
CSE 3400/5850 - Introduction to Cryptography and
Cybersecurity
/ Introduction to Cybersecurity

Lecture 8
Shared Key Protocols – Part I

Ghada Almashaqbeh

UConn

Adapted from the Textbook Slides

Outline

- ❑ Cryptography protocols.
- ❑ Session or record protocols.
- ❑ Entity authentication protocols.

Modeling Cryptographic Protocols

- ❑ A protocol is a set of PPT (efficient) functions or algorithms
 - ❑ Each receiving (input), outputting (output)
 - ❑ Stateful protocols will have current state as additional input, and updated state as an additional output
 - ❑ Two (or more) parties (each has its own state)
 - ❑ Including *Init* (*short for initialize*)
 - ❑ Set the initial state of a party.
 - ❑ The execution process is a series of function invocations based on which the protocol proceeds.
 - ❑ Our discussion (from here) is focused on shared-key, two-party protocols, (man in the middle) MitM adversary.
-

Record Protocols

Secure communication between two parties using shared keys.

Two-party, shared-key **Record protocol**

- Parties/peers: *Alice* (sender), *Bob* (receiver)

- Simplest – yet applied – protocol
- Simplify: only-authentication for what Alice sends to Bob
 - Goal: Bob outputs m only if Alice had $Send(m)$

- Let's design the protocol! define the protocol functions

- $Init(k)$ [Initialize Alice/Bob with secret key k]
- $Send(m)$: Alice sends message m and a tag over m (to Bob)
- $Receive(m)$: Bob receives (m, tag) and accepts m if the tag is valid.

Two-party, shared-key **Record** protocol

- ❑ Design has many simplifications, easily avoided:
 - ❑ Only message authentication
 - ❑ No confidentiality!
 - ❑ Only ensure same message was sent
 - ❑ Does not address duplication, out-of-order, 'stale' messages, losses
- ❑ To add confidentiality: use encryption
 - ❑ Namely, employ EtA (encrypt then authenticate).

Two-party record protocol with Confidentiality

- ❑ *Init(k)* [Initialize Alice/Bob with secret key k]
 - ❑ Generate keys for encryption and MAC
 $(k_E = F_k(E), k_A = F_k(A))$
- ❑ *Send(m)*: Alice sends message m (to Bob)
 - ❑ $\{Output\ x = (E_{k_E}(m), MAC_{k_A}(E_{k_E}(m))) ; \}$
- ❑ *Receive(c, tag)* : Bob receives (c, tag) from the network
 - ❑ $\{Output\ D_k(c) \text{ if } (tag = MAC_{k_A}(c)) ; \}$

So, security guarantees ...

What does a secure shared-key two-party
record protocol mean?

How about the security of the one with
confidentiality?

Shared-key Entity Authentication Protocols

Ensure the identity of an entity (or a peer) involved in communication.

Mutual Authentication Protocols

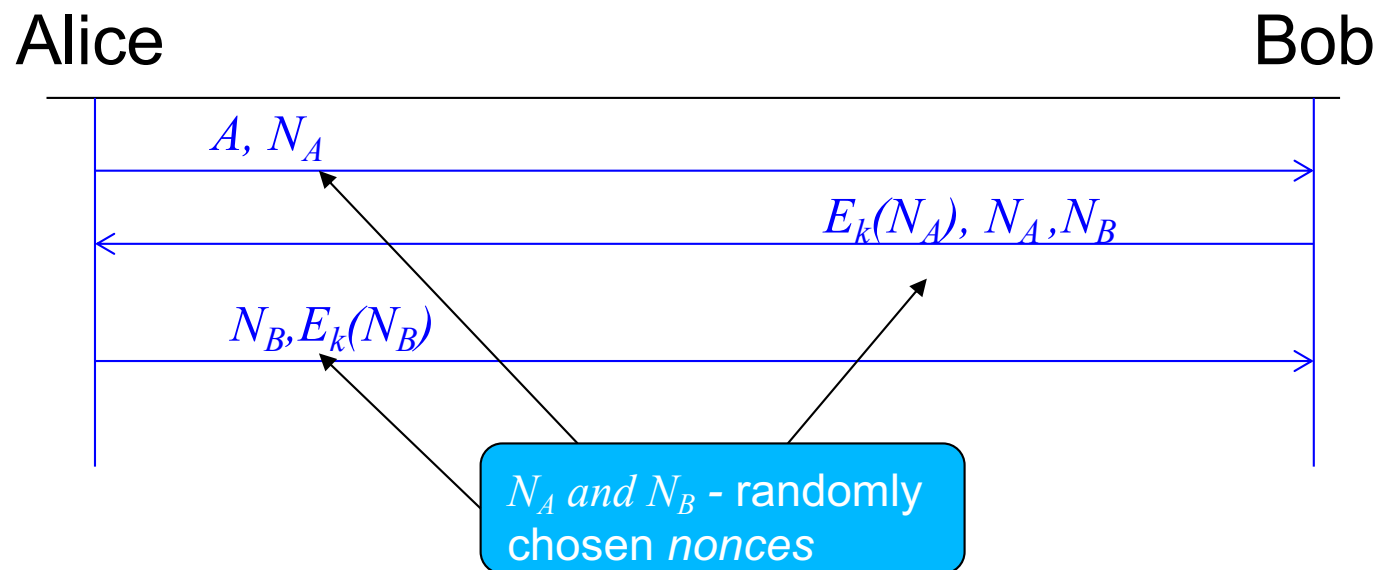
- ❑ In mutual authentication, each party authenticates herself to the other.
 - ❑ Alice knows that she is communicating with Bob, and vice versa
- ❑ This requires, at least, one exchange of messages.
 - ❑ A message from Alice and a response from Bob (or vice versa).
- ❑ Such a flow is called a ***handshake***.

Handshake Entity-Authentication protocol

- ❑ A protocol to open communication **sessions** between parties
- ❑ Protocol functions
 - ❑ *Init*(k): Initialize Alice/Bob with secret key k
 - ❑ *Open*: Alice/Bob open a session
 - ❑ *Send*(x) : party sends x to peer
 - ❑ *Receive*(x) : party receives x from the network channel
- ❑ Protocol outputs
 - ❑ *Open*(i): party opened session i
 - ❑ (and received messages).

Example : IBM's SNA Handshake

- ❑ First dominant networking technology
- ❑ Handshake uses encryption with shared key k



Insecure !! Why ?

SNA (Systems Network Architecture): IBM's proprietary network architecture, dominated market @ [1975-1990s], mainly in banking, government.

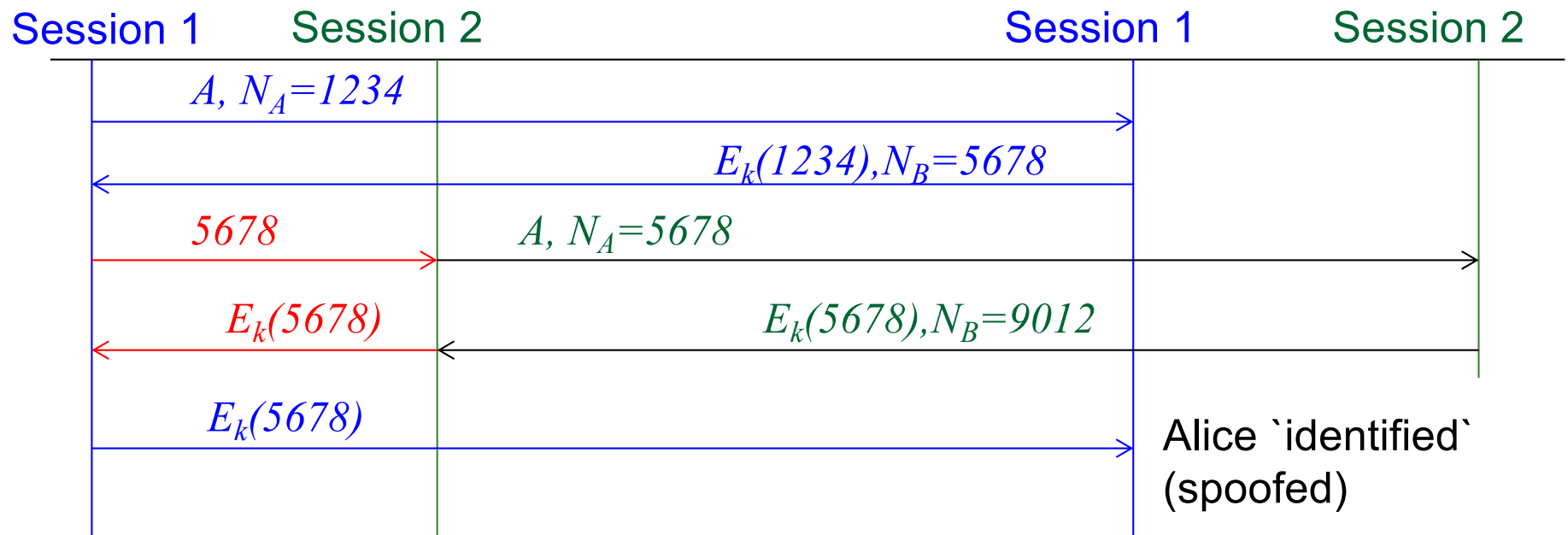
Attack on SNA's Handshake

❑ **MitM** opens **two** sessions with Bob, sending N_B to Bob in 2nd connection to get $E_k(N_B)$

❑ SNA is secure for sequential mutual authentication handshakes but not concurrent ones.

MitM (spoofing as Alice)

Bob



Fixing Mutual Authentication

- Encryption does not ensure authenticity
 - Use MAC to authenticate messages!
- Prevent redirection
 - Identify party in challenge
 - Better: use separate keys for each direction
- Prevent replay and reorder
 - Identify flow and connection
 - Prevent use of old challenge: randomness, time or state
- Do not provide the adversary with an oracle access!
 - Do not compute values from Adversary
 - Include self-chosen nonce in the protected reply

Secure Two-Party Handshake Protocol (2PP)



Alice

A, N_A

$N_A, N_B, \text{Mac}_k(2 \parallel A \leftarrow B \parallel N_A \parallel N_B)$

$N_B, \text{Mac}_k(3 \parallel A \rightarrow B \parallel N_A \parallel N_B)$



Bob

- ✓ Use MAC rather than encryption to authenticate
- ✓ Prevent redirection: include identities (A, B)
- ✓ Prevent replay and reorder:
 - Nonces (N_A, N_B)
 - Separate 2nd and 3rd flows: 3 vs. 2 input blocks
 - Provably secure [formal proof is out of scope]

Covered Material From the Textbook

- ❑ Chapter 5
 - ❑ Sections 5.1 and 5.2

Thank You!

