Key Establishment

Daniel Zappala

CS 465 · Introduction to Security and Privacy · Fall 2026

BYU



01

Key establishment



How do Alice and Bob end up with a key only they share?

01

Key establishment

How Alice and Bob can share a symmetric key

In person

Copy the key to a USB drive and hand it over

Catch: a meeting for every pair of people

Key transport

Alice picks the key and encrypts it under Bob's public key

Catch: Bob's private key guards every session

Key agreement

Each side contributes a value and both derive the same key

Upside: neither side chooses the key alone, the key is unique for each session

01

Key establishment

Sending a key under Bob’s public key fails once his private key is stolen

Today

Alice → Bob: $E_B(K)$
then messages under K

Eve records all of it

Months later

Eve steals Bob’s private key

Then

Eve decrypts $E_B(K)$ to get K and reads every recorded message

Replay

Eve resends an old $E_B(K)$ to push Bob back onto an old key

No sender

Anyone can encrypt under Bob’s public key, so $E_B(K)$ doesn’t show it came from Alice

01

Key establishment

Proving you hold a shared key needs a fresh value, or the proof can be replayed

Replay: send H(S) to prove you know S

1. Alice → Bob: H(S)
2. Eve records H(S)
3. Later, Eve → Bob: H(S)

Bob accepts, and Eve never learned S

Fix: make every proof fresh

Random number (nonce)

Bob sends a new challenge; Alice answers with a function of it and S

Sequence number

Bob rejects any number he has already seen

Timestamp

Bob rejects anything outside a short window; clocks must agree

01

Key establishment

Key agreement uses a short-lived key, so a stolen long-term key exposes no past sessions

	Key transport (RSA)	Key agreement
Who picks the key	Alice alone	Both sides contribute
How long the key lasts	As long as Bob's key pair	One session, then destroyed
Bob's private key leaks later	**Every recorded session opens**	**Recorded sessions stay secret**

Ephemeral key: used for one session and destroyed afterward, so no long-term secret can unlock it later

02

Diffie-Hellman

Really cool idea – create a shared secret key by exchanging information in public

Almost seems like this should be impossible

02

Diffie-Hellman

Everyone knows p and g; only a and b stay private

Public, fixed for all users

p

A huge prime

$p - 1$ needs at least one very large prime factor

g

A generator mod p (a primitive root)

Every value from 1 to $p - 1$ equals g^k mod p for some k

> **Why:** g is a generator, so g^1, g^2, ..., g^{p-1} mod p hits every value from 1 to $p - 1$ exactly once

Private, fresh for each run

a, b

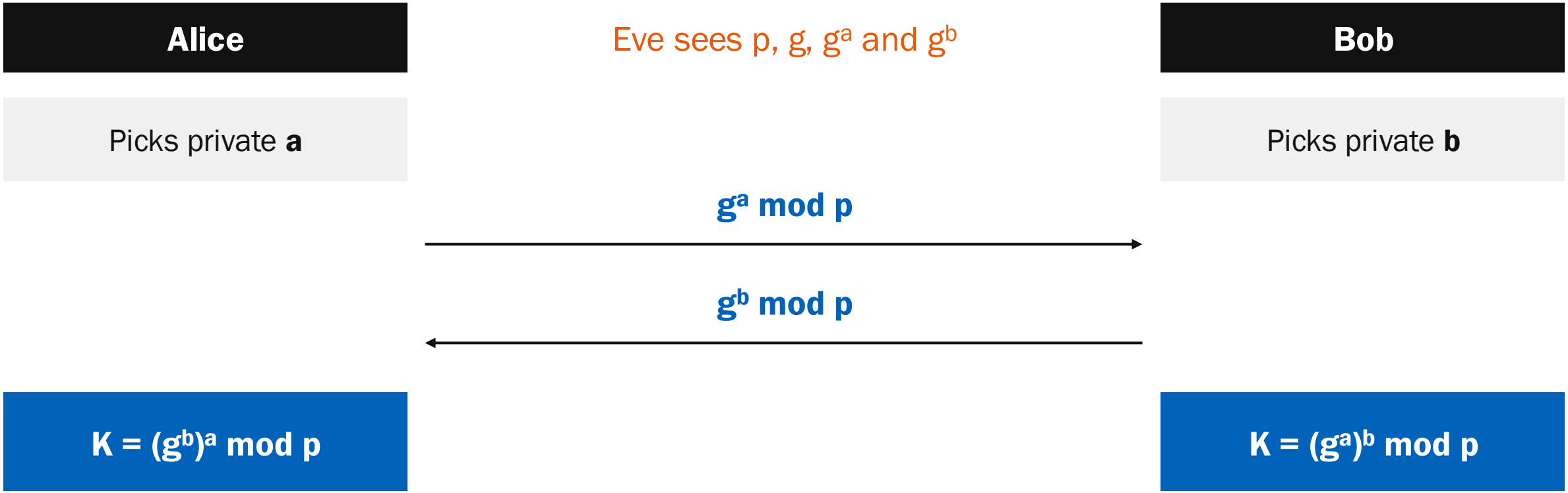
Alice picks a and Bob picks b, random in $[1, p - 2]$

Neither value is ever sent

02

Diffie-Hellman

Alice and Bob each send one number and both compute the same key, $g^{ab} \bmod p$



Both equal $g^{ab} \bmod p$, because $(g^a)^b = (g^b)^a$

02

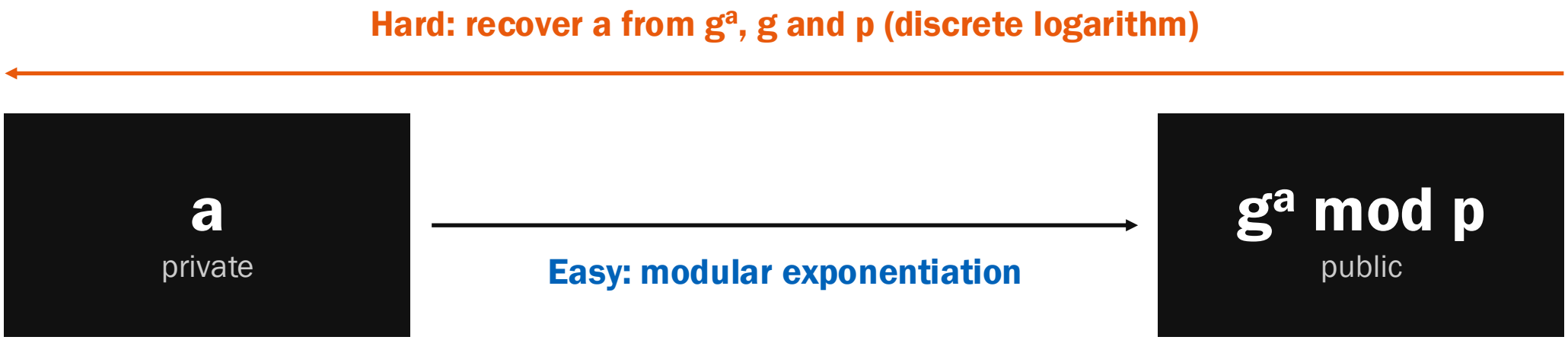
Diffie-Hellman

With p = 11 and g = 2, Alice and Bob both arrive at K = 4

Alice	Public: p = 11, g = 2	Bob
Picks a = 3		Picks b = 4
Sends 2^3 mod 11 = 8	8 →	Sends 2^4 mod 11 = 5
Receives 5	← 5	Receives 8
$K = 5^3$ mod 11 = 4 (125 = 11 × 11 + 4)		**$K = 8^4$ mod 11 = 4** (4096 = 11 × 372 + 4)

Your turn: pick your own a and b, compute K by hand, then check with a neighbor

Modular exponentiation is a one-way function



Raw K isn't the session key

Feed K into a key derivation function (KDF) to get the key actually used

Practical DH adds authentication

Textbook DH alone is safe only against eavesdroppers, so real protocols such as STS and TLS authenticate the exchange

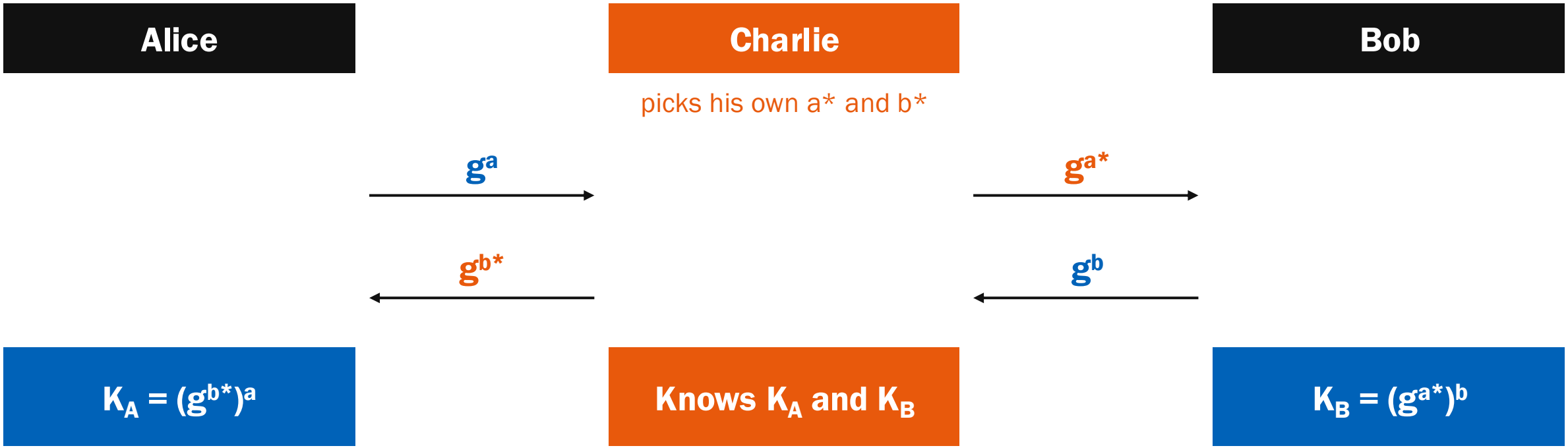
03

Attacking textbook DH

03

Attacking textbook DH

Charlie runs one exchange with Alice and another with Bob, then reads everything



Charlie decrypts with K_A, re-encrypts with K_B and forwards. Both sides see a working channel while Charlie reads, changes or injects messages

Adapted from P. C. van Oorschot, Computer Security and the Internet, Fig. 4.4

03

Attacking textbook DH

A novice can draw two grandmasters by relaying their moves

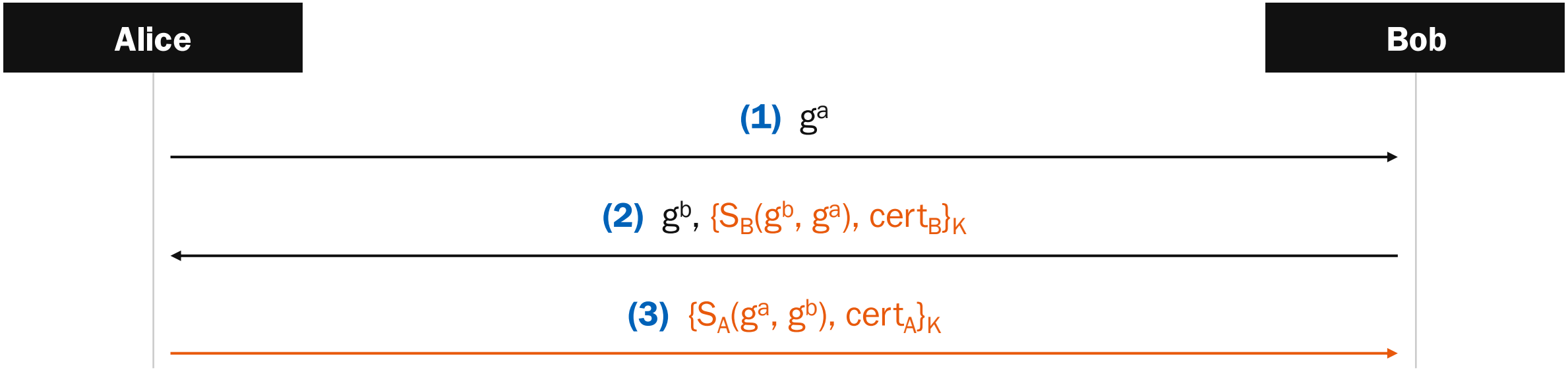
1. Charlie plays black against Alexis and white against Boris, by mail
2. He copies each move from one game onto the other board
3. Alexis and Boris are really playing each other, and Charlie collects two draws against grandmasters

---

Same flaw in Diffie-Hellman

Relaying makes the middle invisible. Asking “Bob” to prove he knows a shared password doesn’t help either: Charlie asks the real Bob and relays the proof

STS signs both exchanged values, so each side knows who shares the key



S_B signs with Bob's private key; **$cert_B$** vouches for his public key

$\{...\}_K$ is encrypted under the DH key K, hiding identities from eavesdroppers

Charlie can't sign as Bob, so a swapped g^{b*} fails Alice's check

03

Attacking textbook DH

Swapping g^a for 0, 1 or p − 1 forces the key into a tiny, guessable set

Charlie sends

0

$K = 0^b = 0$

Charlie sends

1

$K = 1^b = 1$

Charlie sends

p − 1

$K = (-1)^b$, so 1 or p − 1

Defense: check every value you receive

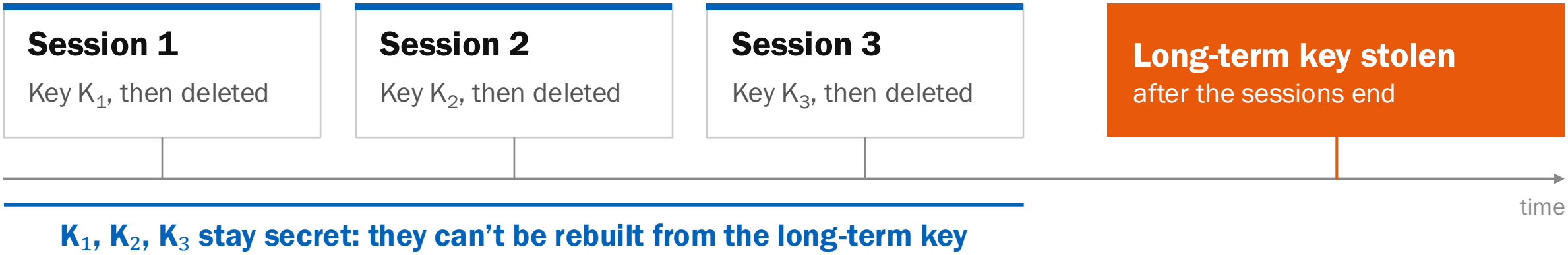
Reject 0, 1 and p − 1, and confirm the value lies in the intended subgroup

Further reading: small-subgroup confinement attacks, linked in the course notes

04

Key authentication properties

Forward secrecy: stealing a long-term key later doesn't expose earlier sessions



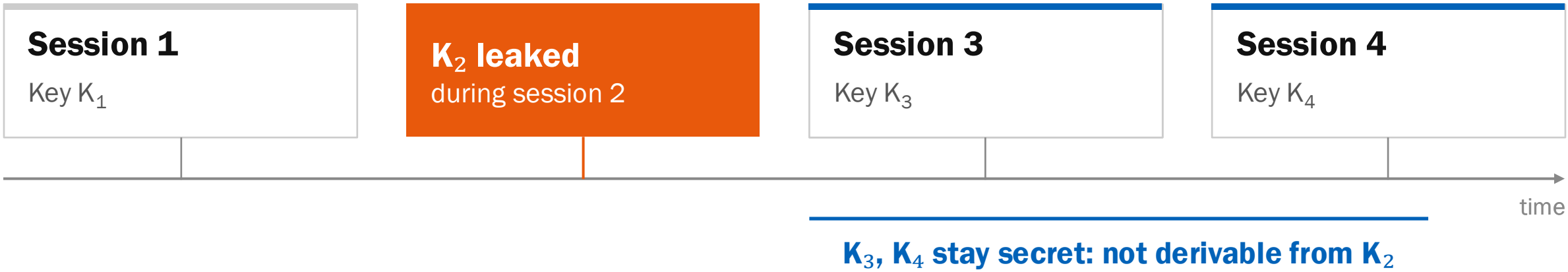
No forward secrecy: RSA key transport

Each session key travels encrypted under Bob's long-term key, so stealing it later decrypts every recorded session

How Diffie-Hellman provides it

Alice and Bob pick fresh *a* and *b* each run and securely delete them when the session ends

Known-key security: a leaked session key doesn't expose future sessions or allow impersonation



No later impersonation

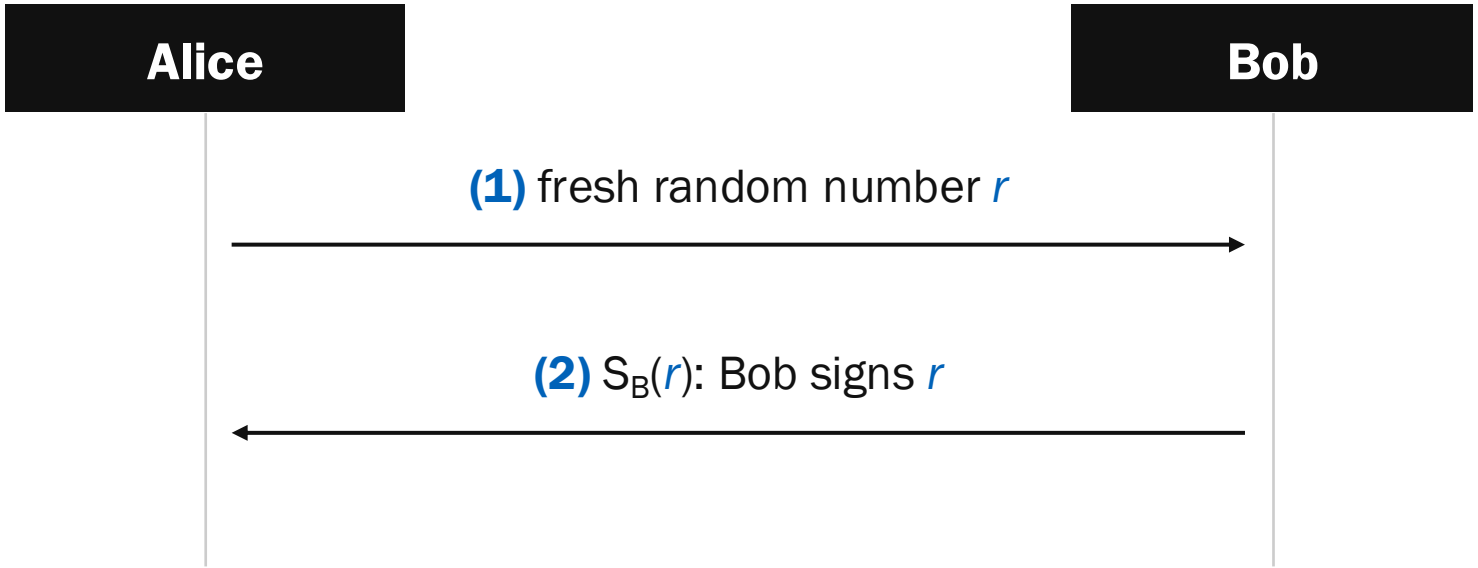
Knowing K_2 doesn't let an attacker pass as Alice or Bob in a new run

Long-term keys stay safe

Long-term keys set up session keys, never the other way around

Compare: forward secrecy protects past sessions from a stolen long-term key; known-key security protects future sessions from a leaked session key

Entity authentication shows the other party is taking part in the protocol right now



Liveness: only a Bob who is online now could sign a value that didn't exist before step (1)

Email can't give liveness

Email is store-and-forward: a validly signed message may have been written days ago, by a sender who is no longer online

04

Key authentication

Key-use confirmation proves the other side holds the key, but not who they are

1 Unauthenticated DH

Exchange g^a and g^b with no identities

2 Use the key

The far end sends known data encrypted under K

3 Check it

Alice decrypts it correctly, so the sender holds K

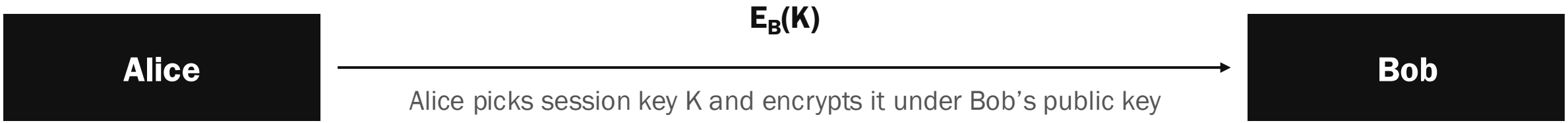
What Alice learns

Someone in this protocol run computed the correct K. This works in real time or store-and-forward

What she doesn't learn

Who that someone is. Without authentication, it could be Charlie in the middle

Implicit authentication: only Bob can recover the key, but Alice can’t tell whether he has



Narrowed to Bob

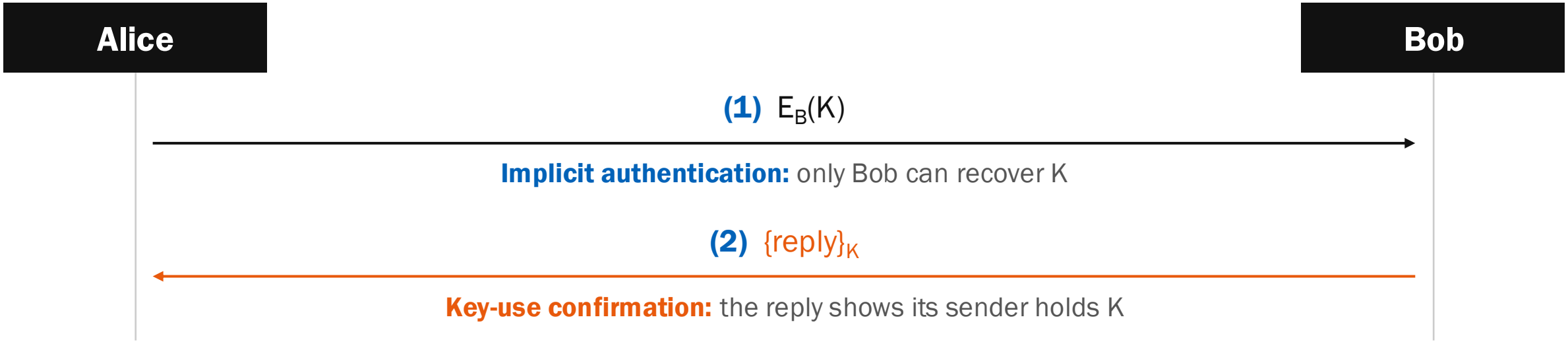
Only Bob’s private key decrypts $E_B(K)$, so no one else can hold K

Not confirmed

Bob may never have received or decrypted it, and nothing in this message tells Alice

Implicit authentication narrows who could have the key to one named party, without confirming that party has it

Explicit authentication needs both implicit authentication and key-use confirmation



Implicit authentication + Key-use confirmation = Explicit authentication

Only Bob can hold K, so only Bob could have sent (2): Alice knows Bob has the key

For Alice only: anyone can encrypt under Bob's public key, so Bob still doesn't know who sent K

STS gives both parties entity authentication and explicit key authentication

What Alice checks in message (2): g^b, $\{S_B(g^b, g^a), \text{cert}_B\}_K$

Bob's signature covers Alice's fresh g^a	**Entity authentication** Bob is taking part right now
The message decrypts under K and the signature verifies	**Key-use confirmation** The far end holds K
The signature is encrypted under K, which only the DH partner can compute	**Implicit key authentication** Only Bob can hold K
Both of the last two together	**Explicit key authentication** Bob holds K

Message (3) gives Bob the same properties about Alice