Cryptographic Protocols

Eike Ritter
Computer Security and Networks

Last Lecture

- How connections over the Internet work
  - Message passed from computer to computer
  - Anyone on the path, or at either end, can read/change the messages.

- Computer, Networks can be scanned for open connection.

- Network traffic can be sniffed (e.g. WireShark) and altered.

Today’s Lecture

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

A Simple Protocol

“A” sends message “M” to “B”:

written as:

Alice $\rightarrow$ Bob : “I’m Alice”

Rules

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

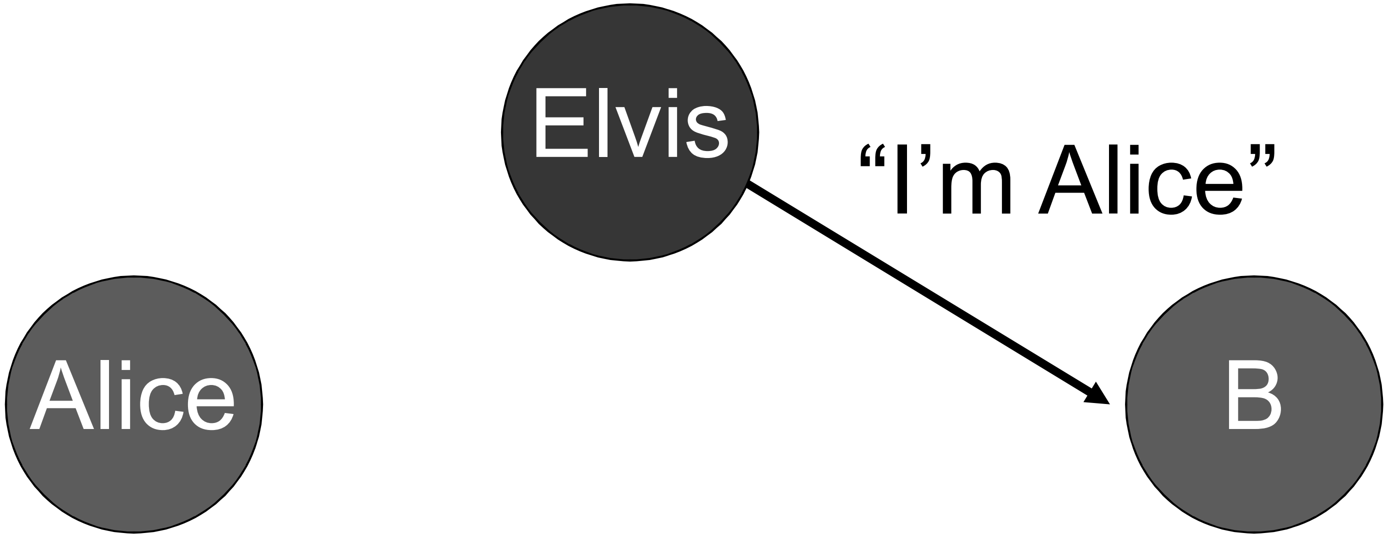
1. $A \rightarrow B$: “Hello”
2. $B \rightarrow A$: “Offer”
3. $A \rightarrow B$: “Accept”

A Simple Protocol

A $\rightarrow$ B : “I’m Alice”

Message “I’m Alice” can be read by “The Attacker”.

A Simple Protocol



"The Attacker" can pretend to be anyone.

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

A Simple Protocol

$\{\ \_\ \}_{Kab}$ means symmetric key encryption

$A \rightarrow B$: {"I'm Alice"}$_{Kab}$

If Alice and Bob share a key "Kab",
then Alice can encrypt her message.

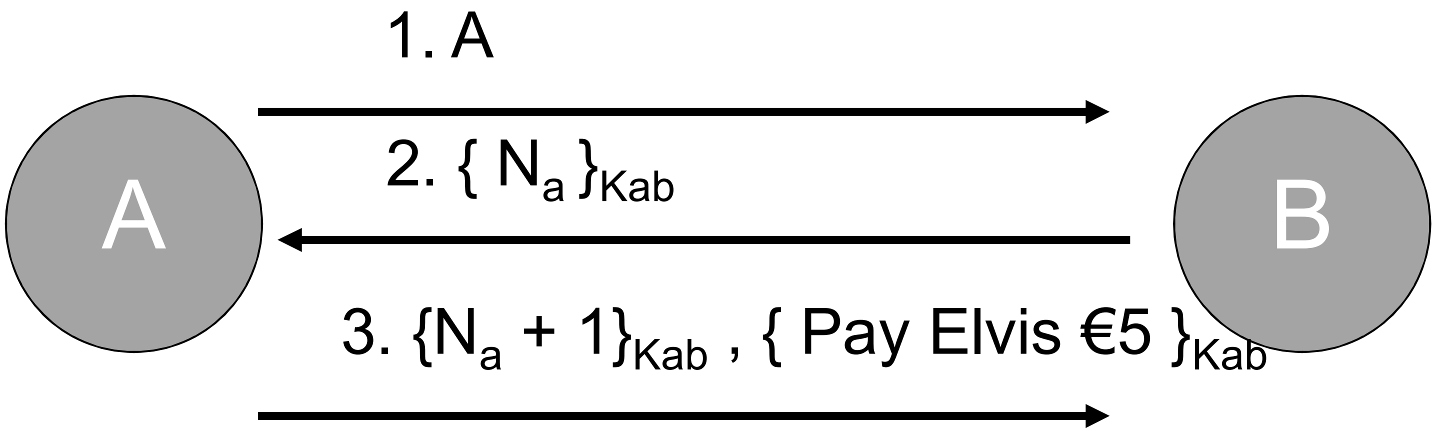
A Simple Protocol

$A \rightarrow E(B) : \{\text{“I’m Alice”}\}_{Kab}$

$E(A) \rightarrow B : \{\text{“I’m Alice”}\}_{Kab}$

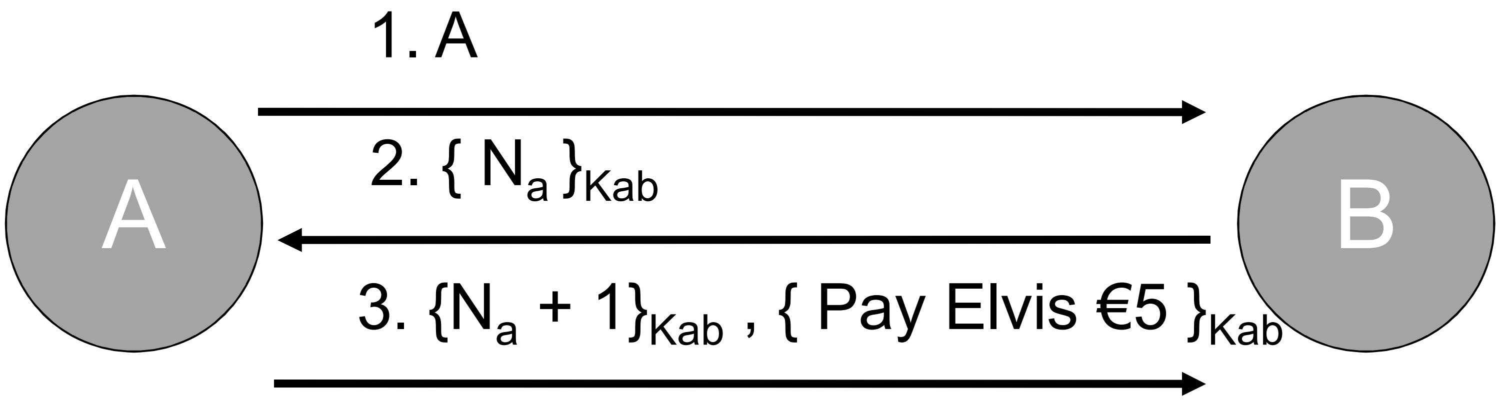
- Attacker can intercept and replay messages.
- Assume the attacker “owns” the network.

A Nonce

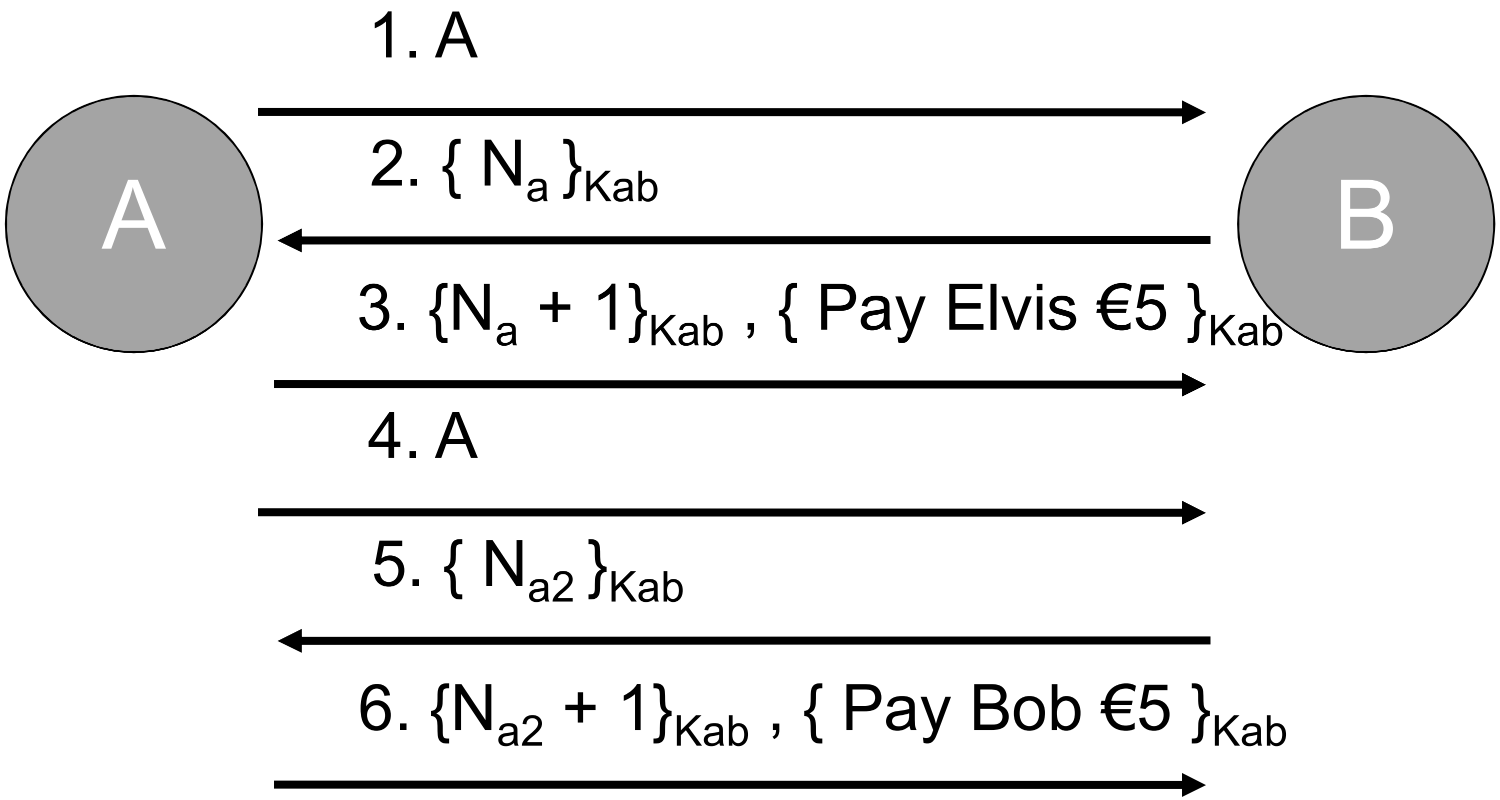


1. $A \rightarrow B : A$
2. $B \rightarrow A : \{ N_a \}_{Kab}$
3. $A \rightarrow B : \{ N_a + 1 \}_{Kab}$, { Pay Elvis €5 $\}_{Kab}$

A Nonce

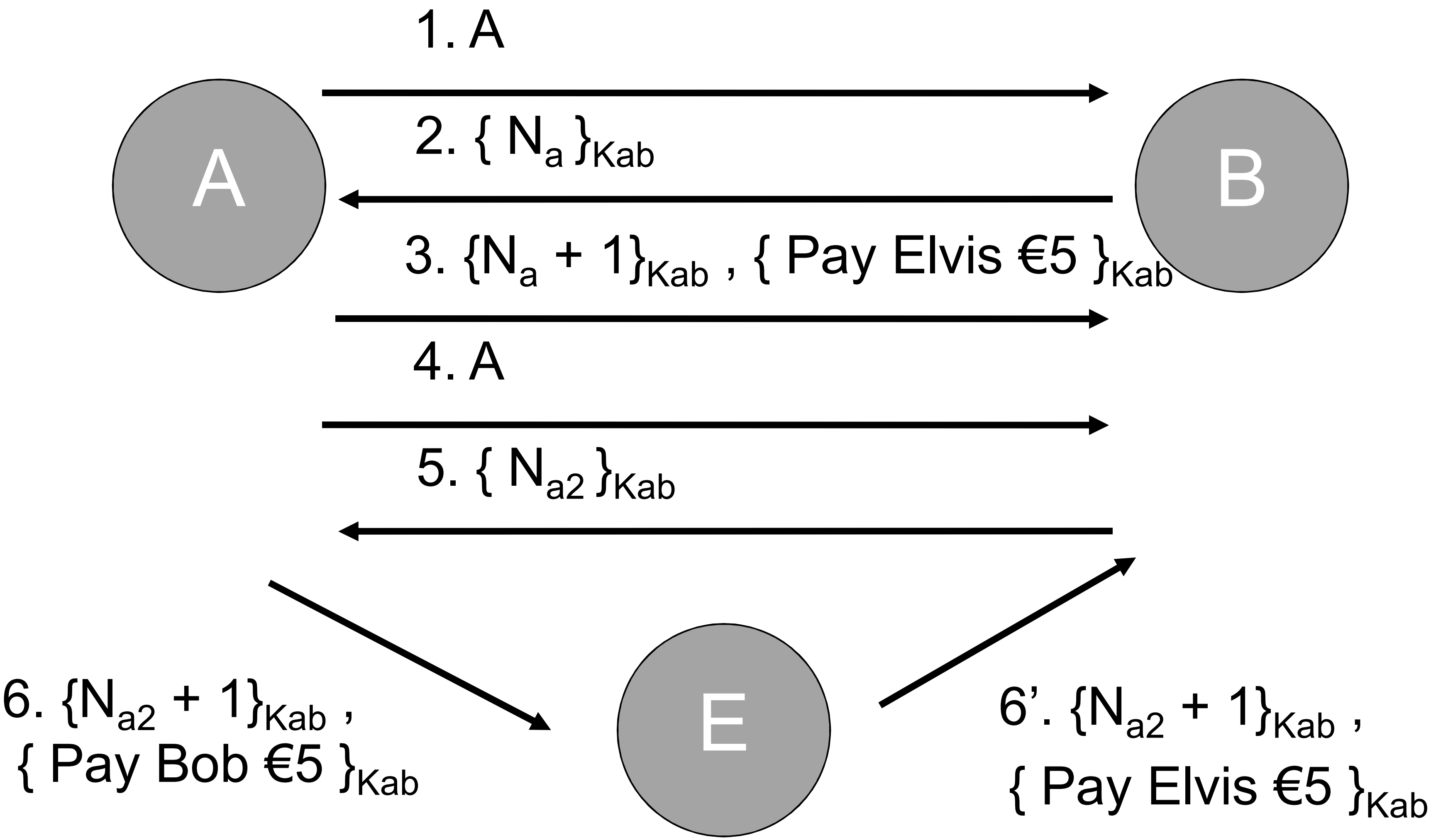


A Nonce

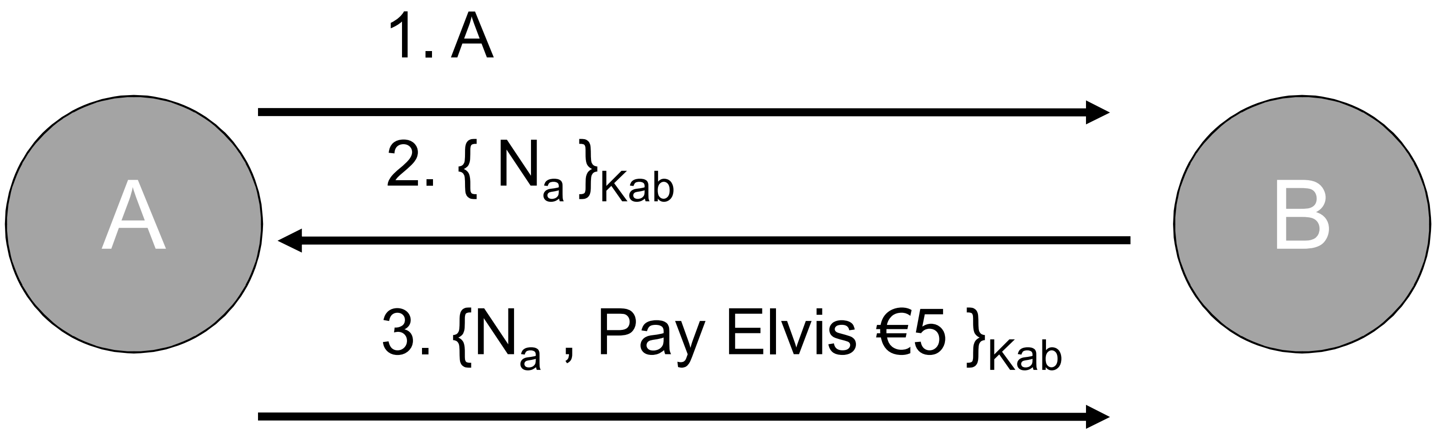


A
B
1. A
2. $\{ N_a \}_{Kab}$
3. $\{N_a + 1\}_{Kab}$, { Pay Elvis €5 $\}_{Kab}$
4. A
5. $\{ N_{a2} \}_{Kab}$
6. $\{N_{a2} + 1\}_{Kab}$, { Pay Bob €5 $\}_{Kab}$

A Nonce

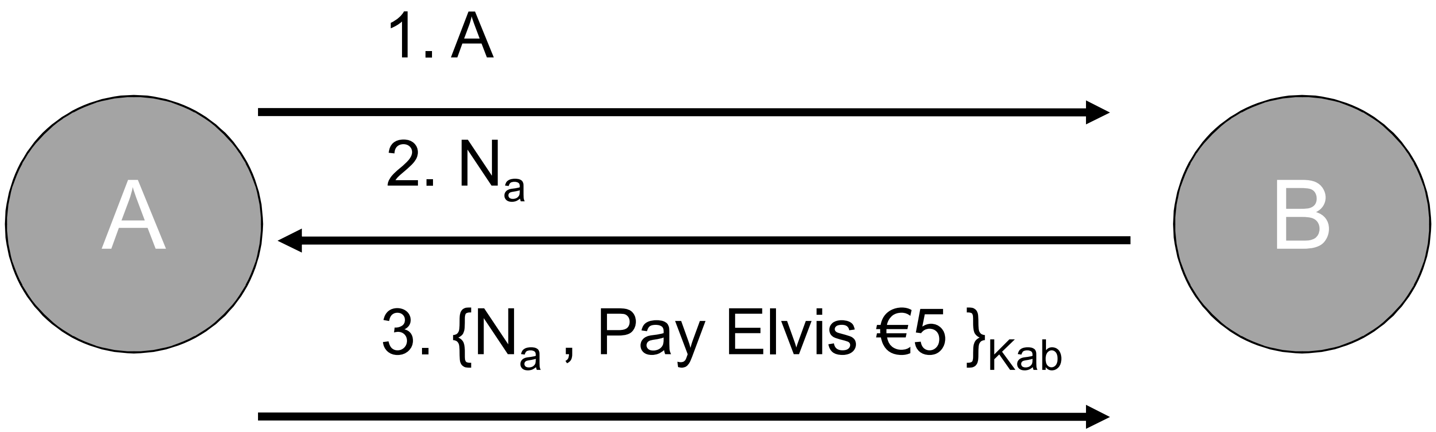

1. A
2. { N_a }$_{Kab}$
3. {N_a + 1}$_{Kab}$, { Pay Elvis €5 }$_{Kab}$
4. A
5. { N_{a2} }$_{Kab}$
6. {N_{a2} + 1}$_{Kab}$,
{ Pay Bob €5 }$_{Kab}$
6'. {N_{a2} + 1}$_{Kab}$,
{ Pay Elvis €5 }$_{Kab}$
A
B
E

A Better Protocol



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

A Better Protocol



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

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”.
- They must either know each others public keys or use a “Trusted Third Party” (TTP).

The Needham-Schroeder Public Key Protocol

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

The Needham-Schroeder Public Key Protocol

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

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

$E_X(\_)$ means public key encryption

The Needham-Schroeder Public Key Protocol

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

$E_X(\_)$ means public key encryption

The Needham-Schroeder Public Key Protocol

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

$E_X(\_)$ means public key encryption

The Needham-Schroeder Public Key Protocol

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

$E_X(\_)$ means public key encryption

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

Demo

An Attack Against the Needham-Schroeder Protocol

The attacker 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 Needham-Schroeder Protocol

The attacker 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\`. $C(A) \rightarrow B : E_B(N_b)$

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

The 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

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

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

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

$B \rightarrow A : \{M\}_{key(Na,\ Nb)}$

Secure against the "standard" attacker.

- intercept, replay, delete, alter.

Forward Secrecy

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

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

$A \rightarrow B : E_B(N_b)$

$B \rightarrow A : \{M\}_{key(Na,\ Nb)}$

Secure against the "standard" attacker.

- intercept, replay, delete, alter.

But what about the governments?

Forward Secrecy

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

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

$A \rightarrow B : E_B(N_b)$

$B \rightarrow A : \{M\}_{key(Na,\ Nb)}$

Secure against the "standard" attacker.

- intercept, replay, delete, alter.

But what about the governments?

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

Forward Secrecy

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

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

$A \rightarrow B : E_B(N_b)$

$B \rightarrow A : \{M\}_{key(Na,\ Nb)}$

Secure against the "standard" attacker.

- intercept, replay, delete, alter.

But what about the governments?

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

Forward Secrecy

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

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

$A \rightarrow B : E_B(N_b)$

$B \rightarrow A : \{M\}_{key(Na,\ Nb)}$

Secure against the "standard" attacker.

- intercept, replay, delete, alter.

But what about the governments?

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

Forward Secrecy

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

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

$A \rightarrow B : E_B(N_b)$

$B \rightarrow A : \{M\}_{key(Na,\ Nb)}$

Secure against the "standard" attacker.

- intercept, replay, delete, alter.

But what about the governments?

After the protocol runs, governments can legally force people to hand over 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

Station-to-Station Protocol

1. $A \rightarrow B : g^x$
2. $B \rightarrow A : g^y$

Station-to-Station Protocol

$S_X(\_)$ means signed by X

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

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. $A \rightarrow B : \{ S_A(g^x, g^y) \}_{g^{xy}}$

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

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

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

Certificates

- What it 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 3rd party (TTP) to sign their identity and public key: a certificate.

See browsers certs

Full Station-to-Station

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

The "full" STS protocol add certificates for A & B.

These contain their public key signed by a TTP, so Alice and Bob don't need to know each others public keys.

The Needham-Schroeder key establishment protocol

A and B use trusted 3rd party S to establish a key K_{ab}:

1. $A \rightarrow S : A, B, N_a$
2. $S \rightarrow A : \{ N_a , B, K_{ab}, \{K_{ab} , A \}_{Kbs} \}_{Kas}$
3. $A \rightarrow B : \{ K_{ab} , A \}_{Kbs}$
4. $B \rightarrow A : \{ N_b \}_{Kab}$
5. $A \rightarrow B : \{ N_b + 1 \}_{Kab}$

Alice can reuse an old key:

1. $A \rightarrow S : A, B, N_a$
2. $S \rightarrow A : \{ N_a , B, K_{ab}, \{K_{ab} , A \}_{Kbs} \}_{Kas}$
3. $A \rightarrow B : \{ K_{ab} , A \}_{Kbs}$
4. $B \rightarrow A : \{ N_b \}_{Kab}$
5. $A \rightarrow B : \{ N_b + 1 \}_{Kab}$

… much later

1’. $A \rightarrow B : \{ K_{ab} , A \}_{Kbs}$

2’. $B \rightarrow A : \{ N_b \}_{Kab}$

3’. $A \rightarrow B : \{ N_b + 1 \}_{Kab}$

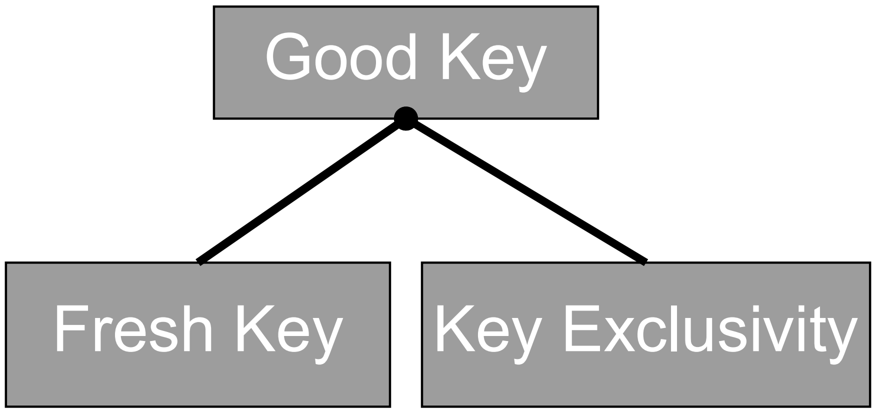
Some Key Establishment Goals

Key Freshness: the key established is new
either from some trusted 3rd 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. $B \rightarrow A : Sign_B (N_a)$

Not enough on its own, e.g. N.S. protocol.

Authentication Goals

Once Authentication: A knows that B wishes to communicate with A

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

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

Entity Authentication

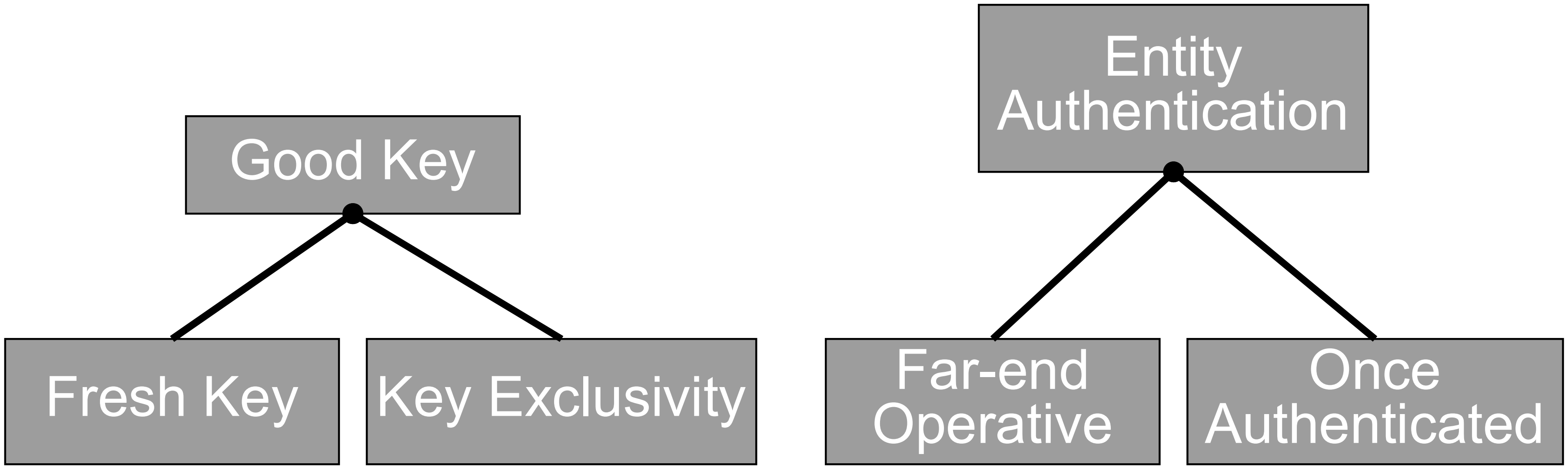
Both of these together give:

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

e.g.

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

A Hierarchy of Goals

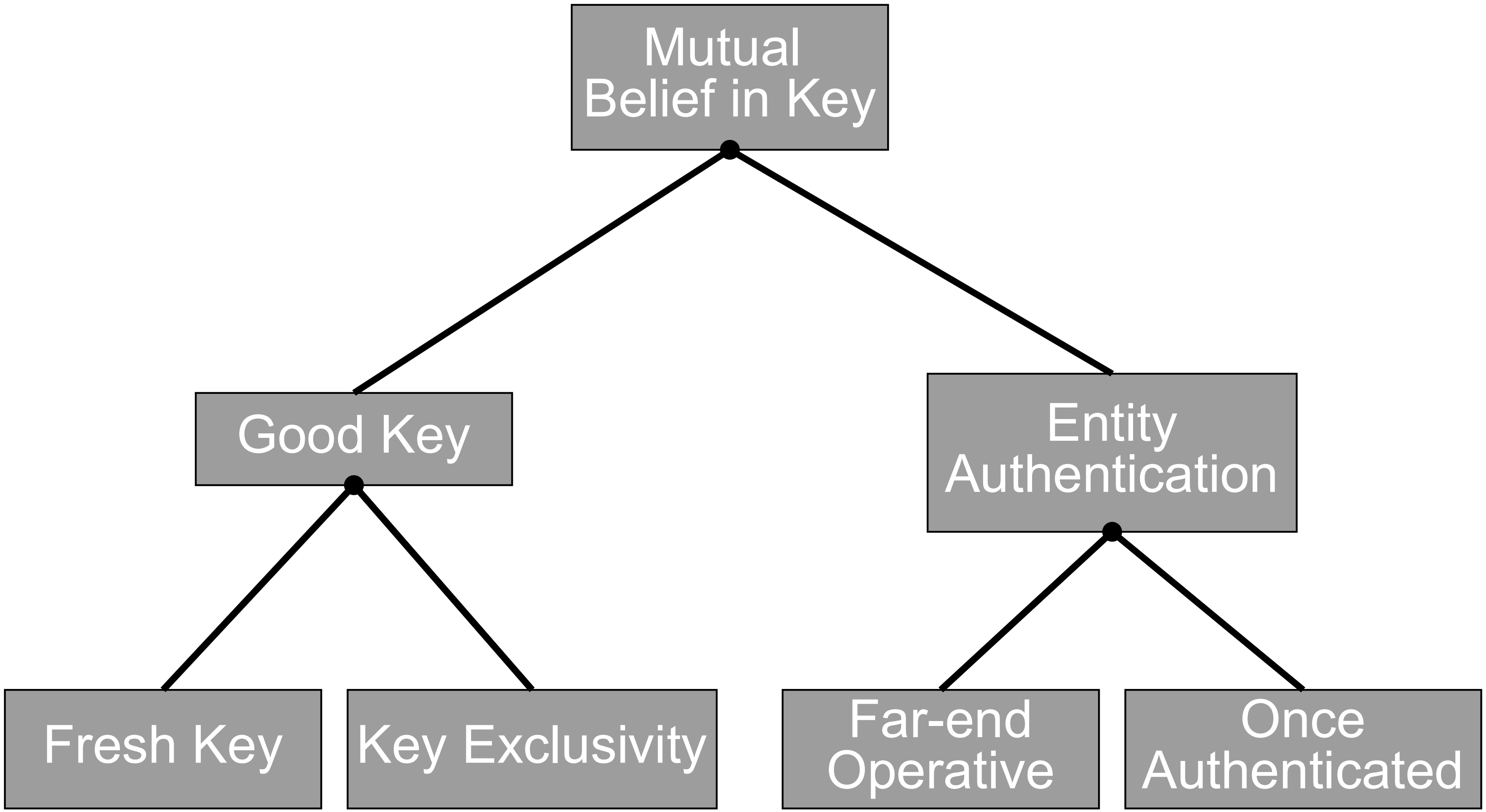


The Highest Goal

A protocol provides ***<u>Mutual Belief</u>*** 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 for Alive and Bob
- 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



Today's Lecture

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