

## Secure Communication

Tom Chothia  
Computer Security, Lecture 8

## Today's Lecture

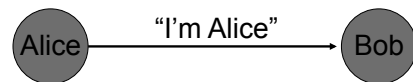
- Protocols in Alice and Bob notation
- Some Key Establishment Protocol
- Secure Sockets Layer (SSL) / Transport Layer Security (TLS)
- Certificates

## 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":



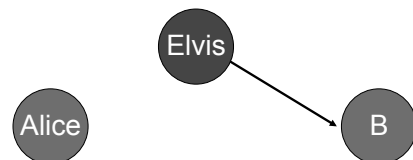
written as:

Alice → Bob : "I'm Alice"

## Rules

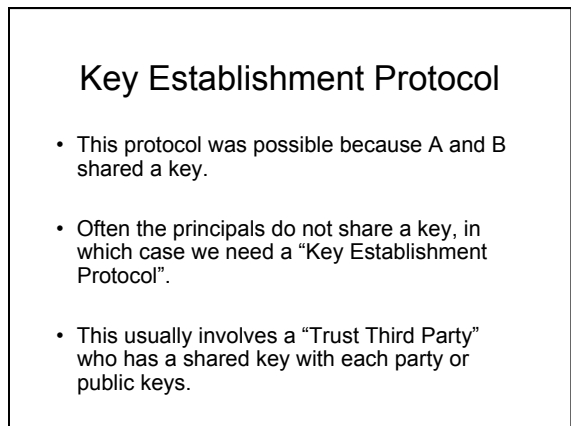
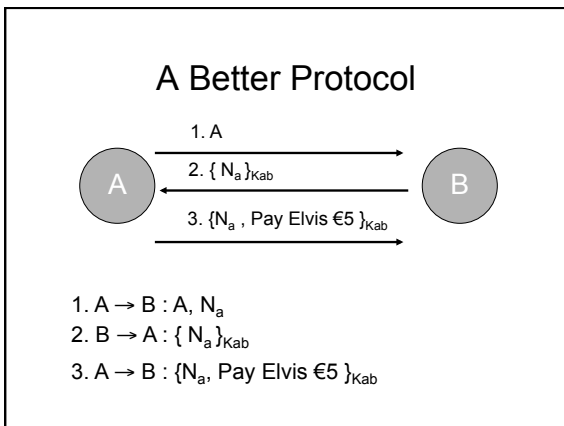
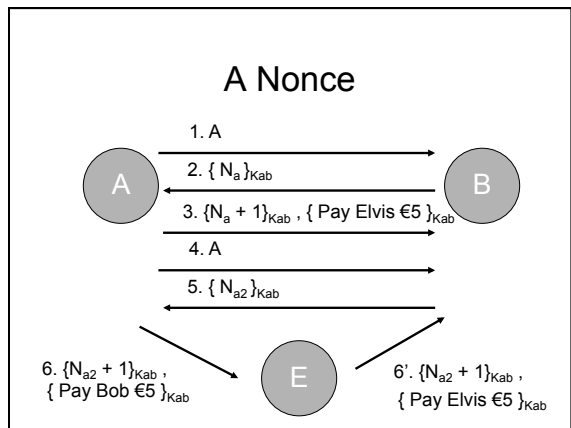
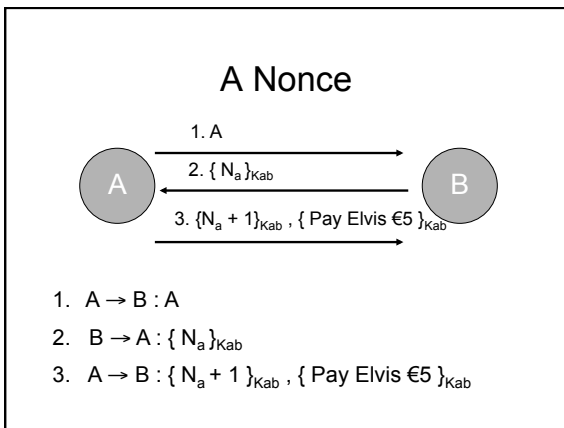
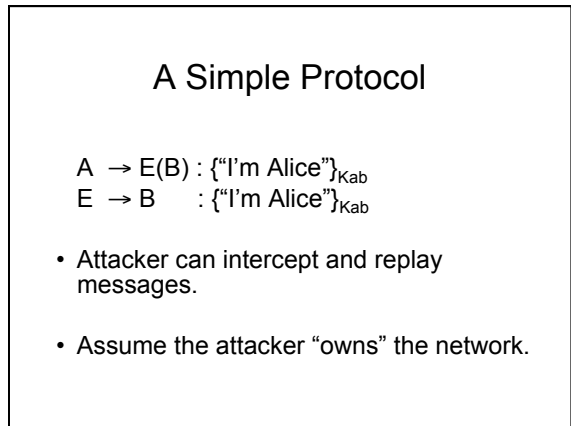
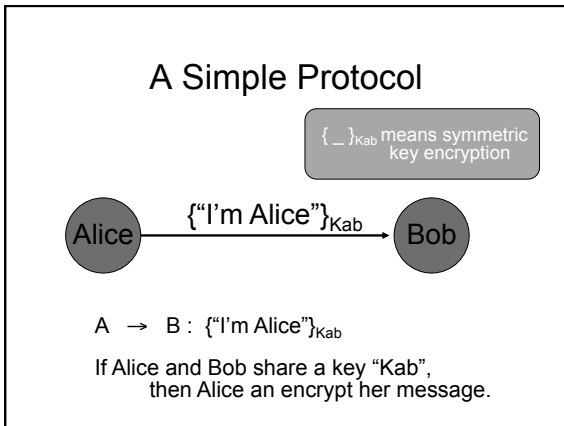
- We write down protocols as a list of messages sent between "principals", e.g.
  1. A → B : "Hello"
  2. B → A : "Offer"
  3. A → B : "Accept"

## A Simple Protocol



"The Attacker" can pretend to be anyone.

E(A) → B : "I'm Alice"



## 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'.  $C(A) \rightarrow B : E_A(N_a, A)$
  - 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.  $A \rightarrow B : E_B(N_b)$

## Needham-Schroeder Symmetric Key Protocol

What if Alice and Bob only have symmetric keys they shared with a server?

- $K_{AS}$  : is a good key for Alice and the server
- $K_{BS}$  : is a good key for Bob and the server

Alice and Bob trust the server.

How can they set up a shared key  $K_{AB}$  with the server doing the least work possible?

## Needham-Schroeder Symmetric Key Protocol

1.  $A \rightarrow S : A, B, N_A$
2.  $S \rightarrow A : \{ N_A, K_{AB}, B, \{ K_{AB}, A \}_{K_{BS}} \}_{K_{AS}}$
3.  $A \rightarrow B : \{ K_{AB}, A \}_{K_{BS}}$
4.  $B \rightarrow A : \{ N_B \}_{K_{AB}}$
5.  $B \rightarrow A : \{ N_B + 1 \}_{K_{AB}}$

Problem: Attacker can force an old key on B by replaying messages 4 & 5.

## Needham-Schroeder Symmetric Key Protocol

4.  $E(B) \rightarrow A : \{ N_B \}_{K_{AB}}$
5.  $E(B) \rightarrow A : \{ N_B + 1 \}_{K_{AB}}$

Problem: Attacker can force an old key on B by replaying messages 4 & 5.

## Kerberos

A and S share the key  $K_{AS}$  and B and S share  $K_{BS}$   
Both A and B trust S to generate a new key for them:  $K_{AB}$   
N is a nonce, T is a timestamp and L is an expiration time.

1.  $A \rightarrow S : A, B, N_A$
2.  $S \rightarrow A : \{K_{AB}, B, L, N_A\}_{K_{AS}}, \{K_{AB}, A, L\}_{K_{BS}}$
3.  $A \rightarrow B : \{A, T_A\}_{K_{AB}}, \{K_{AB}, A, L\}_{K_{BS}}$
4.  $B \rightarrow A : \{T_A + 1\}_{K_{AB}}$

## Kerberos

A protocol for key establishment and authentication used in Windows, MacOS, Apache, OpenSSH, ...

1.  $A \rightarrow S : A, B, N_A$
2.  $S \rightarrow A : \{K_{AB}, B, L, N_A\}_{K_{AS}}, \{K_{AB}, A, L\}_{K_{BS}}$
3.  $A \rightarrow B : \{A, T_A\}_{K_{AB}}, \{K_{AB}, A, L\}_{K_{BS}}$
4.  $B \rightarrow A : \{T_A + 1\}_{K_{AB}}$

## 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.

## Certificates

- But how do you know the public key of the website?
- Every browser comes with the public verification keys for a number of “trusted” companies.
- These companies will verify the identity of others, and sign their public keys.

## 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 issuer's 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.

## Transport Layer Security (TLS)

The core protocol goes:

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

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

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

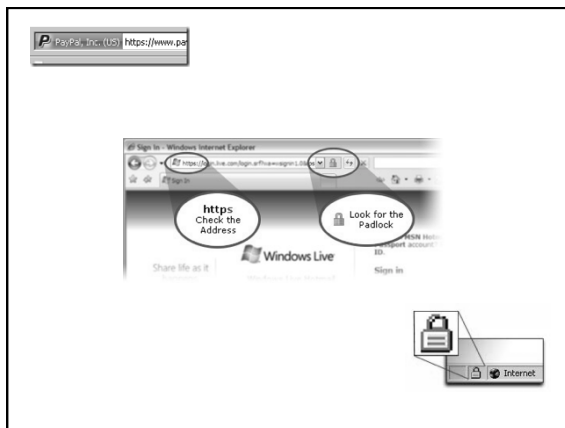
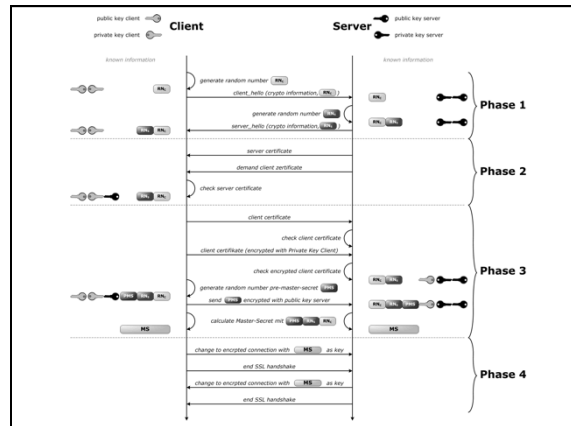
Hash 3 =  $\#(N_C, N_S, E_S(K\_seed), Sign_C(Hash1), \{Hash2\}_{K_{CS}})$

# Transport Layer Security (TLS)

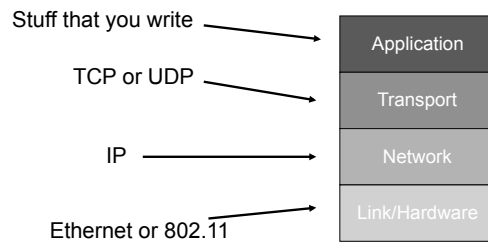
The core protocol goes:

1. C → S :  $N_C$
2. S → C :  $N_S, Cert_S$
3. C → S :  $E_S(K_{seed}), Sign_C(Hash1), \{Hash2\}_{K_{CS}}$
4. S → C :  $\{Hash3\}_{K_{CS}}$

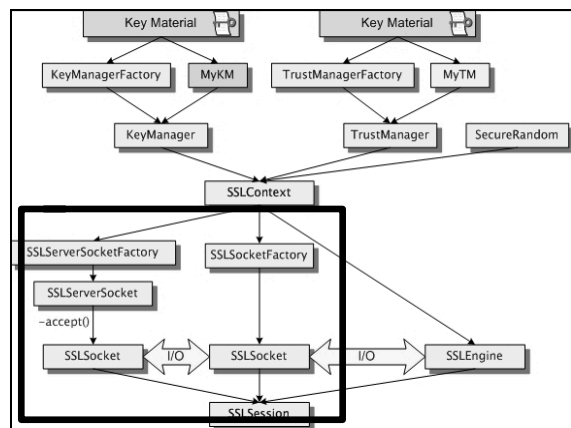
All data is then encrypted with a session key based on  $N_C, N_S$  &  $K_{seed}$ , and hashed for integrity.



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



# The Internet Protocol Stack with TLS



## TLS with no Authentication

- Create a `SSLServerSocketFactory` using  
`sockFact=SSLServerSocketFactory.getDefault();`
- Create a `SSLServerSocket`:  
`secSock=sockFact.createServerSocket(portNo)`
- Set the Ciphers:  
`secSocket.setEnabledCipherSuites(ciphers);`
- Listen on the socket for an encrypted connection:  
`socket = (Socket) secSocket.accept();`

## Cipher Suites

- Cipher Suites with encryptions and authentication:
- Cipher Suites with just 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
...
SSL_RSA_WITH_NULL_MD5
SSL_RSA_WITH_NULL_SHA
...
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
SSL_DH_anon_WITH_AES_128_CBC_SHA
SSL_DH_anon_WITH_AES_256_CBC_SHA
```

## Public key infrastructure (PKI)

- X.509 certificates are an example of a PKI.
  - Bad point: you need to pay a trusted third party.
- Another system is known as “web of trust”
  - This lets you sign the public keys of any of your friends.
  - Then anyone that trusts you learns all of your friend’s keys.



## Next Lecture

The basic building blocks of the web:

- HTTP
- HTML
- JavaScript
- JSP
- SQL