Cryptographic Protocols

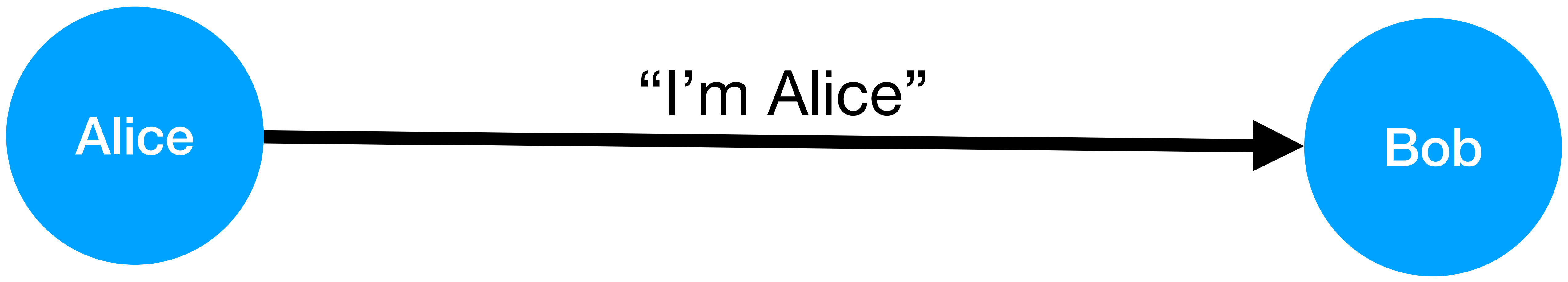
Security & Networks

Today's Lecture

- Protocols in Alice and Bob notation
- Attacks on Protocols
- Forward Secrecy
- Goals and Protocols

A Simple Protocol

- A sends a message m to B



written as:

$A \rightarrow B$: “I'm Alice”

Rules

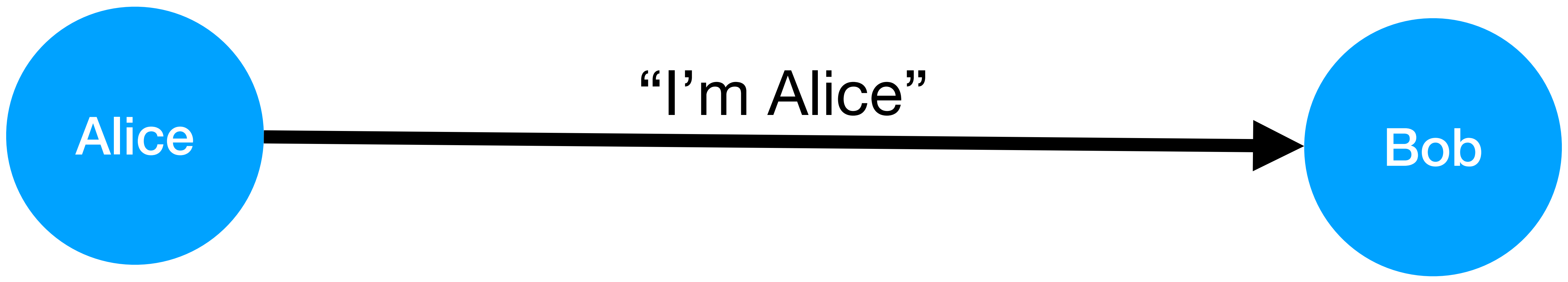
- We write down protocols as a list of messages sent between *principals*, e.g.

  1. $A \to B$: “Hello”

  2. $B \to A$: “Offer”

  3. $A \to B$: “Accept”

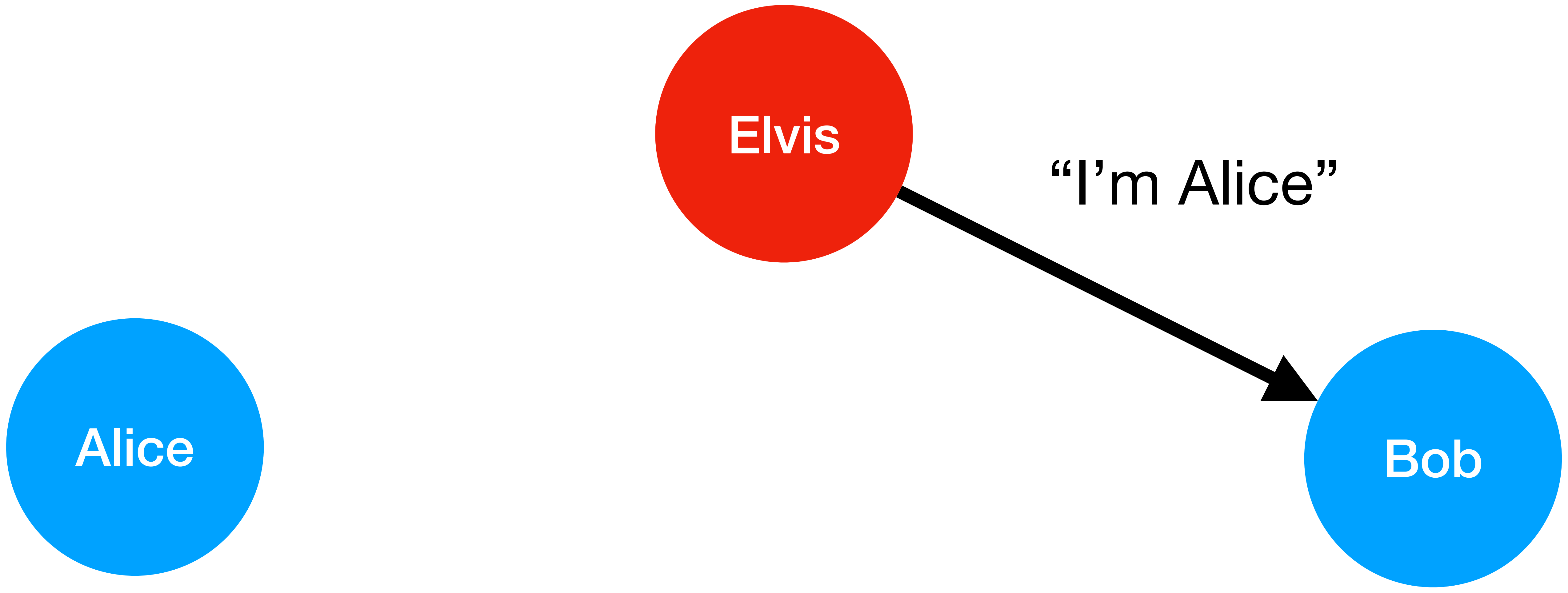
A Simple Protocol



$A \rightarrow B$: “I’m Alice”

Message “I’m Alice” can be read by an attacker.

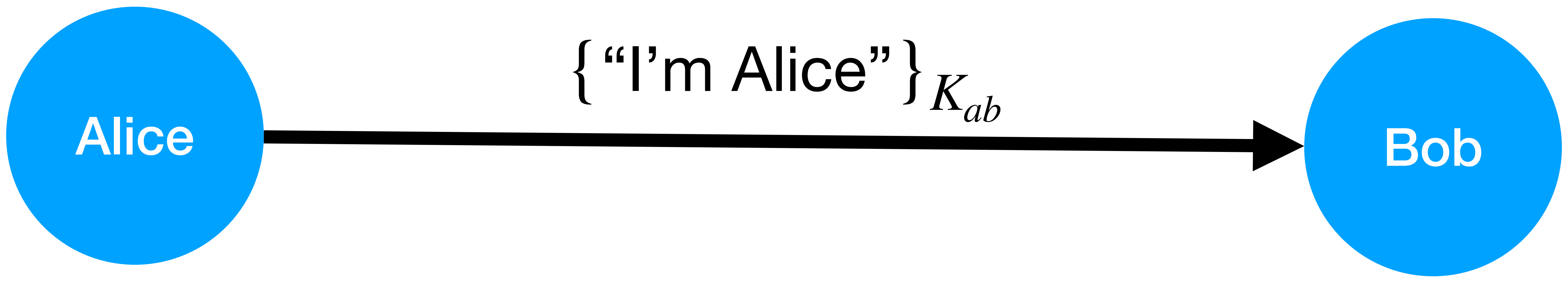
A Simple Protocol



The attacker can pretend to be anyone.

$E(A) \rightarrow B$: "I'm Alice"

A Simple Protocol



$\{\_\}_{K_{ab}}$ **means symmetric key encryption**

$$A \rightarrow B : \{\text{“I’m Alice”}\}_{K_{ab}}$$

If Alice and Bob share a key K_{ab}, then Alice can encrypt her message.

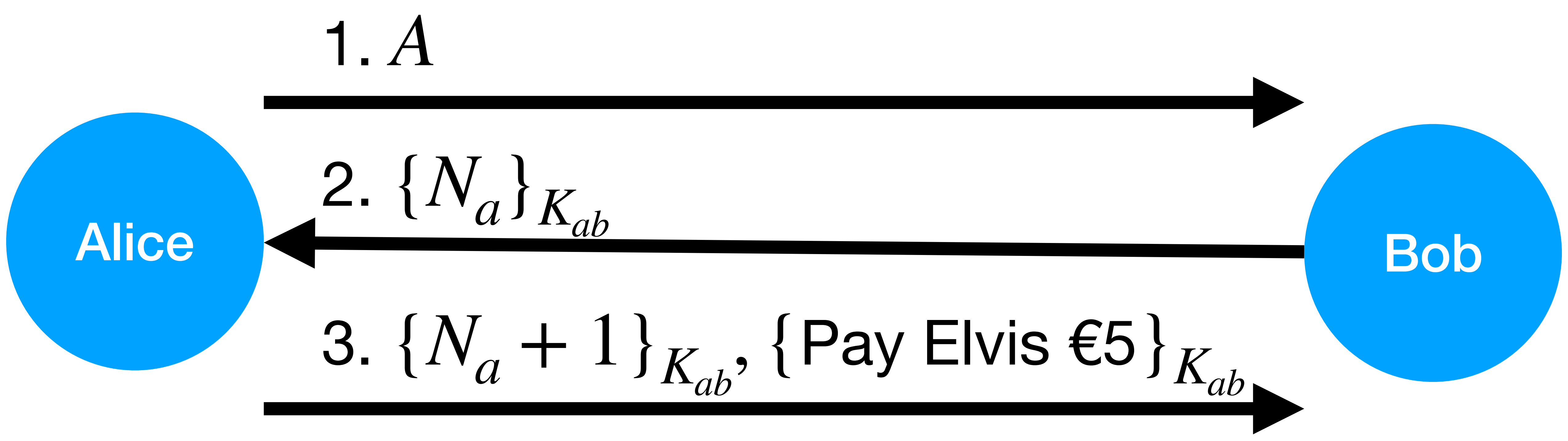
A Simple Protocol

$$A \rightarrow B : \{\text{“I'm Alice”}\}_{K_{ab}}$$
$$E(A) \rightarrow B : \{\text{“I'm Alice”}\}_{K_{ab}}$$

- Attacker can intercept and replay messages.
- Assume the attacker “owns” the network.

A Nonce

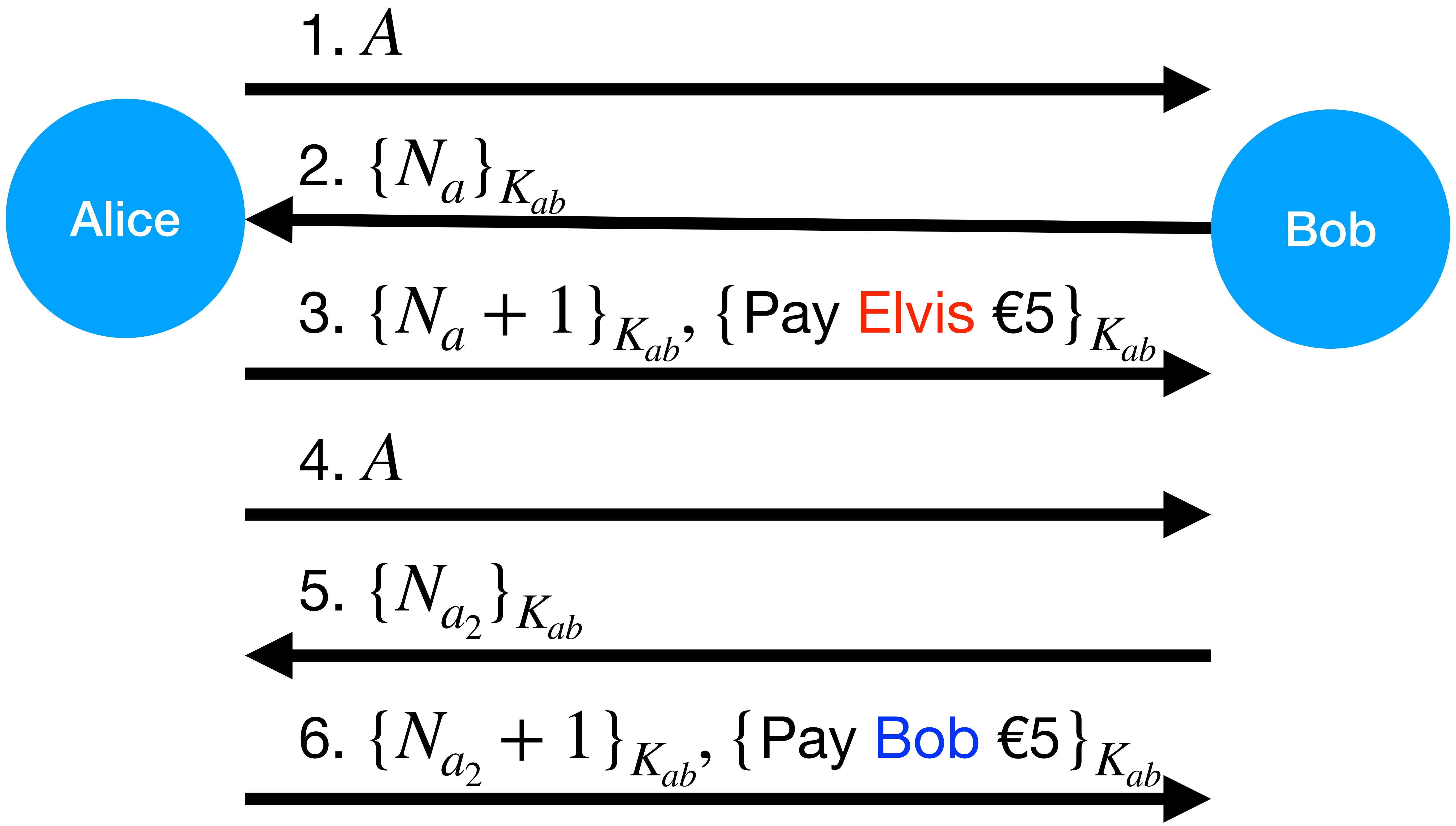
Number that is only used once (often used in a challenge/response setting).



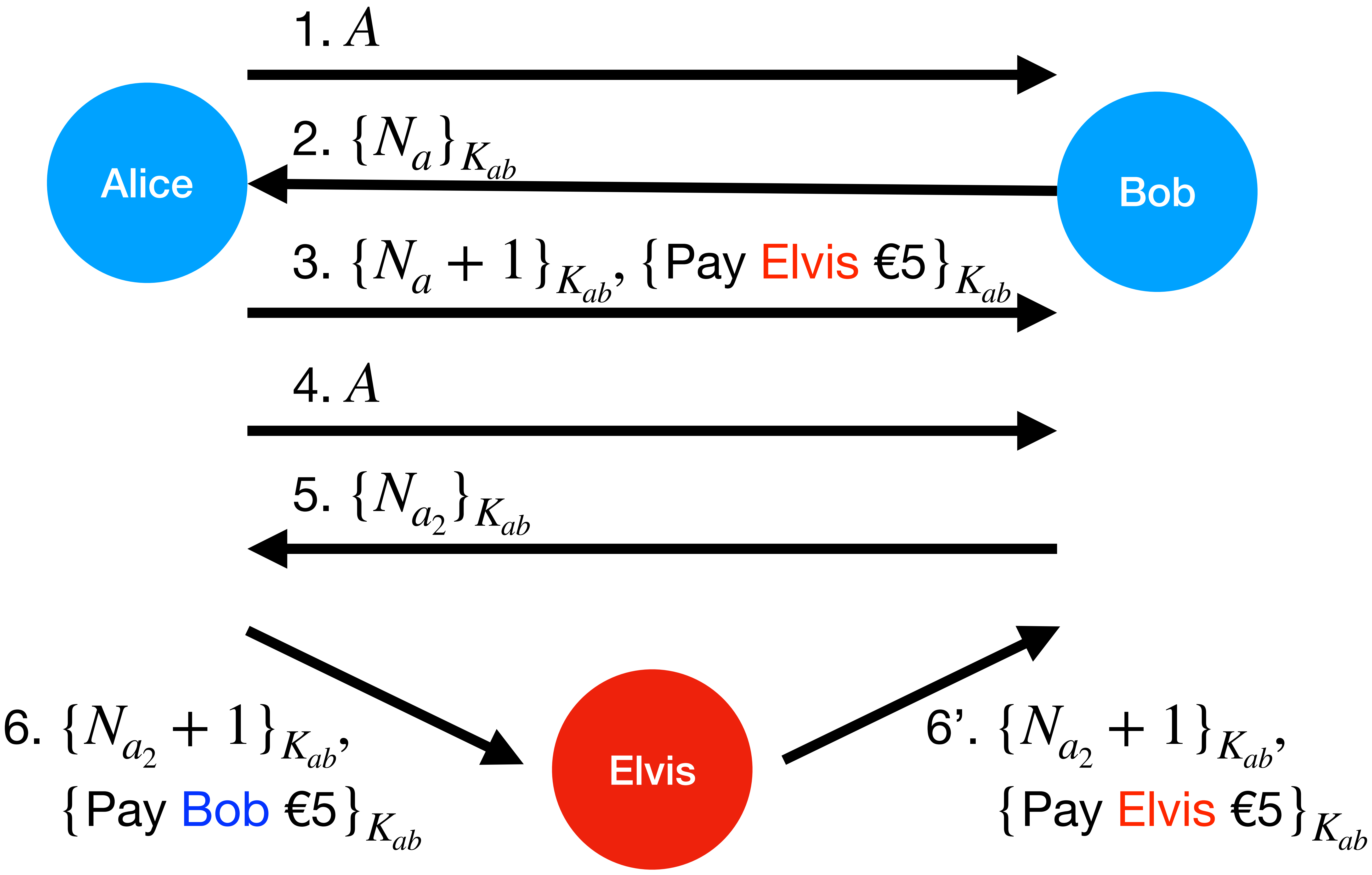
1. $A \rightarrow B : A$
2. $B \rightarrow A : \{N_a\}_{K_{ab}}$
3. $A \rightarrow B : \{N_a + 1\}_{K_{ab}}, \{\text{Pay Elvis €5}\}_{K_{ab}}$

B: Since $N_a + 1$ was encrypted using the shared key with A, I am sure she wants to pay Elvis €5.

A Nonce

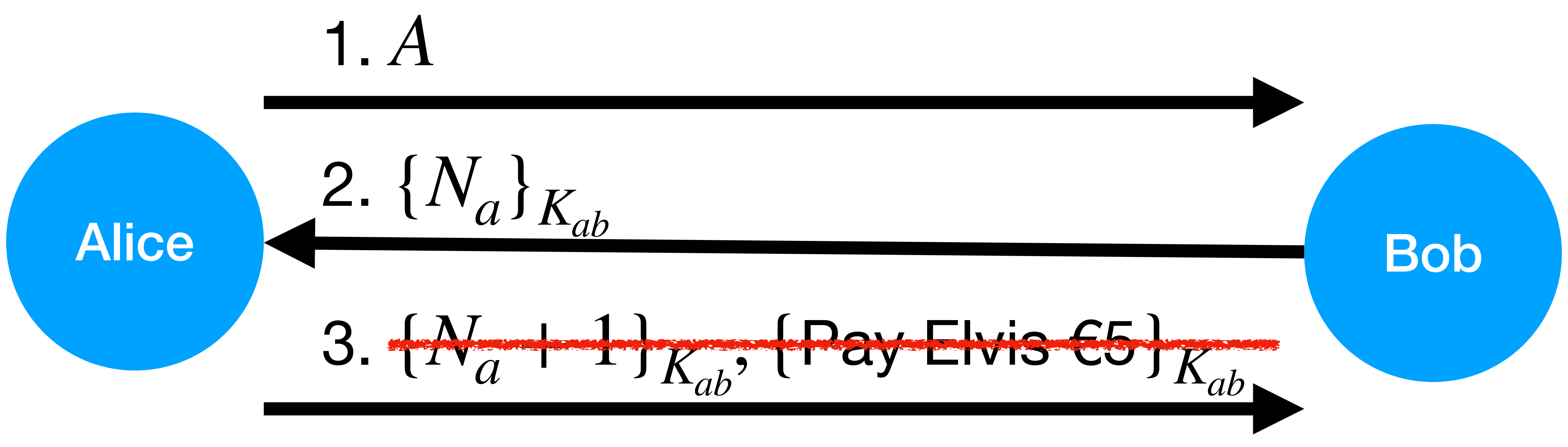

1. A
2. $\{N_a\}_{K_{ab}}$
Alice
Bob
3. $\{N_a + 1\}_{K_{ab}}$, $\{$Pay Elvis €5$\}_{K_{ab}}$
4. A
5. $\{N_{a_2}\}_{K_{ab}}$
6. $\{N_{a_2} + 1\}_{K_{ab}}$, $\{$Pay Bob €5$\}_{K_{ab}}$

A Nonce



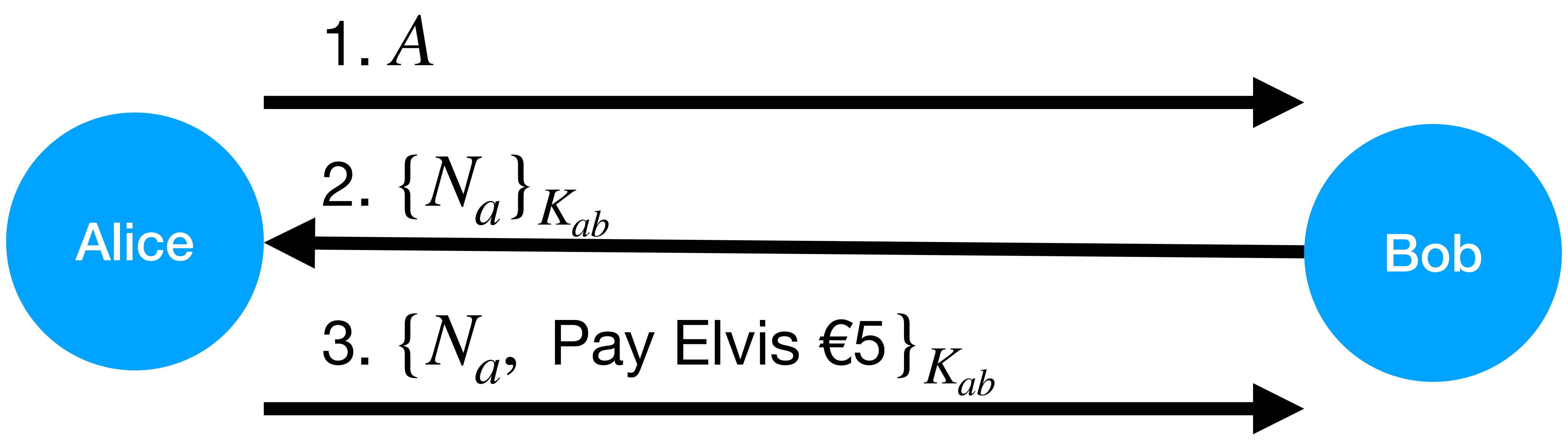
1. A
2. $\{N_a\}_{K_{ab}}$
Alice
Bob
3. $\{N_a + 1\}_{K_{ab}}$, {Pay Elvis €5}$_{K_{ab}}$
4. A
5. $\{N_{a_2}\}_{K_{ab}}$
6. $\{N_{a_2} + 1\}_{K_{ab}}$,
{Pay Bob €5}$_{K_{ab}}$
Elvis
6'. $\{N_{a_2} + 1\}_{K_{ab}}$,
{Pay Elvis €5}$_{K_{ab}}$

A Better Protocol



1. $A \rightarrow B : A$
2. $B \rightarrow A : \{N_a\}_{K_{ab}}$
3. $A \rightarrow B : \cancel{\{N_a + 1\}_{K_{ab}}, \{\text{Pay Elvis €5}\}_{K_{ab}}}$

A Better Protocol



1. $A \rightarrow B : A$

2. $B \rightarrow A : \{N_a\}_{K_{ab}}$

3. $A \rightarrow B : \{N_a,\ \text{Pay Elvis €5}\}_{K_{ab}}$

A Better Protocol

1. $A \rightarrow B : A$
2. $B \rightarrow A : N_a$
3. $A \rightarrow B : \{N_a,$ Pay Elvis €5$\}_{K_{ab}}$

What can Bob be sure of after such a protocol run?

a) He is talking to Alice

b) A wants to send Elvis €5

c) A's messages are fresh (not replayed)

Key Establishment Protocol

- This protocol was possible because A and B shared a key.
- Often, the principals need to set up a session key using a **Key Establishment Protocol**.
- To be sure they are communicating with the correct principal, they must either know each others public keys or use a **Trusted Third Party** (TTP).

The Needham-Schroeder Public Key Protocol

$E_X(\_)$ means public key encryption

Assume Alice and Bob know each others public keys, can they set up a symmetric key?

1. $A \rightarrow B : E_B(N_a, A)$
2. $B \rightarrow A : E_A(N_a, N_b)$
3. $A \rightarrow B : E_B(N_b)$

A: “The only person who could know N_a is the person who decrypted the first message.”

B: “The only person who could know N_b is the person who decrypted the second message.”

N_a and N_b can then be used to generate a symmetric key.

Goals: Alice and Bob are sure they are talking to each other and only they know the key.

An Attack Against the NH Protocol

The attacker C acts as a man-in-the-middle:

1. $A \rightarrow C : E_C(N_a, A)$

    1) $C(A) \rightarrow B : E_B(N_a, A)$

    2) $B \rightarrow C(A) : E_A(N_a, N_b)$

2. $C \rightarrow A : E_A(N_a, N_b)$

3. $A \rightarrow C : E_C(N_b)$

    3) $C(A) \rightarrow B : E_B(N_b)$

An Attack Against the NH Protocol

The attacker C acts as a man-in-the-middle:

1) $C(A) \rightarrow B : E_B(N_a, A)$

2) $B \rightarrow C(A) : E_A(N_a, N_b)$

3) $C(A) \rightarrow B : E_B(N_b)$

Corrected Version

A very simple fix:

1. $A \rightarrow B : E_B(N_a, A)$
2. $B \rightarrow A : E_A(N_a, N_b)$
3. $A \rightarrow B : E_B(N_b)$

Corrected Version

A very simple fix:

1. $A \rightarrow B : E_B(N_a, A)$
2. $B \rightarrow A : E_A(N_a, N_b, B)$
3. $A \rightarrow B : E_B(N_b)$

Forward Secrecy

1. $A \rightarrow B : E_B(N_a, A)$

2. $B \rightarrow A : E_A(N_a, N_b, B)$

3. $A \rightarrow B : E_B(N_b)$

4. $B \rightarrow A : \{M\}_{key(N_a, N_b)}$

Secure against the “standard” attacker:
intercept, replay, delete, alter

What about governments?

After the protocol runs, governments can legally force people to handover their private keys.

Can they read messages encrypted using $key(N_a, N_b)$?

a) Yes ✓
b) No

Forward Secrecy

1. $A \rightarrow B : E_B(N_a, A)$

2. $B \rightarrow A : E_A(N_a, N_b, B)$

3. $A \rightarrow B : E_B(N_b)$

4. $B \rightarrow A : \{M\}_{key(N_a, N_b)}$

Secure against the “standard” attacker:
intercept, replay, delete, alter

What about governments?

After the protocol runs, governments can legally force people to handover their private keys.

Forward Secrecy

1. $A \rightarrow B : E_B(N_a, A)$

2. $B \rightarrow A : E_A(N_a, N_b, B)$

3. $A \rightarrow B : E_B(N_b)$

4. $B \rightarrow A : \{M\}_{key(N_a,N_b)}$

Secure against the “standard” attacker:
intercept, replay, delete, alter

What about governments?

After the protocol runs, governments can legally force people to handover their private keys.

Can we protect against this?

Forward Secrecy

A protocol has **Forward Secrecy** if it keeps the message secret from an attacker who has:

- A recording of the protocol run
- The long term keys of the principals.

Protection against a government that can force people to give up their keys, or hackers that might steal them.

Station-to-Station Protocol

1. $A \to B : g^x$

2. $B \to A : g^y$

Station-to-Station Protocol

$S_X(\_)$ means signed by X

1. $A \rightarrow B : g^x$
2. $B \rightarrow A : g^y, \{S_B(g^y, g^x)\}_{g^{xy}}$
3. $A \rightarrow B : \{S_A(g^y, g^x)\}_{g^{xy}}$
4. $B \rightarrow A : \{M\}_{g^{xy}}$

- x, y, g^{xy} are not stored after the protocol run.
- A and B's keys don't let the attacker read M.
- STS has **forward secrecy**.

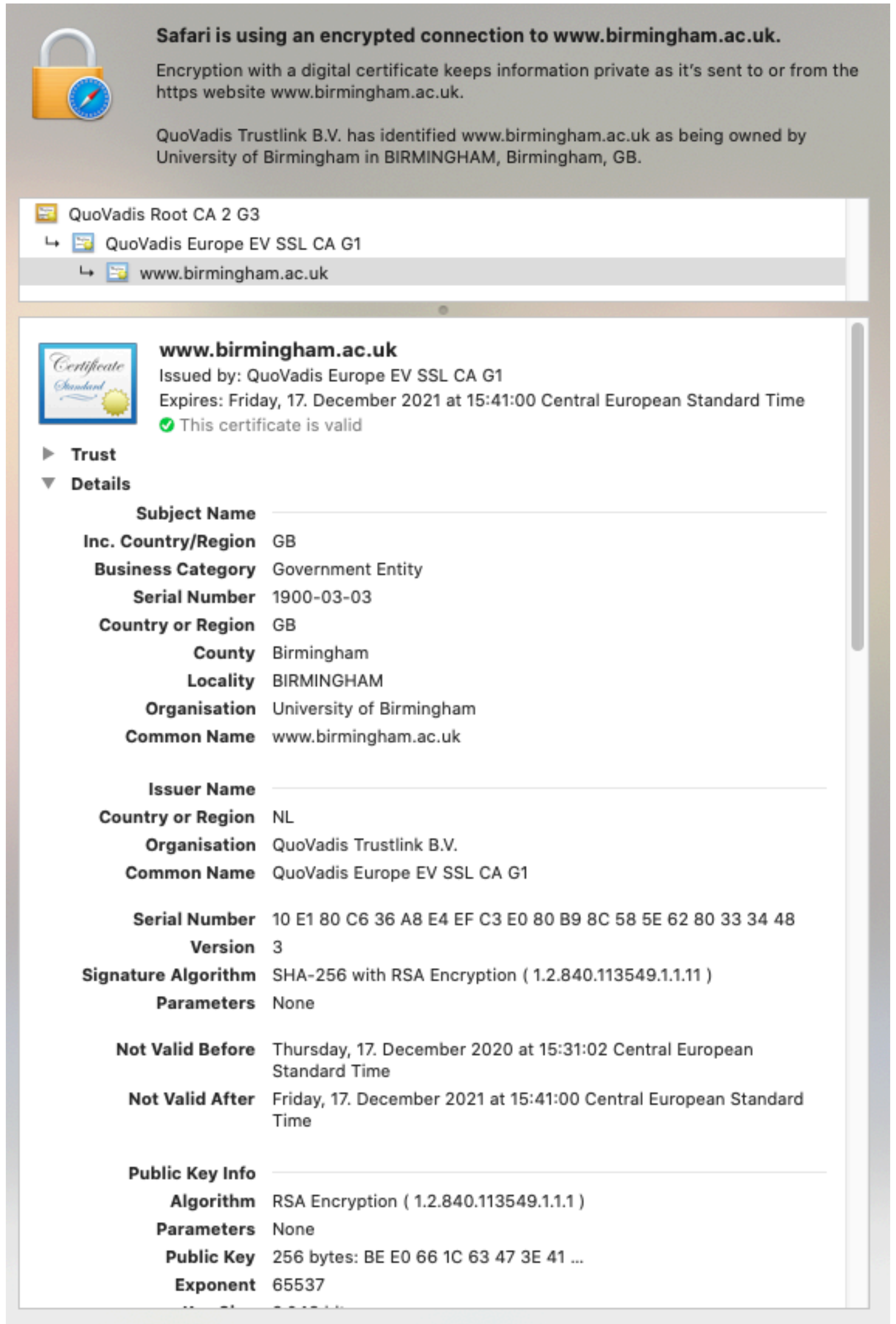
Certificates

- What if Alice and Bob don't know each other's public keys
  to start off with?

- Could meet face-to-face and set up keys.

- Or get a trusted third party (TTP) to sign their identity and public key:
  a certificate.

See browser certs



Full Station-to-Station Protocol

1. $A \rightarrow B : g^x$
2. $B \rightarrow A : g^y, Cert_B, \{S_B(g^y, g^x)\}_{g^{xy}}$
3. $A \rightarrow B : Cert_A, \{S_A(g^y, g^x)\}_{g^{xy}}$

- The “full” STS protocol adds certificates for A and B.
- These contain their public key signed by a TTP,
so Alice and Bob don’t have to know each other’s public key.

The Needham-Schroeder key establishment protocol

A and B use trusted third party S to establish a key K_{ab}:

1. $A \to S : A, B, N_a$

2. $S \to A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$

3. $A \to B : \{K_{ab}, A\}_{K_{bs}}$

4. $B \to A : \{N_b\}_{K_{ab}}$

5. $A \to B : \{N_b + 1\}_{K_{ab}}$

The Needham-Schroeder key establishment protocol

Alice can reuse an old key:

1. $A \to S : A, B, N_a$
2. $S \to A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$
3. $A \to B : \{K_{ab}, A\}_{K_{bs}}$
4. $B \to A : \{N_b\}_{K_{ab}}$
5. $A \to B : \{N_b + 1\}_{K_{ab}}$

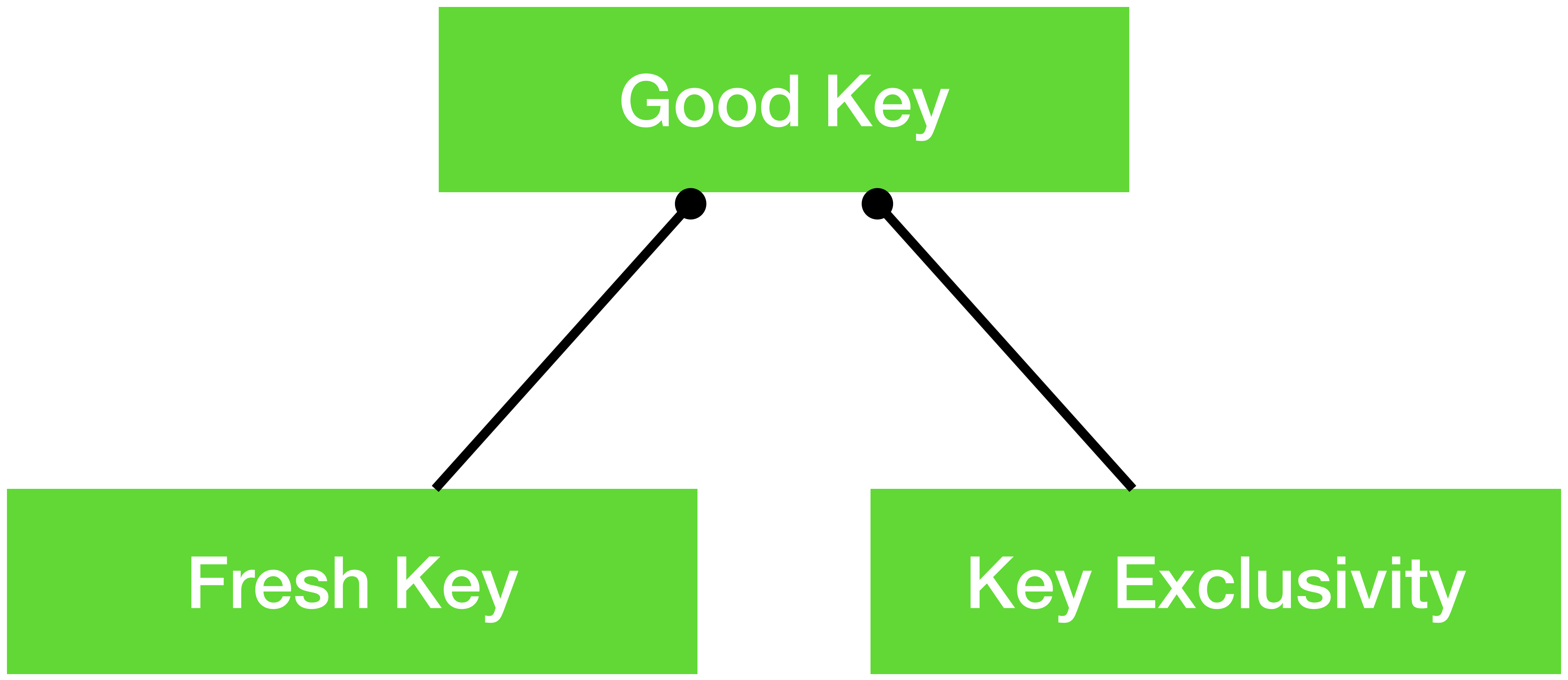
…much later

1) $A \to B : \{K_{ab}, A\}_{K_{bs}}$
2) $B \to A : \{N_b\}_{K_{ab}}$
3) $A \to B : \{N_b + 1\}_{K_{ab}}$

Some Key Establishment Goals

- **Key Freshness:** the key established is new (either from some trusted third party or because it uses a new nonce).
- **Key Exclusivity:** the key is only known to the principals in the protocol.
- **Good Key:** the key is both fresh and exclusive.

A Hierarchy of Goals



Authentication Goals

- **Far-end Operative:** A knows that "B" is currently active.

  For instance B might have signed a nonce generated by A, e.g.

  1. $A \rightarrow B : N_a$
  2. $B \rightarrow A : S_B(N_a)$

Not enough on its own (e.g. Needham-Schroeder protocol).

Authentication Goals

- **Once Authentication:** A knows that B wishes to communicate with A.

  For instance, B might have the name A in the message, e.g.

  1. $B \rightarrow A : S_B(A)$

Entity Authentication

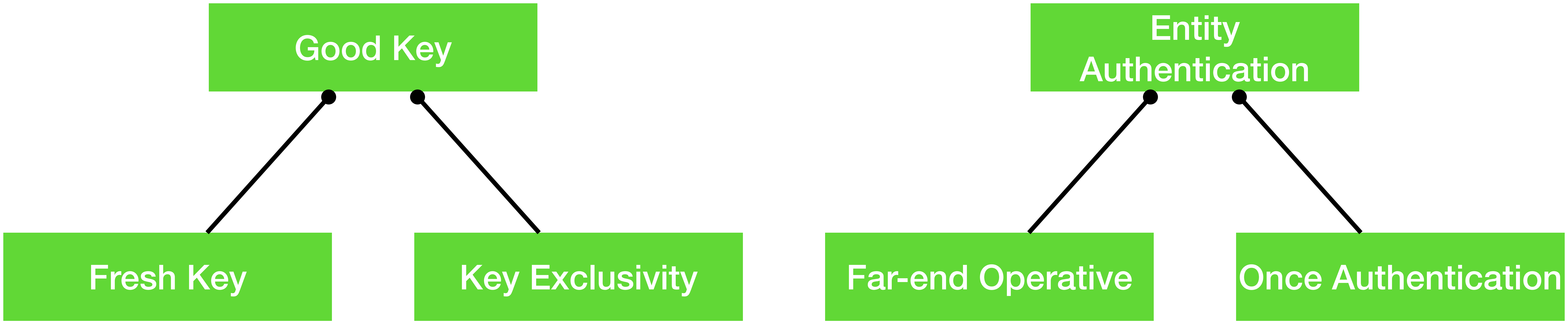
Both of these together give:

- **Entity Authentication:** A knows that B is currently active ***and*** wants to communicate with A.

  e.g.

  1. $A \rightarrow B : N_a$
  2. $B \rightarrow A : S_B(A, N_a)$

A Hierarchy of Goals

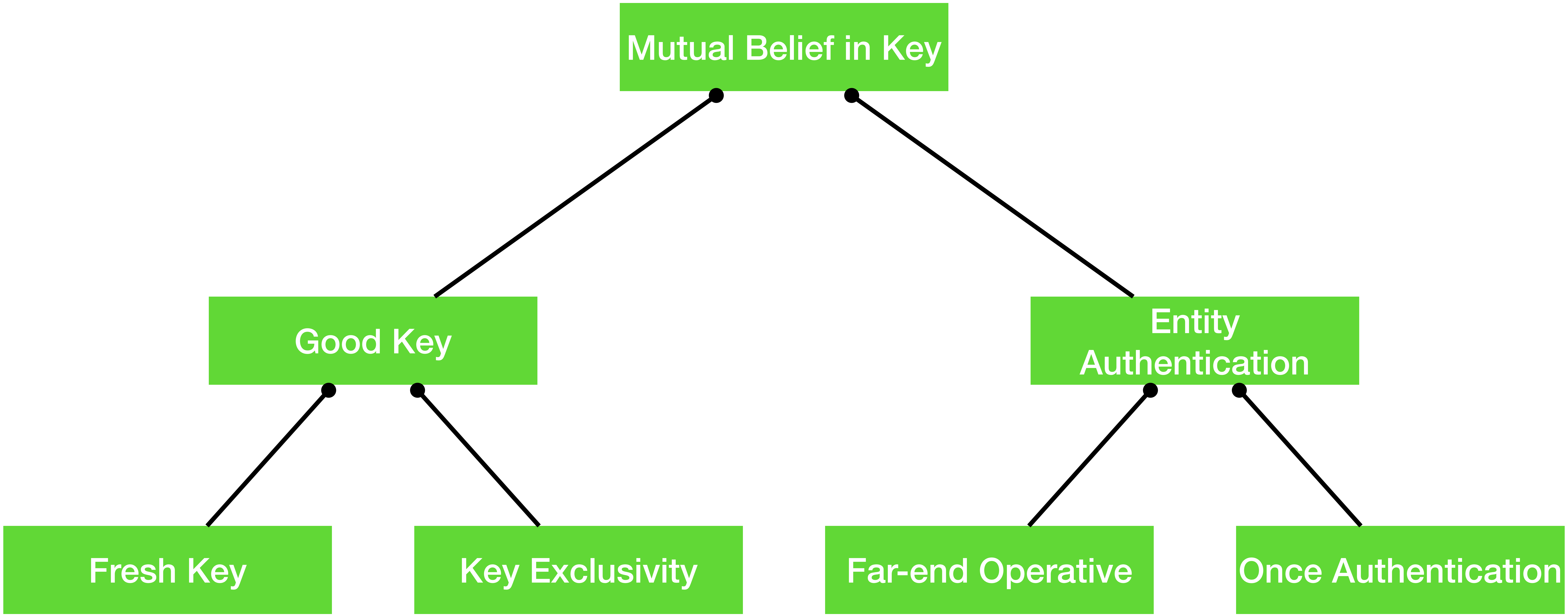


The Highest Goal

A protocol provides **Mutual Belief** in a key K for Alice with respect to Bob if, after running the protocol, Bob can be sure that:

- K is a good key with A
- Alice can be sure that Bob wishes to communicate with Alice using K
- Alice knows that Bob believes that K is a good key for B.

A Hierarchy of Goals



NH
Public Key Protocol

Remember the man-in-the-middle attack against the NH Public Key Protocol:

1. $A \rightarrow C : E_C(N_a, A)$

1) $C(A) \rightarrow B : E_B(N_a, A)$

2) $B \rightarrow C(A) : E_A(N_a, N_b)$

2. $C \rightarrow A : E_A(N_a, N_b)$

3. $A \rightarrow C : E_C(N_b)$

3) $C(A) \rightarrow B : E_B(N_b)$

Which goals does the unfixed protocol provide?

a) Fresh Key ✓
b) Key Exclusivity ✗
c) Far-end Operative ✓
d) Once Authentication ✗

Today's Lecture

- Protocols in Alice and Bob notation
- Attacks on Protocols
- Forward Secrecy
- Goals and Protocols