



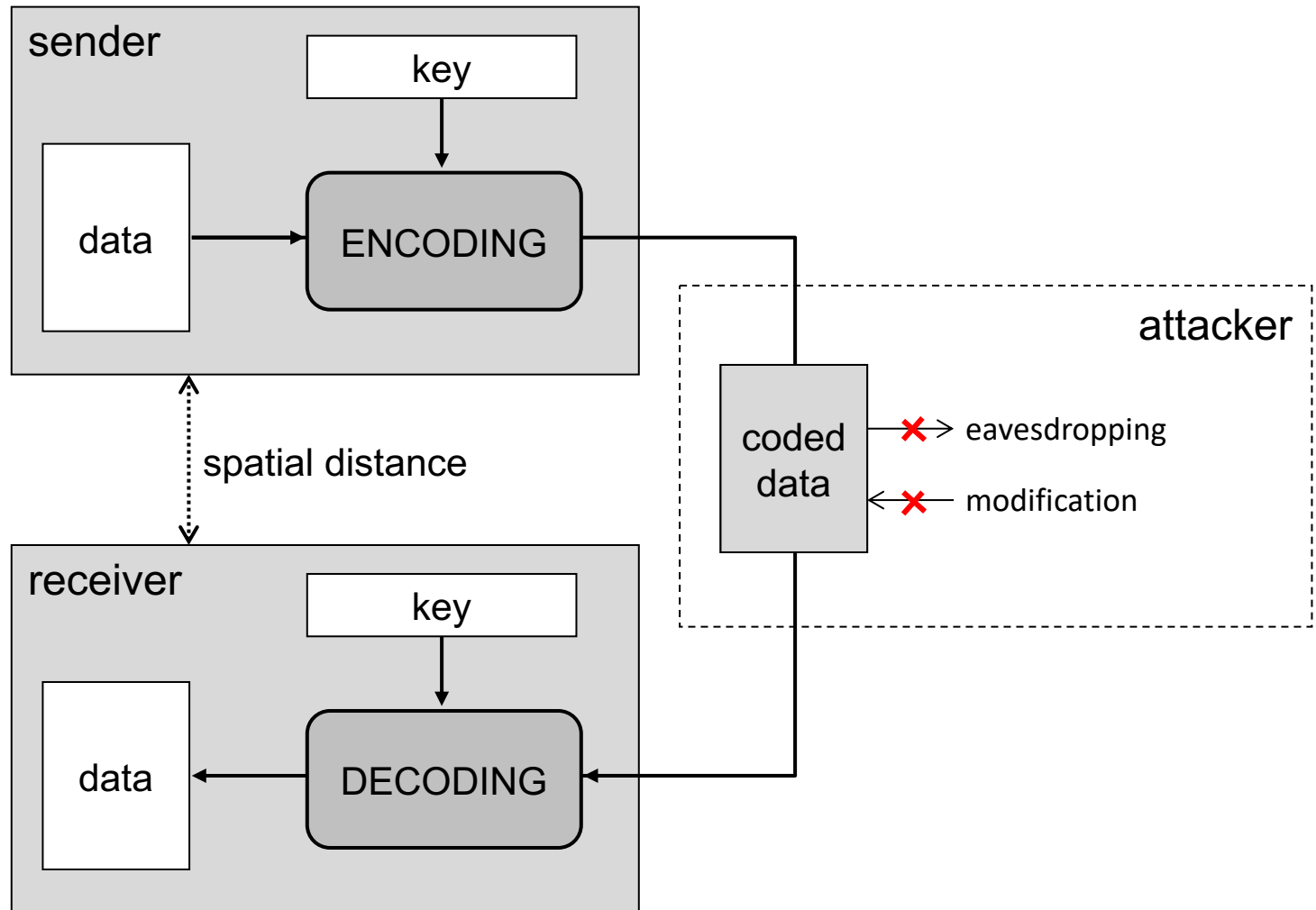
Key Exchange Protocols

Levente Buttyán

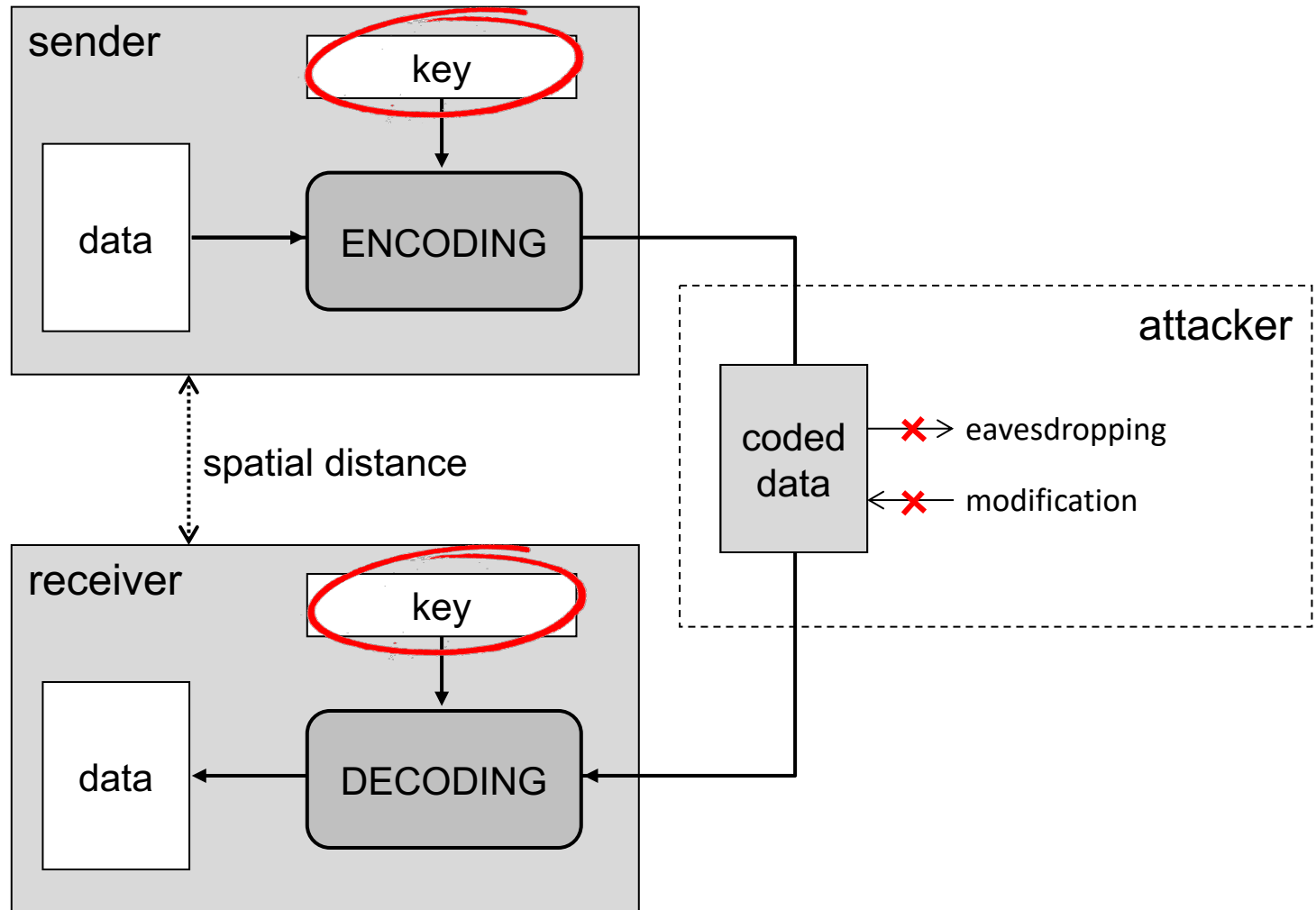
CrySyS Lab, BME

buttyan@crysys.hu

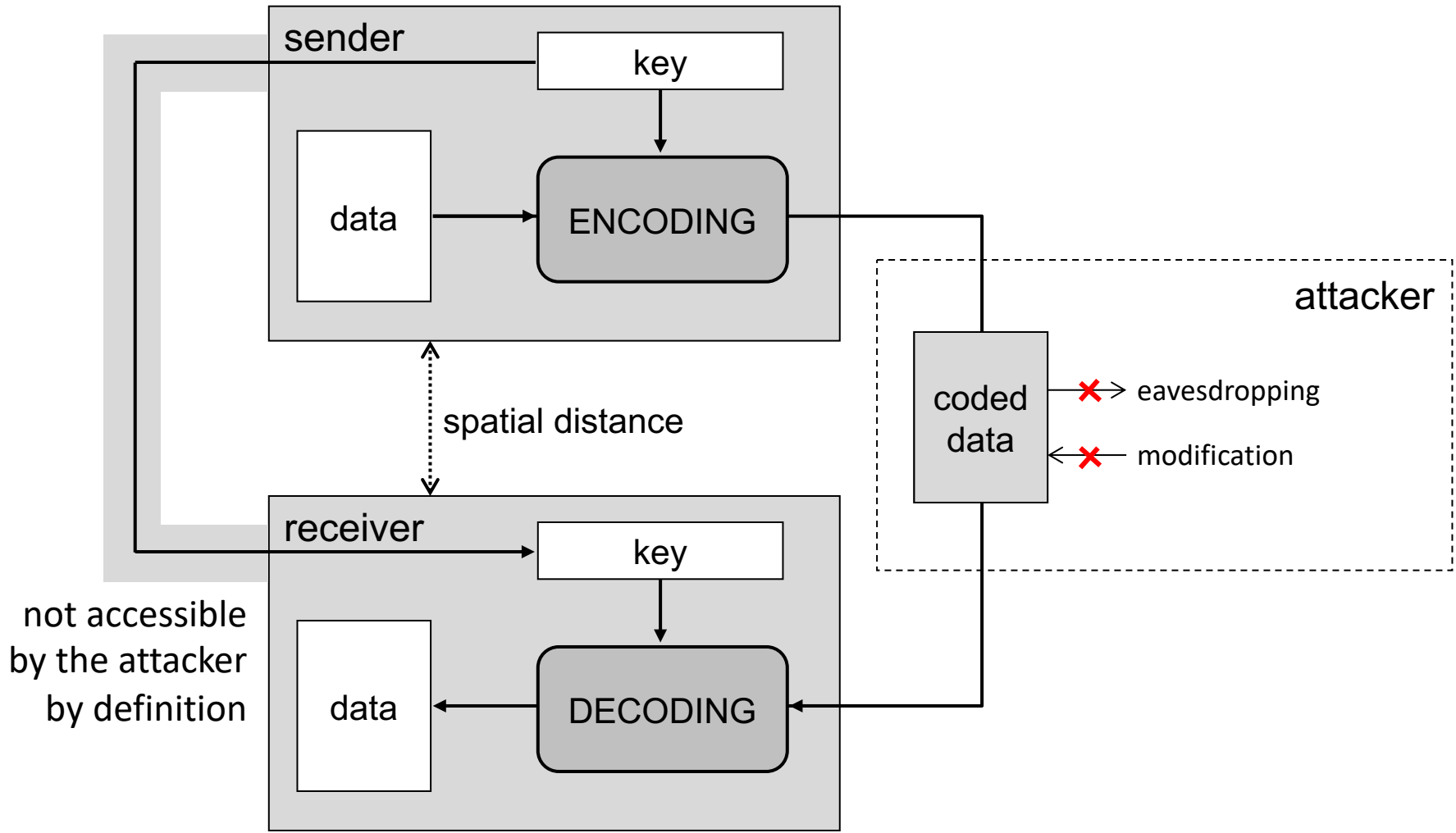
Model of cryptographic coding



How to establish the key?

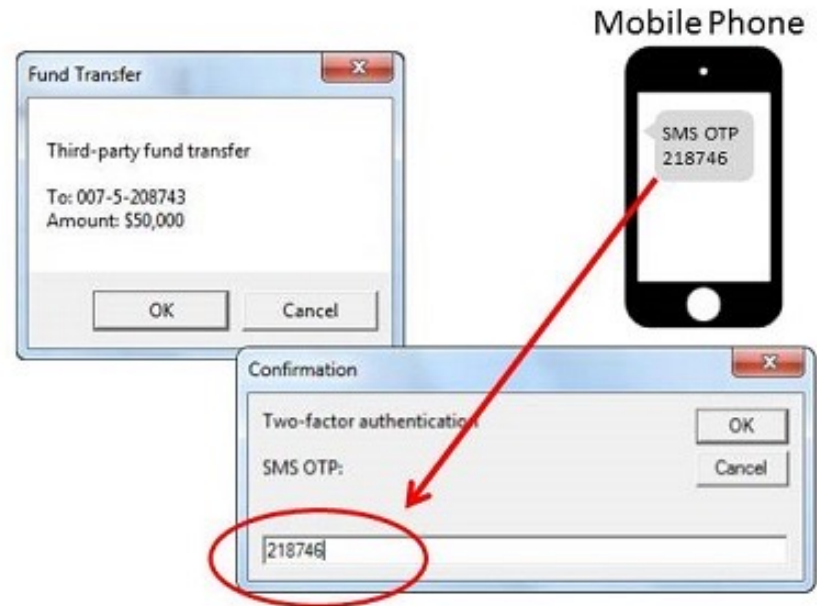


Out-of-band channels



Out-of-band channels

- examples
 - meeting in-person
 - separate trusted network



- could work in some applications, but...
- have some drawbacks
 - meeting in person: does not scale, inflexible, ...
 - separate network: expensive, needs extra technology, ...

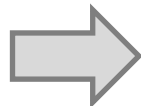
Can we solve the key establishment problem without an out-of-band channel?

Bad news:

No, not really ...

Good news:

We can limit the use of out-of-band channels to setting up a few keys, and then we can use these already established keys for setting up new keys without using out-of-band channels



key establishment protocols

Casting

- Alice and Bob
 - the parties that want to establish a shared key
- Trent
 - a third party trusted by both Alice and Bob
 - helps them accomplishing the task of setting up a key



Casting

- Eve (a.k.a. Mallory, Trudy)
 - the attacker who wants to
 - » obtain the key established between Alice and Bob
 - » impersonate Alice to Bob, or vice versa
 - she can be a legitimate protocol participant too!
 - in addition, she has full control over the communications of the honest parties Alice, Bob, and Trent
 - » can eavesdrop, modify, delete, inject, and replay messages
 - however, she cannot break the underlying cryptographic primitives used in the protocol

= attacker model for
key exchange protocols

Main design objectives

1. **Secrecy of the key:** When the protocol is executed by Alice and Bob, no other parties (with the possible exception of Trent) should learn the value of the established key.
2. **Key authentication:** If Alice believes that she successfully executed the protocol and established a new key K with Bob, then Bob was indeed present and he should believe that he executed the protocol and established the same key K with Alice.
3. **Key freshness:** Both parties should believe that the established key is fresh (new, not used before).
(if freshness cannot be verified, then Mallory may force Alice and Bob to re-establish and re-use an old key, which may lead to problems...)

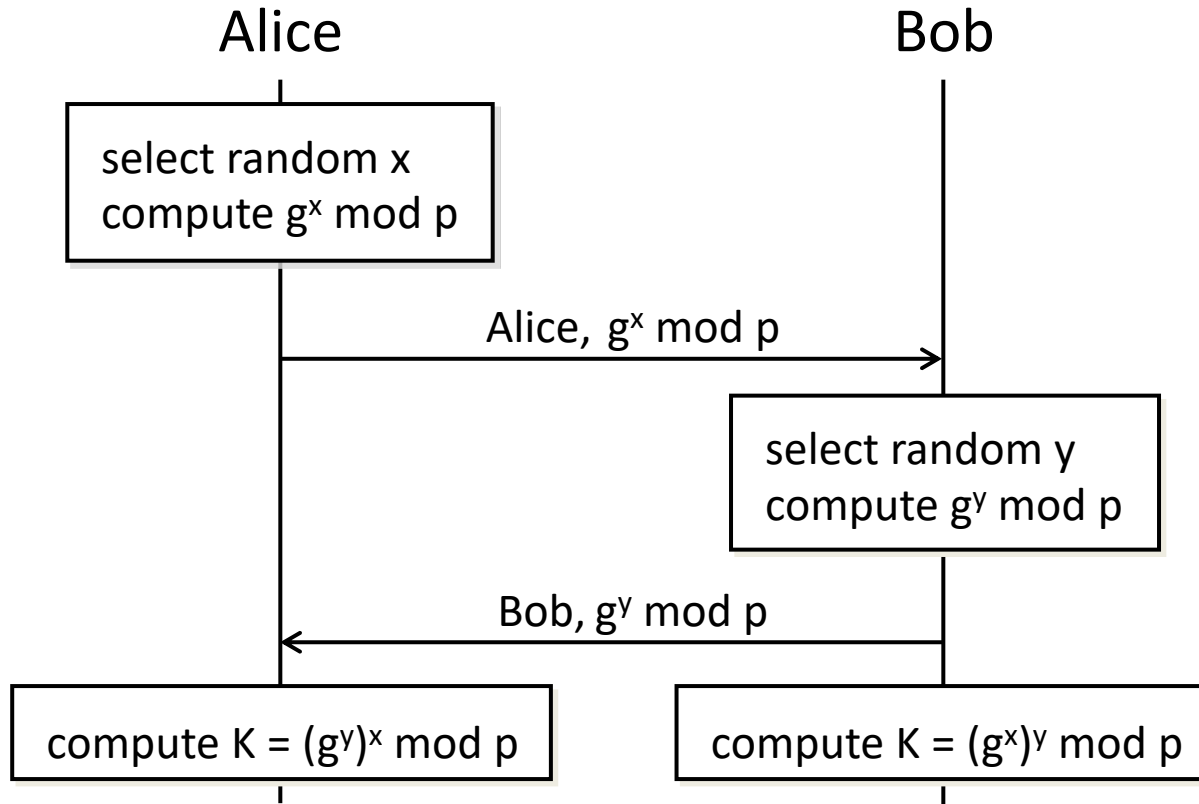
Basic classification of protocols

- key agreement protocols
 - the new key is derived by Alice and Bob as a function of information contributed by each of them, such that neither Alice nor Bob can predetermine the resulting value
 - example: Diffie-Hellman protocol
- key transport protocols
 - one party (Alice, Bob, or maybe Trent) creates a new key, and securely transfers it to the other party
 - example: Kerberos protocol, RSA based key exchange in TLS

Reminder on DH key agreement

public parameters:

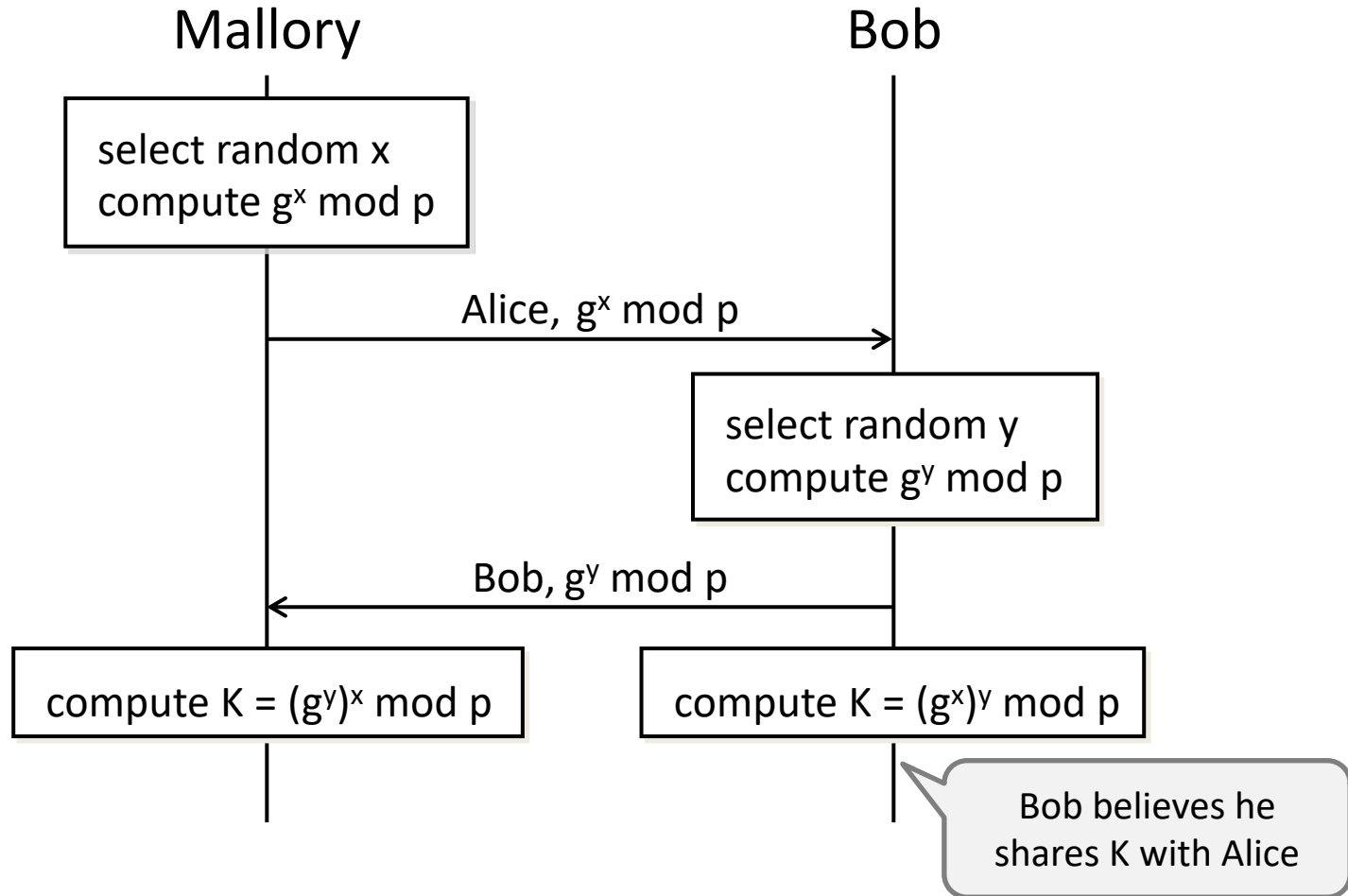
a large prime p and a generator g of a cyclic subgroup of $Z_p^* = \{1, 2, \dots, p-1\}$



An attack against the DH protocol

public parameters:

a large prime p and a generator g of a cyclic subgroup of $Z_p^* = \{1, 2, \dots, p-1\}$



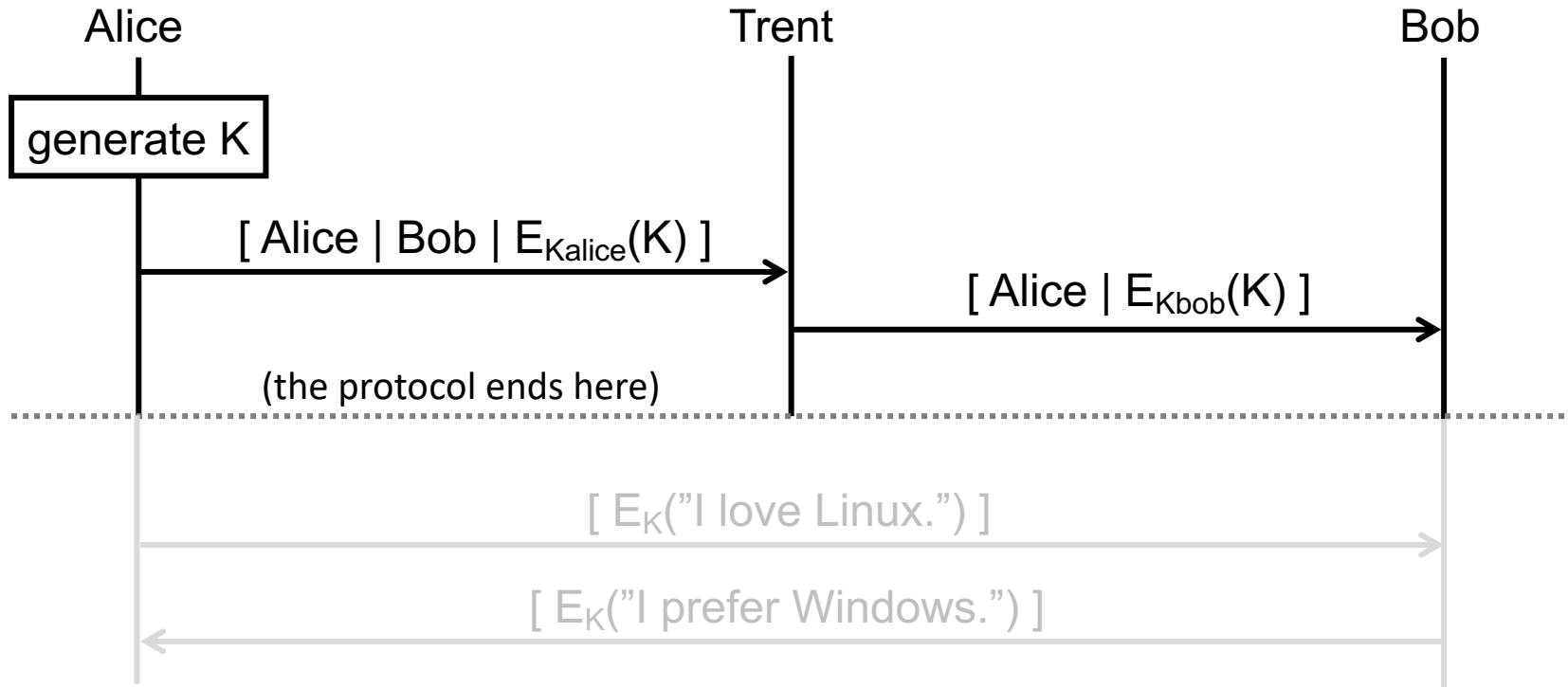
Key transport protocols

- Alice and Bob will rely on Trent, with whom they already share the long-term keys K_{alice} and K_{bob} , respectively (established via some out-of-band channel)
- the new key K will be protected from Mallory by encoding it with K_{alice} and/or K_{bob}

notes:

- it may seem that coding with K_{alice} and/or K_{bob} is actually an out-of-band channel
- but it is not, because we do not abstract away this encoding and we do allow Mallory to access messages encoded with K_{alice} and K_{bob}

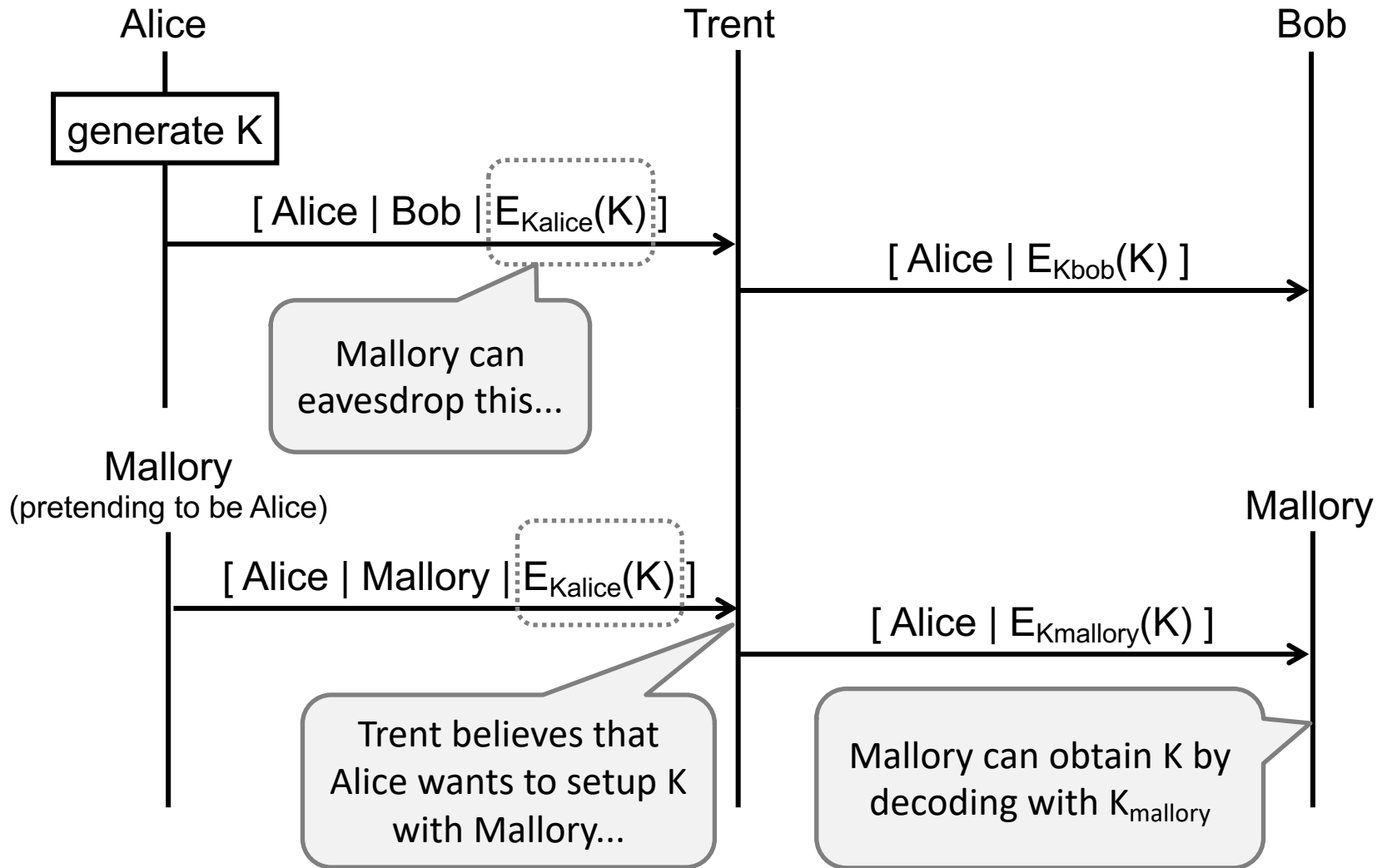
An example protocol



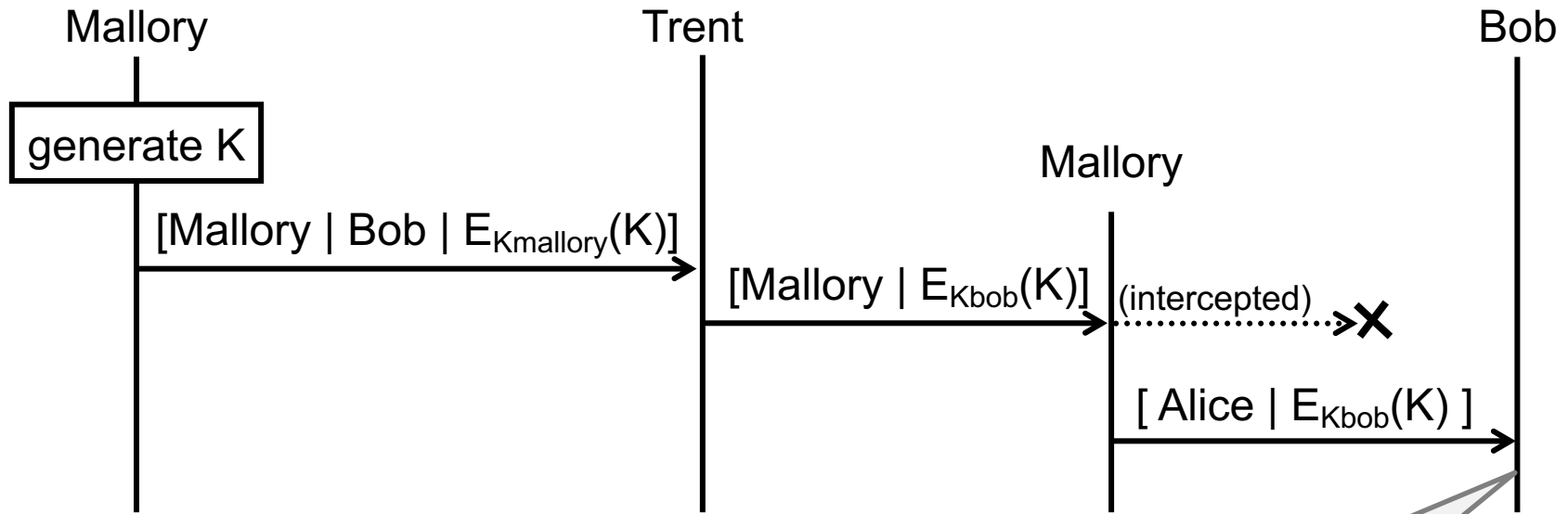
note: the protocol definition is just a *blueprint* (Alice and Bob are *roles*, rather than specific entities)

Is this protocol secure?

An attack against key secrecy



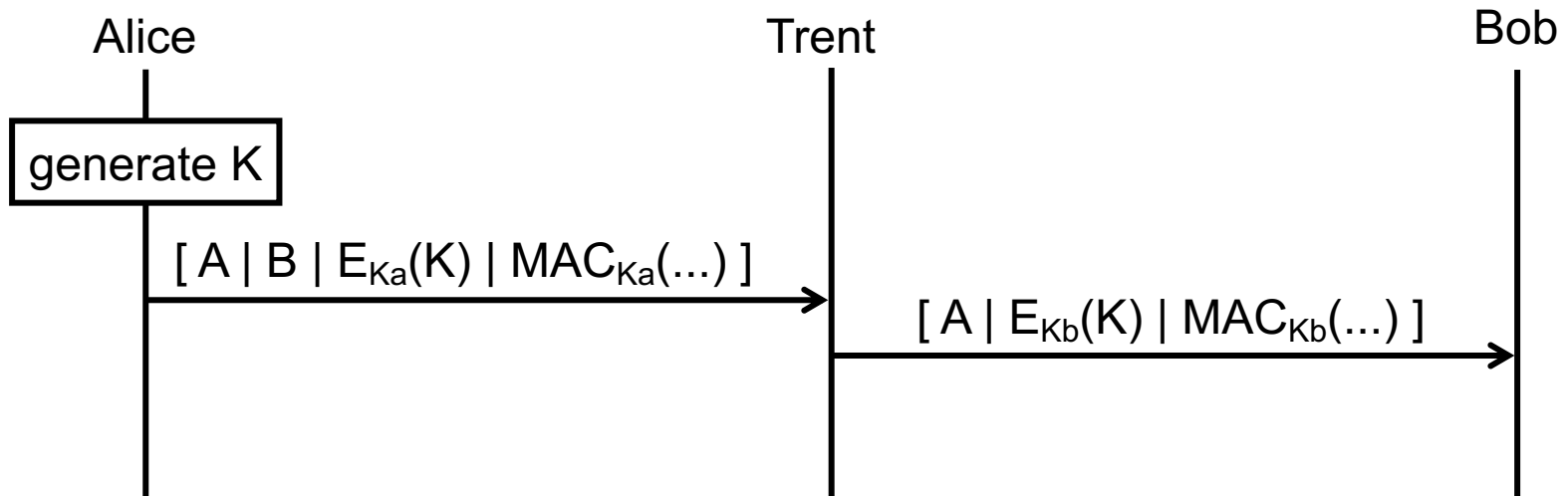
An attack against key authentication



Mallory makes Bob believe that he established K with Alice, while actually Alice is not even present...




Bob believes he shares K with Alice

Fixing the protocol

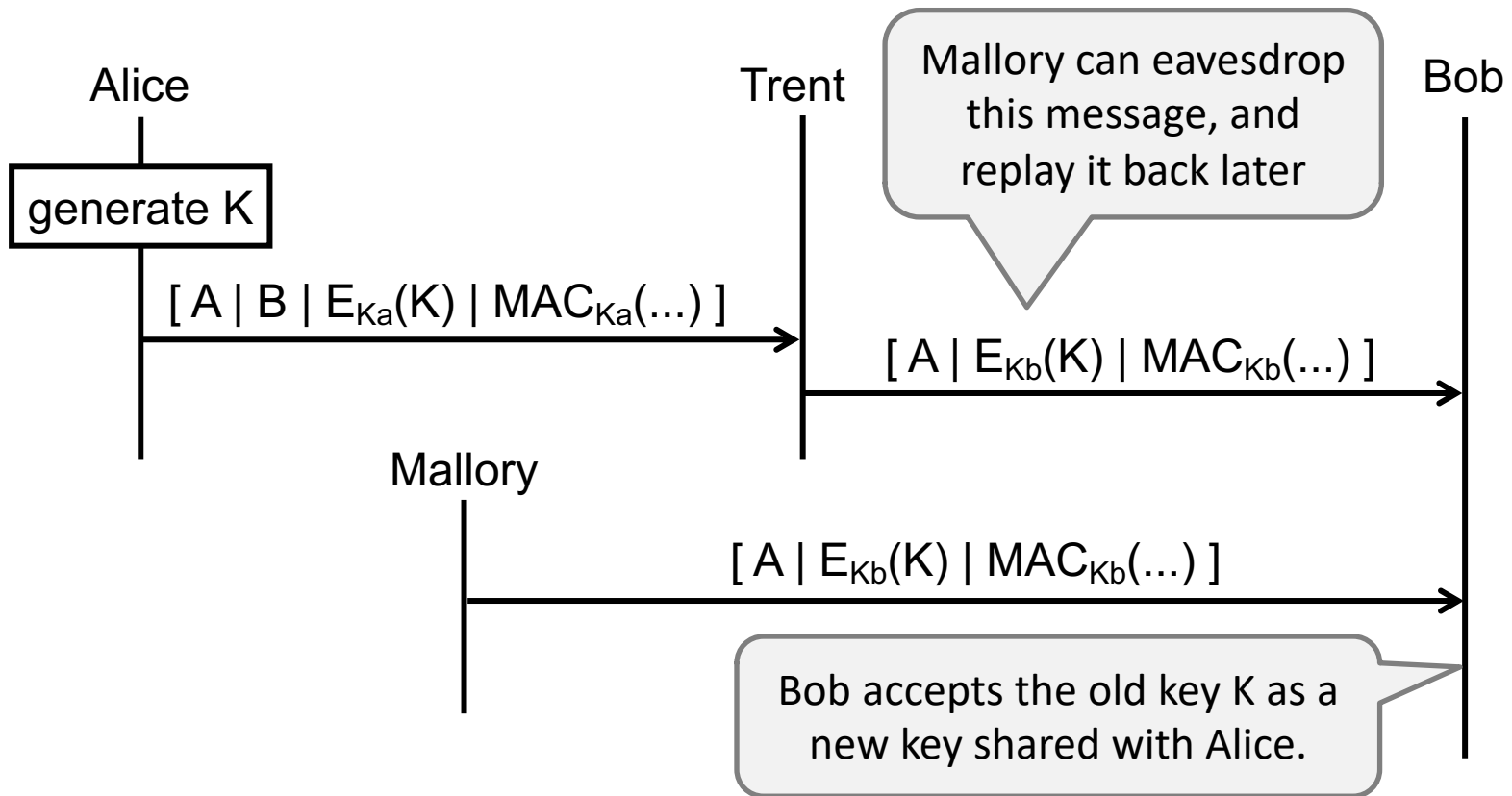


Is this protocol secure?

Reminder on the design objectives

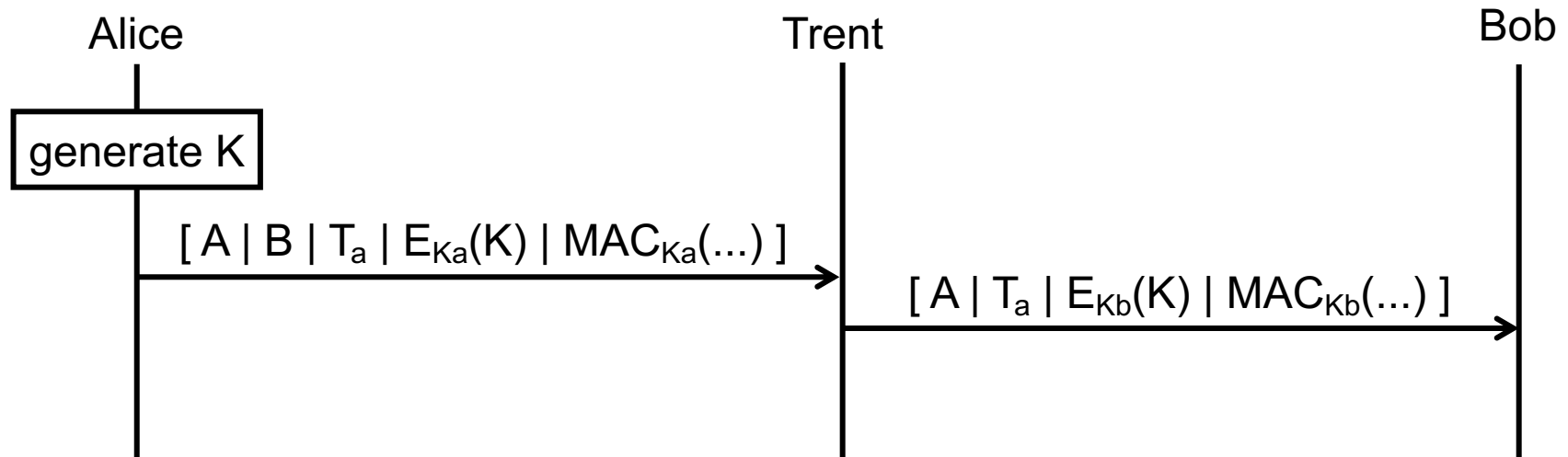
1. **Secrecy of the key:** When the protocol is executed by Alice and Bob, no other parties (with the possible exception of Trent) should learn the value of the established key. 
2. **Key authentication:** If Alice believes that she successfully executed the protocol and established a new key K with Bob, then Bob was indeed present and he should believe that he executed the protocol and established the same key K with Alice. 
3. **Key freshness:** Both parties should believe that the established key is fresh (new, not used before). 
(if freshness cannot be verified, then Mallory may force Alice and Bob to re-establish and re-use an old key, which may lead to problems...)

A replay attack



- if K is known to Mallory, then she can engage in a communication with Bob, who will believe he is talking to Alice
- even if K is not known to Mallory, she can still replay messages encrypted with K from the prior session where K was used

Providing key freshness with timestamps

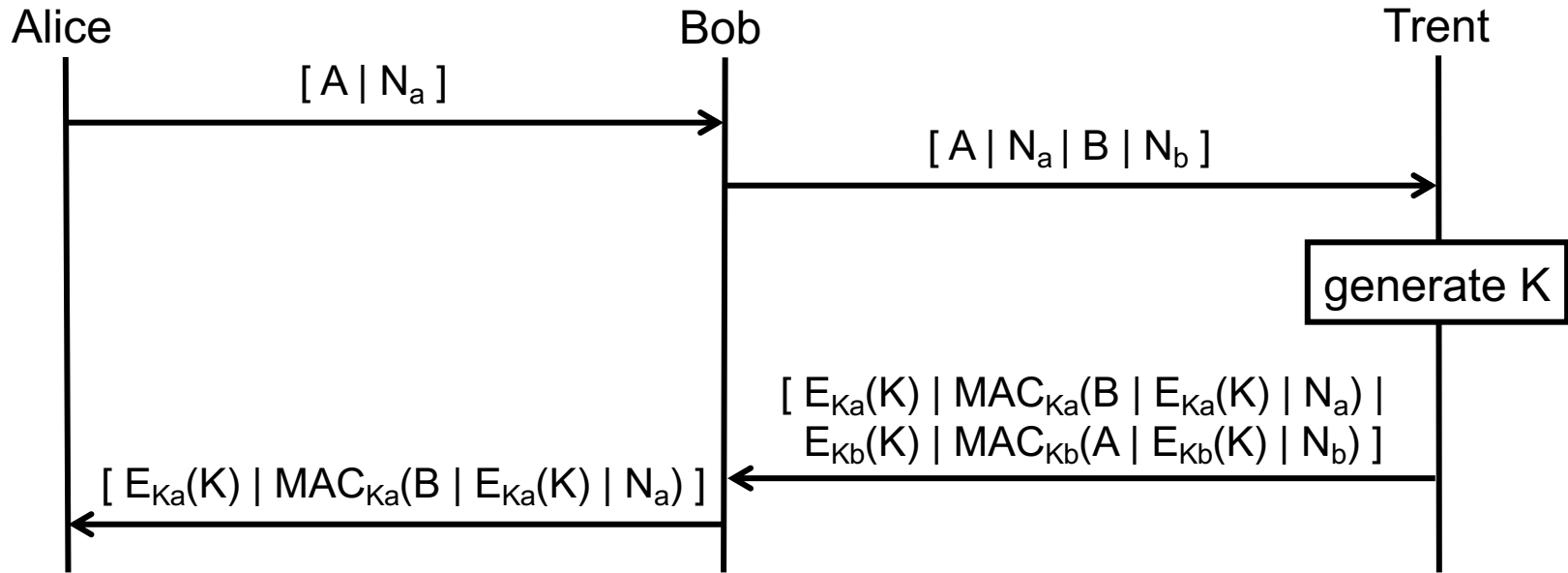


- T_a is the current time on the clock of Alice
- the key K is accepted by Bob only if T_a is within some (small) window of the current time on his clock
- replay is only possible within a small window of time \rightarrow Bob should be prepared for that
- clocks must be synchronized!
- secure clock synchronization is not trivial!

Notes on clock synchronization

- both slow and fast clocks can be a problem:
 - if a party's clock is slow, then (s)he may accept old (possibly replayed) messages
 - if a party's clock is advanced, then (s)he may generate messages that will be considered fresh *in the future* (although they may be dropped near the time of their generation)
- secure clock synchronization usually requires communication with a trusted time source
 - freshness of the time synch protocol messages cannot be ensured by timestamps (the synchronizing party does not know the time and therefore cannot produce and verify timestamps)
 - usually, random nonces are used...

Providing key freshness with nonces



- N_a and N_b are unpredictable random numbers (also called nonces) generated by Alice and Bob, respectively
- the key K is accepted by any party if he/she receives it bound to his/her nonce within some (small) window of time after sending the nonce out

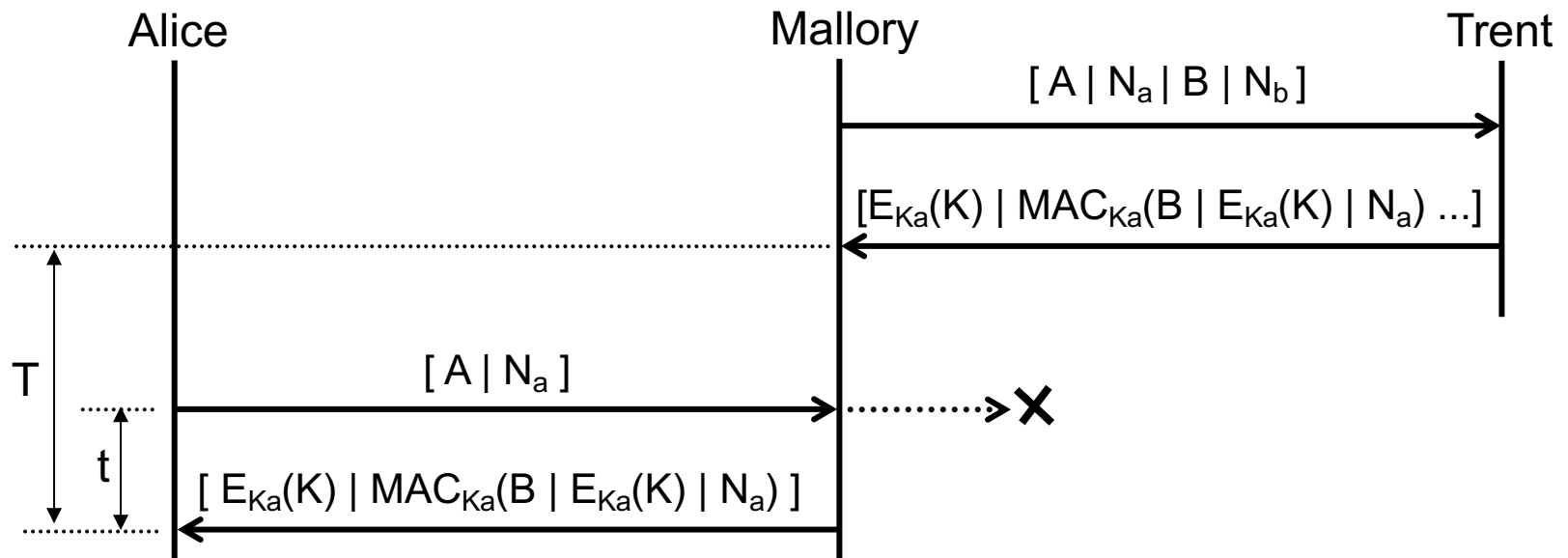
The logic of random nonces

- as N_x is unpredictable, X can be sure that $\text{MAC}_{KX}(Y \mid E_{KX}(K) \mid N_x)$ was generated after X sent out N_x
- if the time that elapsed between sending N_x and receiving $\text{MAC}_{KX}(Y \mid E_{KX}(K) \mid N_x)$ is acceptably short, then K can be considered fresh
- advantage:
 - no need for synchronized clocks, time is measured only locally
- disadvantages:
 - requires an extra message to send the nonce
 - requires some temporary state to store the nonce for verification purposes



Importance of unpredictability

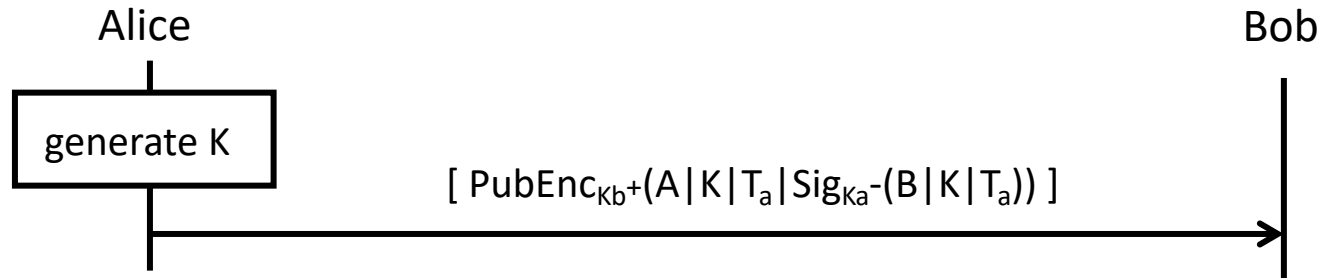
- if nonces were predictable, the adversary could obtain a message containing a future nonce of Alice, which would later be considered as fresh by Alice



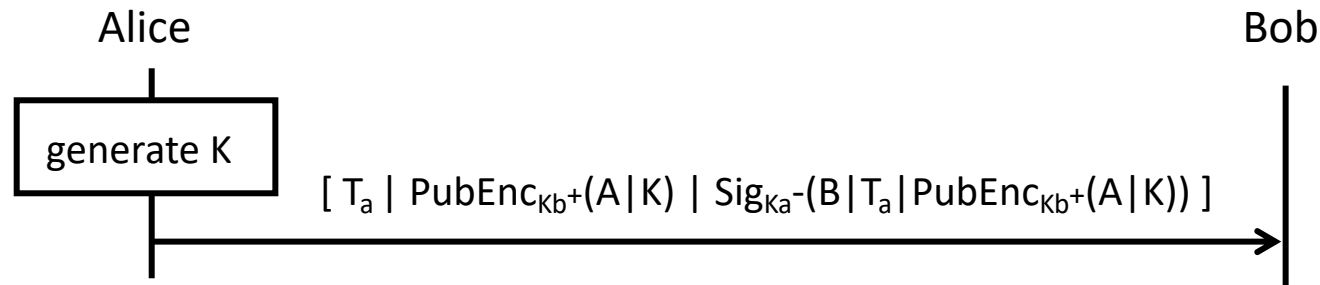
Alice believes that K is younger than t , while in fact, it is older than T

Key transport using public key crypto

- encrypting signed keys (ISO 11770-3/3)



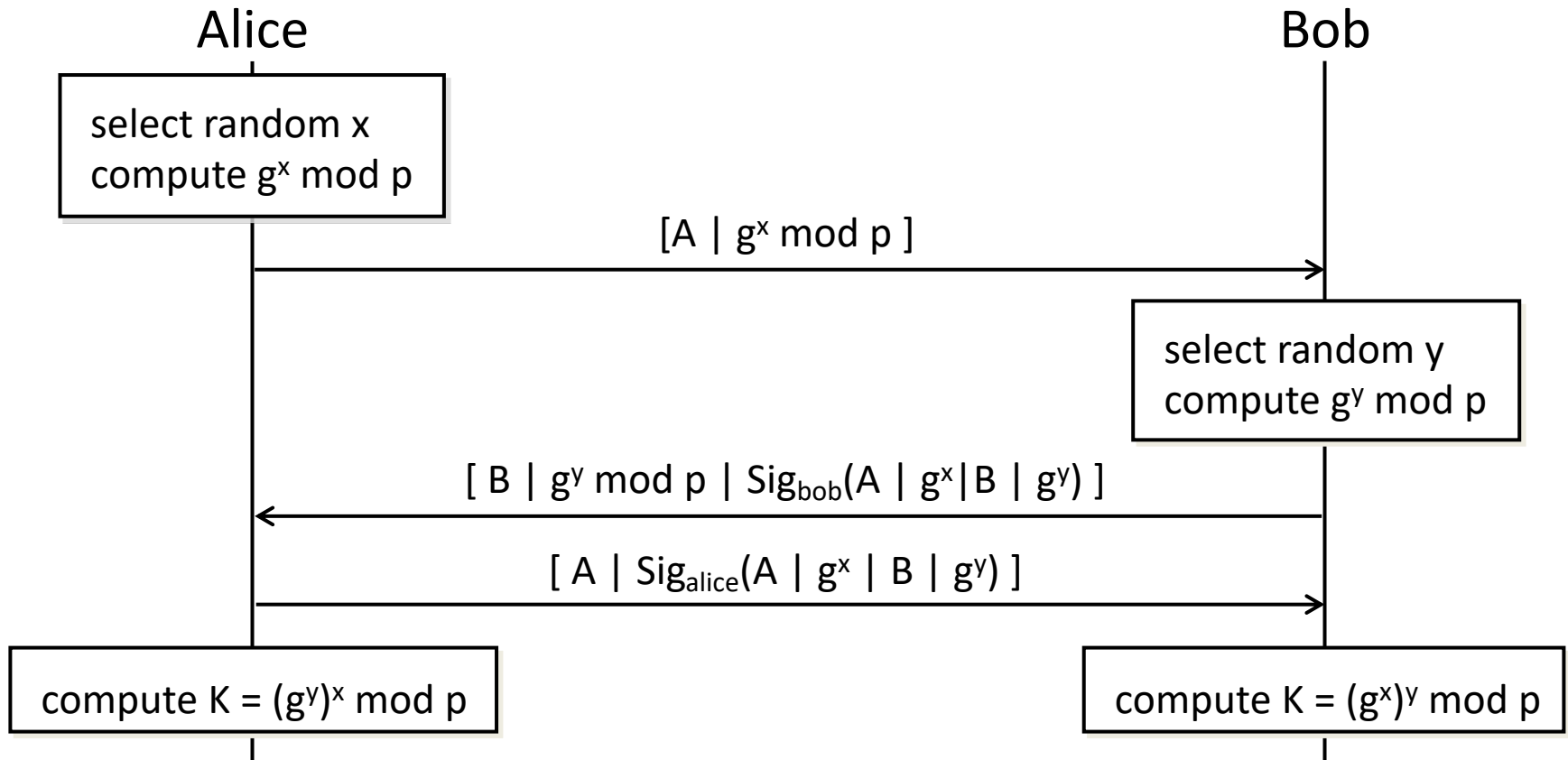
- signing encrypted keys (ISO 11770-3/2)



notes:

- $\text{PubEnc}()$ and $\text{Sig}()$ denote public key encryption and signature, resp.
- it is assumed that the parties know the public key of each other, no need for Trent (at least no need for him being on-line)

DH key agreement in practice (example)



- DH parameters must be authenticated (e.g., by digital signatures)
- key freshness is provided by the key agreement itself

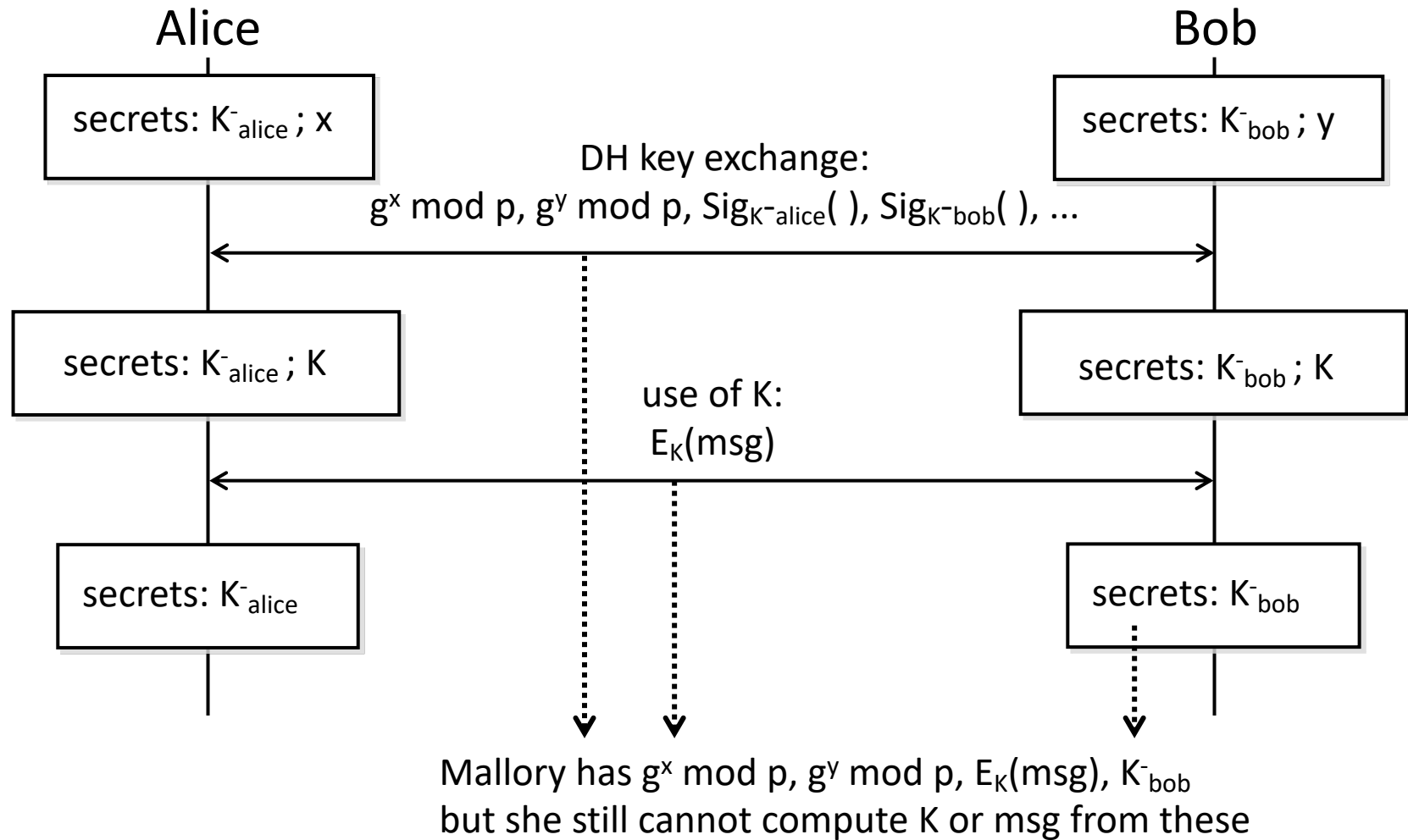
Key freshness with key agreement

- $K = f(k_A, k_B)$, where k_A and k_B are the contributions of Alice and Bob, respectively
 - if $f(x, \cdot)$ is a one-way function (for any x), then once Alice has chosen k_A , Bob cannot find any k_B , such that $f(k_A, k_B)$ has a pre-specified value (e.g., an old key)
 - similarly, if $f(\cdot, y)$ is a one-way function (for any y), then once Bob has chosen k_B , Alice cannot find any k_A , such that $f(k_A, k_B)$ has a pre-specified value
- if the contribution of a party is fresh, then (s)he can be sure that the resulting shared key is fresh too

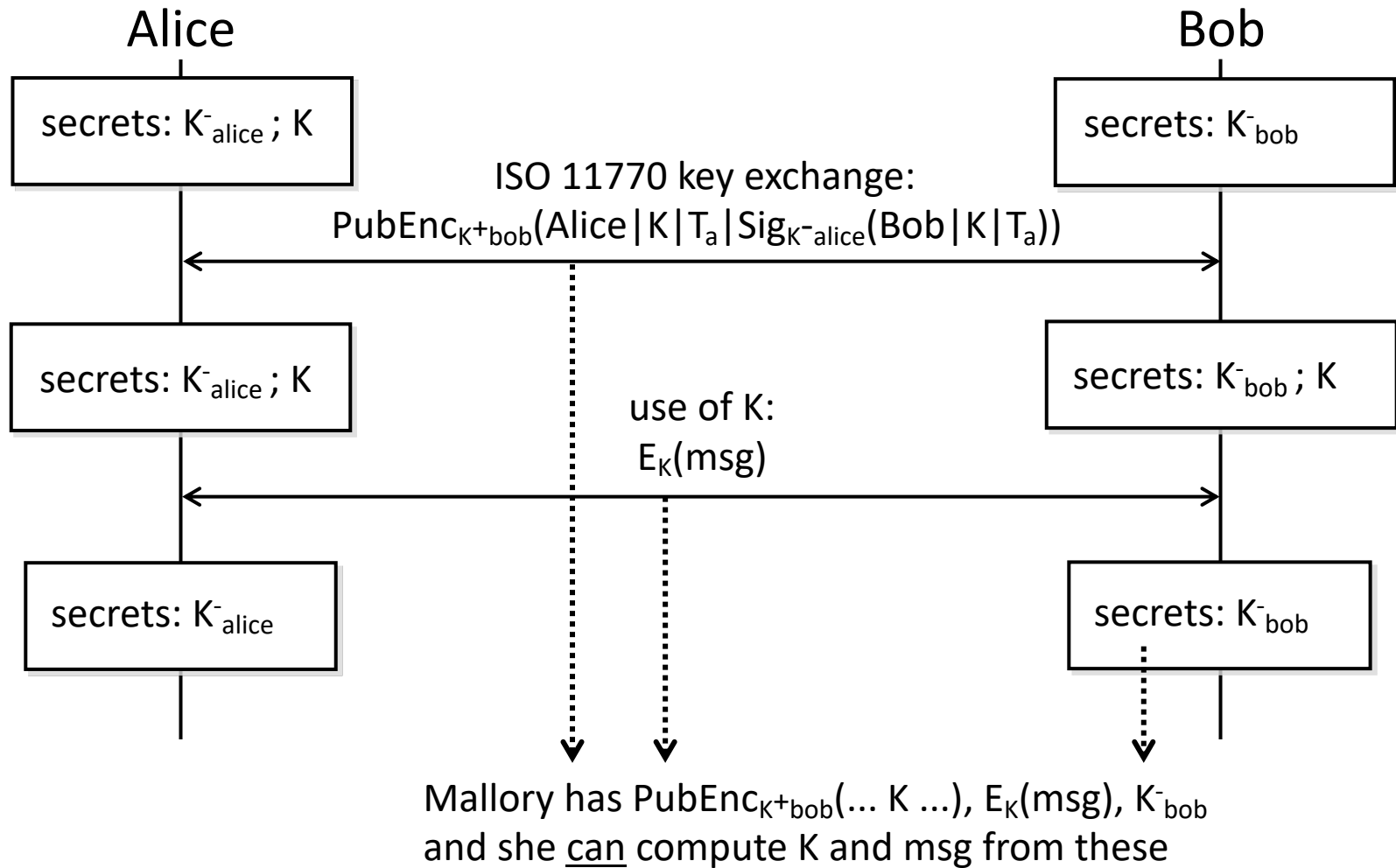
Perfect forward secrecy

- let's assume that Mallory eavesdropped and recorded a run of the authenticated DH key agreement protocol between Alice and Bob
- let's further assume that Alice and Bob delete their secret DH parameters x and y , respectively, as well as the shared key K when they are no longer needed (i.e., after the communication session between Alice and Bob)
- some time later, Mallory may compromise Alice or Bob and read their current secrets, including their signature generation key
- Mallory would still not be able to figure out K and decode any messages from the past that were protected by K

Perfect forward secrecy – illustrated



No perfect forward secrecy – illustrated



Key points

- cryptographic coding requires shared keys
- setting up shared keys via a secure out-of-band channel is possible, but setting up all keys (e.g., short-term session keys) in this way is not practical
- key exchange protocols can be used to setup short-term shared session keys by relying on some pre-established long-term keys (limited use of out-of-band channels)
- key exchange protocols define format of messages and rules of interaction between the protocol participants
- main design objectives:
 - secrecy of the key
 - key authentication (matching protocol runs)
 - key freshness

Key points

- attacker model
 - can be a legitimate protocol participant, but malicious
 - full control over the communication between the honest parties
 - cannot break cryptographic primitives
- there are two basic types of protocols:
 - key transport (key is generated by one of the parties)
 - key agreement (none of the parties can pre-determine the key)
- approaches for key transport
 - secrecy: encryption of the new key with already existing keys
 - authenticity: authenticating all parts that are needed for the correct interpretation of messages with already existing keys
 - freshness: timestamps or unpredictable nonces

Key points

- approaches for key agreement
 - secrecy: encryption with already existing keys or relying on one-way functions
 - authenticity: authenticating all parts that are needed for the correct interpretation of messages with already existing keys
 - freshness: fresh key contributions
- the notion of perfect forward secrecy
- some examples for attacks
 - impersonating a party
 - replay of old messages
 - (wo)man-in-the-middle and on-the-fly modification

A final note on key exchange protocols

- many protocols were proposed in the literature, but most of them have been found flawed later on (sometimes years after the publication of the protocol)
- flaws are often very subtle and hard to find
- considerable amount of research was done on applying formal methods to model and verify key establishment protocols
 - logics, theorem provers, process calculi, model checking, ...
 - formal methods are less error-prone, because they allow for systematic analysis
 - however, the modeling step still requires human creativity (and hence it is prone to errors)

Alice and Bob song*



Alice is sending her message to Bob
Protecting that transmission is Crypto's job
Without the help of our good friend Trent,
It's hard to get that secret message sent
Work tries to deposit the check of your salary
But with no crypto, it'll be changed by Mallory
You think no one will see what it is, you believe?
But you should never forget, there's always an Eve...

[Chorus]

'Cause I'm encrypting shit like every single day
Sending data across the network in a safe way
Protecting messages to make my pay
If you hack me, you're guilty under DMCA

...

*appears on the album *Algorhythms* (2005) by MC Plus+

Control questions

- What is the purpose of key exchange protocols?
- What are the design objectives of key exchange protocols?
- What are the two main types of key exchange protocols?
- What do we assume about the attacker?
- What kind of attacks do you know against key exchange protocols?
- How do key transport protocols ensure the secrecy of the established key?
- How is key authenticity achieved?
- What methods do you know for providing key freshness?
Explain how they work, and what their advantages and disadvantages are!
- What is perfect forward secrecy?