational Diffie-Hellman (CDH) Assumption

W.r.t. a random prime: for every p.p.t. algorithm A, there is a negligible function μ s.t.

$$\Pr\left[\begin{array}{l} p \leftarrow PRIMES_n; g \leftarrow GEN\left(\mathbb{Z}_p^*\right); \\ x, y \leftarrow \mathbb{Z}_{p-1}:\ A\left(p, g, g^x, g^y\right) = g^{xy} \end{array}\right] = \mu(n)$$

$$\begin{tikzcd} \textbf{CDH} \arrow[r, bend left] & \textbf{DLOG} \arrow[l, bend left, dotted, "\textbf{OPEN}"] \end{tikzcd}$$

Further readings

- Merkle Puzzles are Optimal,
  B. Barak, M. Mahmoody-Ghidary, Crypto ’09

- On formal models of key exchange (sections 7-9)
  V. Shoup, 1999

DLOG: more generally

Let $\mathbb{G}$ be a finite cyclic group and g a generator of $\mathbb{G}$

$$\mathbb{G} = \{ 1, g, g^2, g^3, \ \dots \ , g^{q-1} \}$$ (q is called the order of G)

<u>**Def**</u>: We say that **DLOG is hard in G** if for all efficient alg. A:

$$\Pr_{g \leftarrow G,\, x \leftarrow Z_q} \left[A(G, q, g, g^x) = x \right] < \text{negligible}$$

Example candidates:

(1) $(Z_p)^*$ for large p, (2) Elliptic curve groups mod p

Computing Dlog in $(Z_p)^*$ (n-bit prime p)

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

cipher key size	modulus size	Elliptic Curve group 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