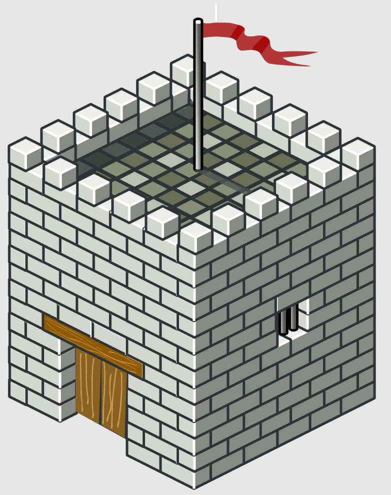
Foundations of Cybersecurity

X-Protocols



Paweł Szałachowski

2017

Cryptographic Protocols

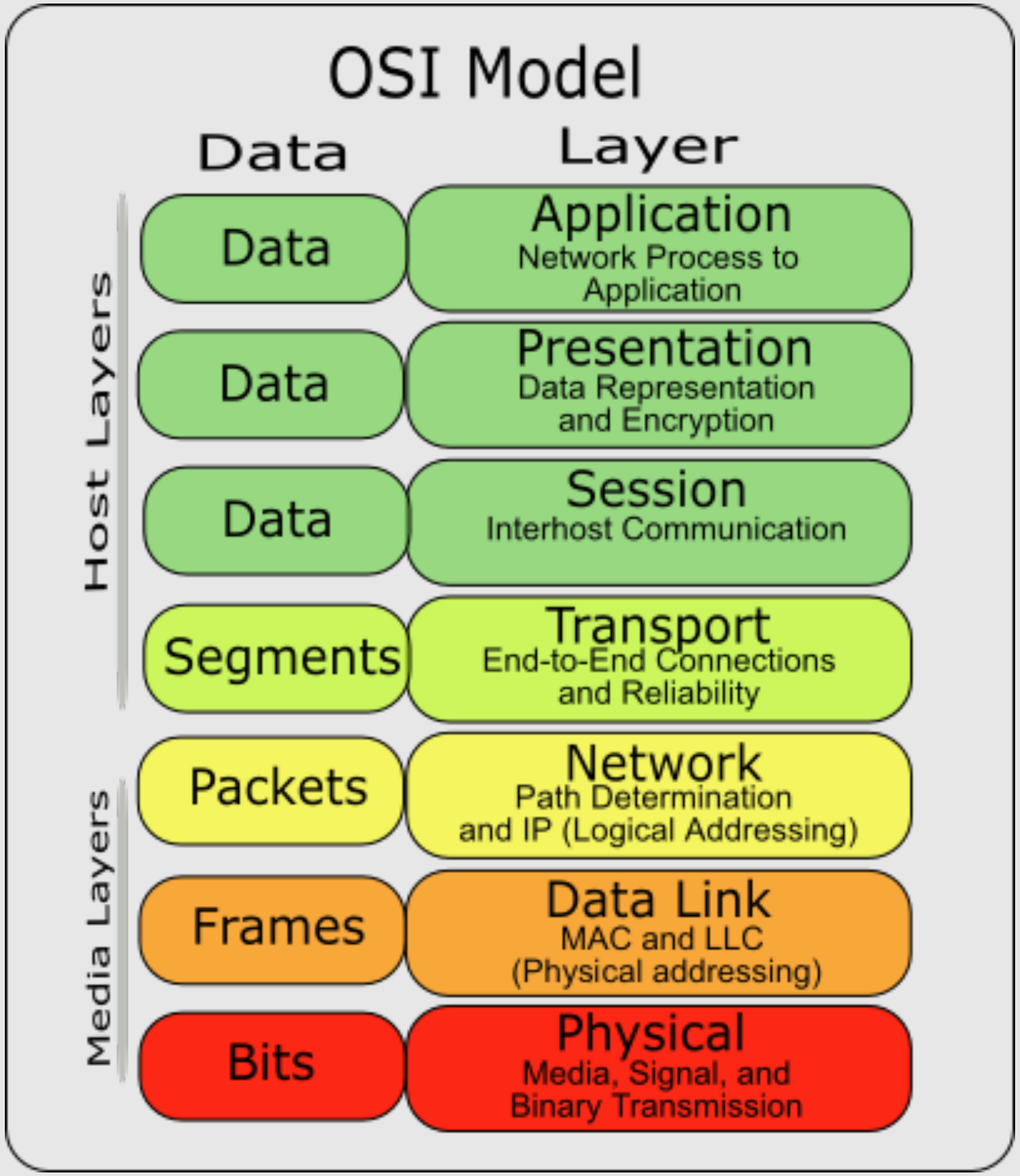
- Roles
  - Alice and Bob (client-server, customer-merchant, ...)
  - Adversary (passive, man-in-the-middle, able to steal secrets,...)
- Trust
  - Ethics, reputation, law, ...
- Incentives
  - Influence behavior, deployment, ...

Cryptographic Protocols

- Trust in cryptographic protocols
  - Cryptography tries to minimize the amount of trust required (usually, by replacing it by mathematics)
    - Number of trusted parties
    - Scope of trust
  - Paranoia model
    - Alice assumes that all participants are colluding against her

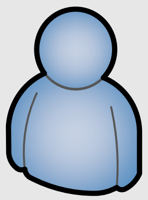
Messages and Steps

- High level of abstraction
- Transport Layer
- Protocol and Message Identity
- Message Encoding and Parsing
- Protocol State Machine
- Errors
- Replay and Retries

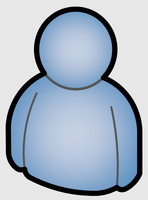


Key Negotiation

Recap: Basic DH



Alice



Bob

Random secret a

$g^a \bmod p$ →

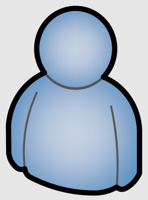
Random secret b

← $g^b \bmod p$

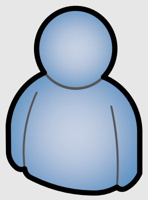
$K = (g^b)^a \bmod p$

$K = (g^a)^b \bmod p$

Authenticated DHv1



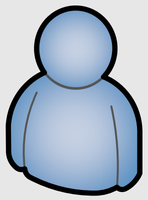
Alice



Bob

known (g, p, q)		known (g, p, q)
$a =$ random$(1, q\text{-}1)$		
$A = g^a \bmod p$	$A \longrightarrow$	
		$b =$ random$(1, q\text{-}1)$
		$B = g^b \bmod p$
	$\longleftarrow B$	
$K = (B)^a \bmod p$		$K = (A)^b \bmod p$
	$AUTH_{Alice}(K) \longrightarrow$	
		check $AUTH_{Alice}(K)$
	$\longleftarrow AUTH_{Bob}(K)$	
check $AUTH_{Bob}(K)$		

Authenticated DHv2



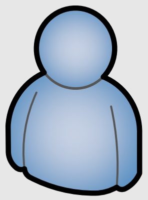
Alice

choose (g, p, q)

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

$A = g^a \bmod p$

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



Bob

check (g, p, q)

check A, $AUTH_{Alice}$

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

$B = g^b \bmod p$

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

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

check B, $AUTH_{Bob}$

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

Choosing (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 DHv3

Alice | **Bob**

Alice:

s = min p size

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

Alice → Bob: s, N

Bob:

choose (g, p, q)

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

$B = g^b \bmod p$

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

Alice:

check (g, p, q)

check $B, AUTH_{Bob}$

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

$A = g^a \bmod p$

Alice → Bob: $A, AUTH_{Alice}$

Bob:

check $A, AUTH_{Alice}$

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

Alice:

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

Authenticated DH (final, short)

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$(1, q-1)$

$A = g^a \bmod p$

$A, AUTH_{Alice}$ →

check $A, AUTH_{Alice}$

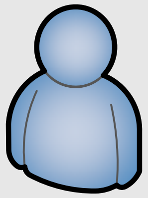
$K' = (A)^b \bmod p$

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

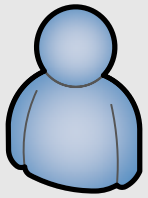
$K' = (B)^a \bmod p$

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

Authenticated DH (final, long)



Alice



Bob

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

s_a, N →

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

← (g, p, q), B, $AUTH_{Bob}$

check $AUTH_{Bob}$
assert $s_a - 1 \leq \log_2 p \leq 2 * s_a$
assert $255 \leq \log_2 q \leq 256$
check (p, q) both prime
assert $q \mid (p-1) \wedge g \neq 1 \wedge g^q = 1$
assert $B \neq 1 \wedge B^q == 1$
a = random$(1, q-1)$
$A = g^a \bmod p$

A, $AUTH_{Alice}$ →

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

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

The Clock

Uses

- Expiration
    - Limit validity period of a document or credential
- Unique Value
    - Timestamp + make sure it is unique
    - It is predictable
- Monotonicity
    - Replay protection
    - Auditing and logging
- Real-Time Operations
    - Payments

Security

- Manipulating the Clock
  - Setting back, stopping, setting forward
- Reliable Clock
  - No simple solution (CPU, network time, atomic clock, GPS, ...)
- The Same-State Problem
  - Embedded devices
- Time Standard
  - UTC (issues the leap seconds)

Key Servers

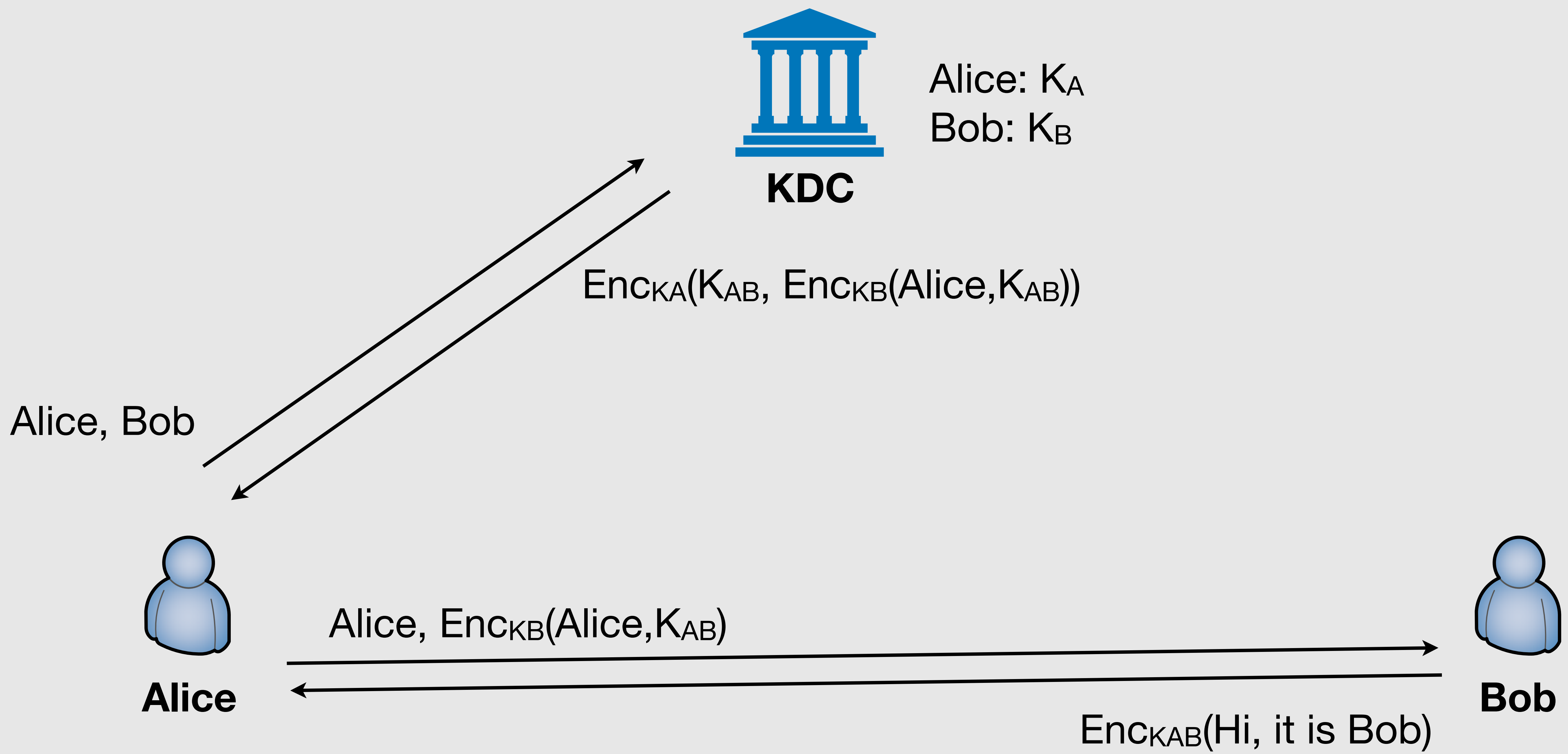
Key Management

- How Alice and Bob recognize each other?
- Challenging as people are involved
  - Hard to understand and predict
- Key server
  - A trusted entity 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 ask as a proxy, but due to scalability issue it is much better when the server establish a key for Alice and Bob

Kerberos (simplified)



Kerberos

- Complicated
  - What is authenticated?
  - What is encrypted?
  - What is timestamped/nonced?
- Replay attacks
  - Needham–Schroeder protocols
- Key-Distribution Center (KDC)
  - Trusted
  - It is a single point of failure
  - Significant overheads

Alternatives

- Alice can simply establish a secure channel with KDC
  - The secure channel provides authenticated, confidentiality, replay protection, ...
  - K_A can be used to derive a new key(s), used for the secure channel establishment
  - encrypted K_{AB} can be sent by the server within the channel
- Now a protocol to pass K_{AB} to Bob is only needed
  - That task seems to be simpler

Discussion&Classwork