Secure Communication

Tom Chothia

Computer Security, Lecture 8

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
- Forward Secrecy
- Secure Sockets Layer (SSL) / Transport Later Security (TLS)

Remote Authentication

- How can you tell who you are talking to over the Internet?
  - No online shopping, banking, e-mail, facebook, … without remote authentication.
- Simple authentication protocols.
  - Writing down protocols

A Simple Protocol

"A" sends message "M" to "B":

written as:

Alice → Bob : "I'm Alice"

Rules

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

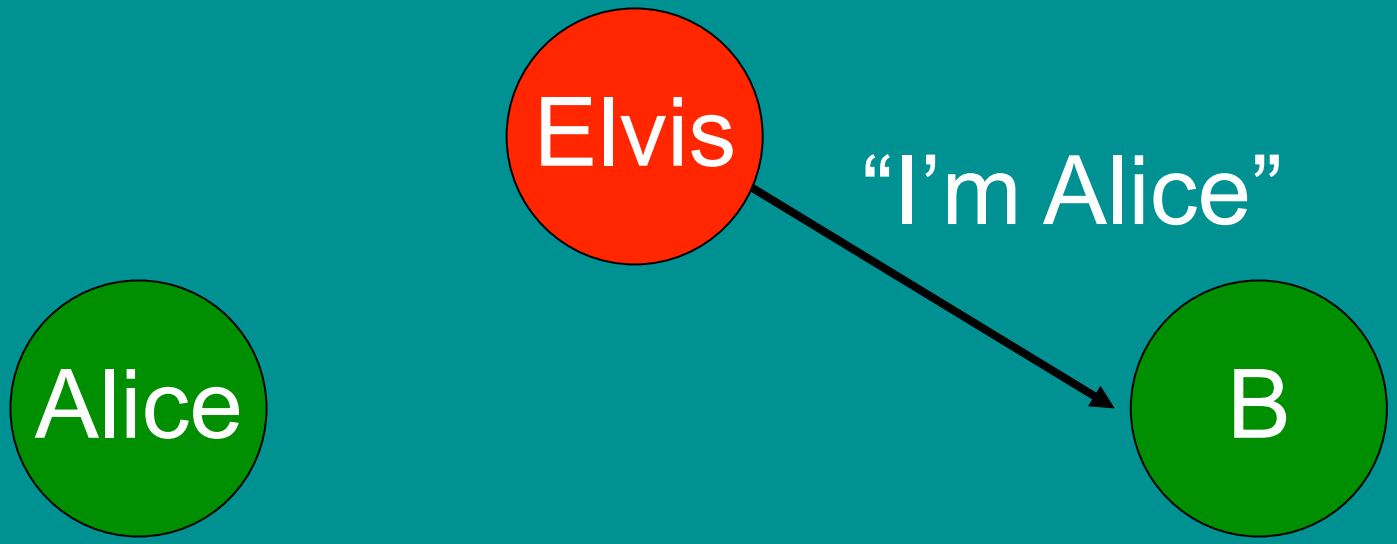
1. A → B : “Hello”
2. B → A : “Offer”
3. A → B : “Accept”

A Simple Protocol

A → 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) → B : "I'm Alice"

A Simple Protocol

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

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

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

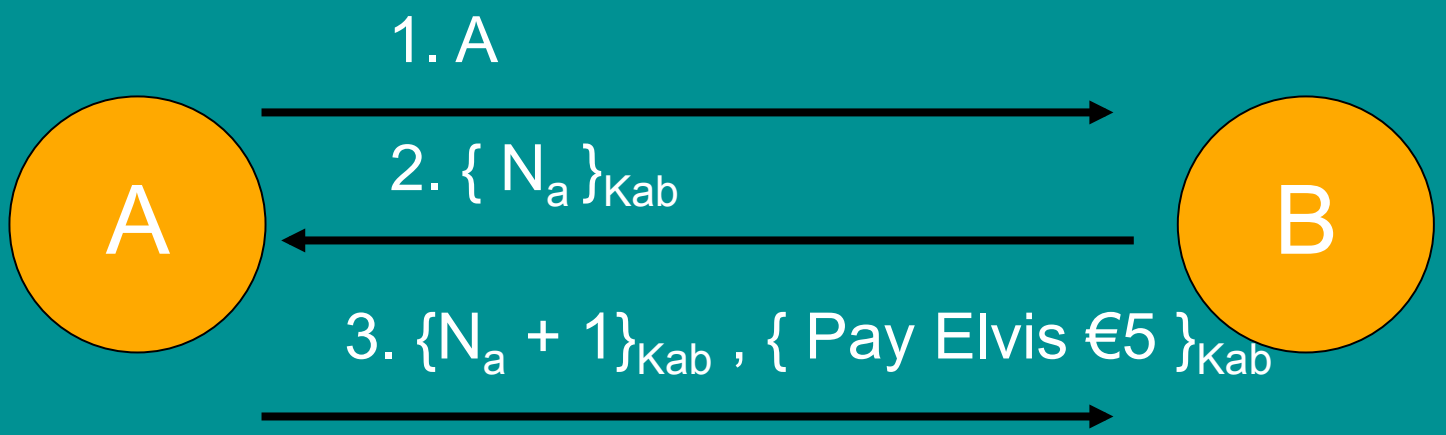
A Simple Protocol

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

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

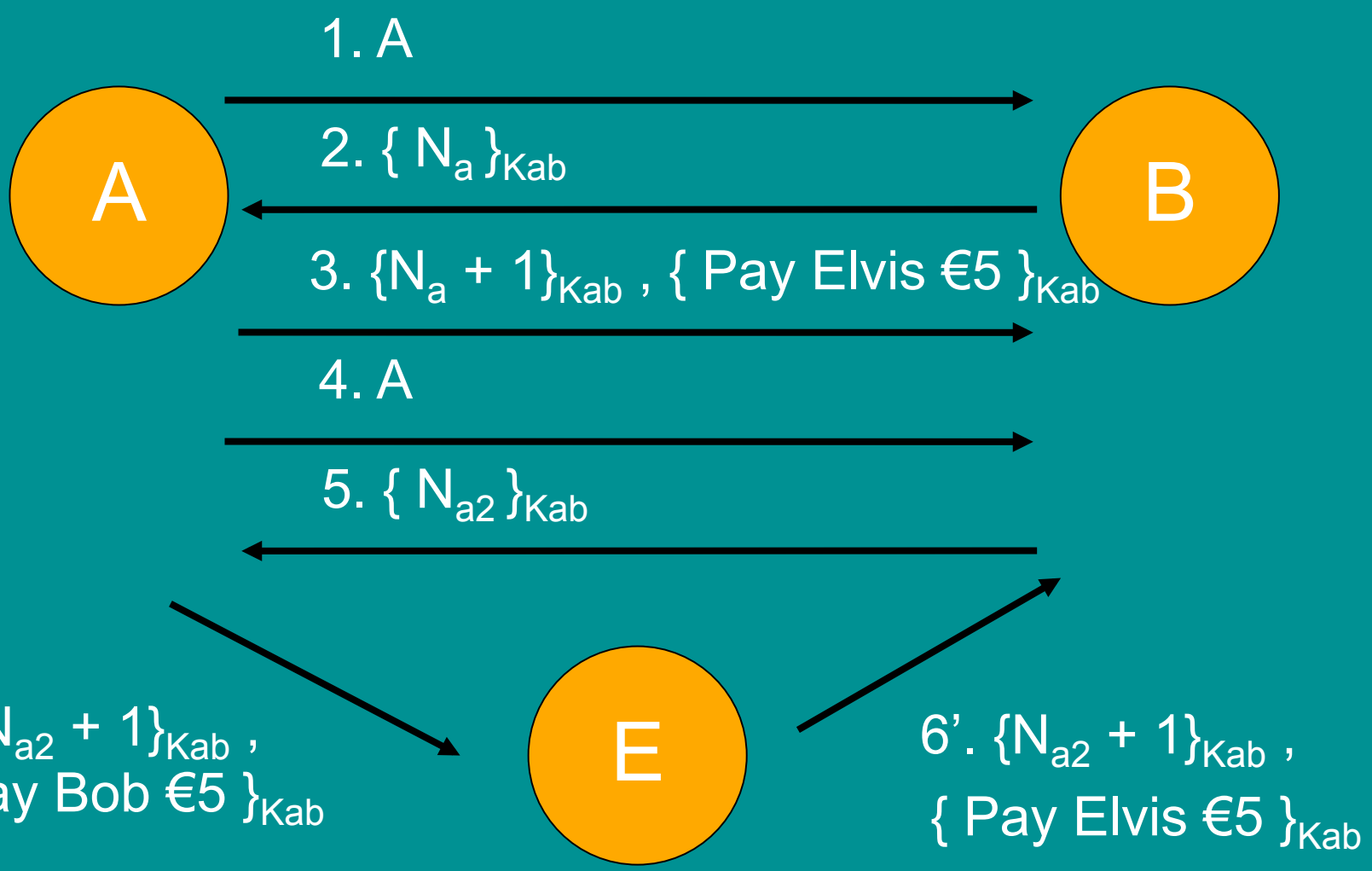
- Attacker can intercept and replay messages.
- Assume the attacker “owns” the network.

A Nonce



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

A Nonce


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

A Better Protocol

1. A → B : A, N_a
2. B → A : { N_a }$_{Kab}$
3. A → B : {N_a, 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?

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

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

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.

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

$S_X(\_)$ 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.

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 SSL/TLS Protocol

- The Secure Sockets Layer (SSL) protocol has been renamed the Transport Layer Security (TLS).
- It provides encrypted socket communication and authentication, based on public keys.
- It may use a range of ciphers (RSA,DES,DH,..)
  - These are negotiated at the start of the run.

X.509 Standard for Certificates

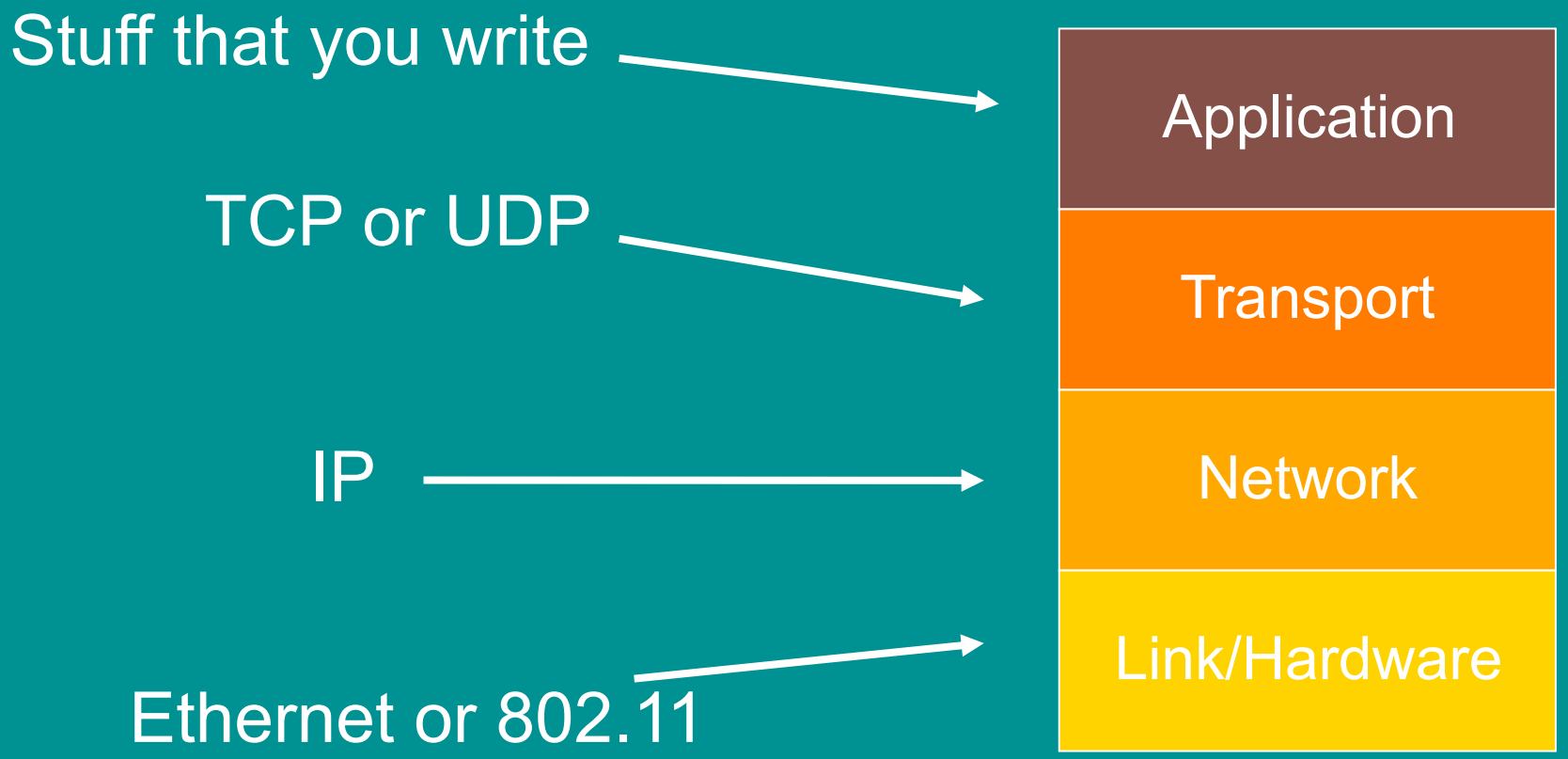
X.509 certificates contain a subject, subject's public key, Issuer name, etc

The issuer signs the hash of all the data

To check a certificate I hash all the data and check the issuers public key.

If I have the issuer's public key, and trust the issuer, I can then be sure of the subject's public key.

The Internet Protocol Stack, (Most of the Time):

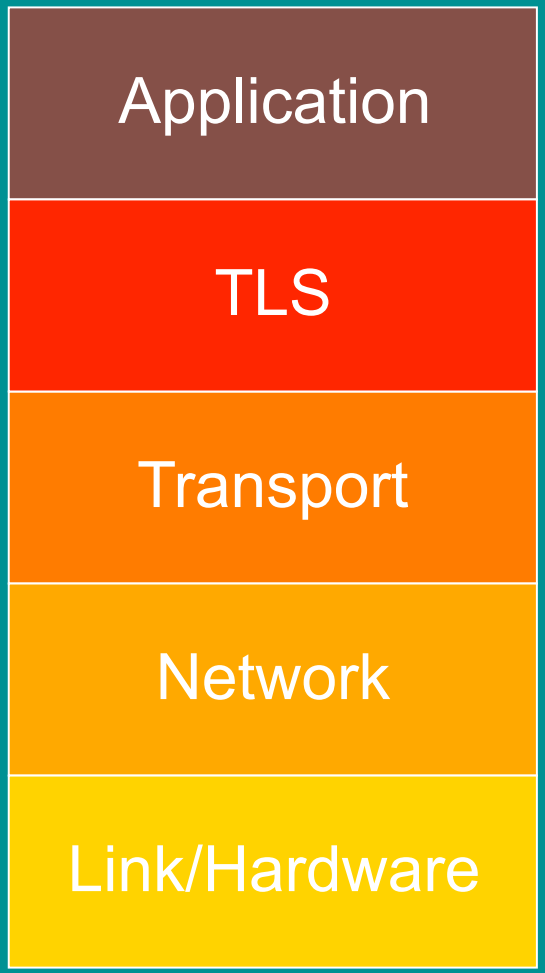


The Internet Protocol Stack with TLS

The TLS layer runs between the Application and Transport layer.

The encryption is transparent to the Application layer.

Normal TCP and IP protocols etc. can be used at the low layers



Transport Layer Security (TLS)

1. $C \rightarrow S : N_C$
2. $S \rightarrow C : N_S , Cert_S$
3. $C \rightarrow S : E_S(K\_seed), Sign_C(Hash1), \{Hash2\}_{KCS}$
4. $S \rightarrow C : \{Hash3\}_{KCS}$

Hash 1 = $\#(N_C, N_S, E_S(K\_seed))$

Hash 2 = $\#(N_C, N_S, E_S(K\_seed), Sign_C(Hash1))$

Hash 3 = $\#(N_C, N_S, E_S(K\_seed), Sign_C(Hash1), \{Hash2\}_{KCS})$

All data is then encrypted with a session key based on N_C, N_S & K_seed, and hashed for integrity.

Cipher Suites

- Cipher Suites with encryptions and authentication:

```
SSL_RSA_WITH_3DES_EDE_CBC_SHA
SSL_RSA_WITH_DES_CBC_SHA
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5
TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA
...
```

- Cipher Suites with just authentication:

```
SSL_RSA_WITH_NULL_MD5
SSL_RSA_WITH_NULL_SHA
```

- Cipher Suites with just encryptions:

```
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA
SSL_DH_anon_WITH_DES_CBC_SHA
SSL_DH_anon_WITH_RC4_128_MD5
TLS_DH_anon_WITH_AES_128_CBC_SHA
TLS_DH_anon_WITH_AES_256_CBC_SHA
```

Today's Lecture

- Protocols in Alice and Bob notation
- Forward Secrecy
- Secure Sockets Layer (SSL) / Transport Later Security (TLS)

Next Lecture

The basic building blocks of the web:

- HTTP
- HTML
- JavaScript
- JSP
- SQL