Aalto University

Security Protocols

Tuomas Aura
CS-C3130 Information security

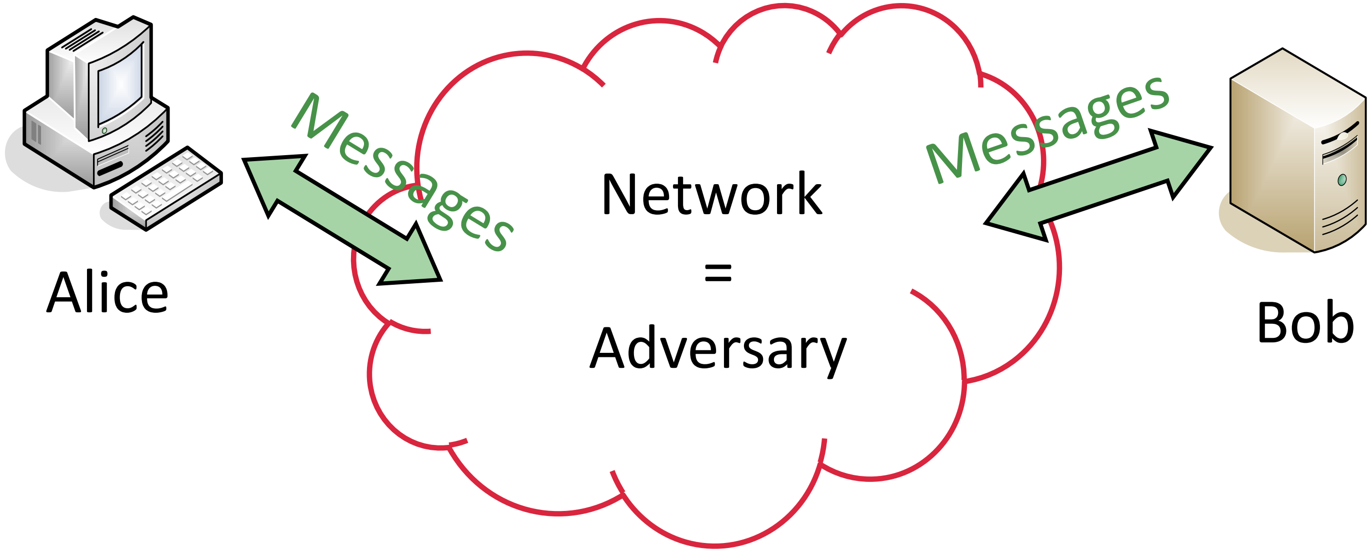
Aalto University, 2022 course

Outline

- Network threat model
- Replay and freshness
- Authenticated Diffie-Hellman

NETWORK THREAT MODEL

Network-security threat model



Dolev-Yao adversary model:

- Endpoints are trusted; network is the attacker
- The network may deliver, delete, modify, and send fake messages

Network security goals

- Data confidentiality: secrets only revealed to intended parties
- Data integrity: receiver can detect data modification
- Data-origin authentication: receiver verifies who sent the data
- Data and service availability: communication successful

- Questions:
  - Can there be confidentiality without authentication, or authentication without secrets?
  - Can there be integrity without authentication, or authentication without integrity?
  - Can availability be achieved in the Dolev-Yao adversary model?

Basic attack types

- Data confidentiality

  ↔ sniffing = eavesdropping = interception = spying
- Data integrity

  ↔ unauthorized data modification = tampering
- Data-origin authentication

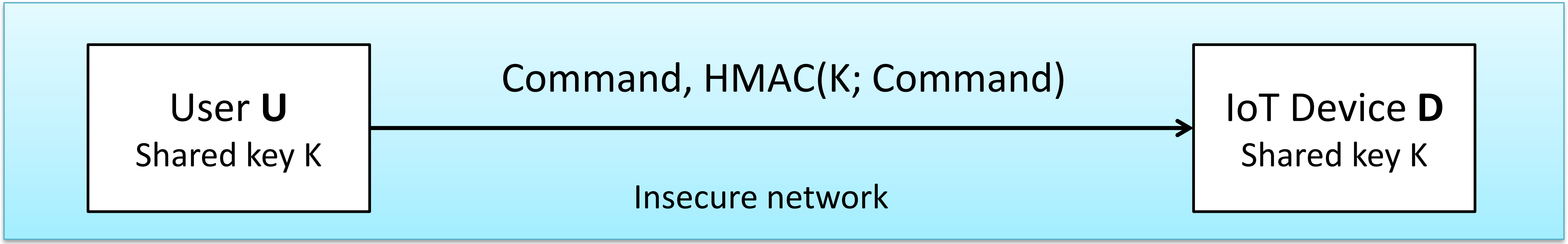
  ↔ spoofing or impersonation
- Data and service availability

  ↔ denial of service (DoS)

REPLAY AND FRESHNESS

Example: broken authentication v1

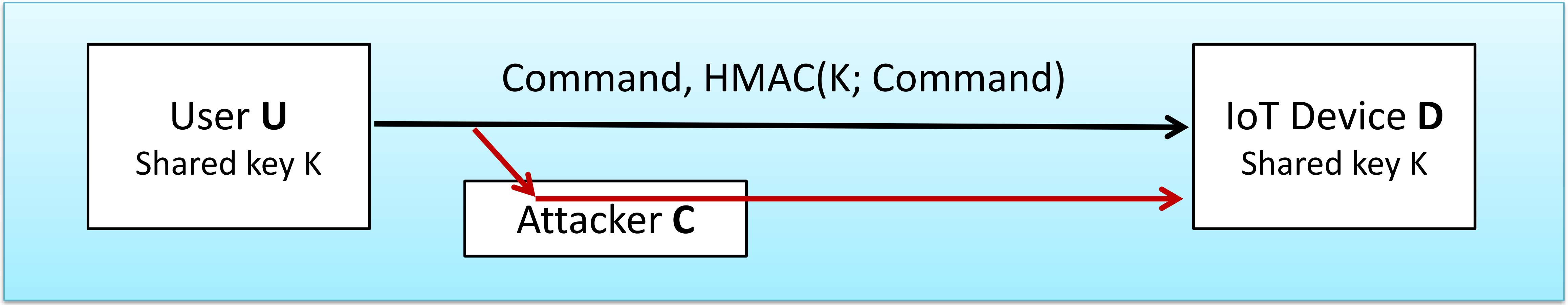
Course exercise: *"IoT device […] listens on a TCP port and accepts command messages, which are authenticated with a message authentication code (HMAC-SHA256)."*



U → D: Command, HMAC(K; Command)

Why is this not secure?

Replay attack



- Replay attack: attacker records the message and resends later

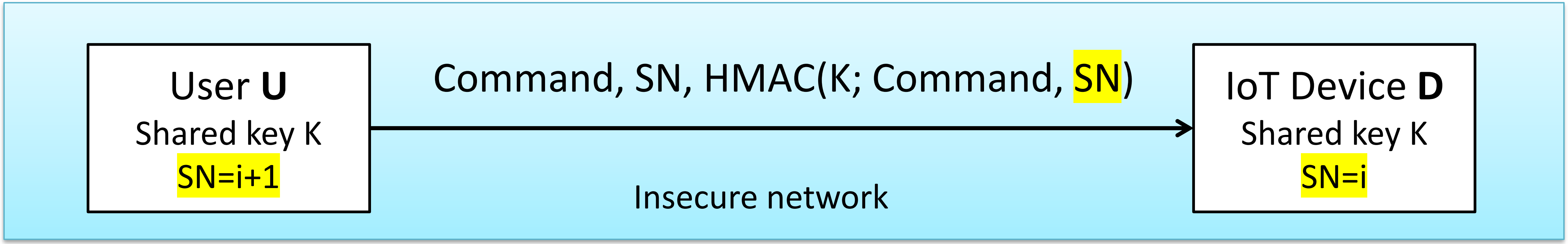
U → D: Command, HMAC(K; Command)

C → D: Command, HMAC(K; Command)

e.g. "increase speed by 10 RPM", "transfer €100 to C"

Example: broken authentication v2

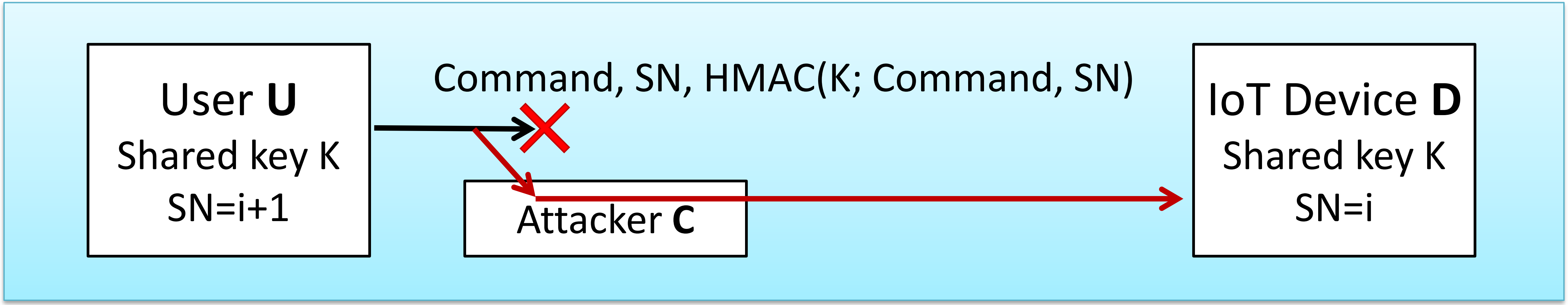
- Sequence number prevents replays
  - Receiver checks that the number increases and never repeats



U → D: Command, SN, HMAC(K; Command, SN)

Why is this still not secure?

Replay attack

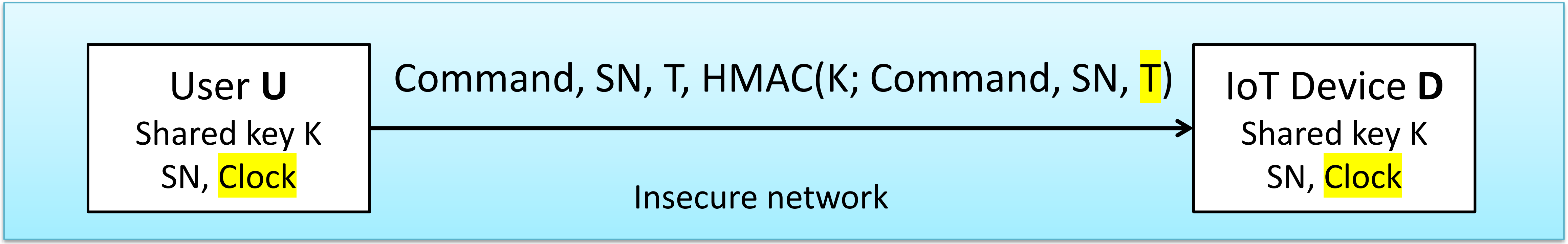


- Attacker cannot copy the message but can delay it

e.g. “open door”, “launch rocket”

Example: broken authentication v3

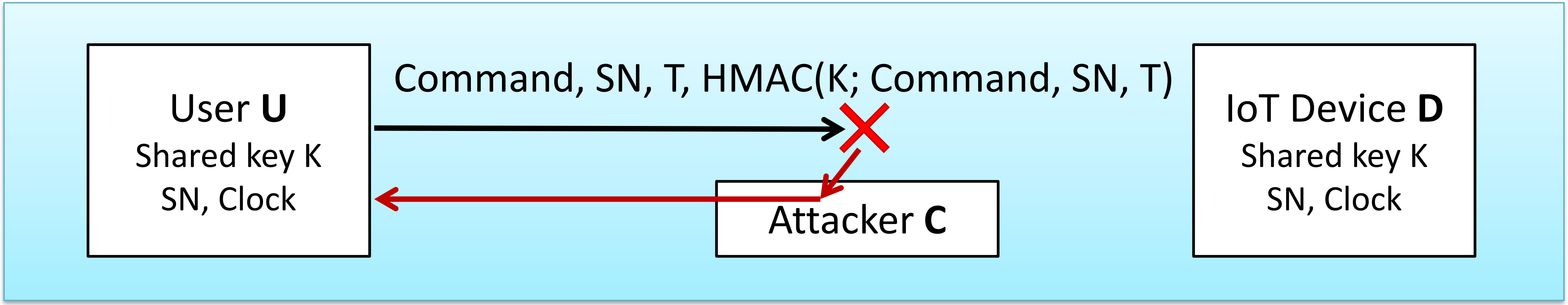
- Timestamp prevents delaying of messages
  - Receiver does not accept messages older than e.g. one minute



U → D: Command, SN, T, HMAC(K; Command, SN, T)

Why is this still not secure?

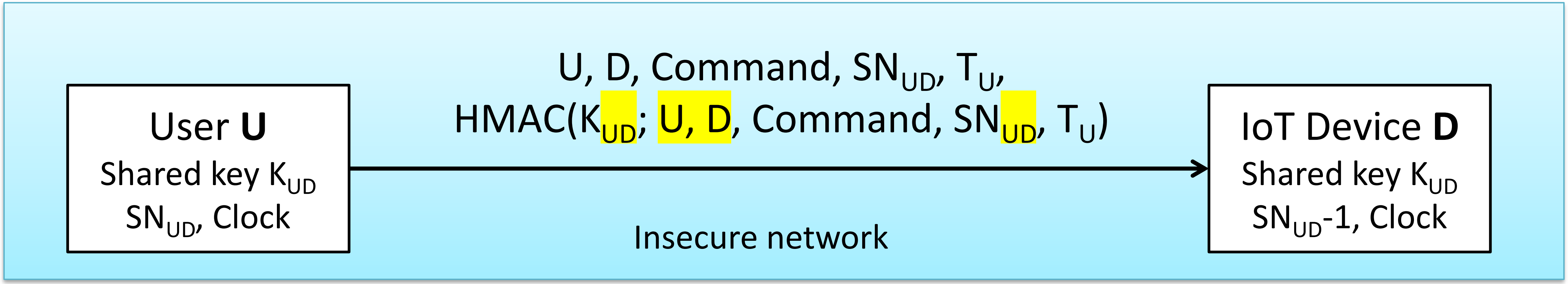
Replay back to sender



- Can the message be replayed back to the sender?
  - Can the same entity act as both user U and device D? Often possible
- Selfie attack against TLS 1.3 PSK mode
  https://eprint.iacr.org/2019/347.pdf

Example: authentication v4

- Explicit direction, or sender and receiver identity
- Separate key (and counter) for each direction

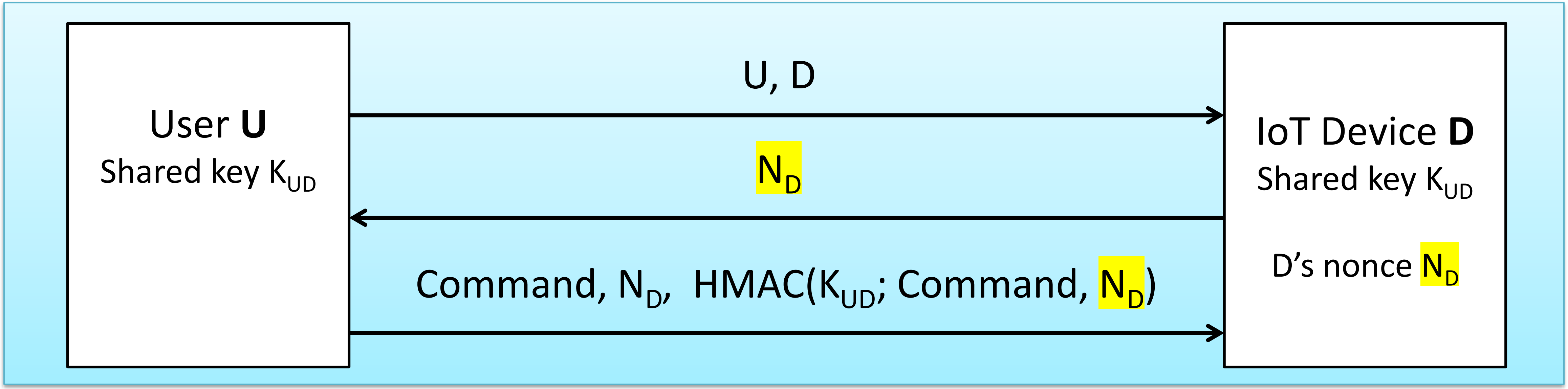


U → D: U, D, Command, SN_{UD}, T_U, HMAC(K_{UD}; U, D, Command, SN_{UD}, T_U)

Is this ok? Maybe the device does not have a reliable clock

Example: authentication v5

- Nonce = fresh random number



U → D: U, D
D → U: N_D
U → D: Command, N_D, HMAC(K_{UD}; Command, N_D)

+ No clock or counter synchronization
– More messages

A MORE REALISTIC PROTOCOL: AUTHENTICATED DIFFIE-HELLMAN

Unauthenticated Diffie-Hellman

- A and B have previously agreed on g and p
- All operations are modulo p

A chooses a random x. B chooses a random y.

1. A → B: A, g^x
2. B → A: B, g^y

A calculates shared secret $SK = (g^y)^x = g^{xy}$.

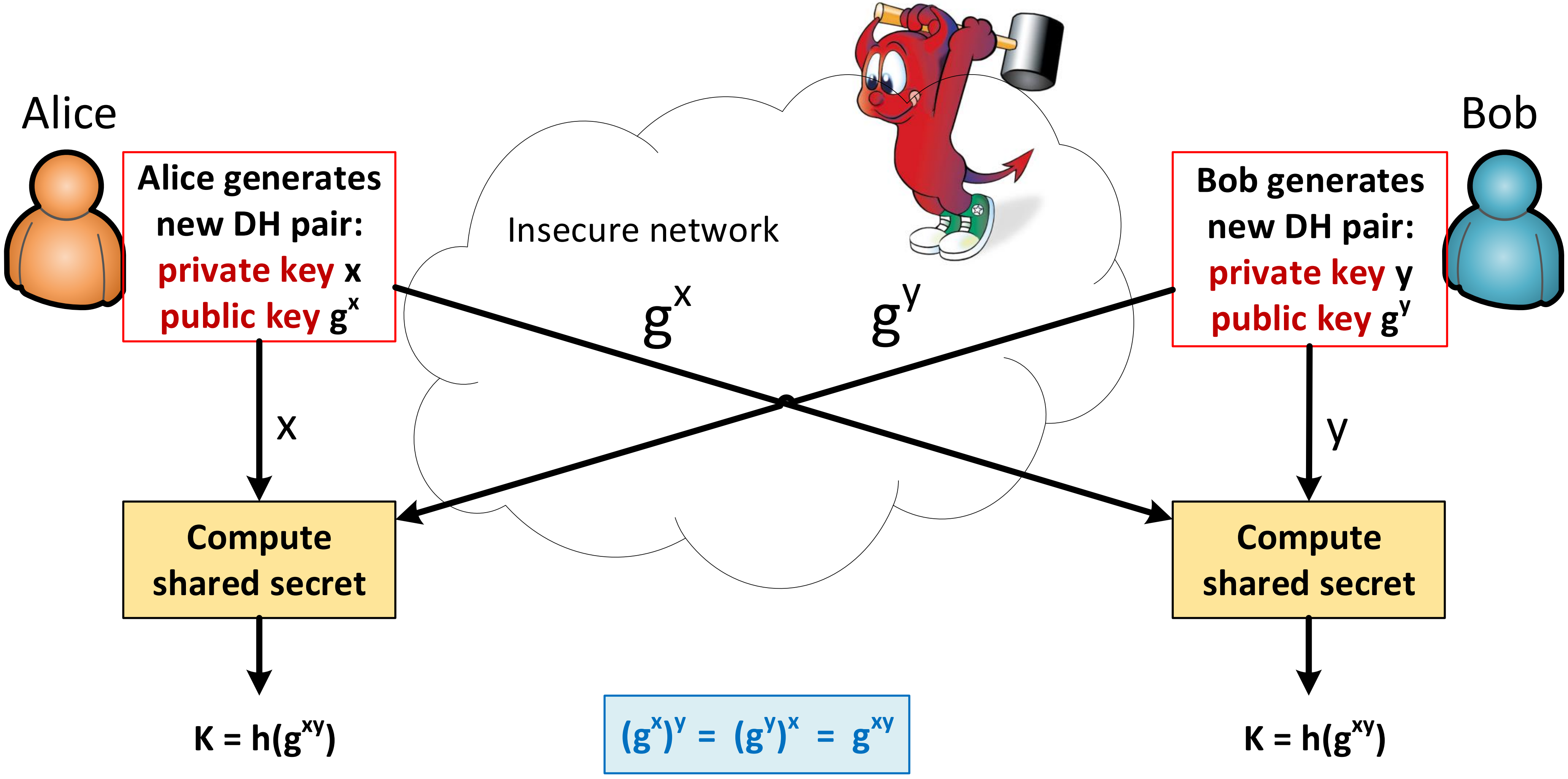
B calculates shared secret $SK = (g^x)^y = g^{xy}$.

So-called Alice-and-Bob notation for security protocols

- Sniffer learns g^x and g^y, cannot compute x, y, or g^{xy}

Recall from earlier

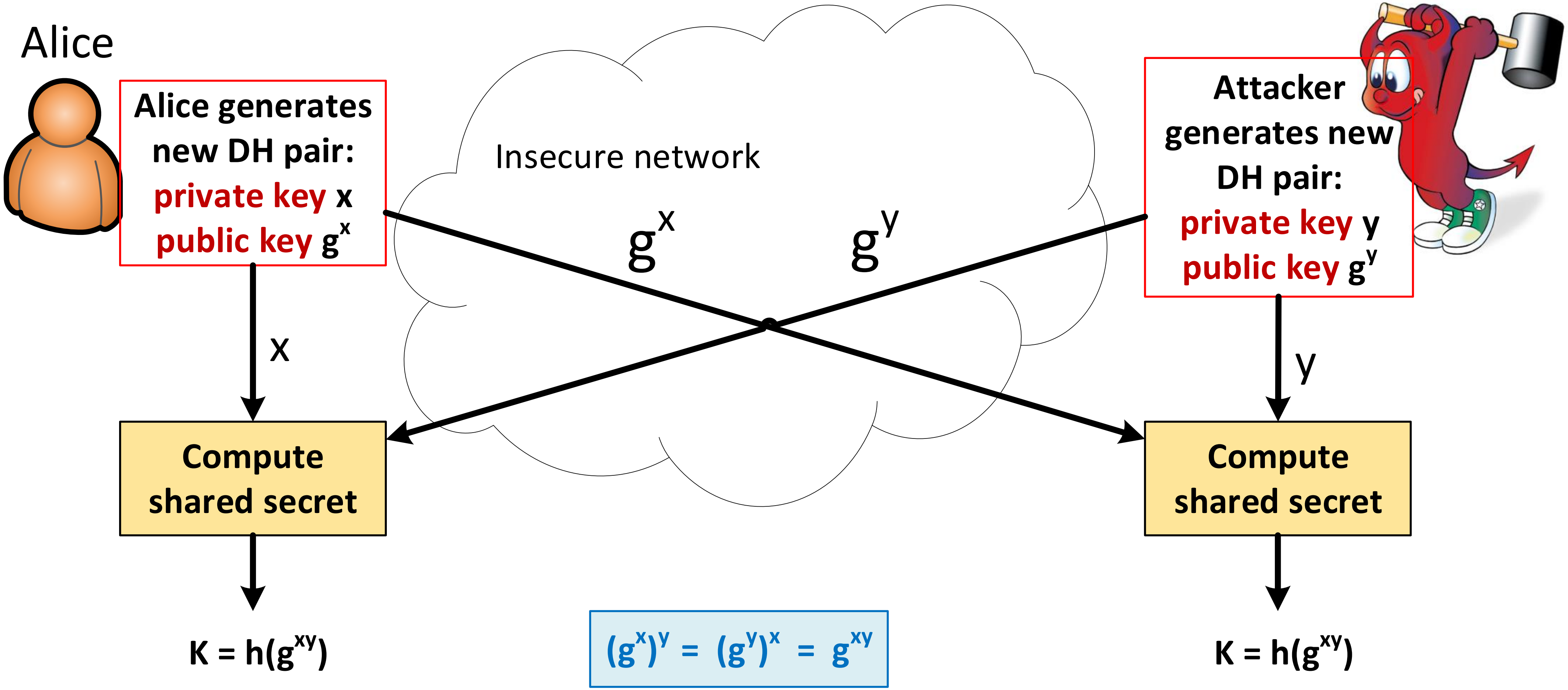
Diffie-Hellman key exchange



[Daemon graphics: www.freebsd.org]

Recall from earlier

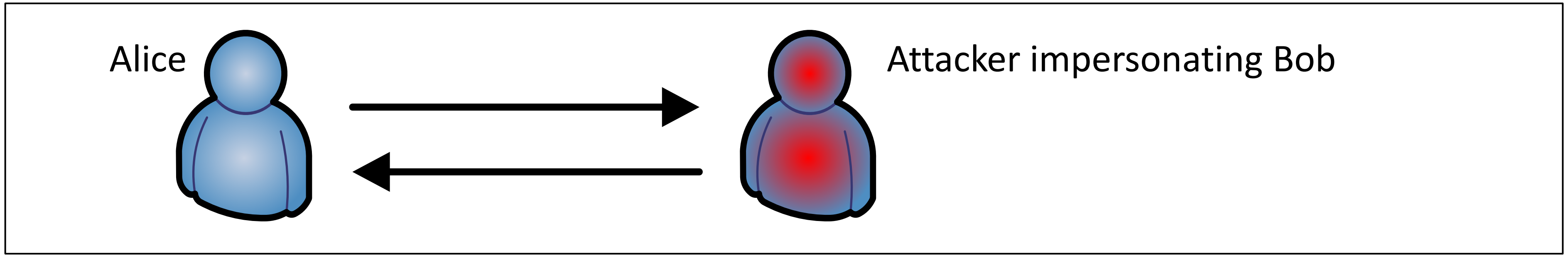
Impersonation attack



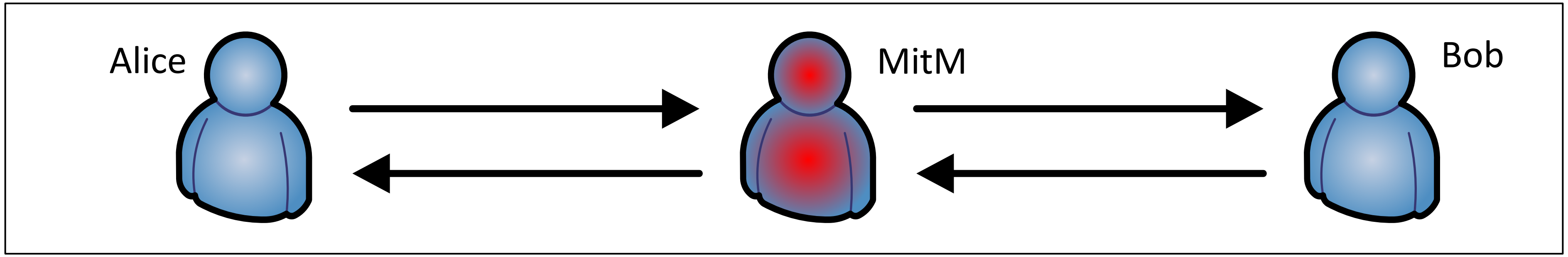
[Daemon graphics: www.freebsd.org]

Man-in-the-middle

- Unauthenticated Diffie-Hellman is secure against passive sniffing but insecure against active attackers
- Impersonation



- Man-in-the-middle (MitM):
  - Attacker impersonates Alice to Bob and vice versa, and modifies messages



Authenticated DH

1. $A \rightarrow B$: A, B, N_A, g, p, g^x, $Sign_A$("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. $B \rightarrow A$: A, B, N_B, g^y, $Sign_B$("Msg2", A, B, N_B, g^y), $Cert_B$,
   MAC_{SK}(A, B, "Responder done.")
3. $A \rightarrow B$: A, B, MAC_{SK}(A, B, "Initiator done.")

$SK = h(N_A, N_B, g^{xy})$

- Prevents impersonation and MitM attacks
- Why so complicated?

Authenticated DH

1. $A \rightarrow B$: A, B, N_A, g, p, g^x, $Sign_A$("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. $B \rightarrow A$: A, B, N_B, g^y, $Sign_B$("Msg2", A, B, N_B, g^y), $Cert_B$,
   MAC_{SK}(A, B, "Responder done.")
3. $A \rightarrow B$: A, B, MAC_{SK}(A, B, "Initiator done.")

$SK = h(N_A, N_B, g^{xy})$

- Signatures for authentication, nonces for freshness, MAC for key confirmation
- How do A and B know each other's public signature keys?

Authenticated DH

1. A → B: A, B, N_A, g, p, g^x, $Sign_A$("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. B → A: A, B, N_B, g^y, $Sign_B$("Msg2", A, B, N_B, g^y), $Cert_B$, MAC_{SK}(A, B, "Responder done.")
3. A → B: A, B, MAC_{SK}(A, B, "Initiator done.")

SK = h(N_A, N_B, g^{xy})

- Signatures for authentication, nonces for freshness, MAC for key confirmation
- How do A and B know each other's public signature keys?

Authenticated DH

1. A → B: A, B, N_A, g, p, g^x, $Sign_A$("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. B → A: A, B, N_B, g^y, $Sign_B$("Msg2", A, B, N_B, g^y), $Cert_B$,
   MAC_{SK}(A, B, "Responder done.")
3. A → B: A, B, MAC_{SK}(A, B, "Initiator done.")

$SK = h(N_A, N_B, g^{xy})$

- Signatures for authentication, nonces for freshness, MAC for key confirmation
- How do A and B know each other's public signature keys?

Authenticated DH

1. $A \rightarrow B$: A, B, N_A, g, p, g^x, $Sign_A$("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. $B \rightarrow A$: A, B, N_B, g^y, $Sign_B$("Msg2", A, B, N_B, g^y), $Cert_B$,
   MAC_{SK}(A, B, "Responder done.")
3. $A \rightarrow B$: A, B, MAC_{SK}(A, B, "Initiator done.")

$SK = h(N_A, N_B, g^{xy})$

- Signatures for authentication, nonces for freshness, MAC for key confirmation
- How do A and B know each other's public signature keys?

Authenticated DH

1. A → B: A, B, N_A, g, p, g^x, $Sign_A$(“Msg1”, A, B, N_A, g, p, g^x), $Cert_A$
2. B → A: A, B, N_B, g^y, $Sign_B$(“Msg2”, A, B, N_B, g^y), $Cert_B$,
   MAC_{SK}(A, B, “Responder done.”)
3. A → B: A, B, MAC_{SK}(A, B, “Initiator done.”)

$SK = h(N_A, N_B, g^{xy})$

- Signatures for authentication, nonces for freshness, MAC for key confirmation
- How do A and B know each other’s public signature keys?

Authenticated DH

1. $A \rightarrow B$: A, B, N_A, g, p, g^x, $Sign_A$("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. $B \rightarrow A$: A, B, N_B, g^y, $Sign_B$("Msg2", A, B, N_B, g^y), $Cert_B$,
   MAC_{SK}(A, B, "Responder done.")
3. $A \rightarrow B$: A, B, MAC_{SK}(A, B, "Initiator done.")

$SK = h(N_A, N_B, g^{xy})$

Certificates – in the next lecture

- Signatures for authentication, nonces for freshness, MAC for key confirmation
- How do A and B know each other's public signature keys?

SUMMARY

List of key concepts

- Dolev-Yao adversary model
- Security goals: confidentiality (secrecy), integrity, data-origin authentication, availability
- Sniffing (eavesdropping, interception), data modification, spoofing, impersonation, DoS
- Replay attacks, freshness, timestamp, sequence number, nonce
- Unauthenticated Diffie-Hellman, impersonation and MitM attack, passive and active attack
- Authentication, key confirmation

Related reading

- Stallings and Brown: Computer security, principles and practice, 4th ed., chapters 20-21
  - other Stallings books have similar sections