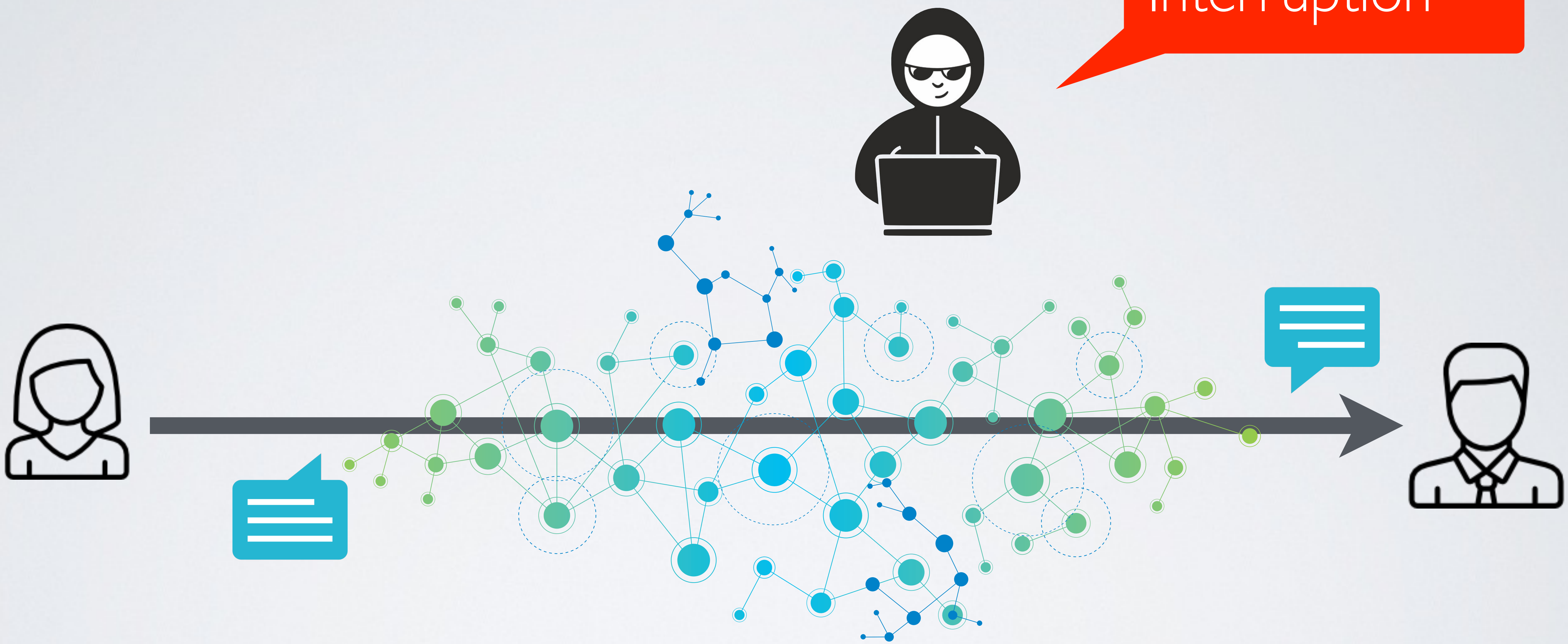
Symmetric Cryptography Protocols

Thierry Sans

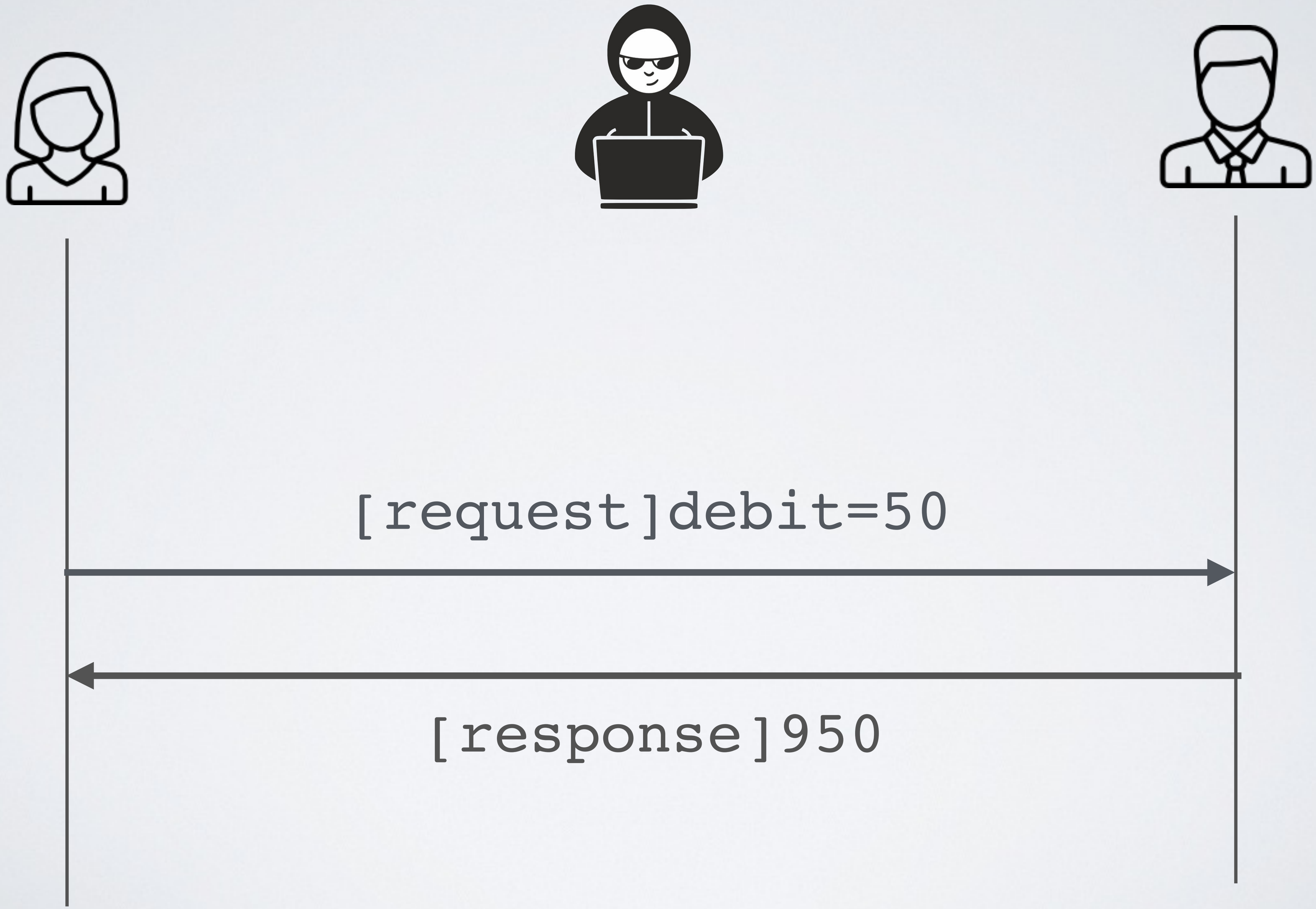
Security goals vs attacker's model

Interception
Modification
Fabrication
Interruption

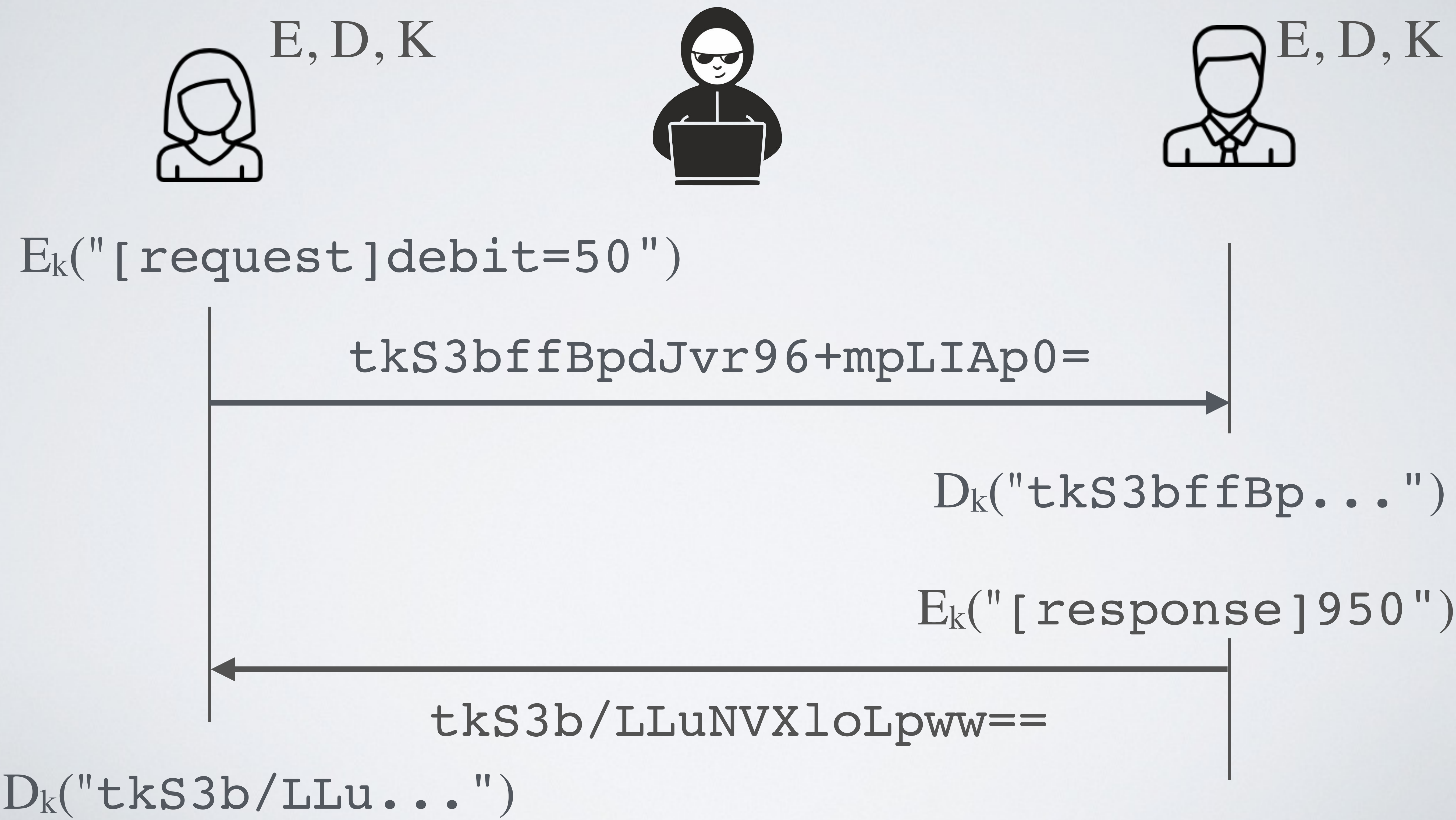


Let us consider **confidentiality, integrity** ~~**and availability**~~

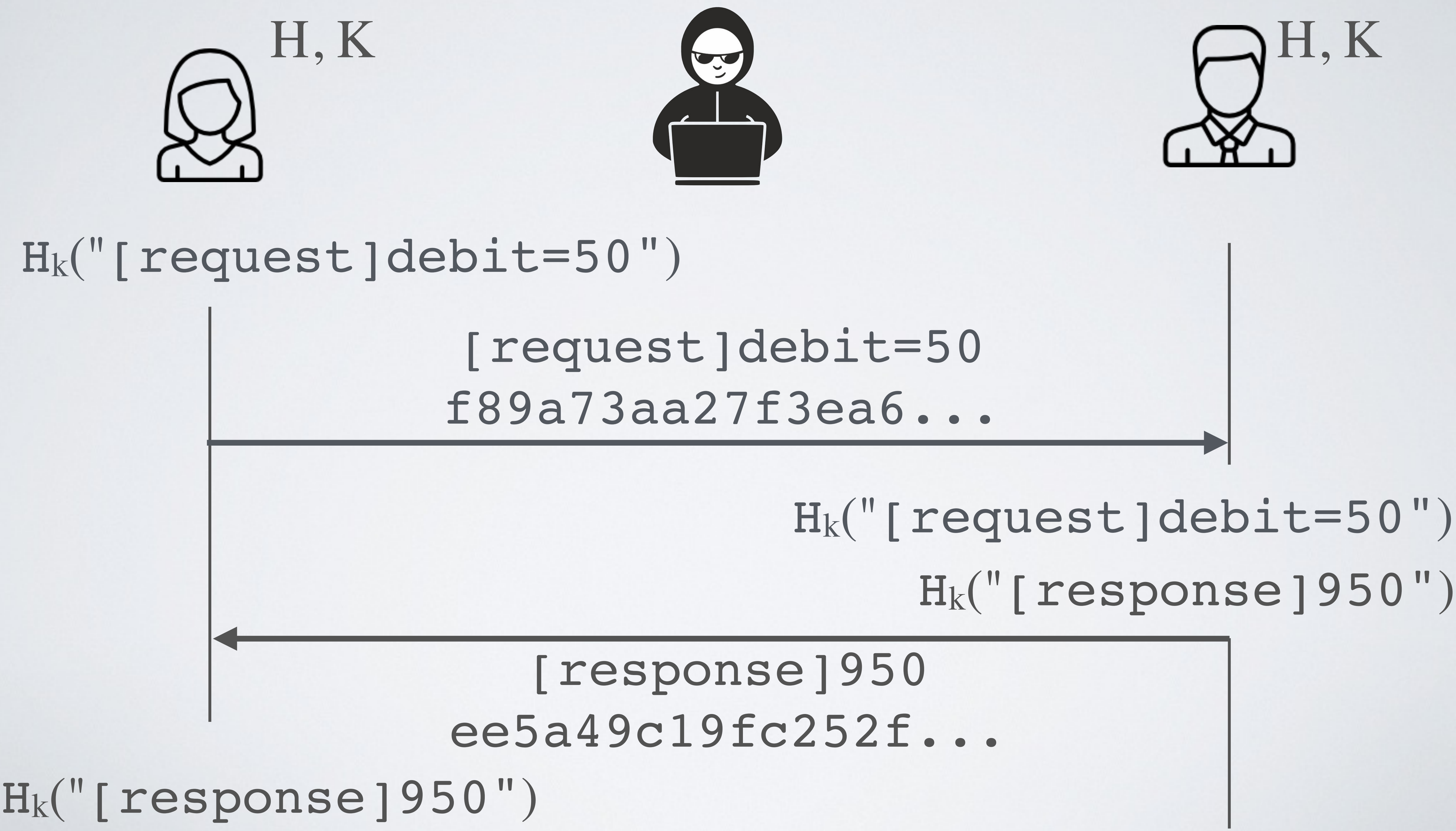
Example



Ensuring confidentiality with encryption



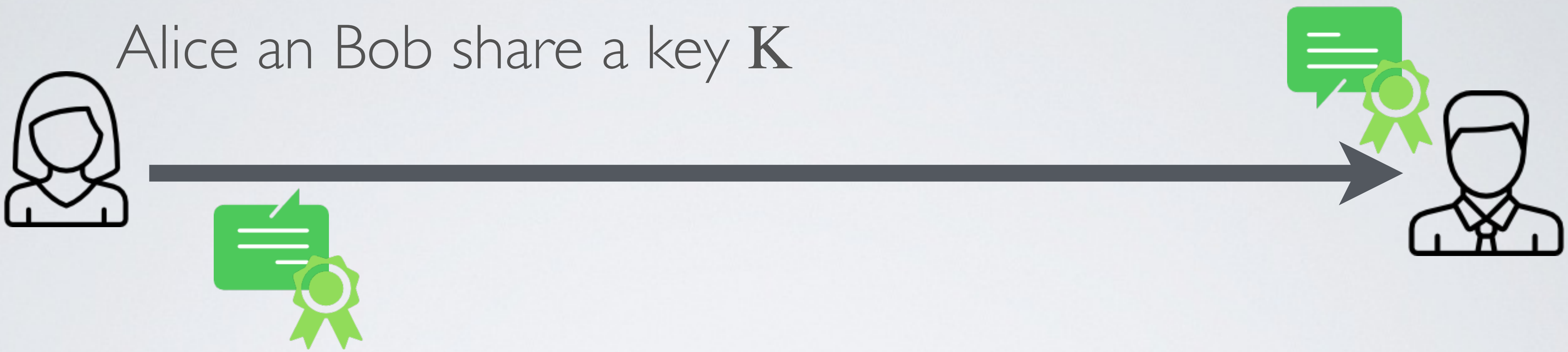
Ensuring integrity with an HMAC



Security mechanisms

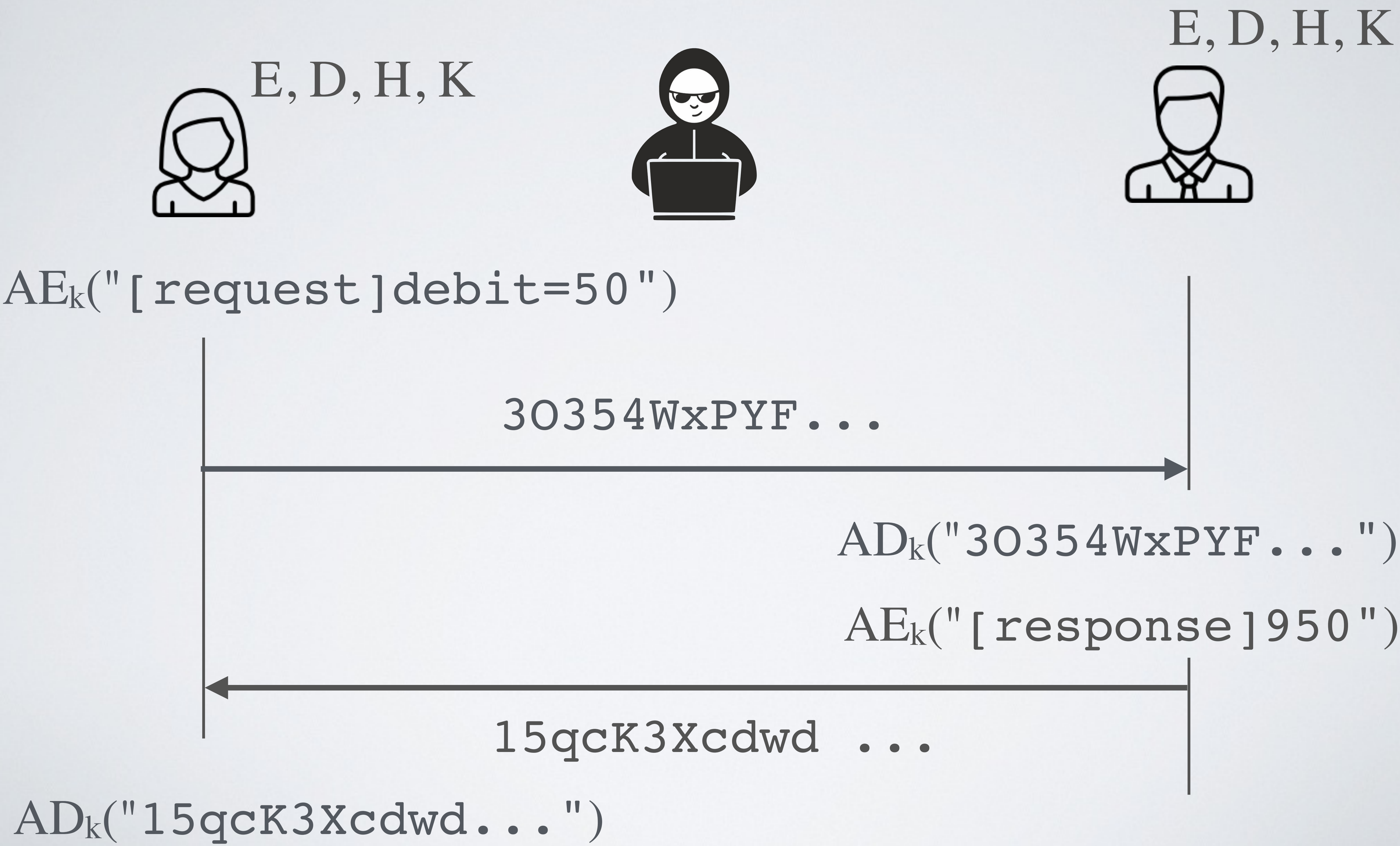
	Encryption	MAC	Authenticated Encryption
Confidentiality	✓	✗	✓
Integrity	✗	✓	✓

Authenticated Encryption (2013)



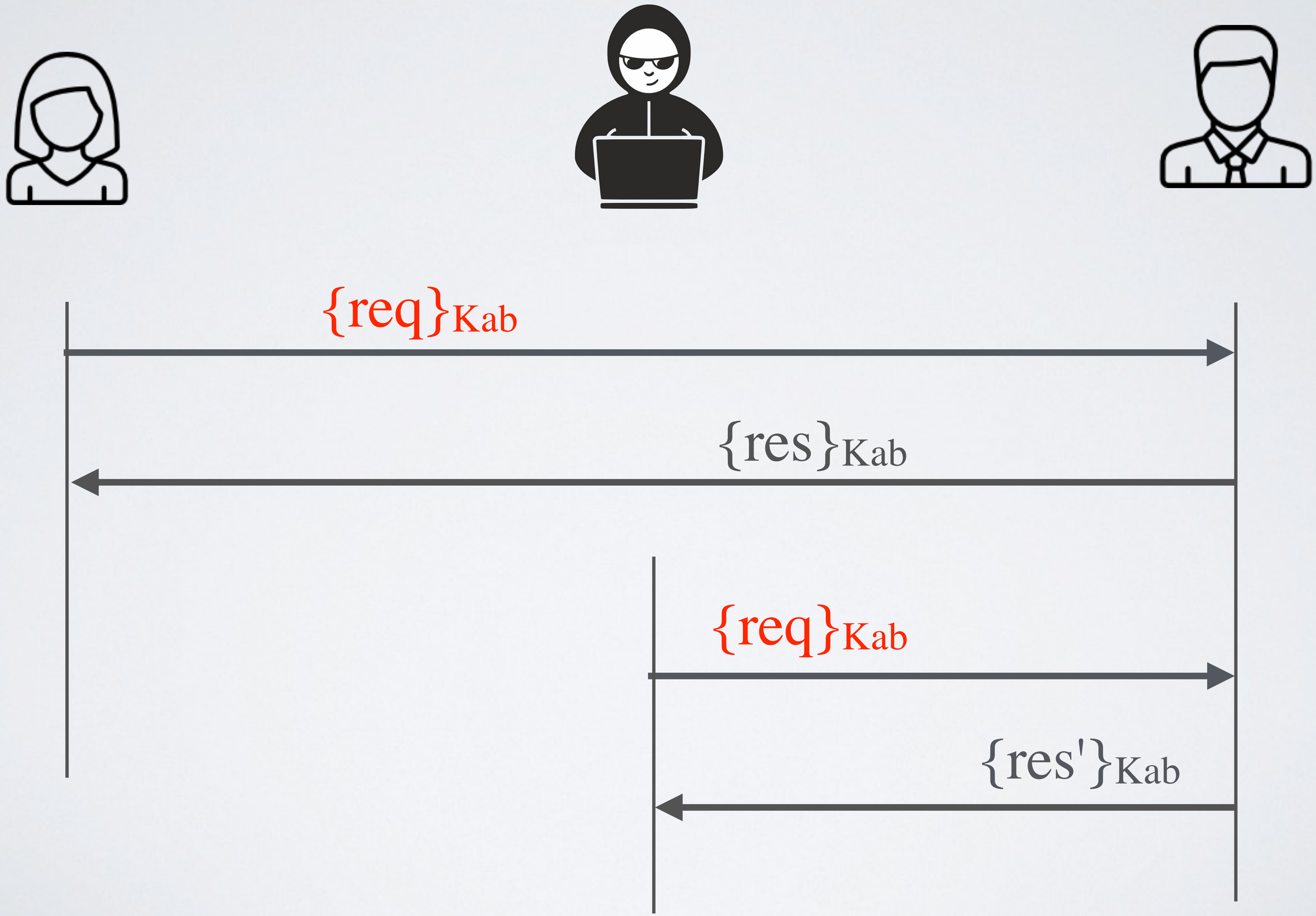
Encrypt-and-MAC (E&M)	$AE_k(m) = E_K(m) \parallel H_K(m)$	*SSH*
MAC-then-Encrypt (MtE)	$AE_k(m) = E_K(m \parallel H_K(m))$	*SSL*
Encrypt-then-MAC (EtM)	$AE_k(m) = E_K(m) \parallel H_K(E_K(m))$	*AES-GCM*

Ensuring confidentiality and integrity with Authenticated Encryption



Replay attacks

Replay attack

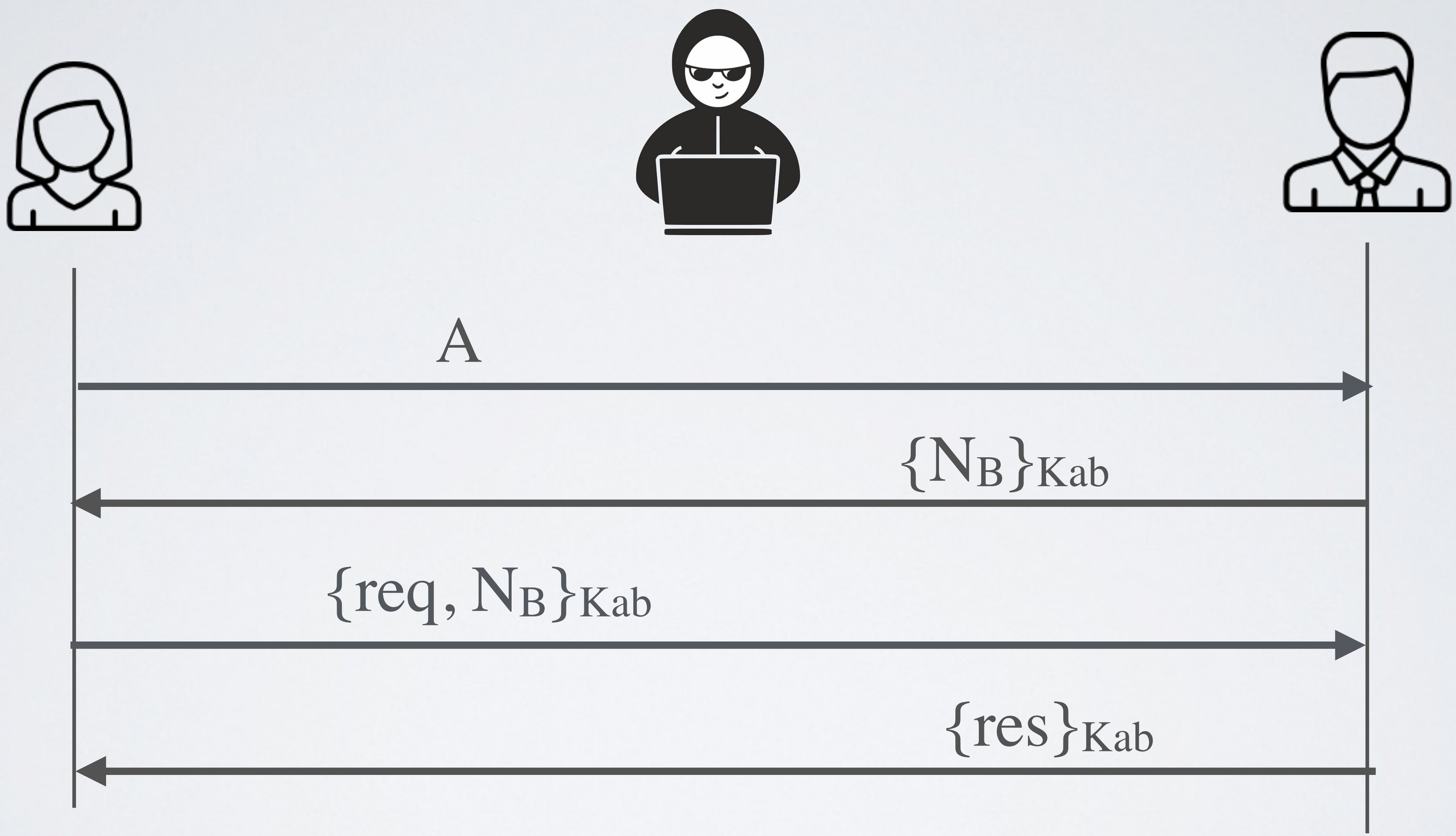


Counter replay attacks

Several solutions:

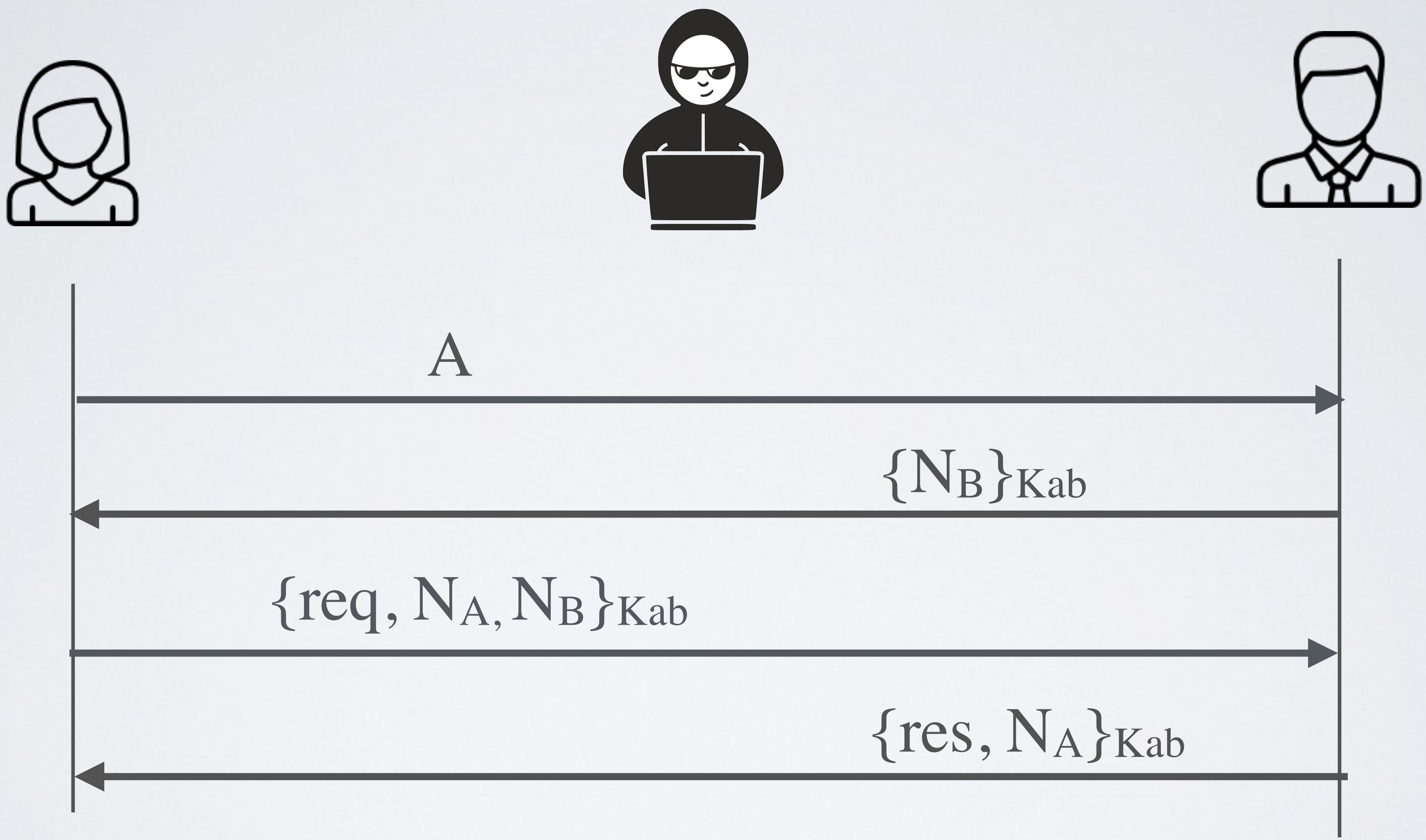
- **use a nonce (random number)**
- use sequence numbers
- use timestamps
- have fresh key for every transaction
  (key distribution problem)

Defeat replay attack with a nonce (not fully secured)



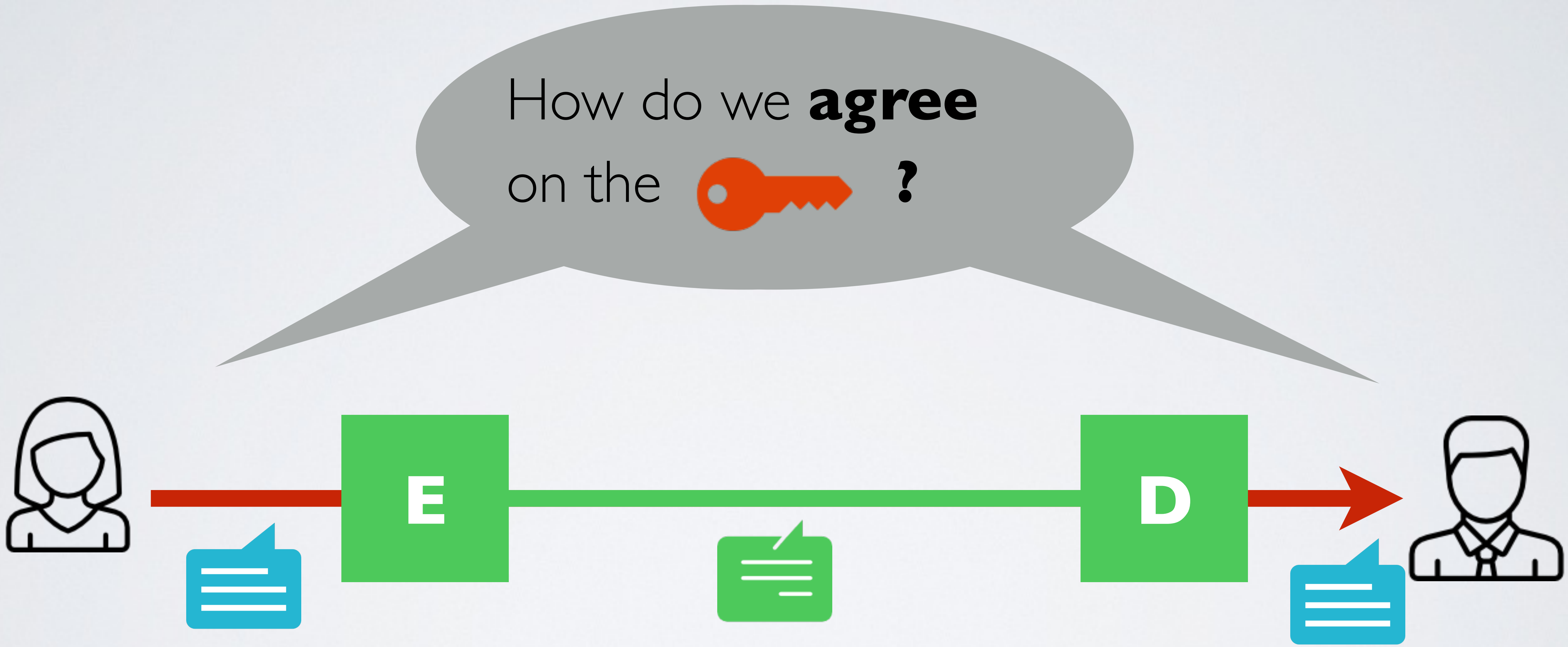
Replay attack on the response!

Defeat replay attack with a double nonce

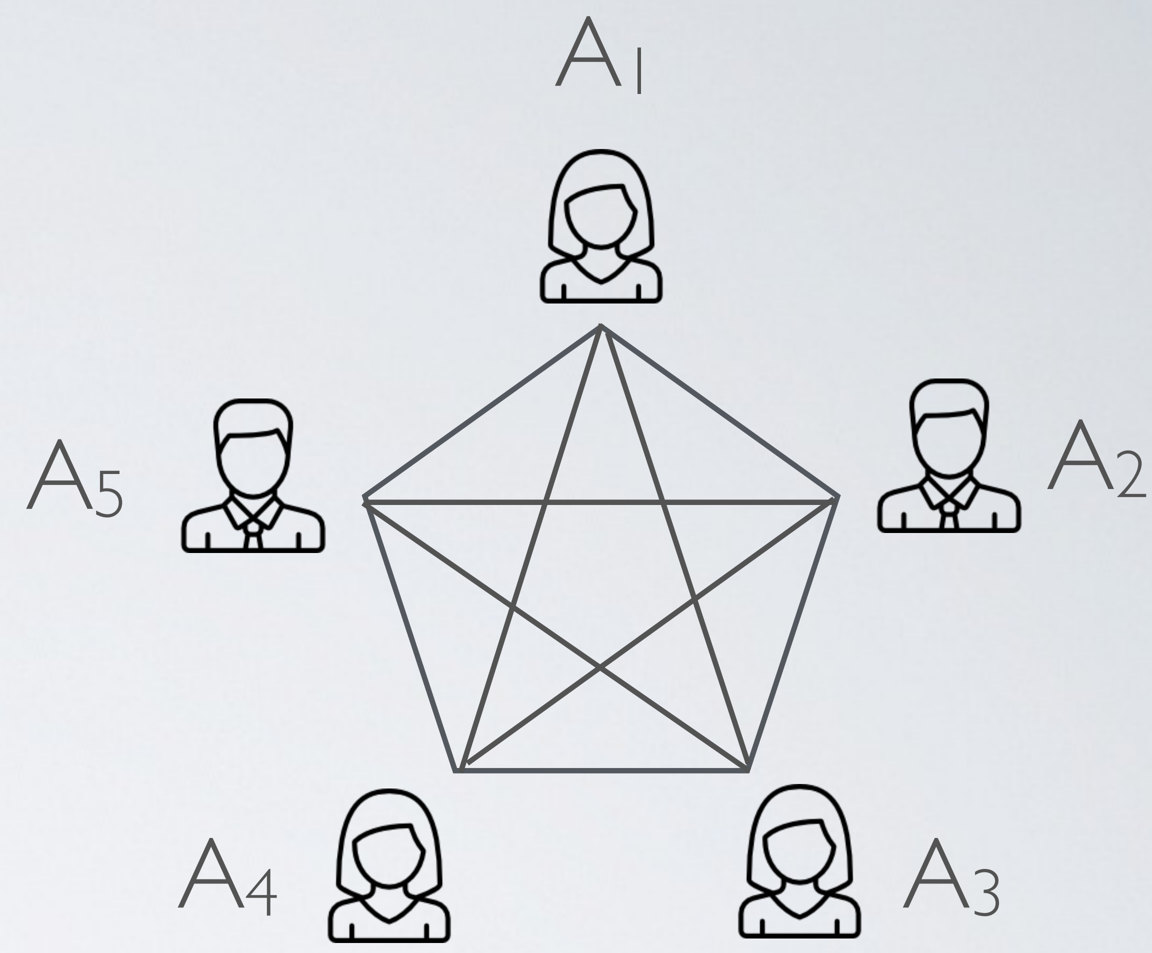


The challenge of key exchange

The big challenge with symmetric cryptosystems?



Naive Key Management

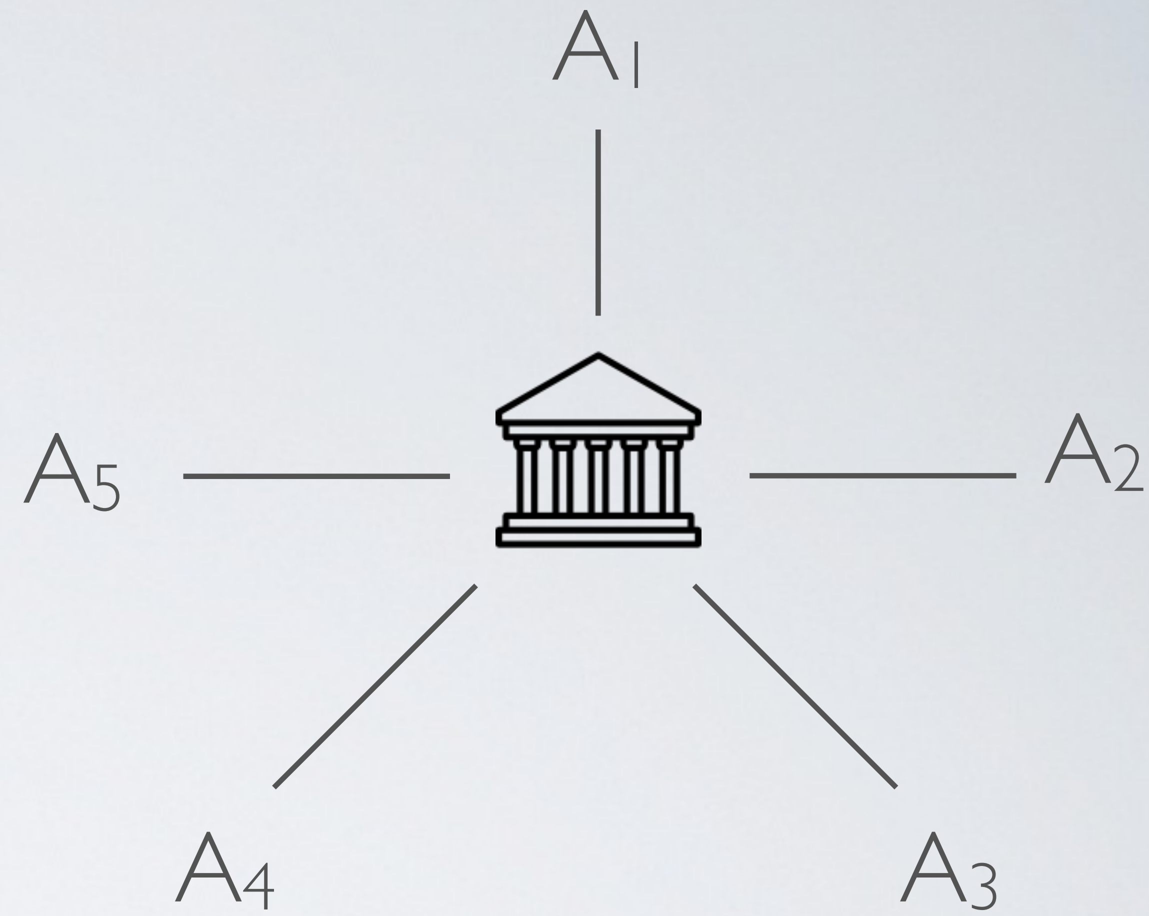


A_1, A_2 … A_5 want to talk

➡ Each pair needs a key : $n (n-1) / 2$ keys

- Keys must be exchanged physically using a secure channel

(Better) centralized solution



$A_1, A_2 \ldots A_5$ can talk to the KDC (Key Distribution Center)

- ➡ When A_i and A_j want to talk, the KDC can generate a new key and distribute it to them

- ◉ We still have n keys to distribute somehow using a secure channel
- ◉ The KDC must be trusted
- ◉ The KDC is a single point of failure

- ➡ The is how *Kerberos* works

The Needham-Shroeder symmetric protocol for key exchange

Assumptions

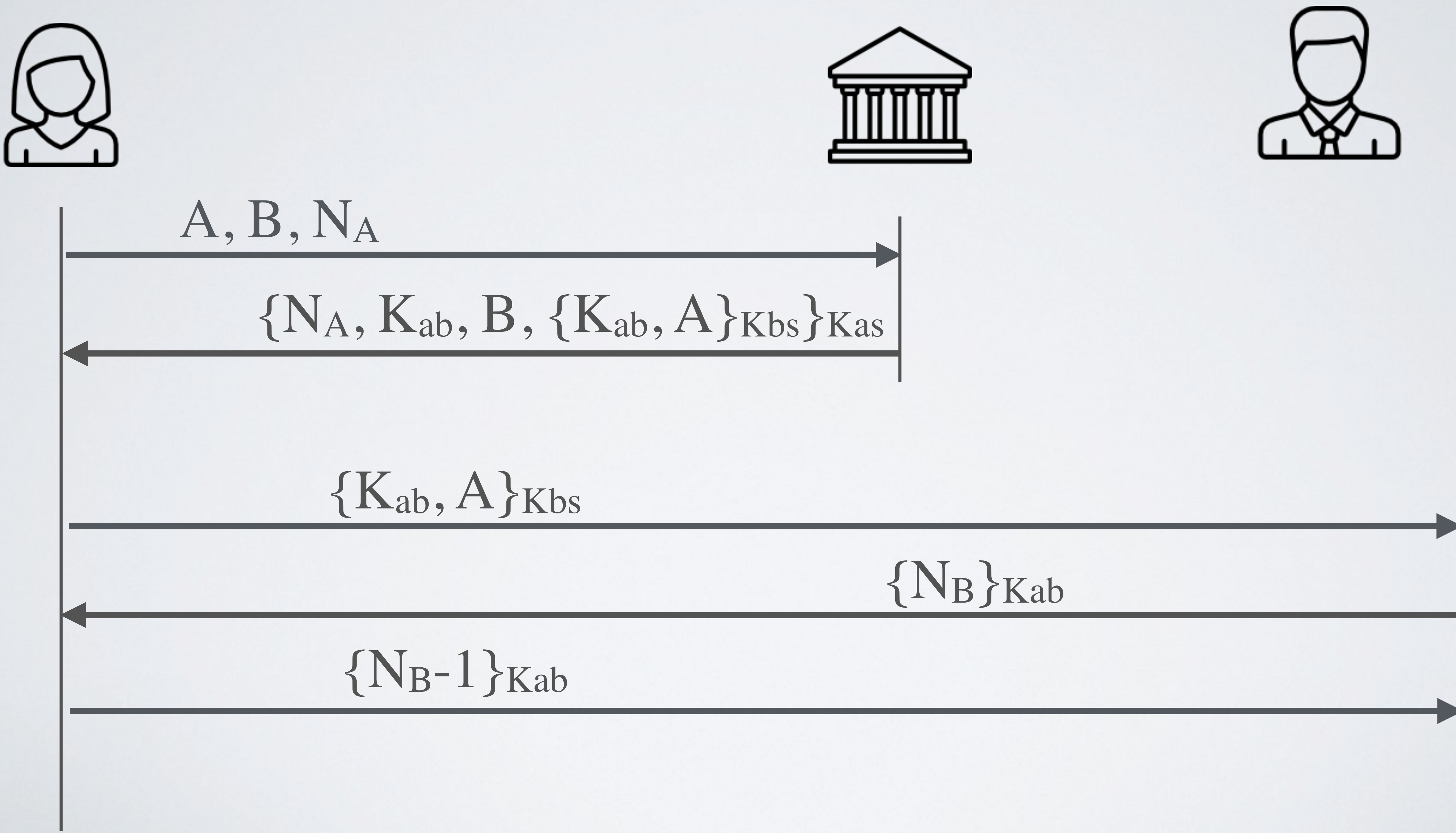
- 4 principals : **A**lice, **B**ob, **M**allory, Key Distribution **S**erver
- **S** shares a key with **A**, **B** and M respectively K_{as}, K_{bs}, K_{ms}
- **A**, **B**, **M** and **S** talk to each other using the same protocol

Goals

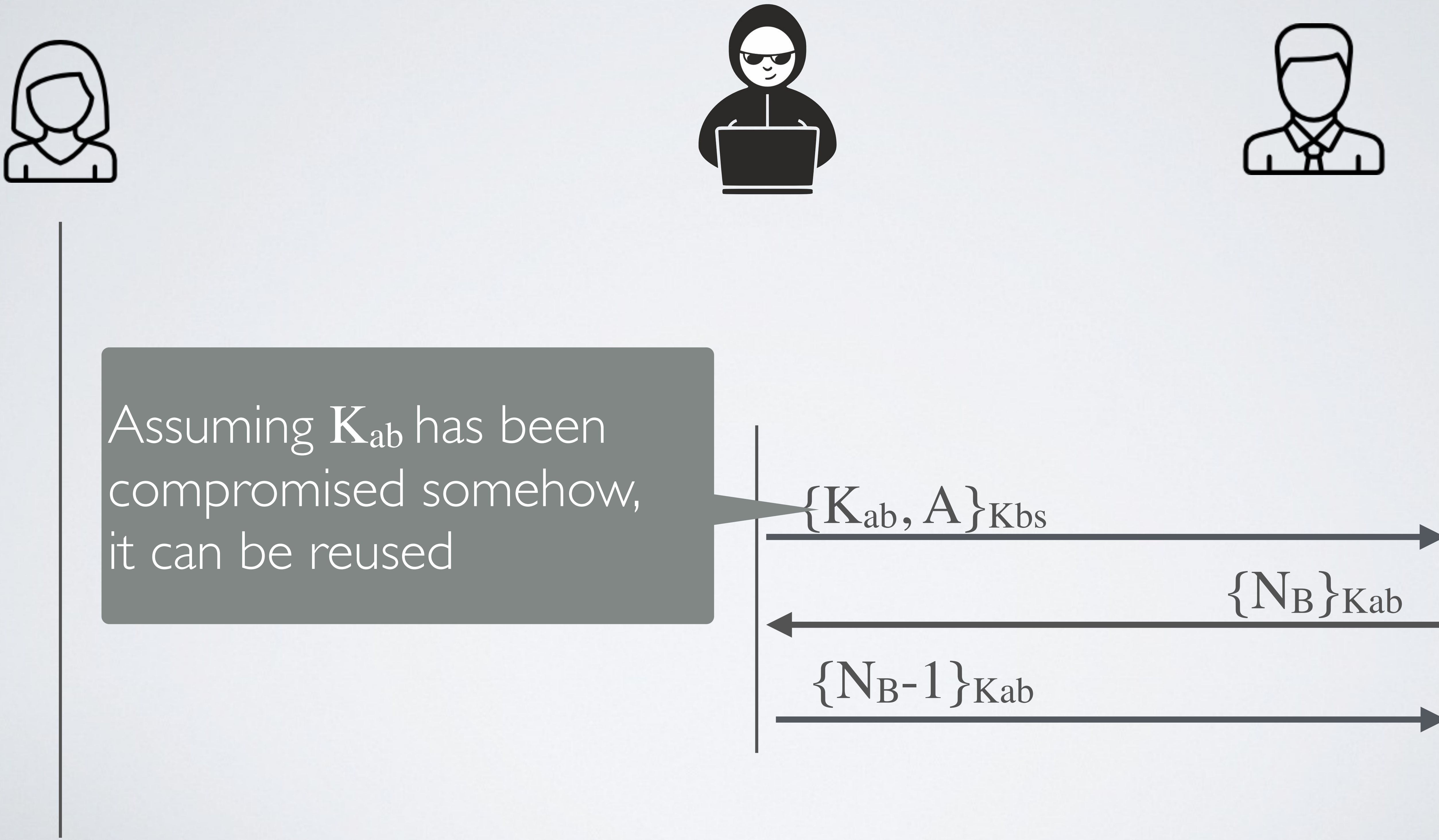
When two parties want to engage in the communication, they want to

1. make sure that they talk to the right person (authentication)
2. establish a session key

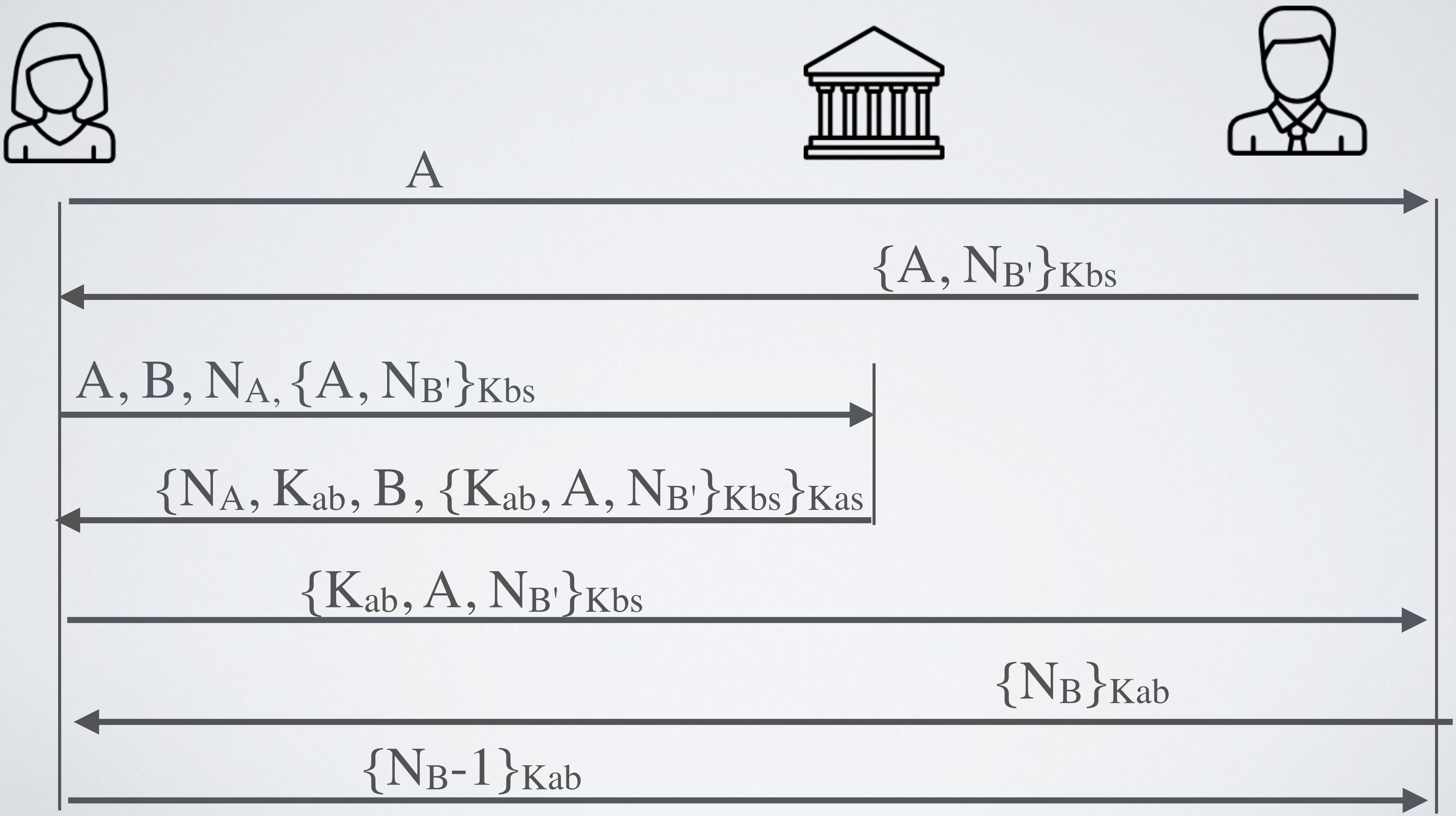
The vulnerable version of the protocol (1978)



Replay attack (1981)



The fix (1987)


A
$\{A, N_{B'}\}_{Kbs}$
$A, B, N_{A}, \{A, N_{B'}\}_{Kbs}$
$\{N_A, K_{ab}, B, \{K_{ab}, A, N_{B'}\}_{Kbs}\}_{Kas}$
$\{K_{ab}, A, N_{B'}\}_{Kbs}$
$\{N_B\}_{Kab}$
$\{N_B-1\}_{Kab}$

Limitations of using a key distribution centre

The key distribution server is a bootleneck and **weak link**

- The attacker could record the key exchange and the encrypted session, if one day either **Kas** or **Kbs** is broken, the attacker can decrypt the session

➡ Having a KDC does not offer "Perfect Forward Secrecy"

Can we avoid having a KDC ?

Could Alice and Bob could magically come up with a key without exchanging it over the network?

➡ The magic is called **Diffie-Hellman-Merkle Protocol**

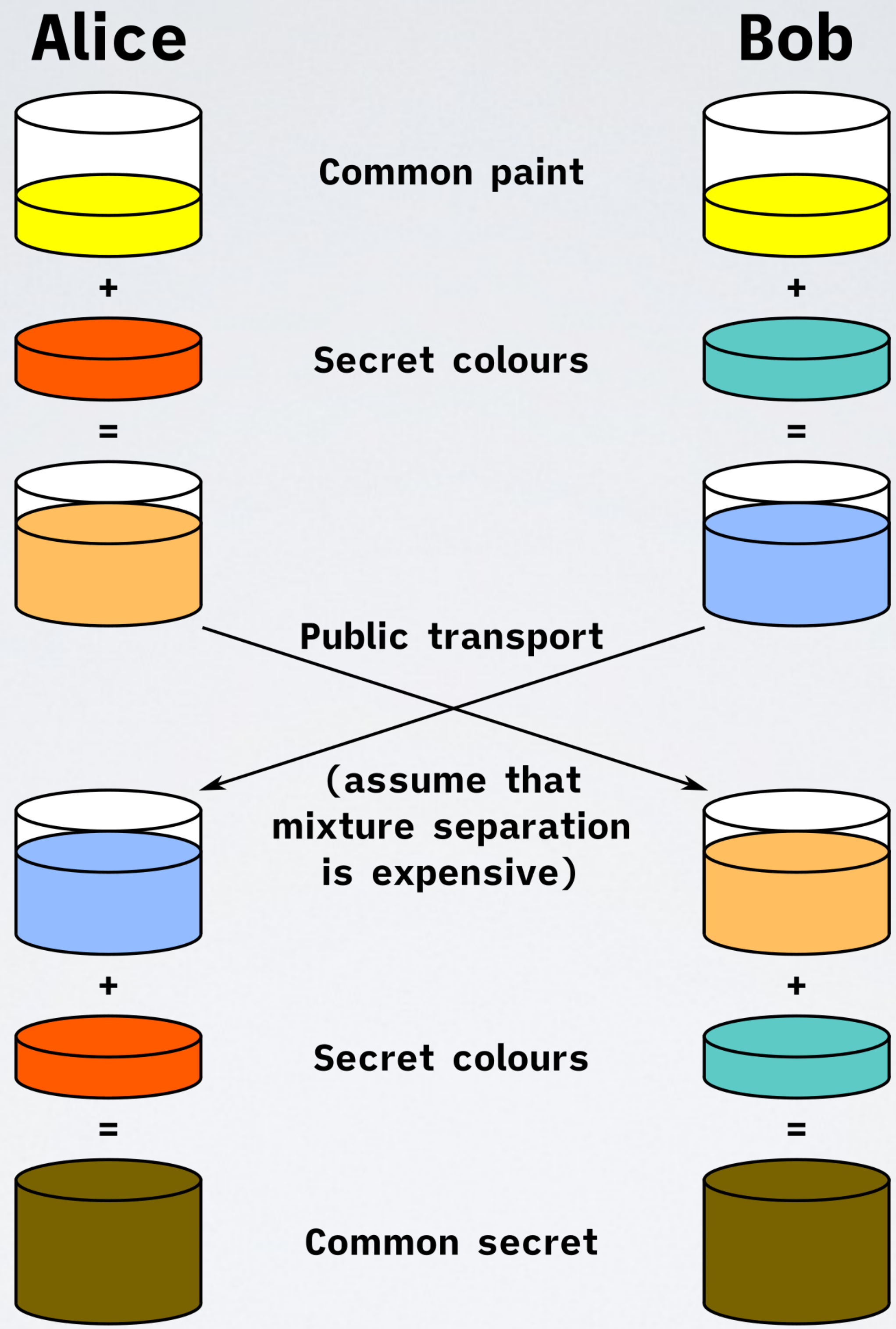
The Diffie-Hellman-Merkel key exchange protocol

p,g

a

$A = g^a \bmod p$

$K = B^a \bmod p$



p,g

b

$B = g^b \bmod p$

$K = A^b \bmod p$

$$\boxed{K = g^{ab} \bmod p = (g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p}$$

The Diffie-Hellman-Merkel key exchange protocol

1. Generates public numbers **p** and **g** such that **g** if co-prime to **p-1**
2. Generates a secret number **a**
3. Sends $A = g^a \bmod p$ to Bob

$\xrightarrow{A,\ p,\ g}$

1. Generates a secret number **b**
2. Sends $B = g^b \bmod p$ back to Alice
3. Calculates the key $K = A^b \bmod p$

$\xleftarrow{B}$

4. Calculates the key $K = B^a \bmod p$

Diffie-Hellman-Merkle in practice

- $\mathbf{g}$ is small (either 3, 5 or 7 and fixed in practice)
- $\mathbf{p}$ is at least 2048 bits (and fixed in practice)
- private keys $\mathbf{a}$ and $\mathbf{b}$ are 2048 bits as well

➡ So the public values $\mathbf{A}$ and $\mathbf{B}$
and the master key $\mathbf{k}$ are 2048 bits

➡ Use $\mathbf{k}$ to derive an AES key using a Key Derivation Function
(usually HKDF - the HMAC-based Extract-and-Expand key derivation function)

A widely used key exchange protocol

Diffie-Hellman-Merkle is in many protocols

- SSH
- TLS (used by HTTPS)
- Signal (used by most messaging apps like Whatsapp)
- and so on …

✓ It is fast and requires two exchanges only

✓ Solves the problem of having a key distribution server

✓ Ensures Perfect Forward Secrecy

- But how to make sure Alice is talking to Bob and vice-versa?
  Diffie-Hellman-Merkle alone **does not ensure authentication**