51.505 – Foundations of Cybersecurity

Week 11 – Protocols

Created by **Pawel Szalachowski** (2017)
Modified by **Jianying Zhou** (2018)

Last updated: 23 Nov 2018

Recap

- Questions on Week 10’s exercises?

Cryptographic Protocols

- Roles

✓ Alice and Bob (client-server, customer-merchant, ...)

  - A single entity can take on *any* of the roles.
  - Especially when doing security analysis of protocols.

✓ Adversary

  - Passive (eavesdropping)
  - Active (man-in-the-middle)

Cryptographic Protocols

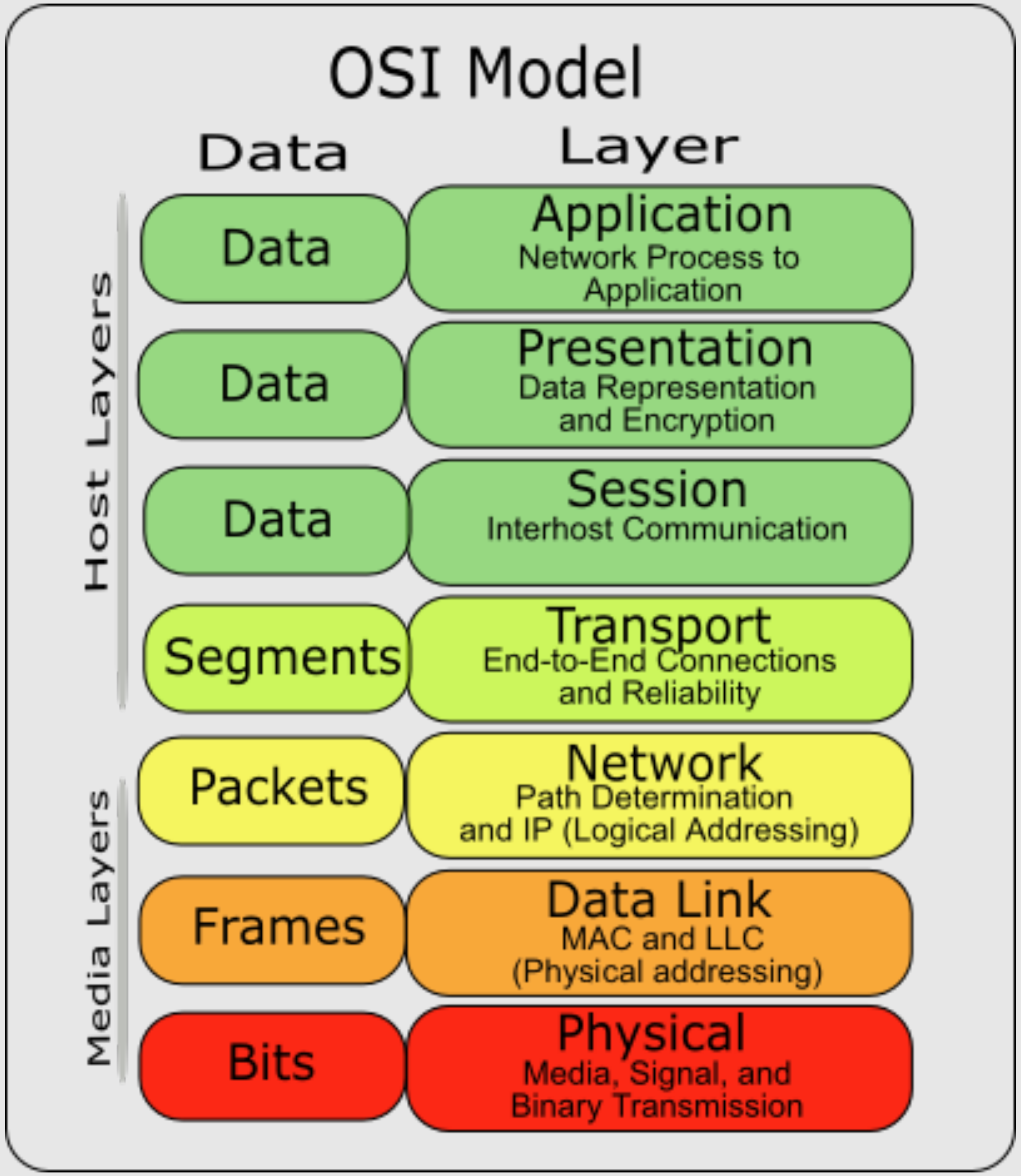
- **Trust** in cryptographic protocols

✓ Try to *minimize* the amount of trust required
  - Number of trusted parties
  - Scope of trust

✓ *Paranoia model*
  - Alice assumes that all participants are colluding against her.
  - Default model when all cryptographic protocols are designed.
  - Any assumption on trust should be expressed explicitly.
  - Each point of trust implies a risk that has to be dealt with.

Cryptographic Protocols

- High level of abstraction
  - ✓ Messages and steps
- Focus on the core functionality of the protocol
  - ✓ No careful specification of all actions each participant should take.
  - ✓ Extremely difficult to create a safe implementation of the protocol.
- **Modularization**
  - ✓ Split the required functionality into several protocol layers.
  - ✓ Each layer works on top of the previous layer.



Cryptographic Protocols

- Transport Layer
  - ✓ Transmit arbitrary strings of bytes
  - ✓ Secure channel at transport layer (confidentiality, authentication, and replay protection)
- Protocol and Message Identity
  - ✓ Protocol identifier (indicating version info, type of crypto protocol)
  - ✓ Message identifier (indicating which message of the protocol)
  - ✓ Horton Principle – avoid the risk that a message is interpreted in the wrong context.
- Message Encoding and Parsing
  - ✓ TVL encoding: *Tag* identifies the field in question; *Length* defined the length of value; *Value* is the actual data to be encoded. Example: ASN.1 or XML.
  - ✓ The receiver must be able to parse the message (not depending on the contextual info).

Cryptographic Protocols

- Protocol State Machine
  - ✓ Multiple protocols running at the same time.
  - ✓ <u>Protocol execution state</u> contains all the info to complete the protocol.
  - ✓ If the expected type of message arrives, it is parsed and processed according to the rules.
- Errors
  - ✓ There is an error if any check fails during protocol execution.
  - ✓ Error might be the cause of attack. Minimize the info of protocol state available to the attacker. Log an error message on a secure log.
- Replay and Retries
  - ✓ Bob can receive *replays* of messages from attacker and *retries* from Alice.
  - ✓ Bob needs to deal properly without introducing a security weakness.

Key Negotiation

Recap: Basic DH

Alice

Bob

Random secret *a*

$A = g^a \bmod p$ →

Random secret *b*

← $B = g^b \bmod p$

$K = (B)^a \bmod p$

$K = (A)^b \bmod p$

No authentication → MITM attack

Authenticated DHv1

Alice

known (g, p, q)

a = random$(1, q\text{-}1)$

$A = g^a \bmod p$

$$\xrightarrow{\quad A \quad}$$

Bob

known (g, p, q)

b = random$(1, q\text{-}1)$

$B = g^b \bmod p$

$$\xleftarrow{\quad B \quad}$$

$K = (B)^a \bmod p$

$$\xrightarrow{\quad AUTH_{Alice}(K) \quad}$$

$K = (A)^b \bmod p$

check $AUTH_{Alice}(K)$

$$\xleftarrow{\quad AUTH_{Bob}(K) \quad}$$

check $AUTH_{Bob}(K)$

Authenticated DHv1

- **Problems**
  - ✓ Based on the assumption that (p, g, q) are known to both Alice and Bob. (Bad idea to choose constant values.)
  - ✓ Use 4 messages. (Can be optimized.)
  - ✓ Two authentication messages have the same format. If the auth function is a simple MAC, Bob (or attacker) can replay $AUTH_{Alice}(K)$. Alice would not be convinced of the last authentication message.

Authenticated DHv2

Alice

choose (g, p, q)

a = random$(1, q\text{-}1)$

$A = g^a \bmod p$

$(g, p, q), A, AUTH_{Alice}$ $\longrightarrow$

Bob

check (g, p, q)

check $A, AUTH_{Alice}$

b = random$(1, q\text{-}1)$

$B = g^b \bmod p$

$\longleftarrow$ $B, AUTH_{Bob}$

$K = (A)^b \bmod p$

check $B, AUTH_{Bob}$

$K = (B)^a \bmod p$

choose/check (g, p, q):

1. $p = 2q + 1$
2. p, q are prime
3. $\alpha = random(2, p\text{-}2)$
4. $g = \alpha^2 \bmod p \wedge g \neq 1 \wedge g \neq p\text{-}1$

Authenticated DHv2

- **Problems**

  ✓ If Bob wants a larger DH prime than Alice, Bob has to abort the protocol.

  ✓ <u>Liveness problem</u> of Alice. The first message can be replayed by an attacker (though K is not disclosed). Bob does not know Alice is alive.

Authenticated DHv3

Alice

Bob

s = min p size
N = random$(0, 2^{256}-1)$

s, N →

choose (g, p, q)
b = random$(1, q-1)$
$B = g^b \bmod p$

← $(g, p, q), B, AUTH_{Bob}$

check (g, p, q)
check $B, AUTH_{Bob}$
a = random$(1, q-1)$
$A = g^a \bmod p$

$K = (B)^a \bmod p$

$A, AUTH_{Alice}$ →

check $A, AUTH_{Alice}$
$K = (A)^b \bmod p$

choose/check (g, p, q):

1. $p = 2q + 1$
2. p, q are prime
3. $\alpha = random(2, p-2)$
4. $g = \alpha^2 \bmod p \wedge g \neq 1 \wedge g \neq p-1$

Authenticated DHv3

- **Problem**
  - ✓ The final result *K* is a variable-sized number, not fitting other parts of the system.
  - ✓ *K* is computed using algebraic relations, which might be exploited by attackers.
- **Solution**
  - ✓ Hash the final key *K*.
  - ✓ This will reduce K to a fixed size, and also destroy any remaining algebraic structure.

Authenticated DH (final, short)

Alice

Bob

Alice: s = min p size

Alice: N = random$(0, 2^{256}-1)$

Alice → Bob: s, N

Bob: choose (g, p, q)

Bob: b = random$(1, q-1)$

Bob: $B = g^b \bmod p$

Bob → Alice: $(g, p, q), B, AUTH_{Bob}$

Alice: check (g, p, q)

Alice: check $B, AUTH_{Bob}$

Alice: a = random$(1, q-1)$

Alice: $A = g^a \bmod p$

Alice → Bob: $A, AUTH_{Alice}$

Bob: check $A, AUTH_{Alice}$

Alice: $K' = (B)^a \bmod p$

Alice: $K = SHA\text{-}256(K')$

Bob: $K' = (A)^b \bmod p$

Bob: $K = SHA\text{-}256(K')$

choose/check (g, p, q):

1. $p = 2q + 1$
2. p, q are prime
3. $\alpha = random(2, p-2)$
4. $g = \alpha^2 \bmod p \wedge g \neq 1 \wedge g \neq p-1$

Authenticated DH (final, long)

Alice **Bob**

Alice:

s_a = min p size
N = random$(0, 2^{256}-1)$

Alice → Bob: s_a, N

Bob:

s_b = min p size
s = max(s_a, s_b)
assert $s \leq 2 * s_b$
choose (g, p, q): $\log_2 p \geq s-1$
b = random$(1, q-1)$
$B = g^b \bmod p$

Bob → Alice: $(g, p, q), B, AUTH_{Bob}$

Alice:

check $AUTH_{Bob}$
assert $s_a - 1 \leq \log_2 p \leq 2 * s_a$
check (p, q) both prime
assert $p = 2q+1 \wedge g \neq 1 \wedge g \neq p-1$
assert $B \neq 1 \wedge B^q = 1 \bmod p$
a = random$(1, q-1)$
$A = g^a \bmod p$

Alice → Bob: $A, AUTH_{Alice}$

Alice:

$K' = (B)^a \bmod p$
$K = SHA\text{-}256(K')$

Bob:

check $AUTH_{Alice}$
assert $A \neq 1$ and $A^q = 1 \bmod p$
$K' = (A)^b \bmod p$
$K = SHA\text{-}256(K')$

Protocol Complexity

- Protocol design is more an art than a science. There is no hard rules to live by.
- Protocol design is very difficult, needs to specify all the rules which might be unknown at the abstraction level.
- Protocols may quickly get too complicated to keep in your head. Once you don’t understand it all, it is almost inevitable that a weakness slips in.
- Even with lots of experience, it is very easy to get wrong. **Treat all protocols with skepticism.**

Key Server

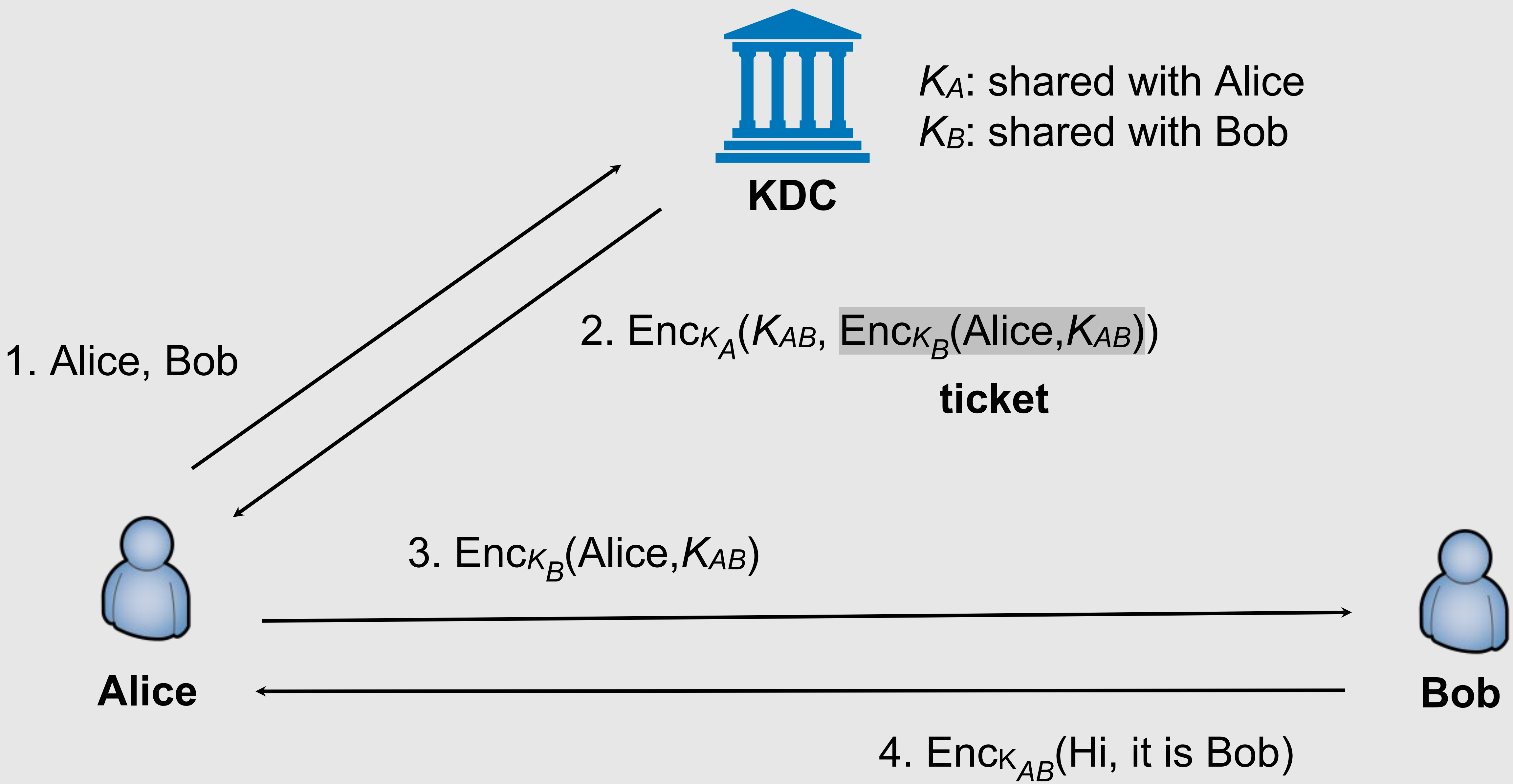
Key Management

- How Alice and Bob recognize each other?
- Challenging as people are involved
  - ✓ Hard to understand and predict
- Key server
  - ✓ A <u>trusted entity</u> that holds keys of all participants

Key Server

- Everybody sets up a shared key with the key server.
  - ✓ The server knows K_A shared with Alice, and K_B shared with Bob.
- Alice wants to talk to Bob.
  - ✓ She has no key shared with Bob.
  - ✓ ... but she can communicate securely with the server, which in turn can communicate securely with Bob.
  - ✓ The server could act as a proxy, but due to scalability issue it is much better when the server establishes a key for Alice and Bob.

Kerberos (simplified)



- Why not have KDC send $Enc_{K_B}(Alice, K_{AB})$ directly to Bob?

Kerberos

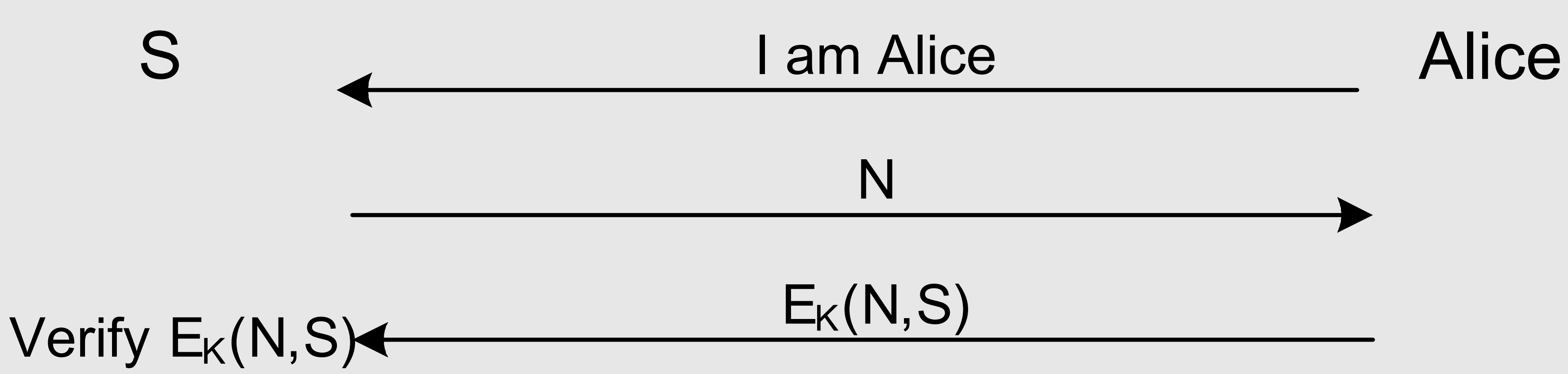
- Key-Distribution Center (KDC)
  - ✓ <u>Trusted</u>
  - ✓ Could be a single point of failure
- Features:
  - ✓ Rely exclusively on symmetric encryption.
  - ✓ KDC only needs to store a shared key with each user.
  - ✓ KDC does not need to keep any state of a session.
- Actual implementation is complicated and error-prone.

(Entity) Authentication Protocols

(Symmetric Key based)

One-way Authentication

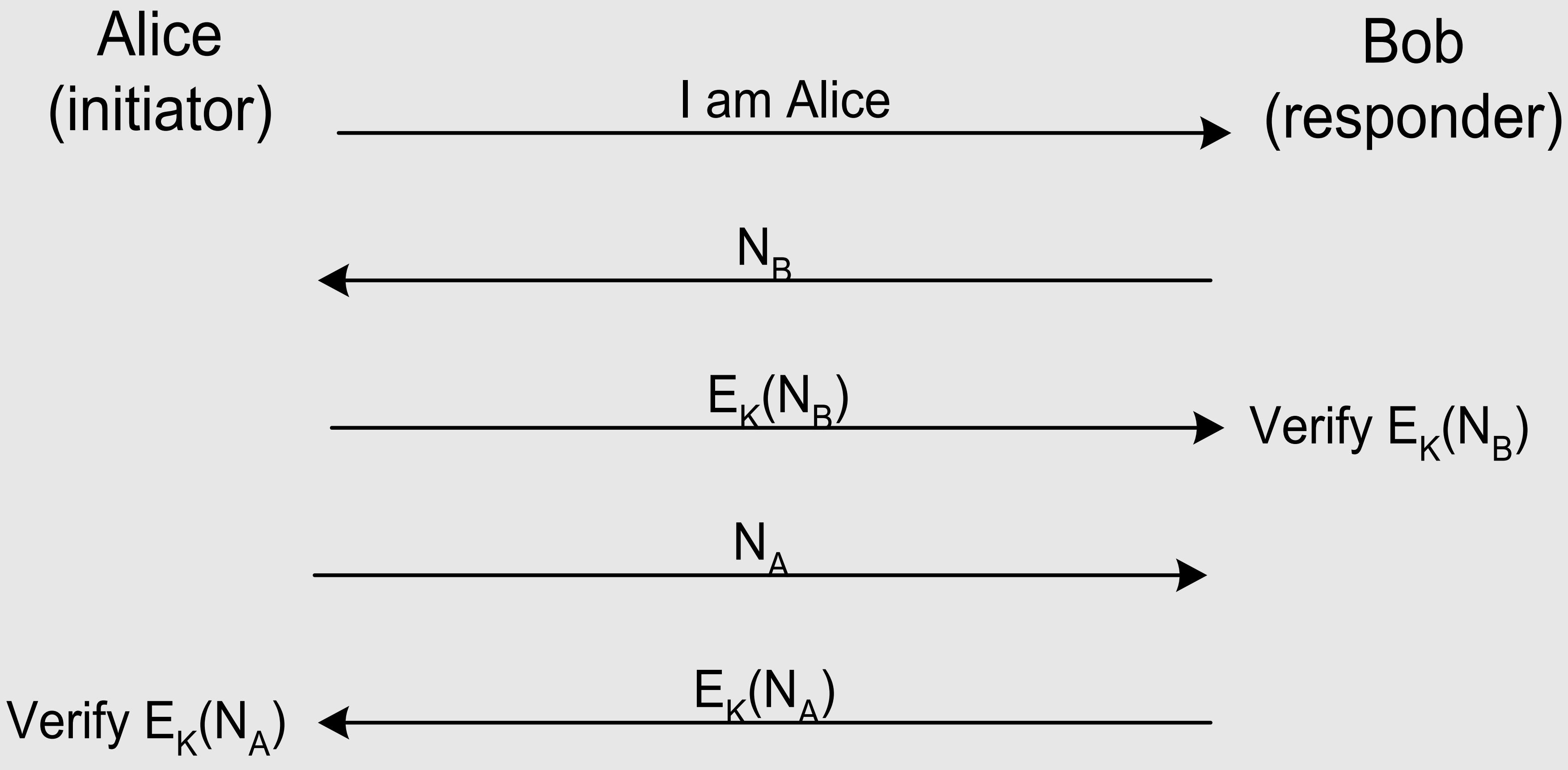
- The server S and Alice share a secret key K, and N is a nonce.
  - ✓ The nonce is to deduce that Alice is live.
  - ✓ The inclusion of S's identity ensures that Alice has the knowledge of S as her entity peer.



(Symmetric Key based)

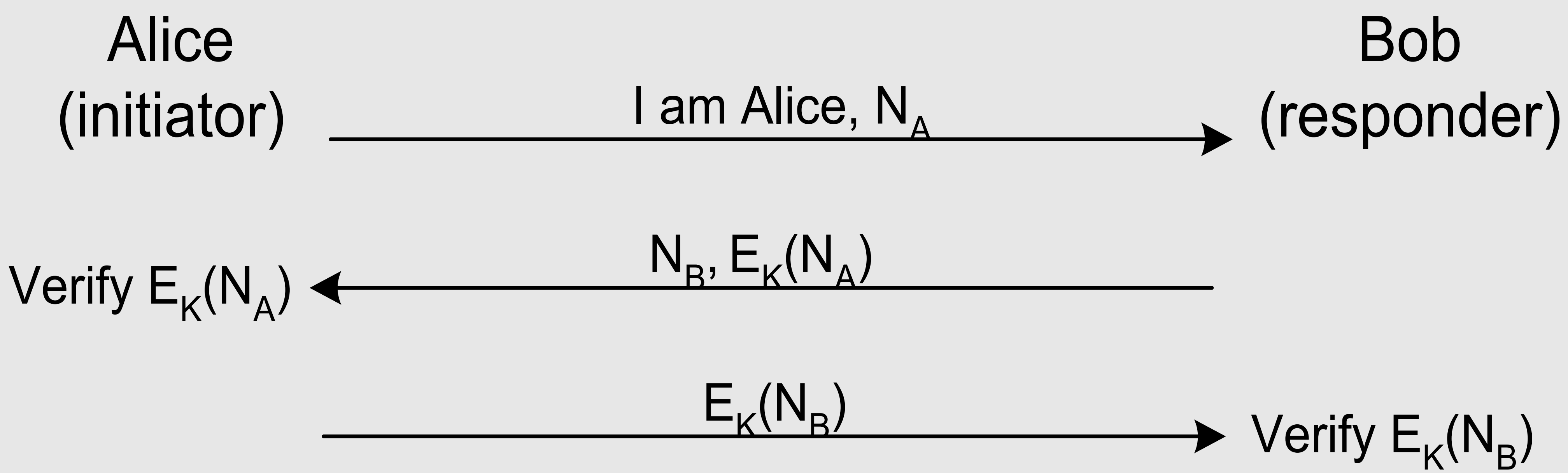
Mutual Authentication

- Alice and Bob share a secret key *K*, and *N* is a nonce.



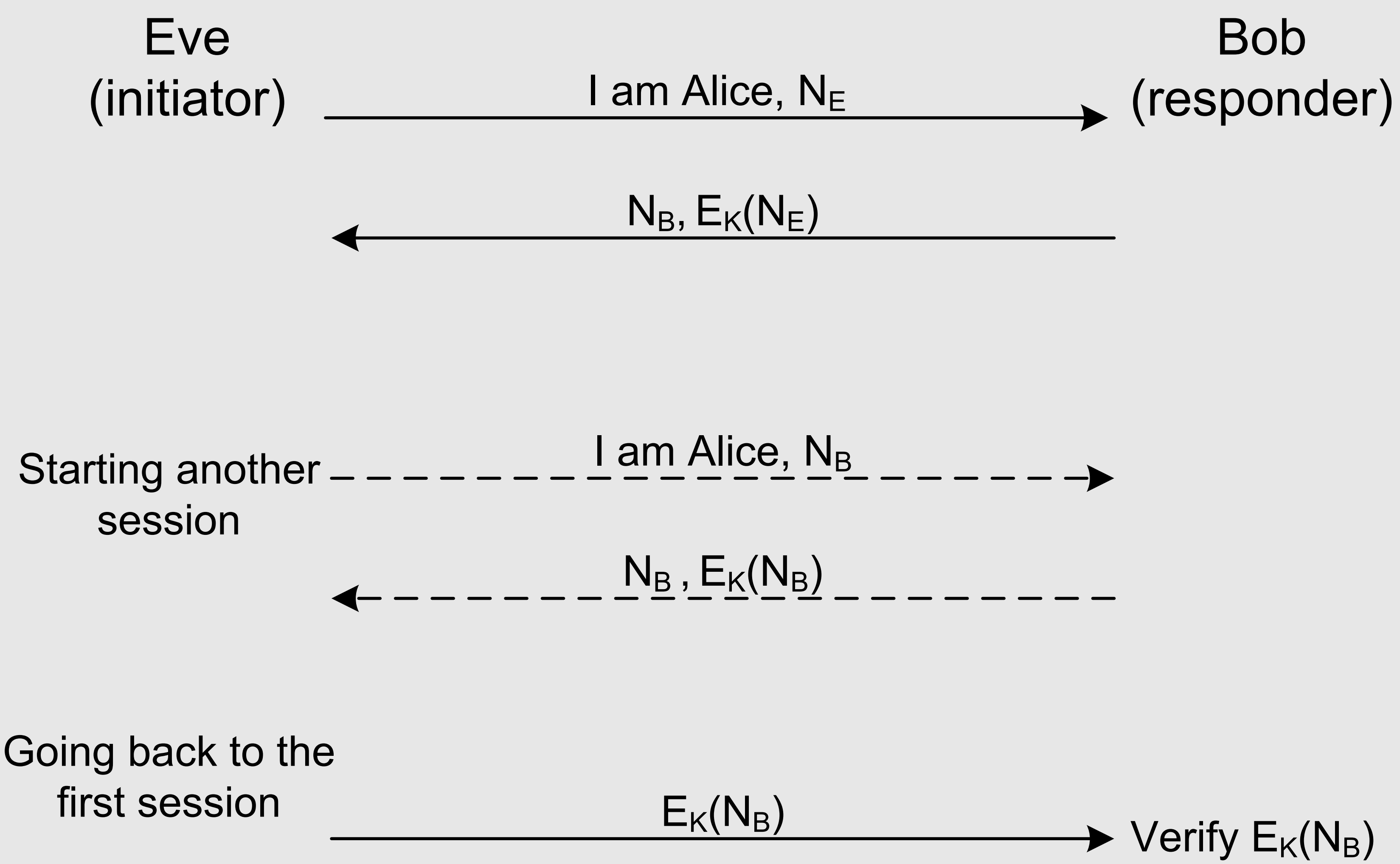
“Optimized” Mutual Authentication

- Alice and Bob share a secret key *K*, and *N* is a nonce.



Reflection Attack

- Assume Eve can open multiple simultaneous sessions with Bob.



- Will the original 5-step mutual authentication protocol be subject to this attack? (Classwork)

A Fix to Reflection Attack

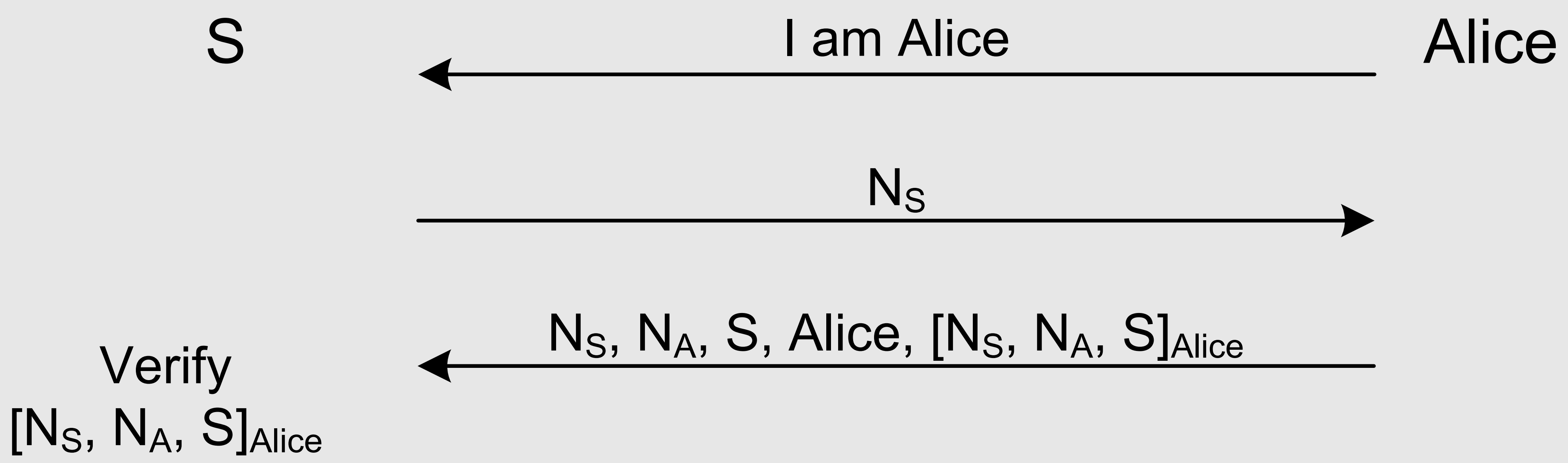
- The main problem is that the encrypted elements in the second and three messages are the same.
- A possible fix:

Alice (initiator)		Bob (responder)
	I am Alice, N_A $\longrightarrow$	
Verify HMAC	$\longleftarrow$ N_B, HMAC(K, N_A, N_B, “Alice”, “Bob”)	
	HMAC(K, N_A, N_B) $\longrightarrow$	Verify HMAC

(Public Key based)

One-way Authentication

- Alice signs the challenge from S. N_S, N_A are nonces picked by S and Alice, respectively.
- It is important that Alice influences what she signs.

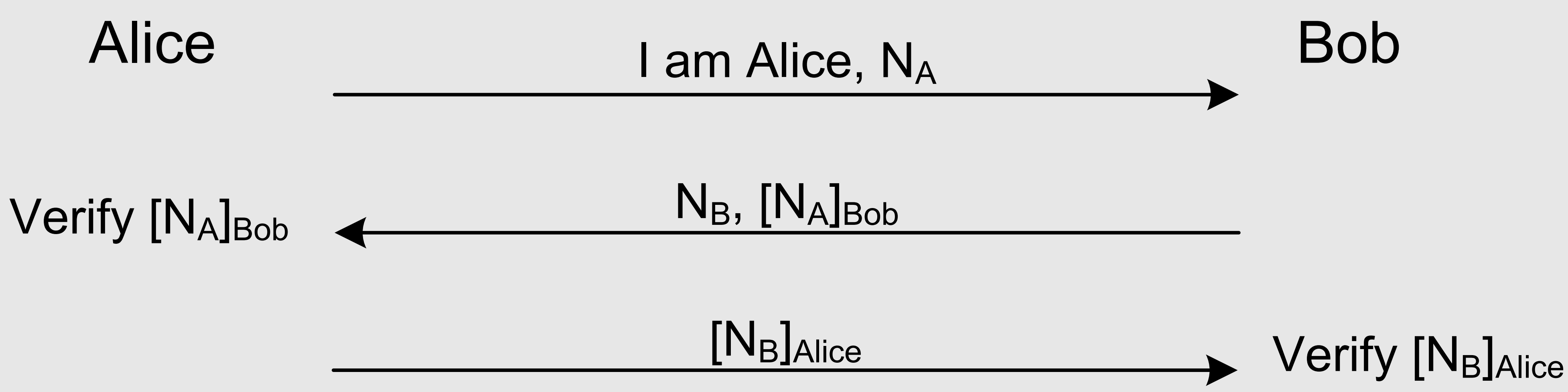


- Is N_A useful in this protocol?

(Public Key based)

Mutual Authentication

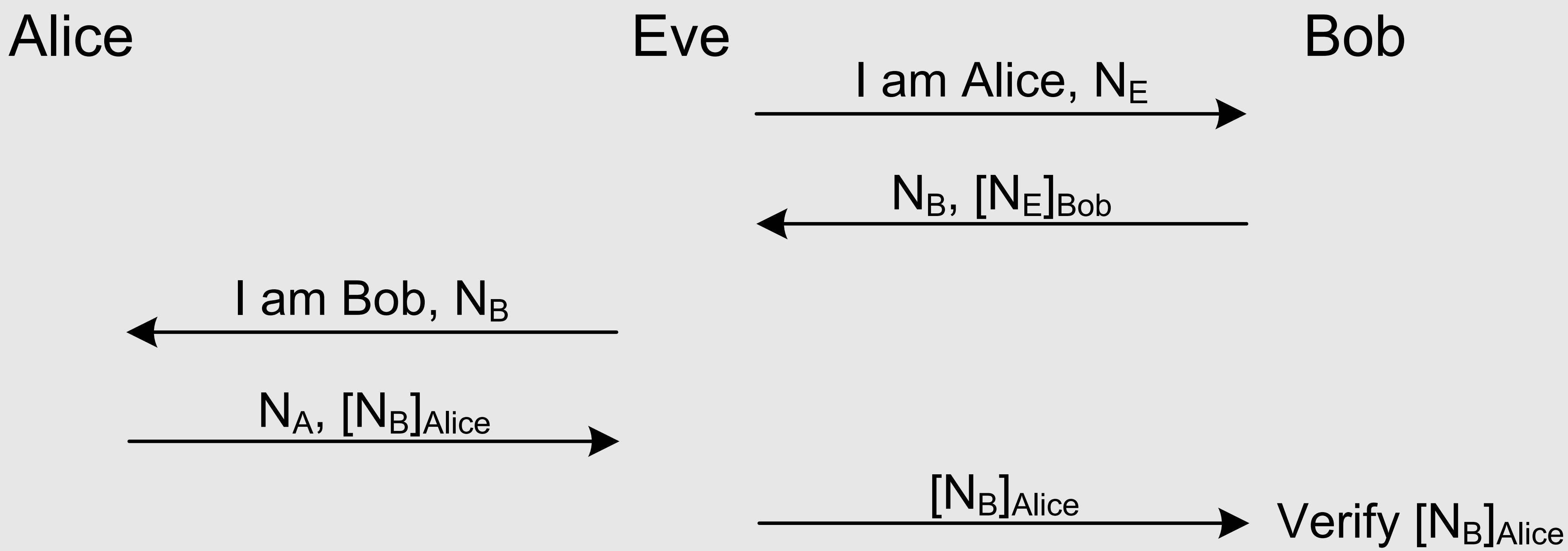
- Each side authenticates the other side by requesting for a correct digital signature.



- If N_B in message 2 is changed by an attacker, what happens?
- Are there other attacks?

MITM Attack

- Eve can impersonate Alice by having Alice's help in signing Bob's nonce.



A Fix to MITM Attack

- The main problem is that Alice and Bob have no influence on what they will sign.
  - ✓ As a general principle, it is better that both parties have some influence over the content being signed.
  - ✓ Otherwise, the protocol can be abused by an attacker.

- A possible fix:

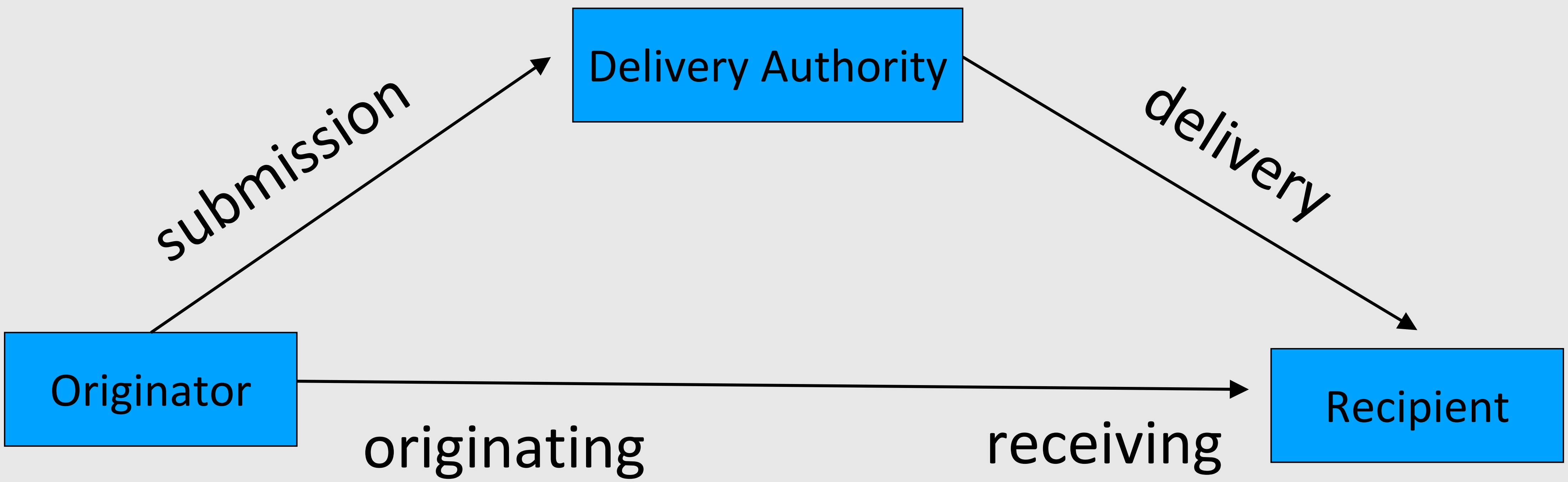
Alice → Bob: I am Alice, N_A

Verify $[N_B, N_A]_{Bob}$ ← Bob: N_B, N_A, Alice, $[N_B, N_A, \text{Alice}]_{Bob}$

Alice → Bob: N_B, N_A, Bob, $[N_B, N_A, \text{Bob}]_{Alice}$ → Verify $[N_B, N_A]_{Alice}$

Non-repudiation Protocols

Non-repudiation Services



- Non-repudiation of Origin
- Non-repudiation of Receipt
- Non-repudiation of Submission
- Non-repudiation of Delivery

Fairness in Non-repudiation

- **ISO/IEC 13888-3**

1. A → B: B, M, EOO
2. B → A: A, ~~EOR~~

- EOO – evidence of origin, signed by A
- EOR – evidence of receipt, signed by B

- **Problem**

✓ Selective Receipt – B may abort the transaction after receiving M, which leaves A without evidence of receipt (EOR).

Protocol Using In-line TTP

	1. A → TTP:	M, EOO
	2. TTP → A:	EOS
	3. TTP → B:	H(M), EOO
IF	4. B → TTP:	EOR
THEN {	5. TTP → B:	M
	6. TTP → A:	EOR, EOD_success **}**
ELSE	7. TTP → A:	EOD_fail

- **Evidence**
  - ✓ B receives EOO
  - ✓ A receives EOS, EOD (and EOR if delivery is successful)

- EOS – evidence of submission, signed by TTP
- EOD – evidence of delivery, signed by TTP

Protocol Using On-line TTP

- **IEEE S&P'96** (simplified)

1. A → B: C, EOO_C
2. B → A: EOR_C
3. A → TTP: K
4. B ← TTP: K, con_K
5. A ← TTP: con_K

- **Evidence**
  - ✓ B receives EOO_C, con_K
  - ✓ A receives EOR_C, con_K

- K – message key defined by A
- $C=Enc_K(M)$ – cipher text of M
- EOO_C, EOR_C – evidence of origin and receipt of C
- con_K – evidence of confirmation of K

- Is it possible to further reduce the TTP's involvement? (Homework)
  - ✓ Using **off-line TTP**, only involved when needed.

Key Points

- Protocol design is more an art than a science.
  - ✓ Minimize the amount of trust required
  - ✓ Paranoia model
- Key negotiation (e.g., authenticated DH)
- Key management (e.g., Kerberos)
- Entity authentication (symmetric key based & public key based)
- Non-repudiation (fairness, TTP involvement)

Exercises & Reading

- Classwork (Exercise Sheet 11): due on Fri Nov 23, 10:00 PM
- Homework (Exercise Sheet 11): due on Fri Nov 30, 6:59 PM
- Reading: FSK [Ch13, Ch14, Ch17]

End of Slides for Week 11