ELTE
EÖTVÖS LORÁND
UNIVERSITY

CRYPTOGRAPHY AND SECURITY

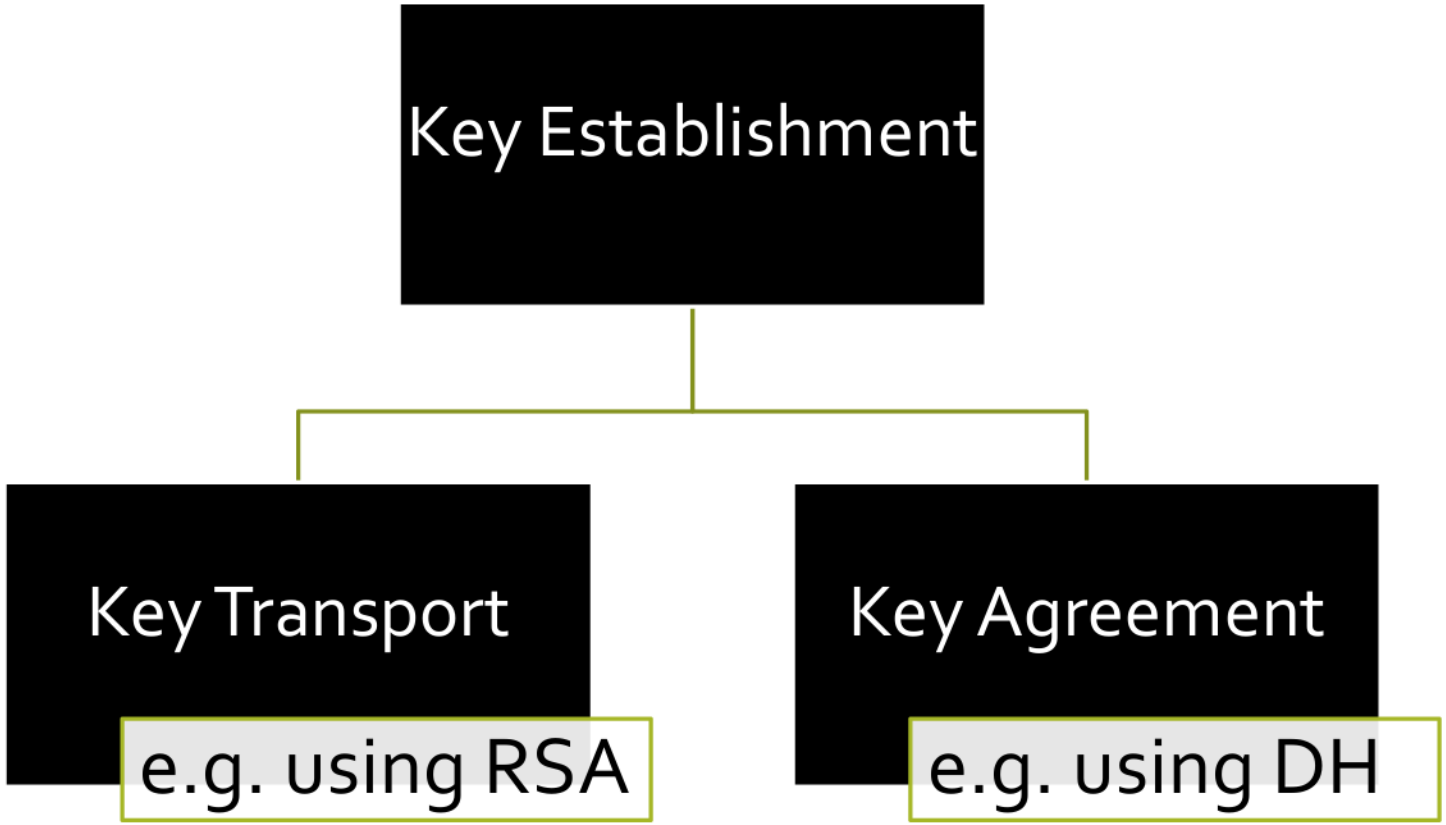
Practice

IP-18FKVKRBG

Lecture 10

2022-2023-2

Key Establishment Approaches



Diffie Hellmann

Diffie–Hellman I

Alice **Bob**

Common Parameters

Prime number p

Generator g

Choose random private key

$k_{prA} = a \in \{1,2,\ldots,p\text{-}1\}$

Choose random private key

$k_{prB} = b \in \{1,2,\ldots,p\text{-}1\}$

Diffie–Hellman II

Alice

…

Compute corresponding public key

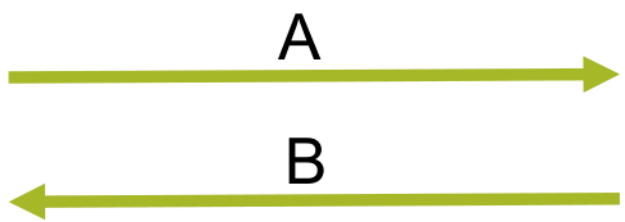
$A = g^a \bmod p$

Bob

…

Compute correspondig public key

$B = g^b \bmod p$



Compute common secret

$k_{AB} = B^a = (g^b)^a \bmod p$

Compute common secret

$k_{AB} = A^b = (g^a)^b \bmod p$

Diffie–Hellman III

Diffie- Hellman is only a key exchange protocol. The joined key can be used in symmetric encryption.

Alice **Bob**

$y = Enc(m, k_{AB})$ $m = Dec(y, k_{AB})$

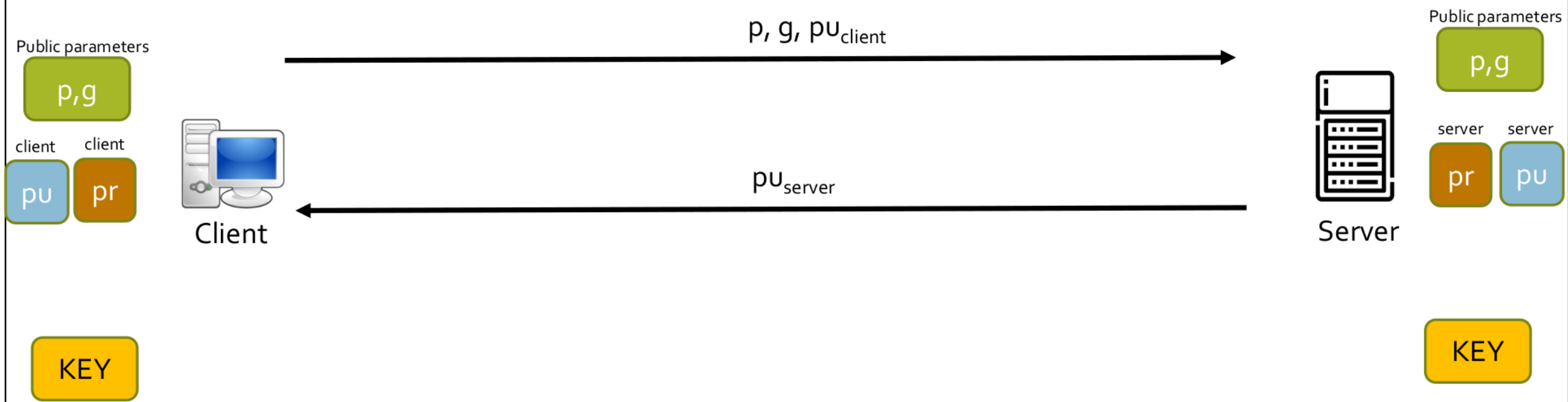
DH: Both parties in a local code

```
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import dh
from cryptography.hazmat.primitives.kdf.hkdf import HKDF
```

```
parameters = dh.generate_parameters(generator=2, key_size=512)

alice_private_key = parameters.generate_private_key()
bob_private_key = parameters.generate_private_key()
bob_public_key = bob_private_key.public_key()

alice_shared_key = alice_private_key.exchange(bob_public_key)

derived_key = HKDF(
    algorithm=hashes.SHA256(),
    length=32,
    salt=None,
    info=b'handshake data',
).derive(alice_shared_key)
```

1. Repeat the (shared key generation) and (key derivation) for the Bob as well.
2. Check whether both derived keys are identical?

DH: Client and Server



DH: Part 1 - Client sender side

Generates the public parameters, and its pair of keys

```
parameters = dh.generate_parameters(generator=2, key_size=512)
p = parameters.parameter_numbers().p
g = parameters.parameter_numbers().g
client_private_key = parameters.generate_private_key()
client_y = client_private_key.public_key().public_numbers().y
client_x = client_private_key.private_numbers().x
```

It stores (p, g, client_x and client_y) as the main parameters for later use.

Then, it sends (p, g and client_y) parameters to the server

DH: Part 2 - Server side

It first regenerate the same DH group, and the public key of the client, using the received parameters: p, g and client_y.

```
pn = dh.DHParameterNumbers(p,g)
client_public_number = dh.DHPublicNumbers(client_y, pn)
client_public_key = client_public_number.public_key()
```

Generates its key pair. Stores server_x to be fixed in later executions.

```
parameters = pn.parameters()
server_private_key = parameters.generate_private_key()
server_y = server_private_key.public_key().public_numbers().y
server_x = server_private_key.private_numbers().x
```

Then, it performs the shared key generation and key derivation.

DH: Part 3 - Client receiver side

It already stored the (p, g and client_y, client_x) parameters, and receives (server_y) from the server.

Regenerates its private key:

```
pn = dh.DHParameterNumbers(p,g)

client_public_number = dh.DHPublicNumbers(client_y, pn)
client_private_number = dh.DHPrivateNumbers(client_x, client_public_number)
client_private_key = client_private_number.private_key()
```

Regenerates the PK of the server:

```
server_public_number = dh.DHPublicNumbers(server_y, pn)
server_public_key = server_public_number.public_key()
```

Then it generate the shared key, and performs the key derivation.