

Key Agreement Schemes

CS 6750 Lecture 9

November 12, 2009

Riccardo Pucella

Key Establishment Problem

PK cryptosystems have advantages over SK cryptosystems

- PKCs do not need a secure channel to establish secret keys
- However, PKCs generally less efficient than SKCs
- So you often want SKCs anyways

The problem: n agents on an insecure network

- Want to establish keys between pairs of agents to communicate securely

Distribution vs Agreement

Last time: **Secret Key Distribution Scheme** (SKDS):

- Assume a Trusted Authority (TA) - a server
- TA chooses a secret key for communicating, and transmits it to parties that wants to communicate

Key Agreement Scheme (KAS):

- Two or more parties want to establish a secret key on their own
- Uses public key cryptography

Main Goal of Key Agreement

At the end of an exchange:

- Two parties share a key K
- The value of K is not known to any other party
 - **Secrecy**

Sometimes want more: mutual identification (chap. 9)

- No honest participant in a session of the scheme will accept after any interaction in which an adversary is active
- A form of **mutual authentication**

Attacker Models

May or may not be a user in the system

- insider vs outsider attacker

May be passive or active

- Alter messages in transit (including intercepting)
- Save messages for later reuse
- Attempt to masquerade as other users

Possible Attacker Objectives

Passive objectives:

- Determine some (partial) information about key exchanged by users

Active objectives:

- Fool U and V into accepting an “invalid” key
 - E.g. an old expired key, or a key known to adv
- Make U and V believe they have exchanged a key with each other when that is not the case

Extended Attacker Models

Known session key attack:

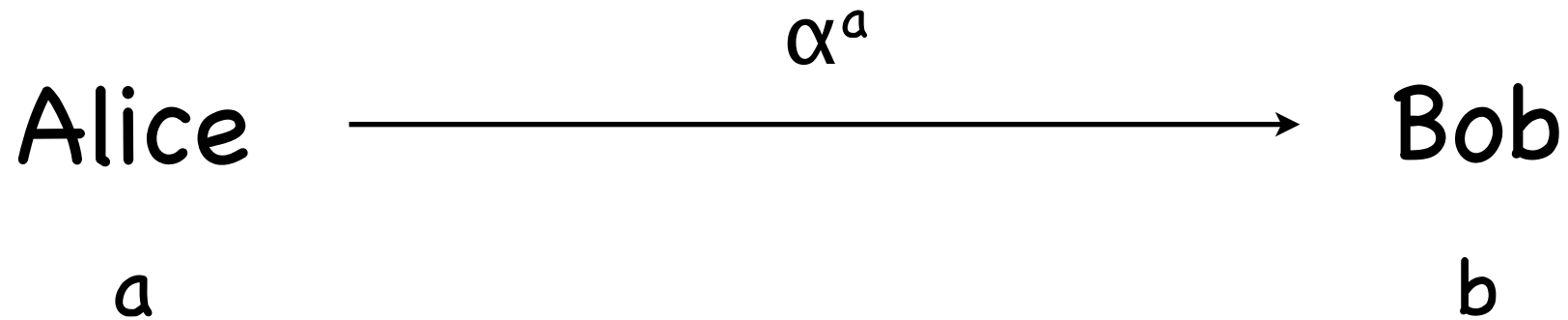
- Attacker learns session keys, you want other session keys (as well as private keys) to remain secret

Known private key attack:

- Attacker learns private key of a participant, you want previous session keys to remain secret
- Perfect forward secrecy
- This is not a property of a cryptosystem, but of how a cryptosystem is used!

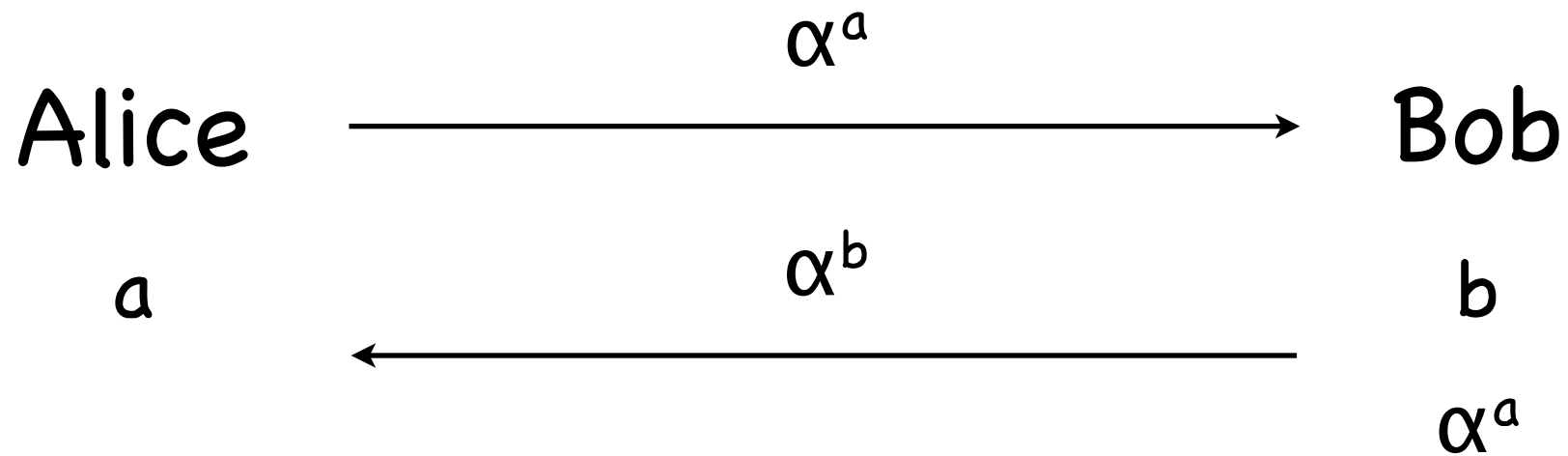
Diffie-Hellman Scheme

G a group and $\alpha \in G$ of order n



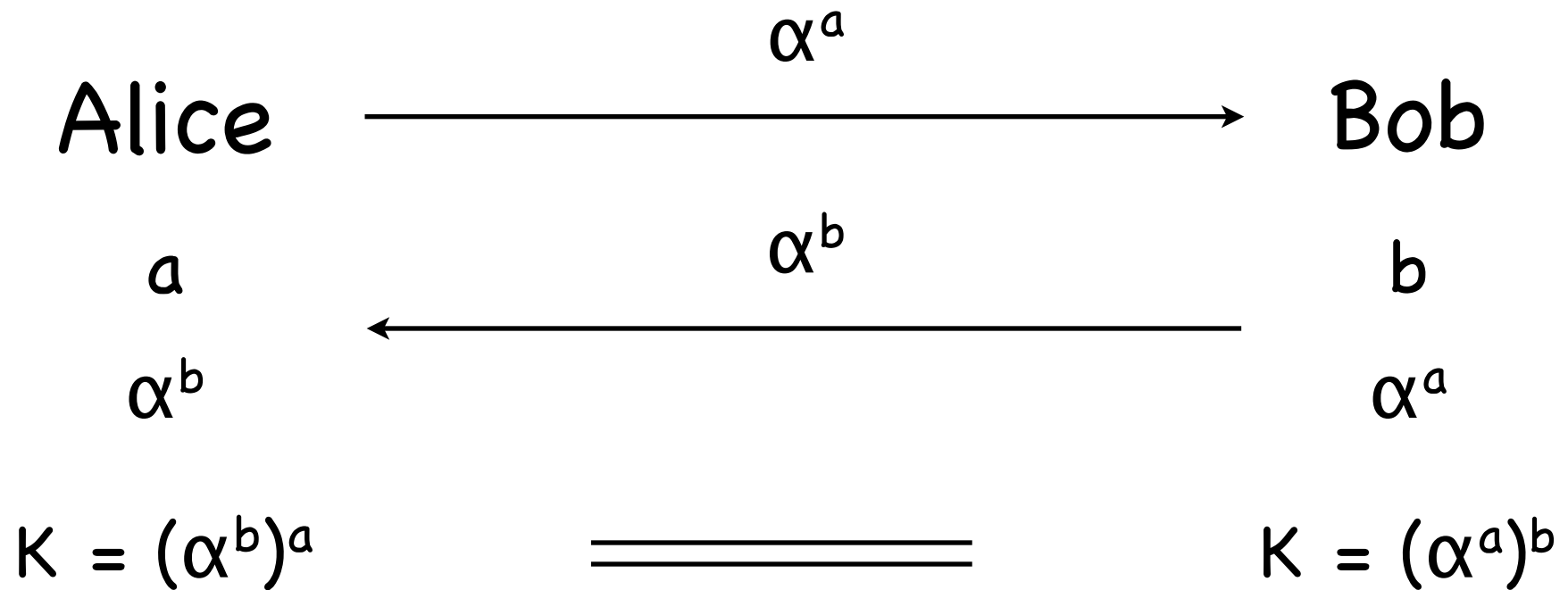
Diffie-Hellman Scheme

G a group and $\alpha \in G$ of order n



Diffie-Hellman Scheme

G a group and $\alpha \in G$ of order n



Computational Diffie-Hellman Problem

For the previous scheme to be secure, need for the group G and α to be such that:

- Given α^a and α^b , it is hard to find α^{ab}

Can show (6.7.3) that if you can solve the CDH problem, then you can solve the discrete log problem in G

Man-in-the-Middle Attack on DHS

Oscar sits between Alice and Bob and substitutes his own messages

Alice

Oscar

Bob

Man-in-the-Middle Attack on DH

Oscar sits between Alice and Bob and substitutes his own messages



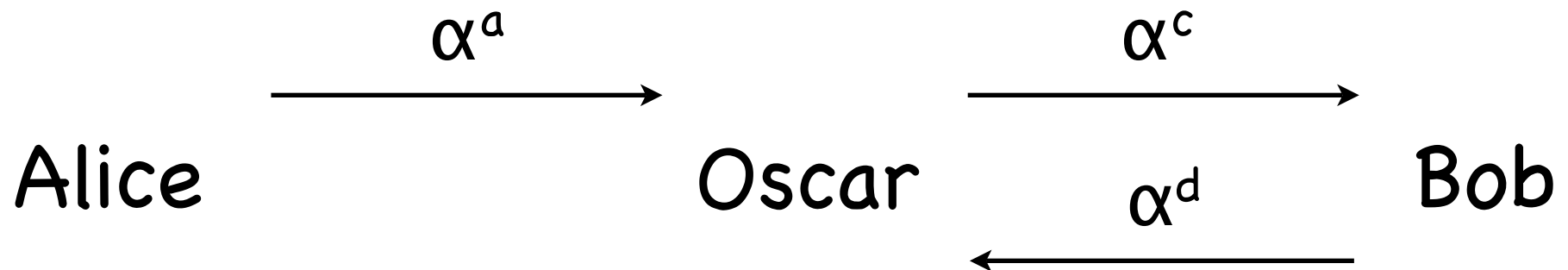
Man-in-the-Middle Attack on DH

Oscar sits between Alice and Bob and substitutes his own messages



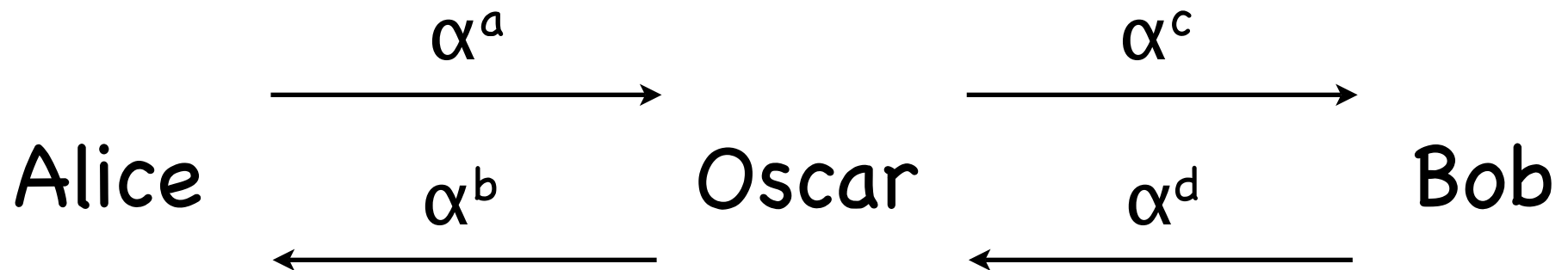
Man-in-the-Middle Attack on DHs

Oscar sits between Alice and Bob and substitutes his own messages



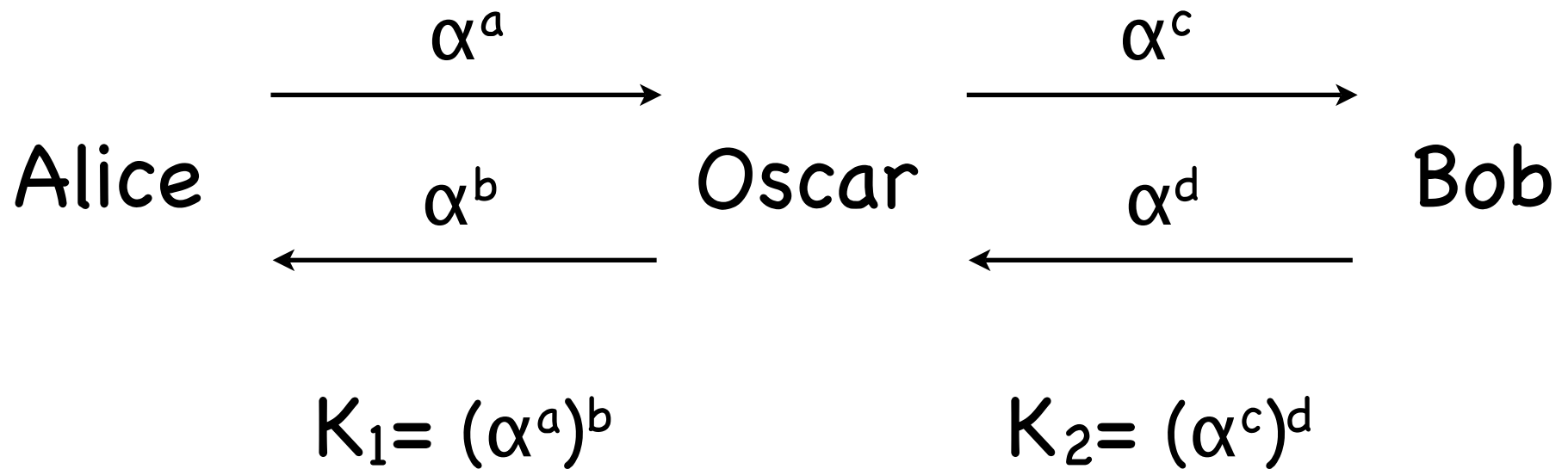
Man-in-the-Middle Attack on DHs

Oscar sits between Alice and Bob and substitutes his own messages



Man-in-the-Middle Attack on DHK

Oscar sits between Alice and Bob and substitutes his own messages



Authenticated DHS

Various ways to authenticate DH Scheme:

- Use signatures
- Use ElGamal-style public keys
- Use passwords

None of these approaches are perfect

Using signatures:

Station-to-Station Scheme (1)

G a group and $\alpha \in G$ of order n

Alice

a

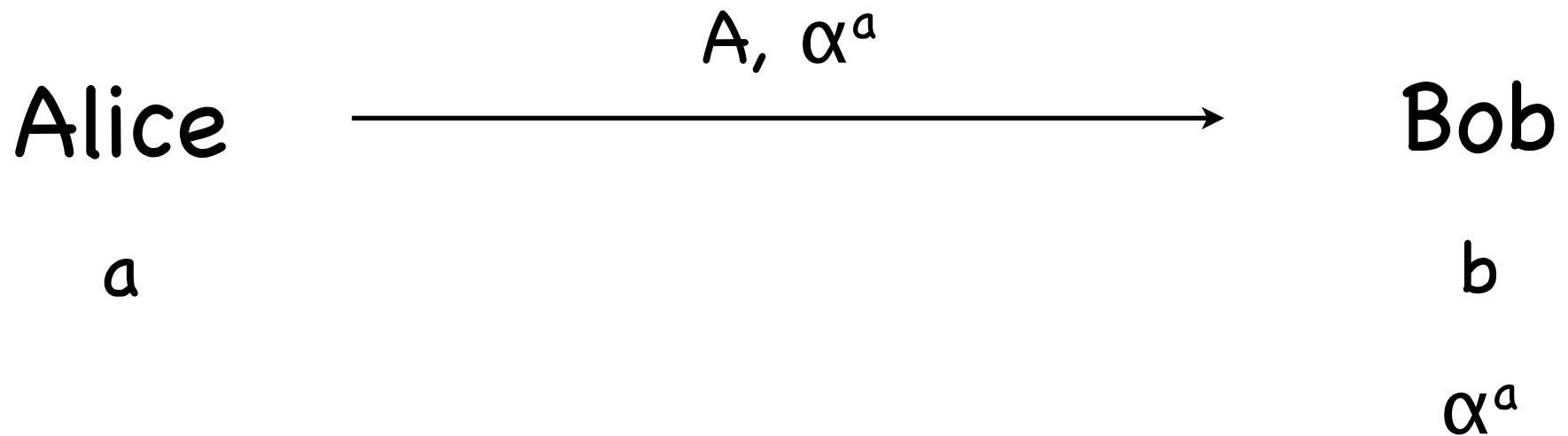
Bob

b

Using signatures:

Station-to-Station Scheme (1)

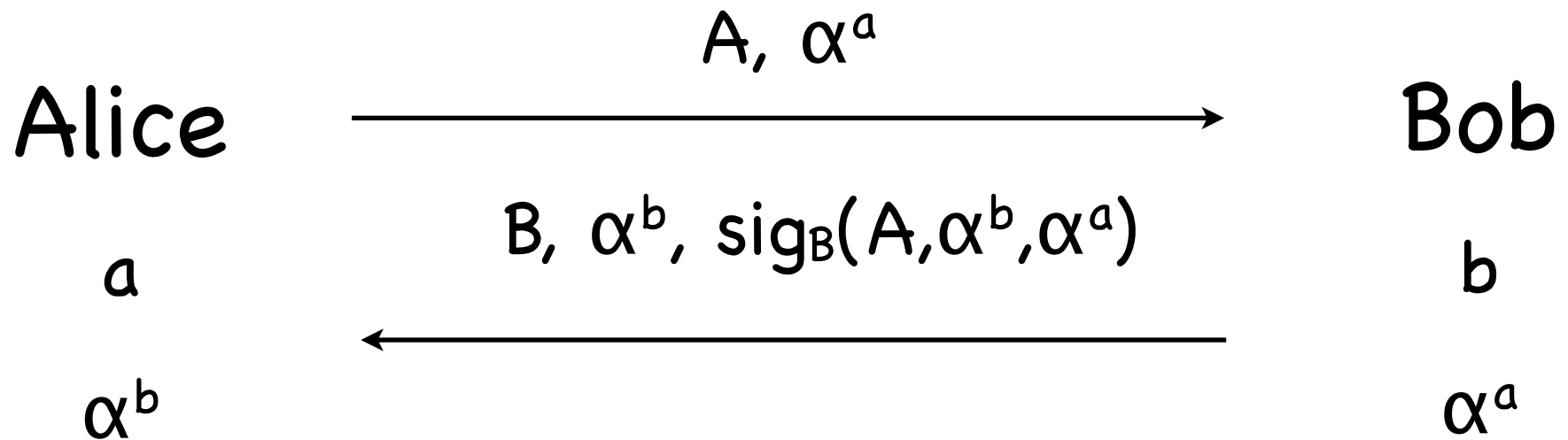
G a group and $\alpha \in G$ of order n



Using signatures:

Station-to-Station Scheme (1)

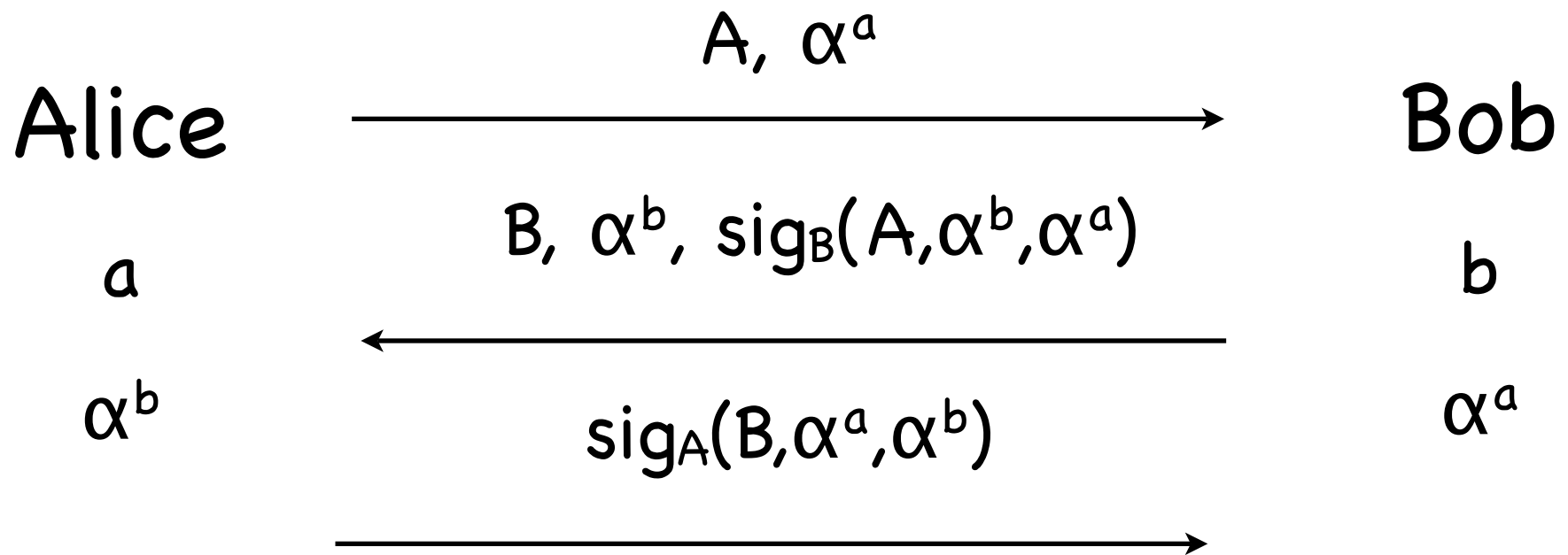
G a group and $\alpha \in G$ of order n



Using signatures:

Station-to-Station Scheme (1)

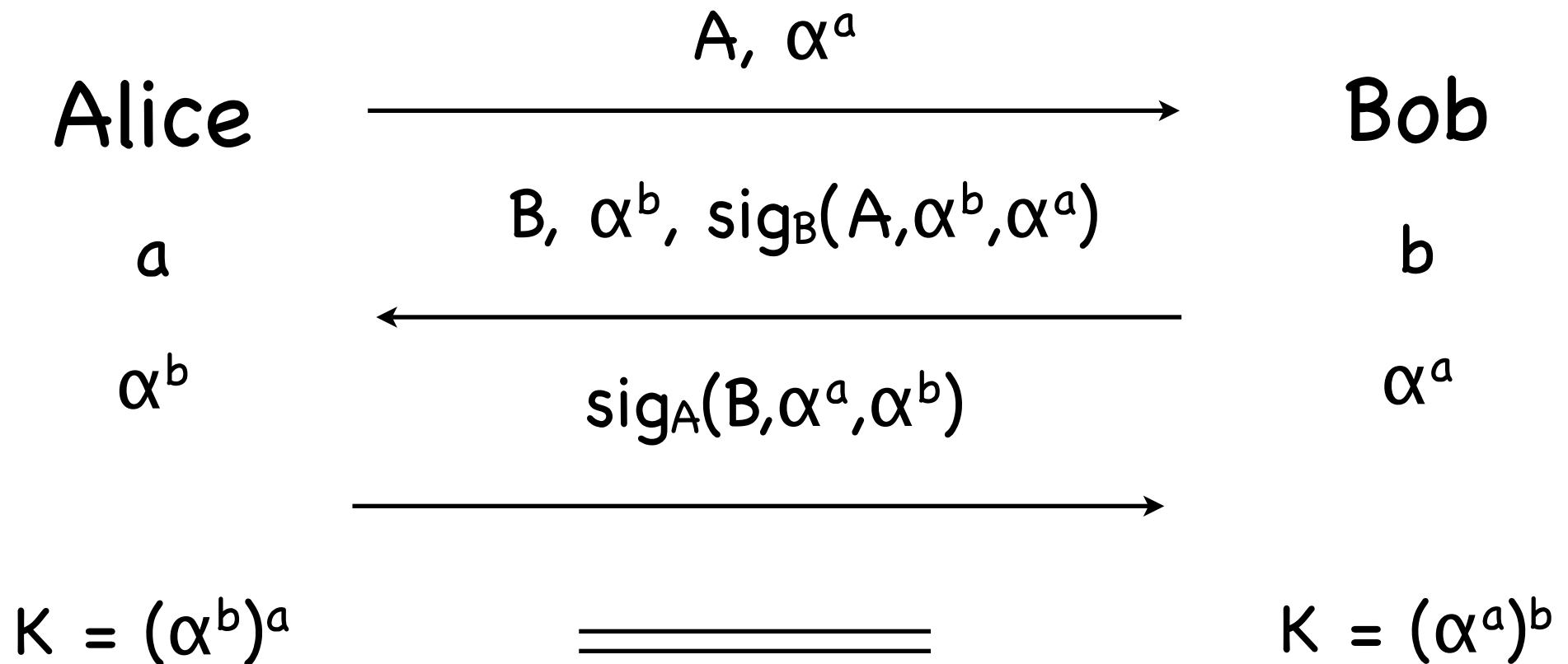
G a group and $\alpha \in G$ of order n



Using signatures:

Station-to-Station Scheme (1)

G a group and $\alpha \in G$ of order n



Using signatures:

Station-to-Station Scheme (2)

G a group and $\alpha \in G$ of order n

Alice

a

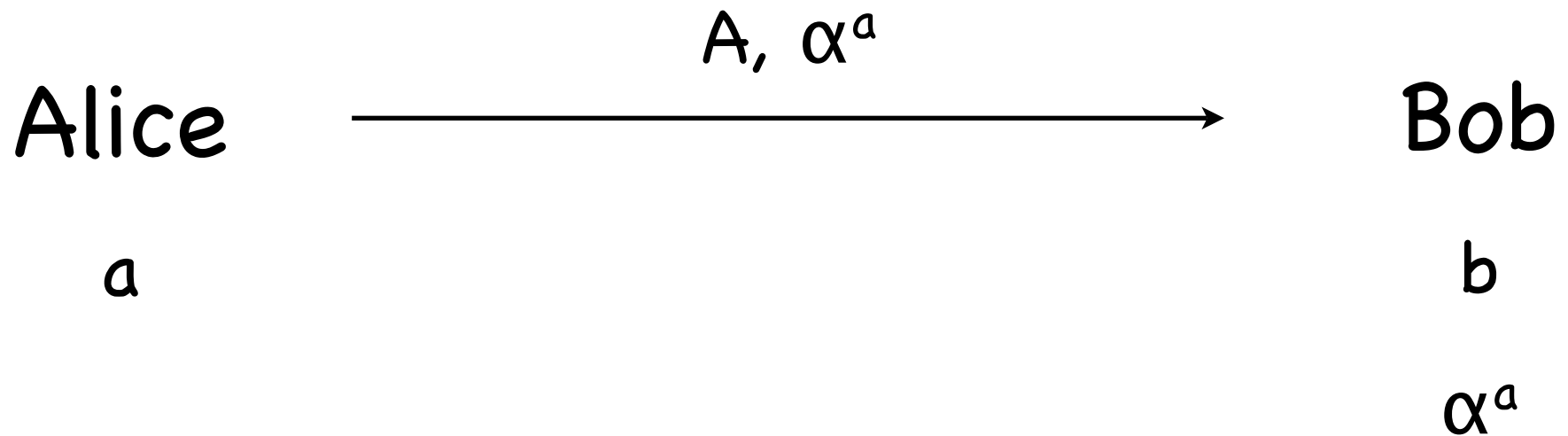
Bob

b

Using signatures:

Station-to-Station Scheme (2)

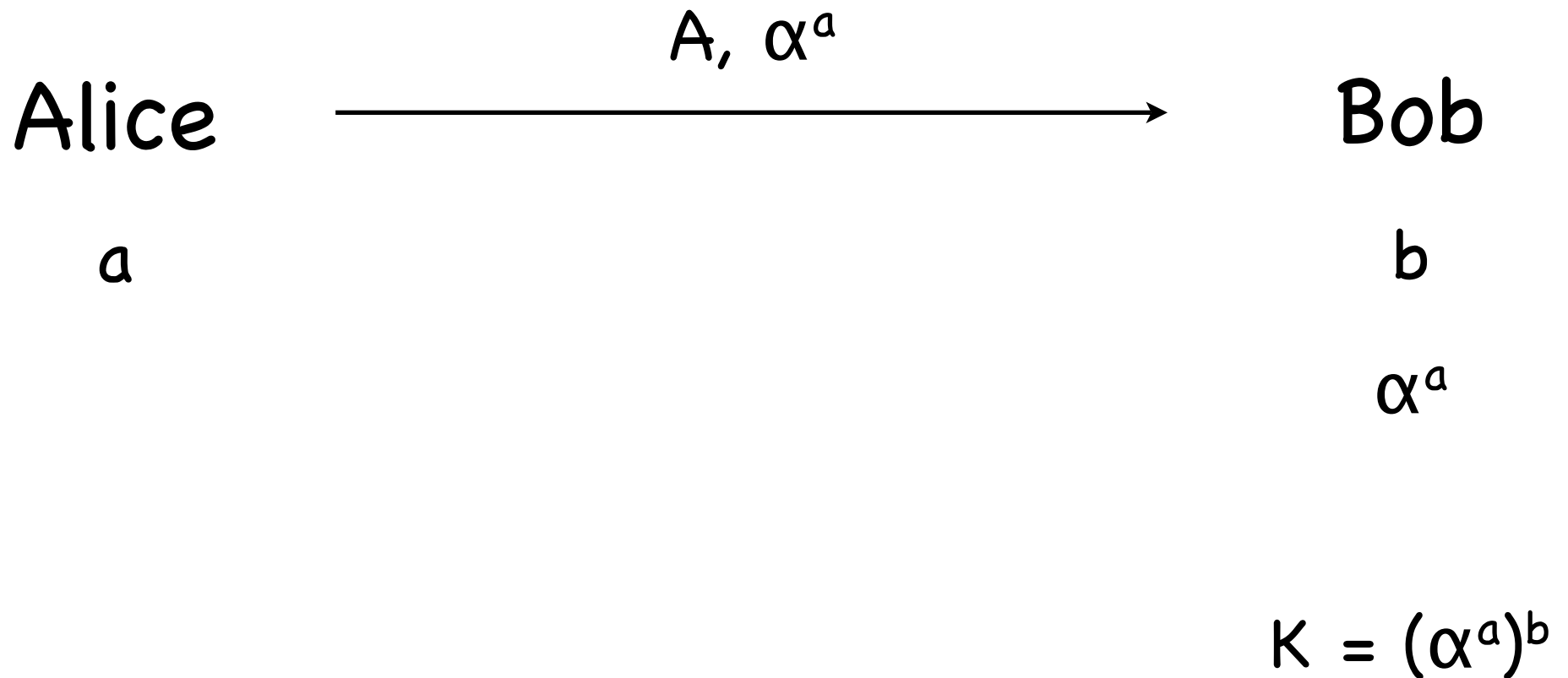
G a group and $\alpha \in G$ of order n



Using signatures:

Station-to-Station Scheme (2)

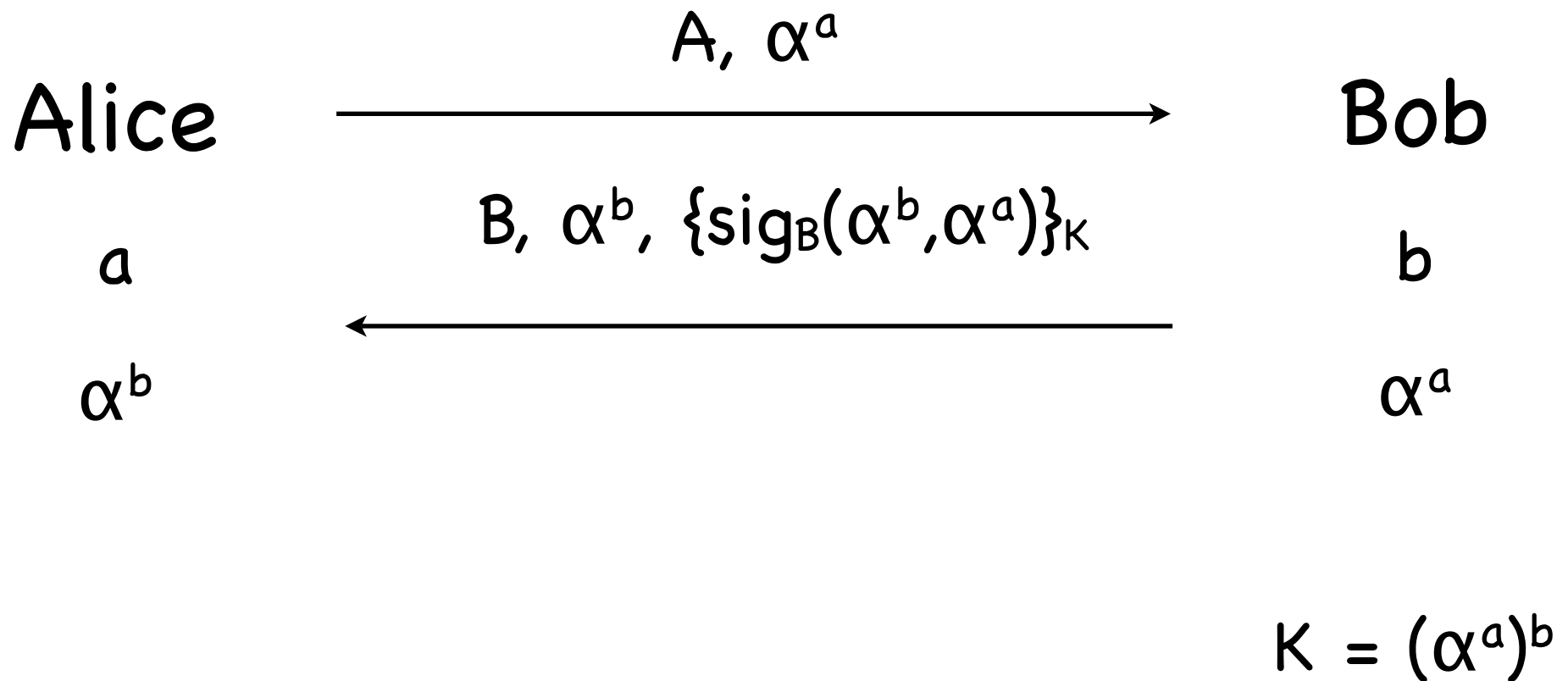
G a group and $\alpha \in G$ of order n



Using signatures:

Station-to-Station Scheme (2)

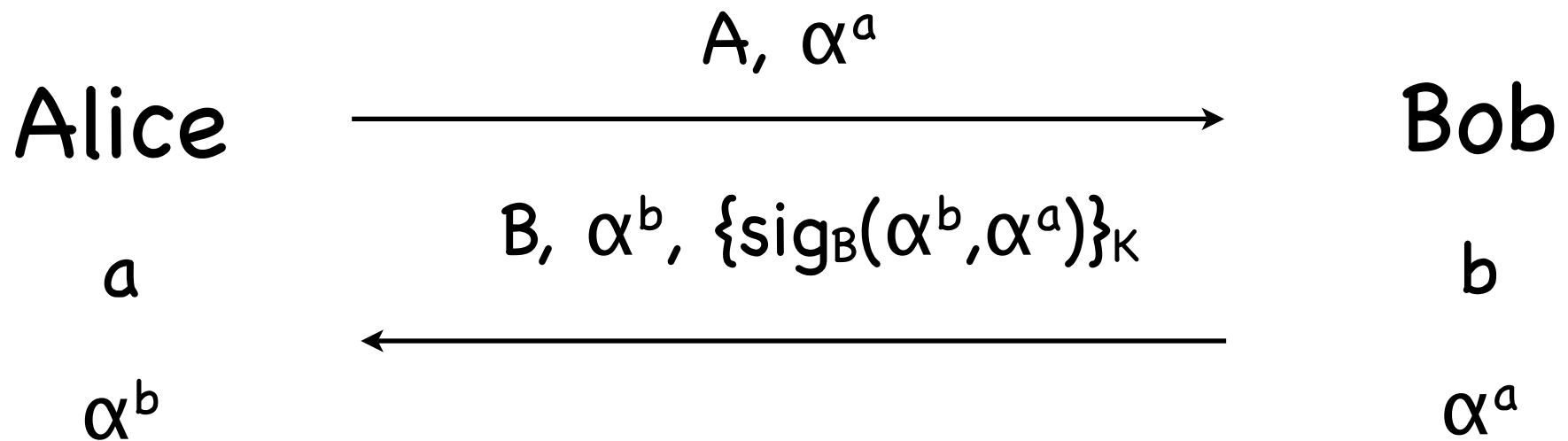
G a group and $\alpha \in G$ of order n



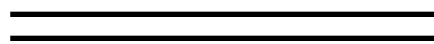
Using signatures:

Station-to-Station Scheme (2)

G a group and $\alpha \in G$ of order n



$$K = (\alpha^b)^a$$

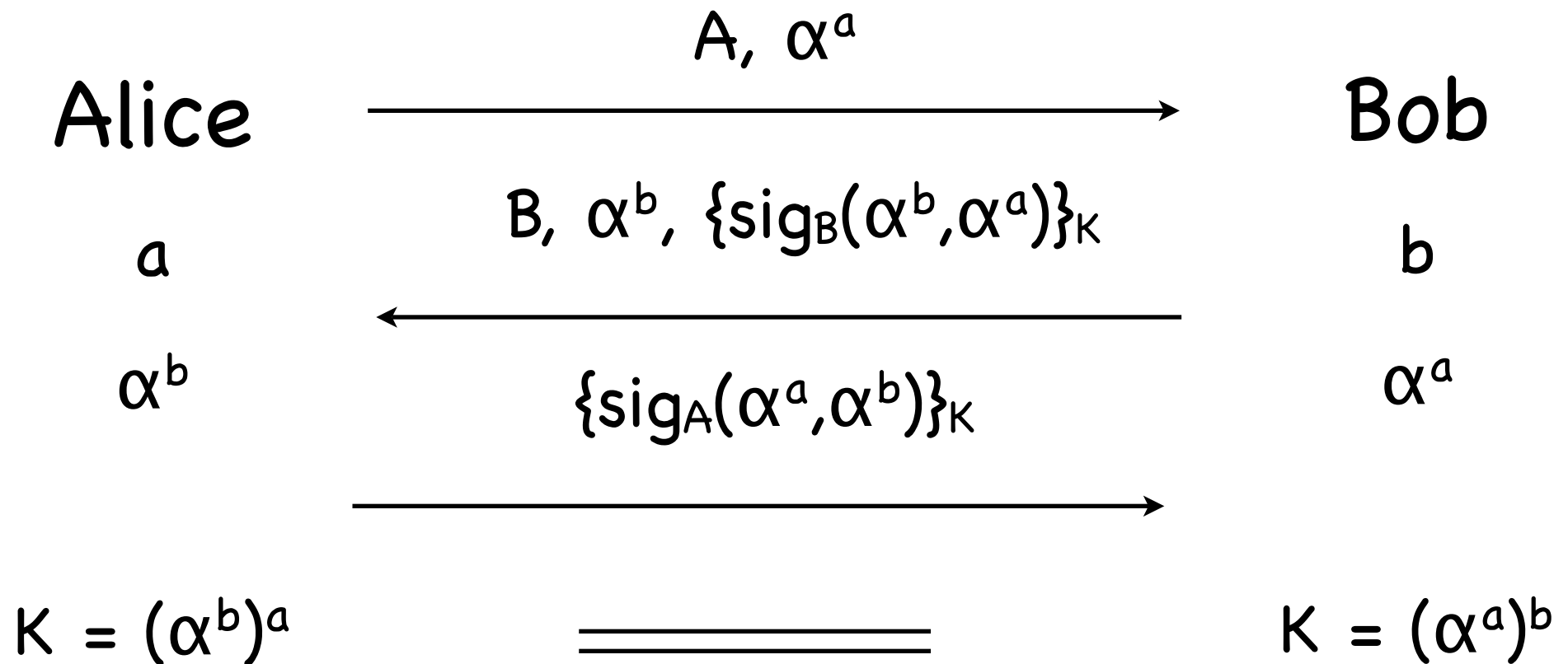


$$K = (\alpha^a)^b$$

Using signatures:

Station-to-Station Scheme (2)

G a group and $\alpha \in G$ of order n



Using ElGamal public keys:

One-Sided Authentication

G a group and $\alpha \in G$ of order n

Let M be Alice's private key, α^M her public key

Alice

M, a

Bob

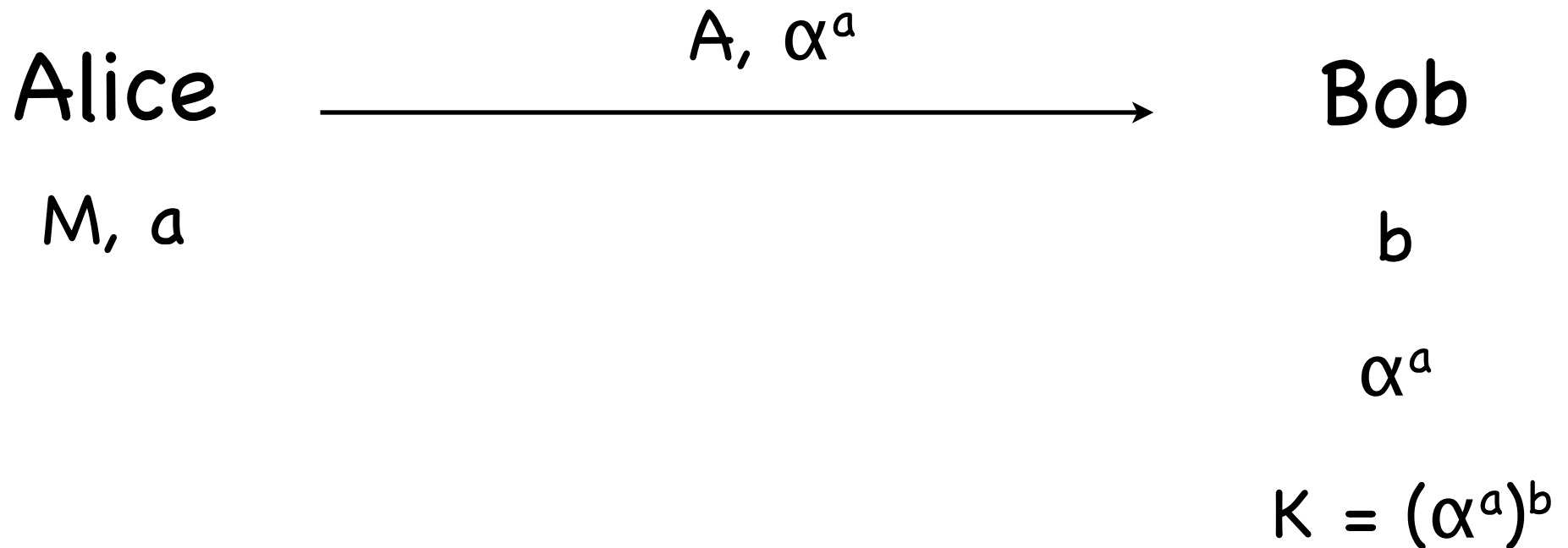
b

Using ElGamal public keys:

One-Sided Authentication

G a group and $\alpha \in G$ of order n

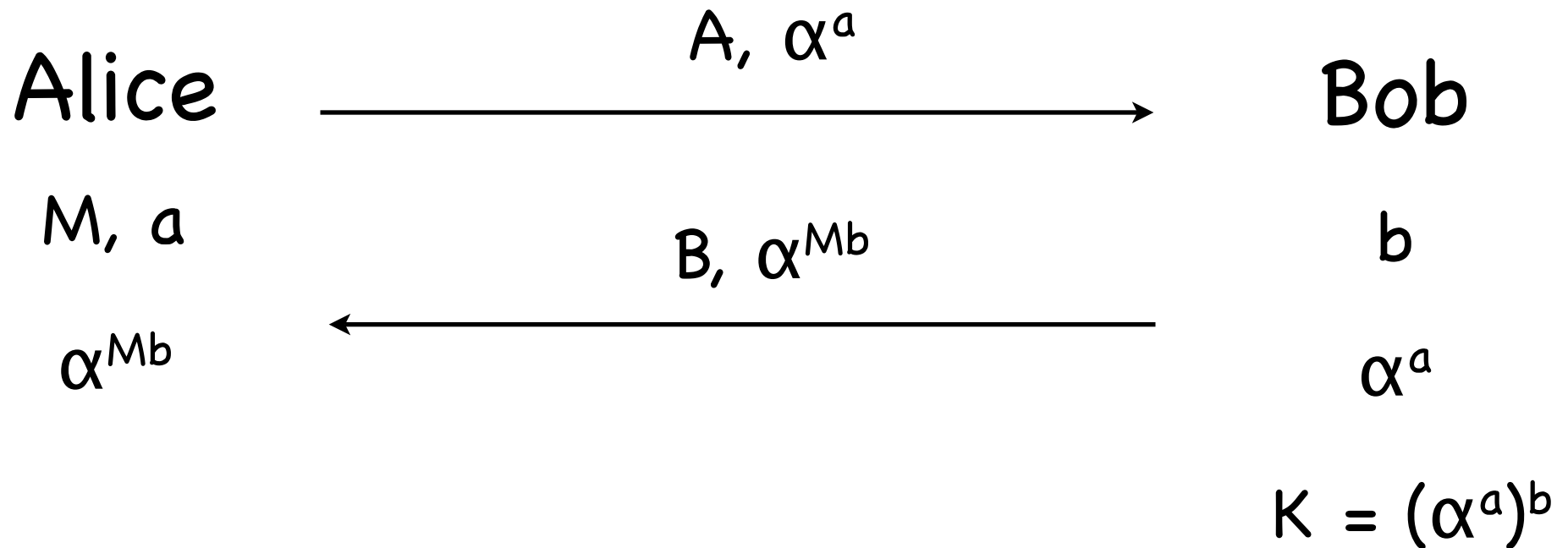
Let M be Alice's private key, α^M her public key



Using ElGamal public keys:
One-Sided Authentication

G a group and $\alpha \in G$ of order n

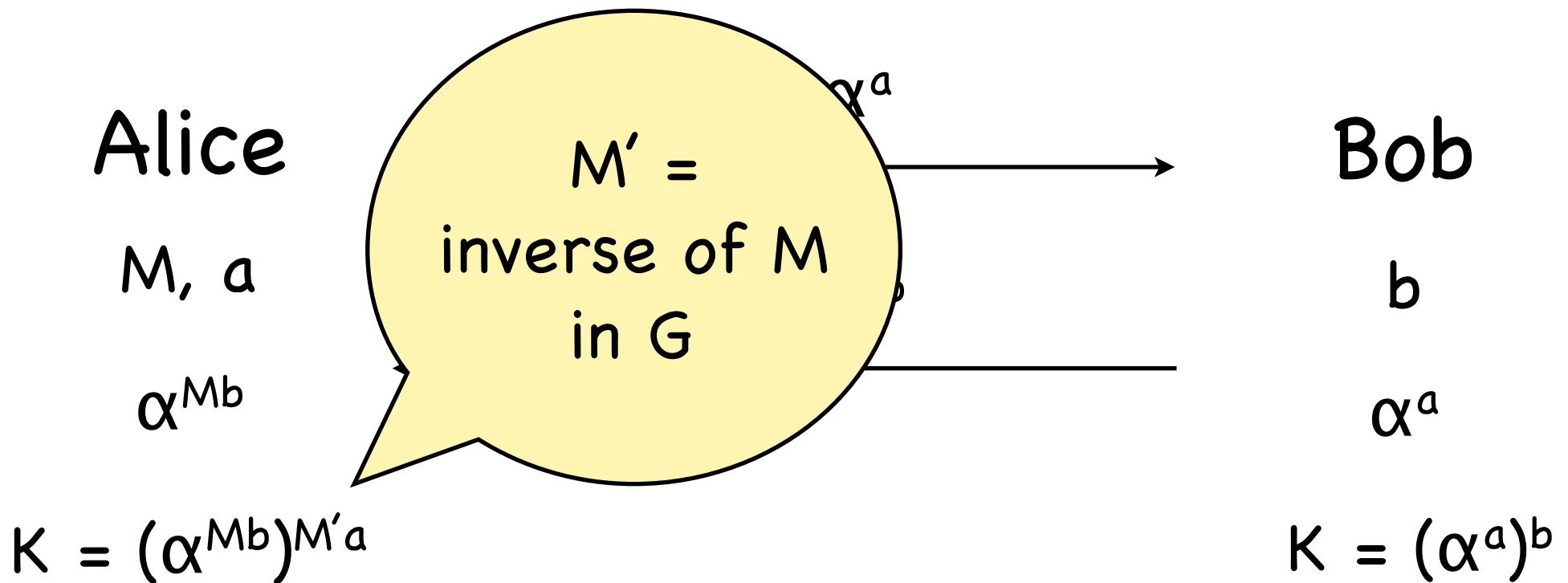
Let M be Alice's private key, α^M her public key



Using ElGamal public keys:
One-Sided Authentication

G a group and $\alpha \in G$ of order n

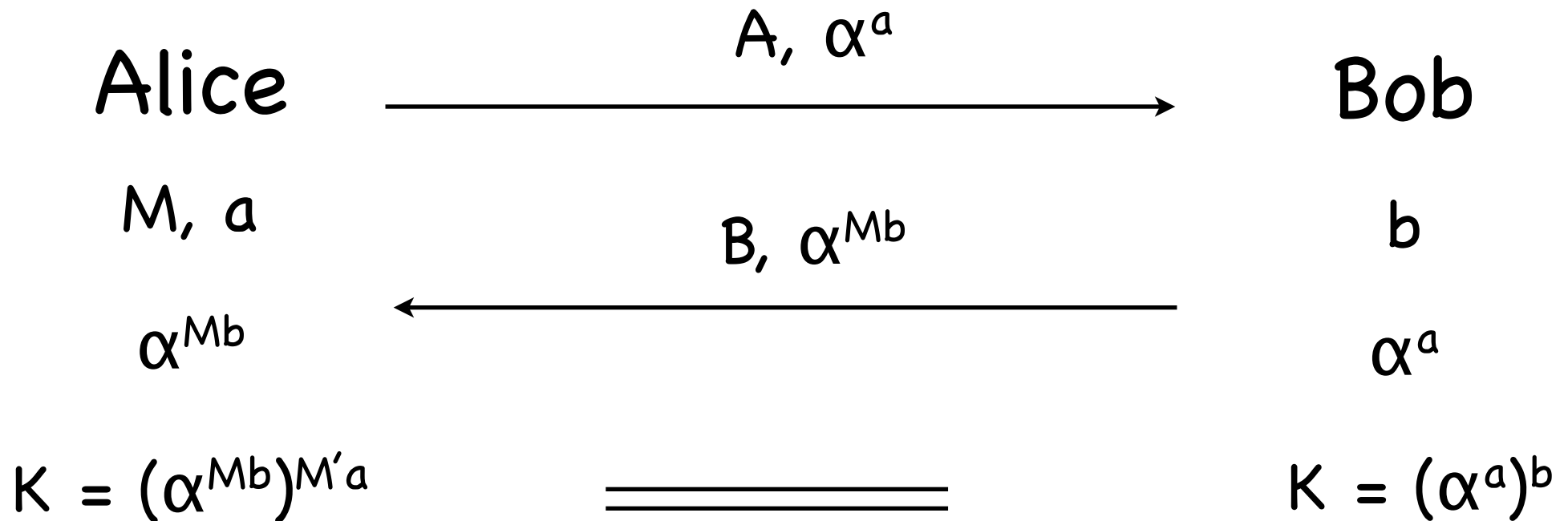
Let M be Alice's private key, α^M her public key



Using ElGamal public keys:
One-Sided Authentication

G a group and $\alpha \in G$ of order n

Let M be Alice's private key, α^M her public key



Using ElGamal public keys:

MTI Scheme

Matsumoto, Takashima, Imai (1986)

G a group and $\alpha \in G$ of order n

Let S_A, S_B be Alice and Bob's private keys

Alice

S_A, a

Bob

S_B, b

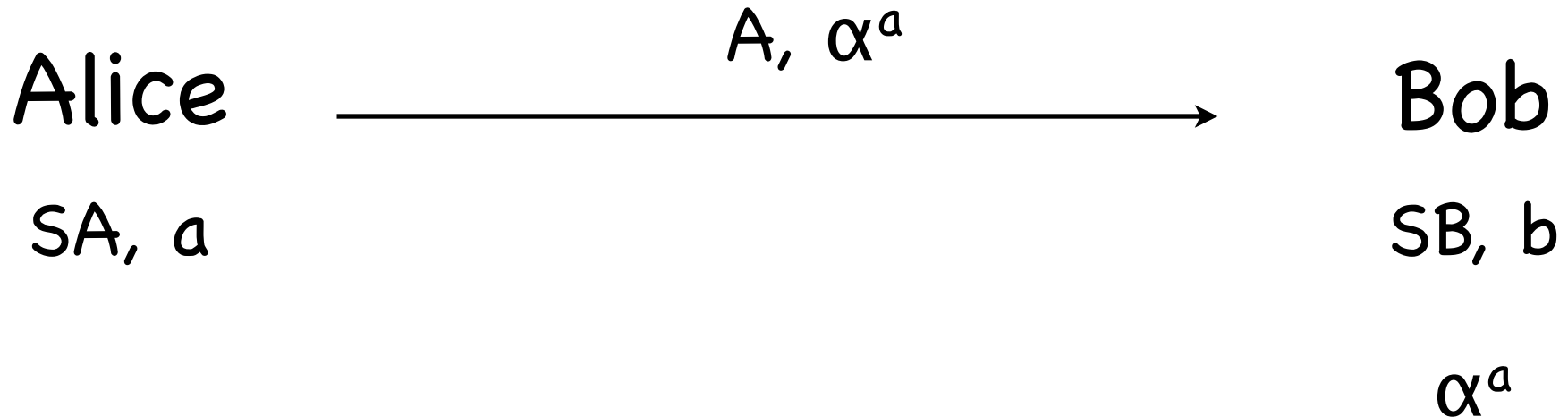
Using ElGamal public keys:

MTI Scheme

Matsumoto, Takashima, Imai (1986)

G a group and $\alpha \in G$ of order n

Let S_A, S_B be Alice and Bob's private keys

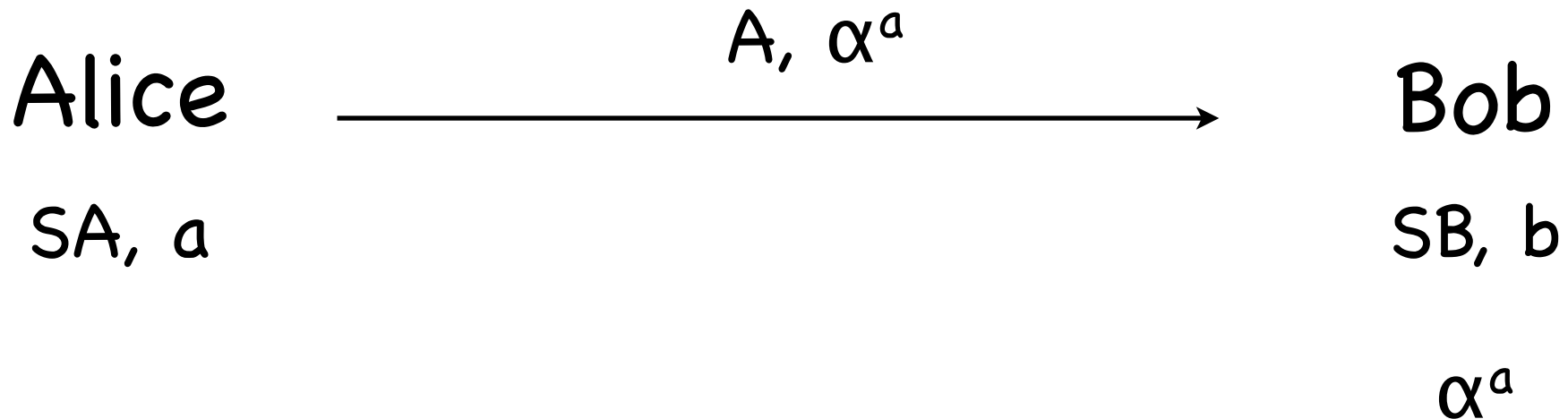


Using ElGamal public keys: MTI Scheme

Matsumoto, Takashima, Imai (1986)

G a group and $\alpha \in G$ of order n

Let S_A, S_B be Alice and Bob's private keys



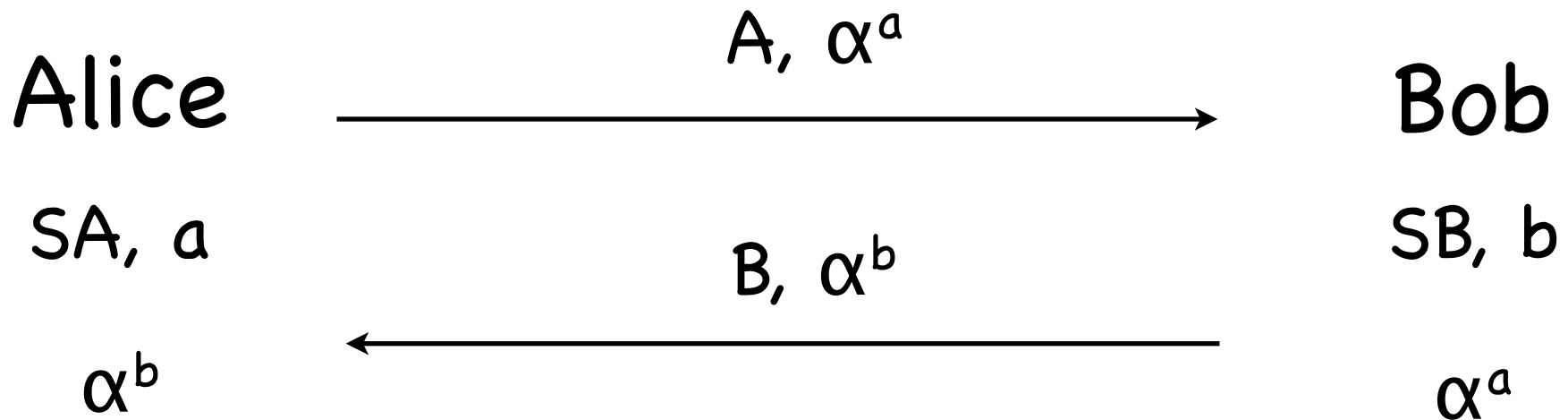
$$\begin{aligned} K &= (\alpha^{S_A})^b (\alpha^a)^{S_B} \\ &= \alpha^{S_A b + a S_B} \end{aligned}$$

Using ElGamal public keys: MTI Scheme

Matsumoto, Takashima, Imai (1986)

G a group and $\alpha \in G$ of order n

Let S_A, S_B be Alice and Bob's private keys



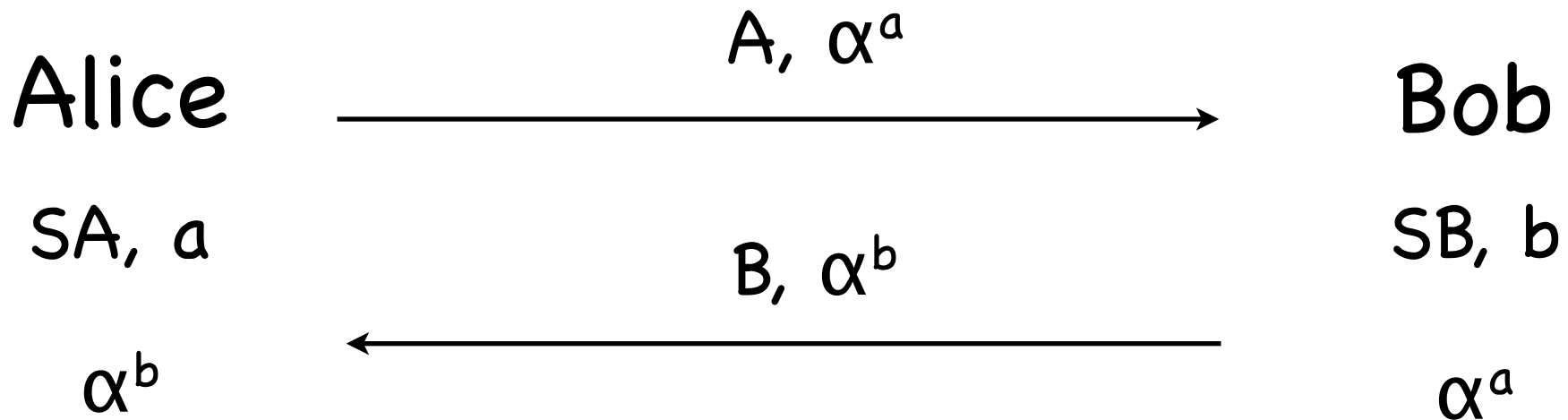
$$\begin{aligned} K &= (\alpha^{S_A})^b (\alpha^a)^{S_B} \\ &= \alpha^{S_A b + a S_B} \end{aligned}$$

Using ElGamal public keys: MTI Scheme

Matsumoto, Takashima, Imai (1986)

G a group and $\alpha \in G$ of order n

Let S_A, S_B be Alice and Bob's private keys



$$K = (\alpha^{S_B})^a (\alpha^b)^{S_A} \\ = \alpha^{S_B a + b S_A}$$

=====

$$K = (\alpha^{S_A})^b (\alpha^a)^{S_B} \\ = \alpha^{S_A b + a S_B}$$

Variants

There are **many** variants of these schemes

- Using a keyed hash (a MAC) instead of encryption in STS(2)
- MQV protocol - MTI with more complex key computation
- IKE - a component of IPSec

Each solves or addresses different issues, or make different tradeoff choices

Using passwords:

Simple Password Encrypted Key Exchange

Jablon (1996)

Suppose Alice and Bob share a password p (not a key)

Alice

a

Bob

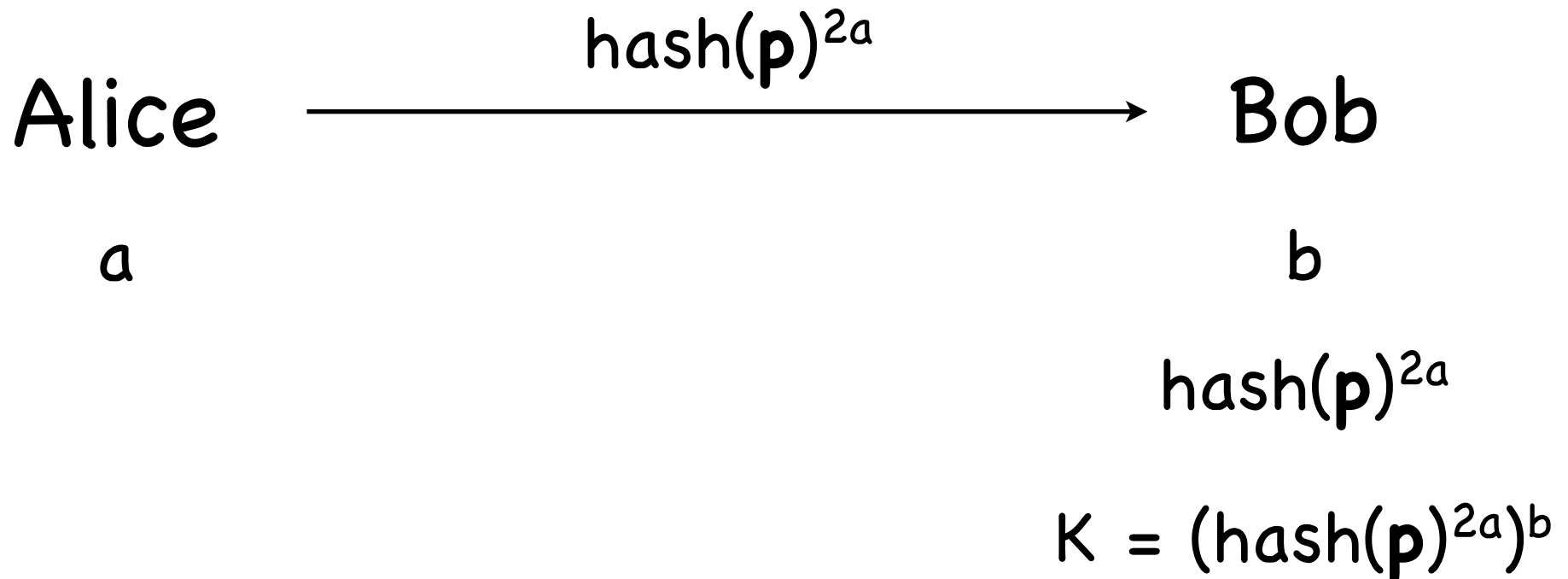
b

Using passwords:

Simple Password Encrypted Key Exchange

Jablon (1996)

Suppose Alice and Bob share a password p (not a key)

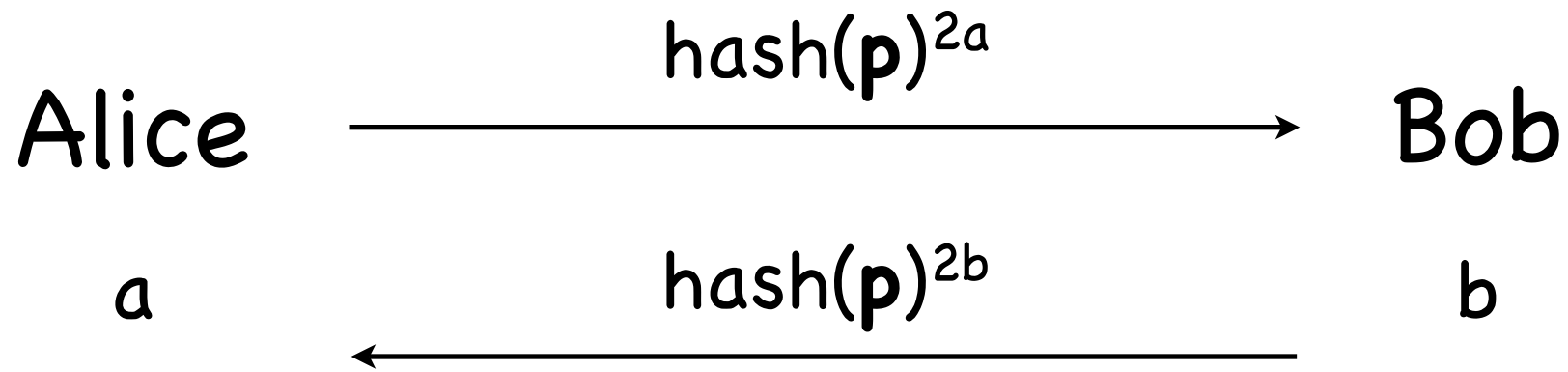


Using passwords:

Simple Password Encrypted Key Exchange

Jablon (1996)

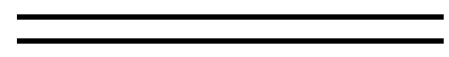
Suppose Alice and Bob share a password p (not a key)



$$\text{hash}(p)^{2b}$$

$$\text{hash}(p)^{2a}$$

$$K = (\text{hash}(p)^{2b})^a$$



$$K = (\text{hash}(p)^{2a})^b$$

Using passwords:

Simple Password Encrypted Key Exchange

Tablén (1996)

Su

Main worry: **dictionary attacks**
(brute force attacks against the password)

Need to make sure that the opponent cannot mount offline dictionary attacks, and cannot use parties as guess validators

hash(p)

hash(p)

$$K = (\text{hash}(\mathbf{p})^{2b})^a$$

====

$$K = (\text{hash}(\mathbf{p})^{2a})^b$$

PKI and Certificates

The main problem with the previous protocols:

- requiring a priori knowledge (public keys, password)

A solution:

- send the public key as part of the protocol
- how do you know the key is the “right” key?

PKI and Certificates

The main problem with the previous protocols:

- requiring a priori knowledge (public keys, password)

A solution:

- send the public key as part of a message
- how do you know the key is the sender's?

X.509 standard

Certificate, signed by a Certification Authority:

$$\text{Cert}_{CA}(U, pk_U) = (U, pk_U, \text{sig}_{CA}(U, pk_U))$$

Using signatures and certificates:

Full Station-to-Station Scheme (1)

G a group and $\alpha \in G$ of order n

Alice

a

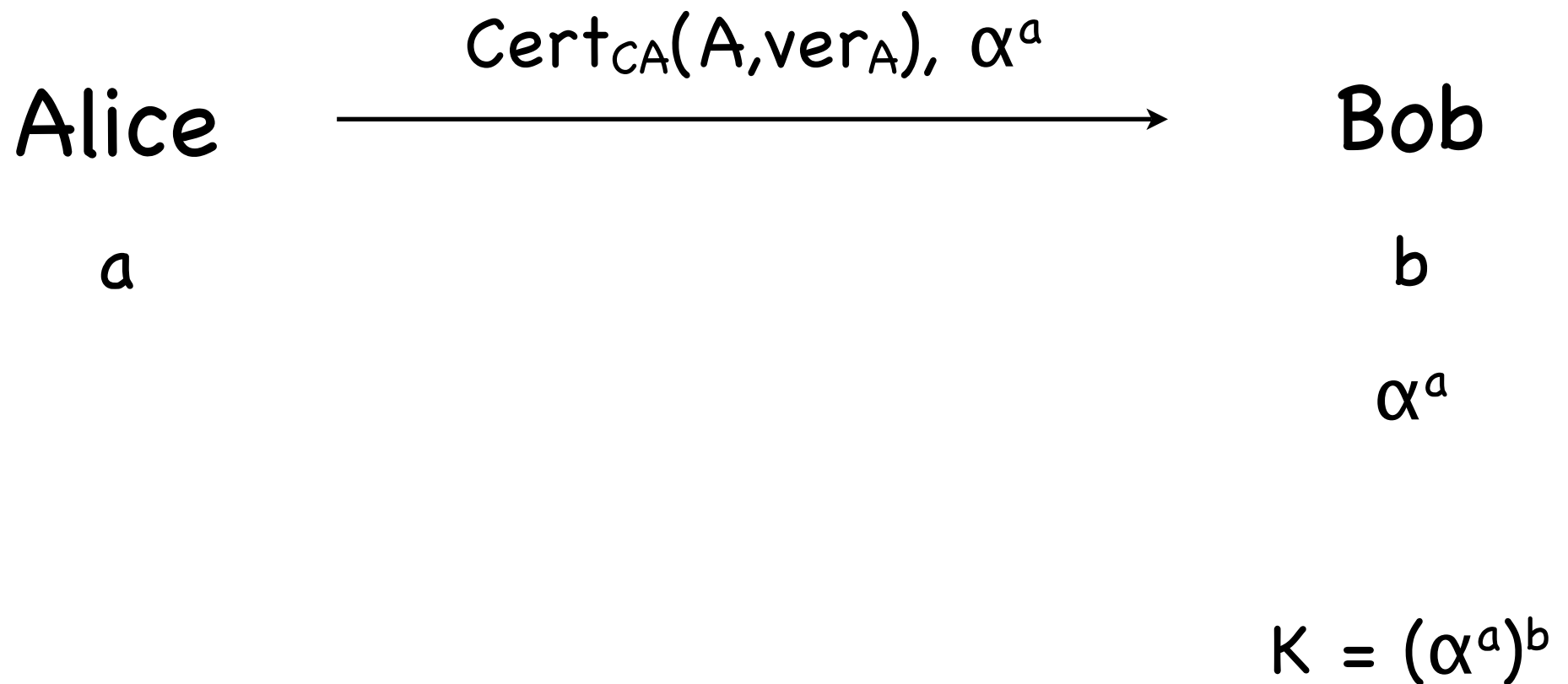
Bob

b

Using signatures and certificates:

Full Station-to-Station Scheme (1)

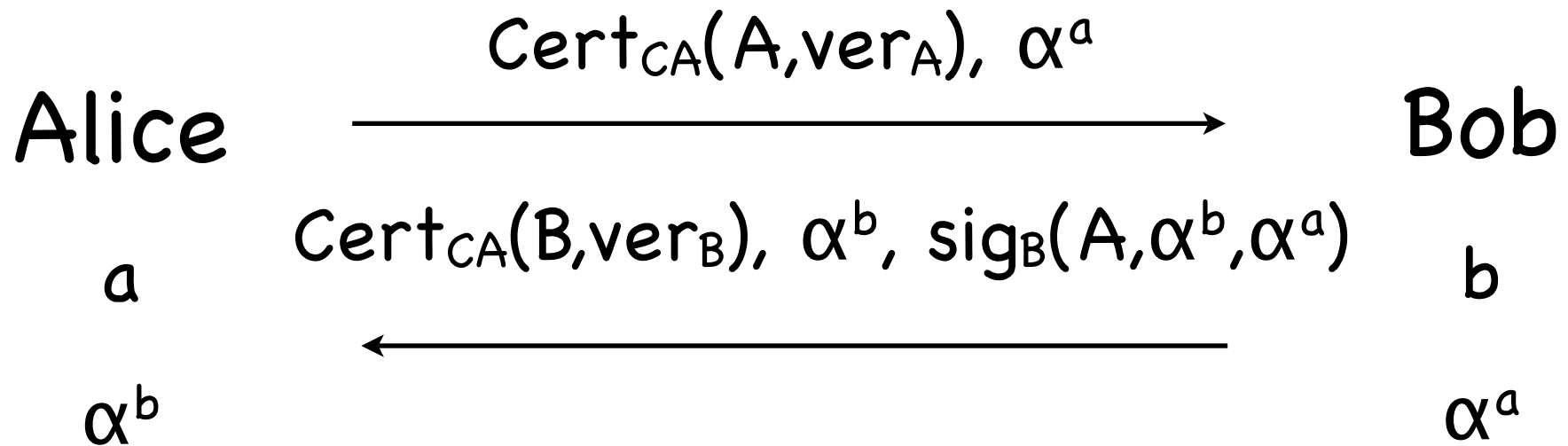
G a group and $\alpha \in G$ of order n



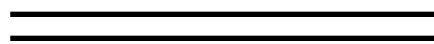
Using signatures and certificates:

Full Station-to-Station Scheme (1)

G a group and $\alpha \in G$ of order n



$$K = (\alpha^b)^a$$

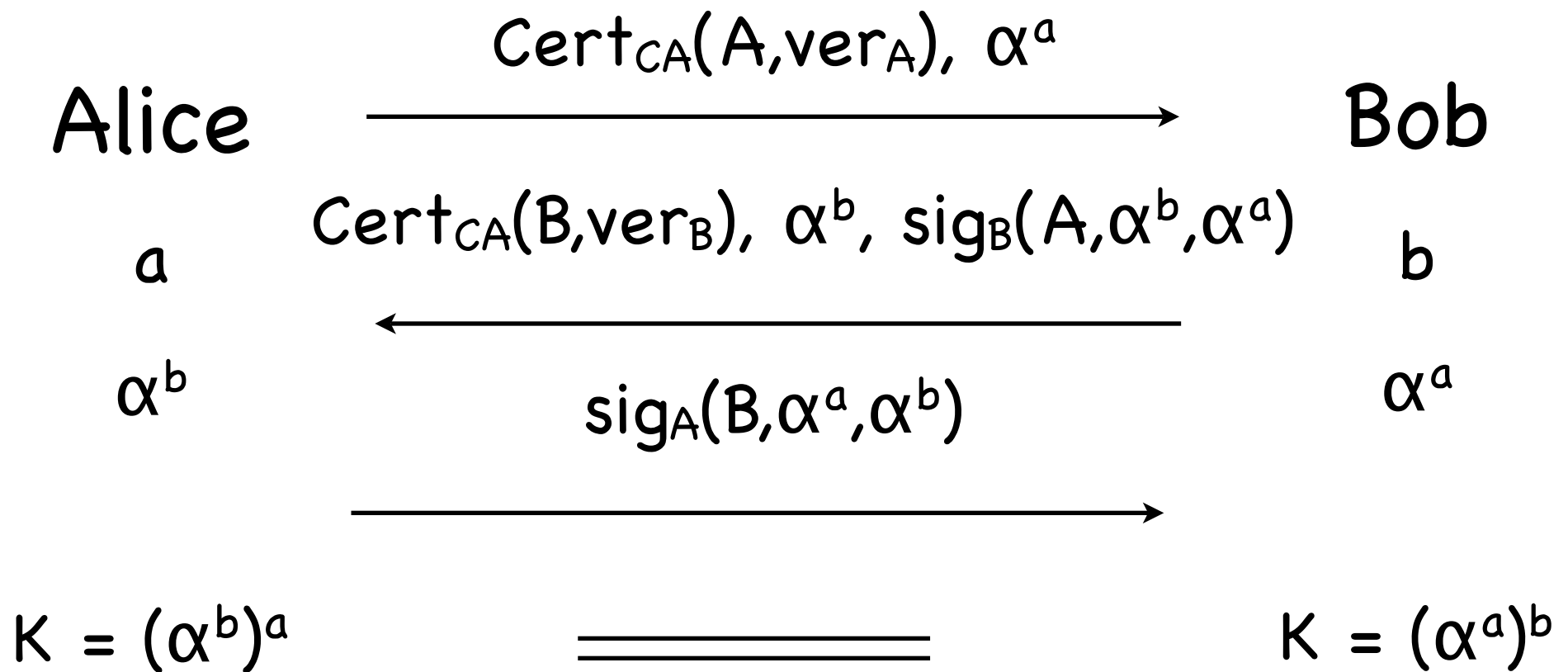


$$K = (\alpha^a)^b$$

Using signatures and certificates:

Full Station-to-Station Scheme (1)

G a group and $\alpha \in G$ of order n



Certificates: Problem 1

Certificate Revocation

- When a secret key is compromised, we should revoke the certificate
- Expiration/TTL is not enough - all expired certificates are invalid, but not all invalid certificates are expired

Use Certificate Revocation Lists (CRL)

- Need to be checked at every certificate validation
- Potential for denial-of-service attacks
- Somewhat defeats the purpose of PK

No comprehensive solution

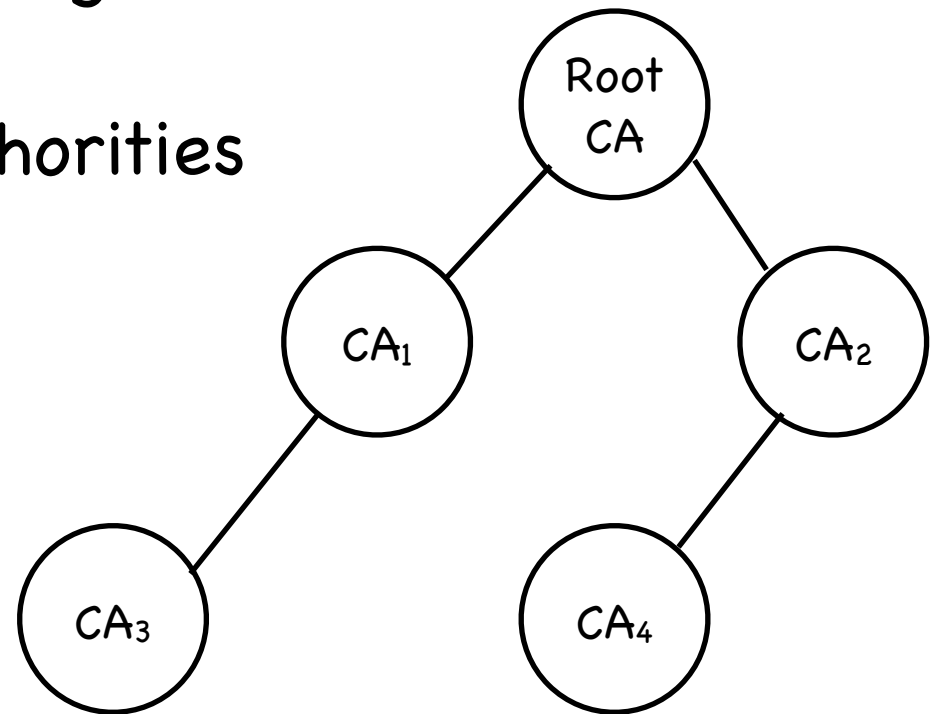
Certificates: Problem 2

Compromised Certification Authorities

- If a certificate authority is compromised, we cannot trust certificates it has signed

Hierarchical certification authorities

- The chain of trust
- CA_3 's verification key is signed by CA_1
- CA_1 's verification key is signed by RootCA



- BUT: Need to send all certificates in a chain

Cer

blem 2

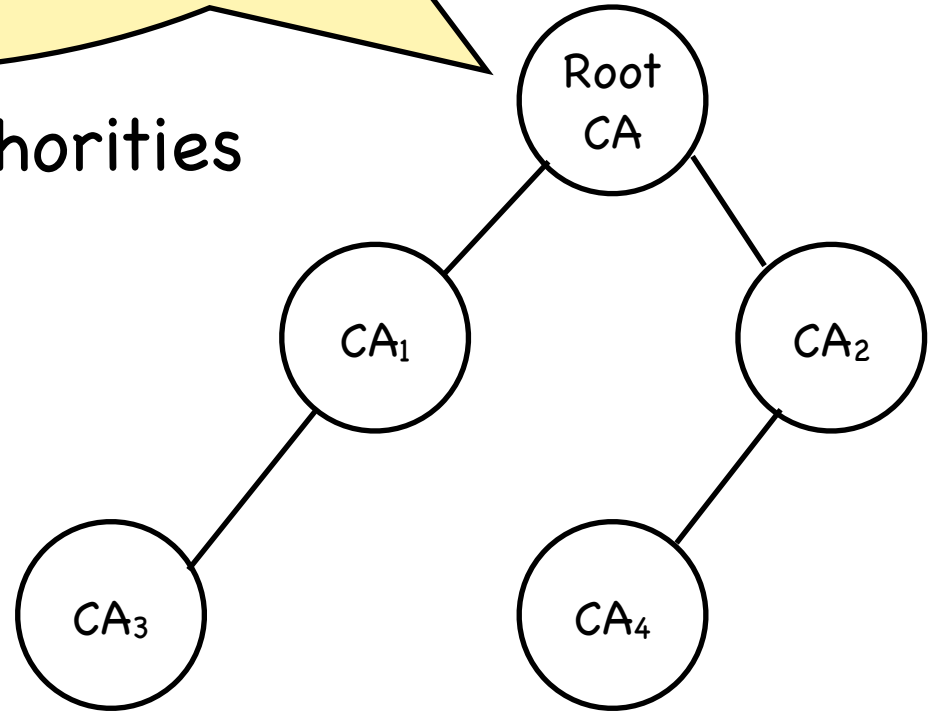
Who vouches for the RootCA's verification key?
No one - has to be trusted by some other means

Compro

- If a trust ca is not trusted, we cannot

Hierarchical certification authorities

- The chain of trust
- CA₃'s verification key is signed by CA₁
- CA₁'s verification key is signed by RootCA



- BUT: Need to send all certificates in a chain