Cryptographic Protocols & Key Management

Security Engineering, Week 5

Dr Arman Khouzani, Dr Na Yao

Learning Outcomes

- Explain the concept of a cryptographic protocol.
- Analyse a simple cryptographic protocol.
- Appreciate the difficulty of designing a secure cryptographic protocol.
- Appreciate the significance of the Diffie–Hellman protocol.
- Identify some fundamental principles of key management.
- Identify the typical goals of AKE protocols.
- Explain the purpose of a public-key certificate.
- Describe the main contents of a public-key certificate.
- Be aware of alternative approaches to certificate-based public-key management.

Cryptographic Protocols

Operational motivation for protocols

Applications:

- Have complex security requirements
- Involve different data items with different security requirements
- Involve information flowing between more than one entity
- Consist of a sequence of logical (conditional) events

Components of a cryptographic protocol

- A *cryptographic protocol* is a specification of all the events which need to take place in order to achieve some required security goals. It should specify:
  - **The protocol assumptions**
  - **The protocol flow**
  - **The protocol messages**
  - **The protocol actions**

Stages of protocol design

- Defining the objectives.

  Merchant Bob wants to make sure a contract he will receive from Alice cannot later be denied.

- Determining the protocol goals.

  At the end of the protocol, Bob requires non-repudiation of the contract received from Alice.

- Specifying the protocol.

- Protocol design is a complex process with challenges, it’s best left to experts!

Standards for Cryptographic Protocols

- The PKCS standards include some cryptographic protocols for implementing public-key cryptography.
- ISO/IEC 11770 specifies a suite of cryptographic protocols for mutual entity authentication and key establishment.
- SSL/TLS specifies a protocol for setting up a secure communication channel.

Analysing a simple protocol

The Objectives

- Alice and Bob have access to a common network. Periodically, at any time of his choosing, Bob wants to check Alice is still ‘alive’ and connected to the network.
- A network consists of many entities, all of whom regularly check the liveness of one another. We thus set a secondary security objective that whenever Bob receives any confirmation of liveness from Alice, he should be able to determine precisely which liveness query she is responding to.

Analysing a simple protocol

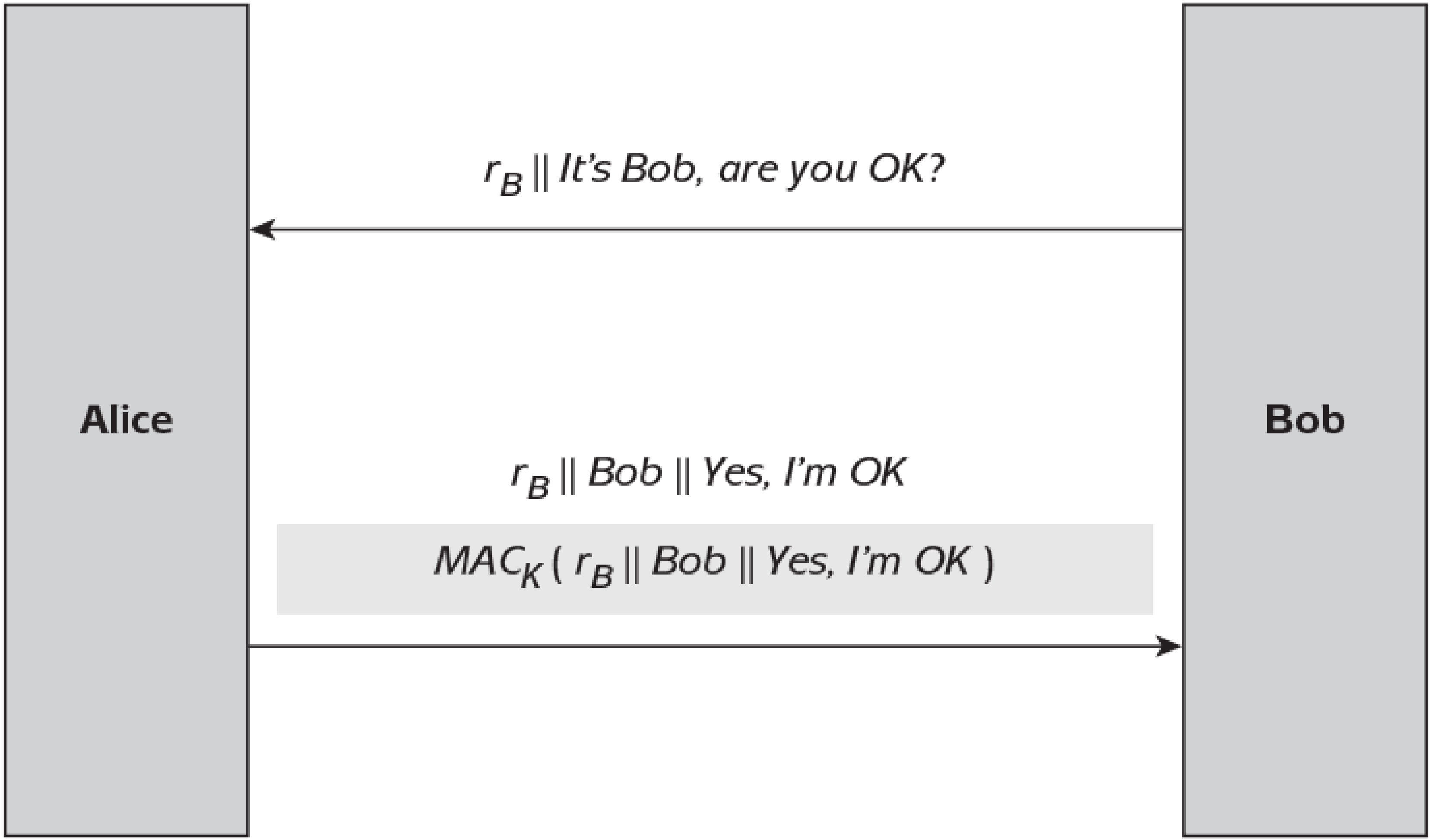
The Protocol Goals

- **Data origin authentication of Alice's reply**.
- **Freshness of Alice's reply**. If this is not provided, then even if there is data origin authentication of the reply, this could be a replay of a previous reply.
- **Assurance that Alice's reply corresponds to Bob's request**. If this is not provided, then it is possible Bob receives a reply which corresponds to a different request (either one of his own, or of another entity in the network).

- Different candidate protocols for analysis.
- Notation used:

r_B	A nonce generated by Bob
$\|\|$	Concatenation
Bob	An *identifier* for Bob (perhaps his name)
$MAC_K(data)$	A MAC computed on *data* using key *K*
$E_K(data)$	Symmetric encryption of *data* using key *K*
$Sig_A(data)$	A digital signature on *data* computed by Alice
T_A	A timestamp generated by Alice
T_B	A timestamp generated by Bob
ID_S	A session identifier

Protocol 1



r_B || It's Bob, are you OK?
Alice
Bob
r_B || Bob || Yes, I'm OK
MAC_K (r_B || Bob || Yes, I'm OK)

Protocol 1 analysis

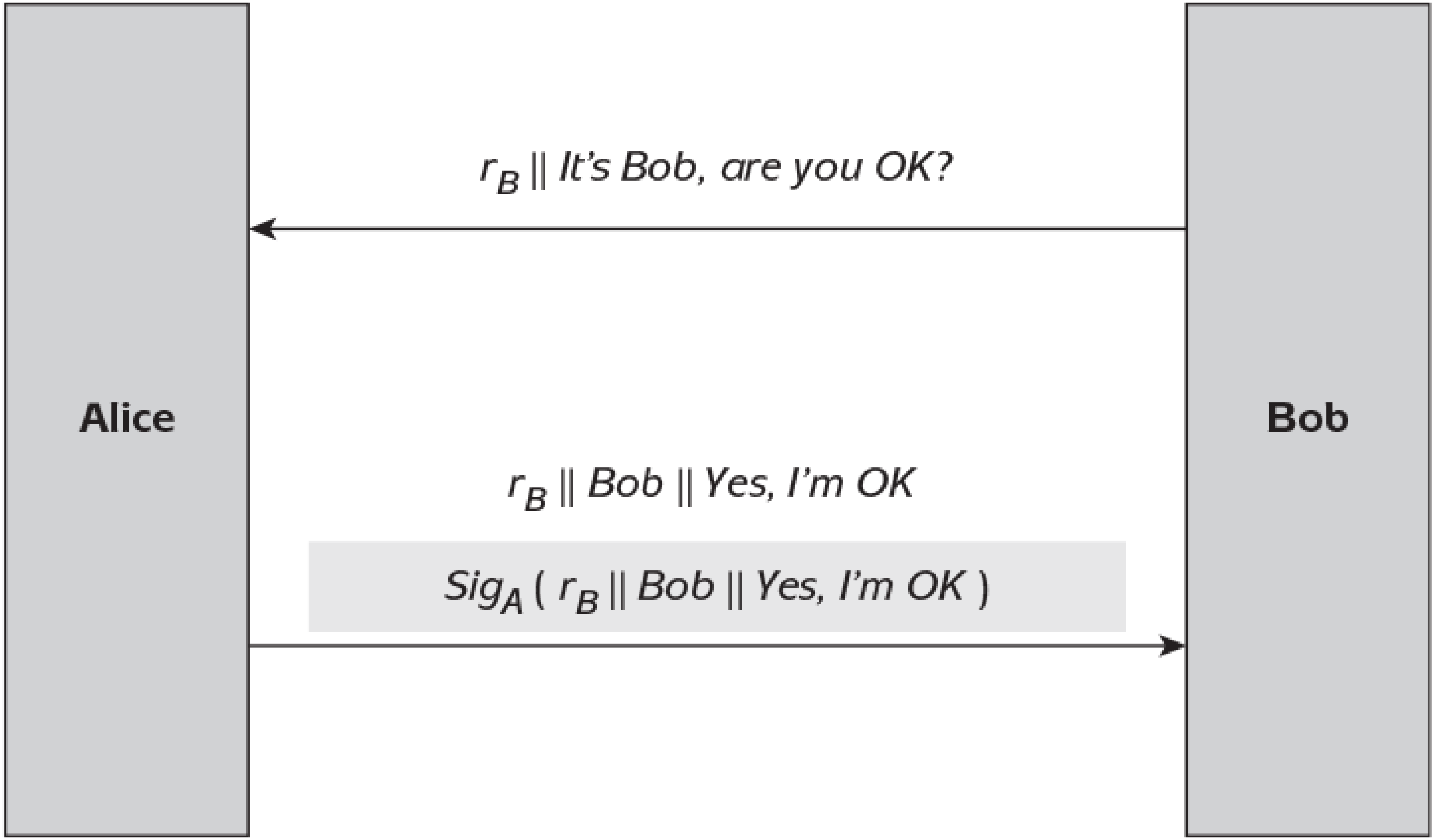
- **Data origin authentication of Alice's reply**: MAC
- **Freshness of Alice's reply**: nonce
- **Assurance Alice's reply corresponds to Bob's request**:
  1. nonce r_B, which Bob generated for this run of the protocol.
  2. The reply contains the identifier Bob.

Protocol Assumptions

1. Bob has access to a source of randomness.
2. Alice and Bob already share a symmetric key K known only to them.
3. Alice and Bob agree on the use of a strong MAC algorithm.

Protocol 1 **meets the security goals** and hence is a suitable protocol to use in our simple application.

Protocol 2



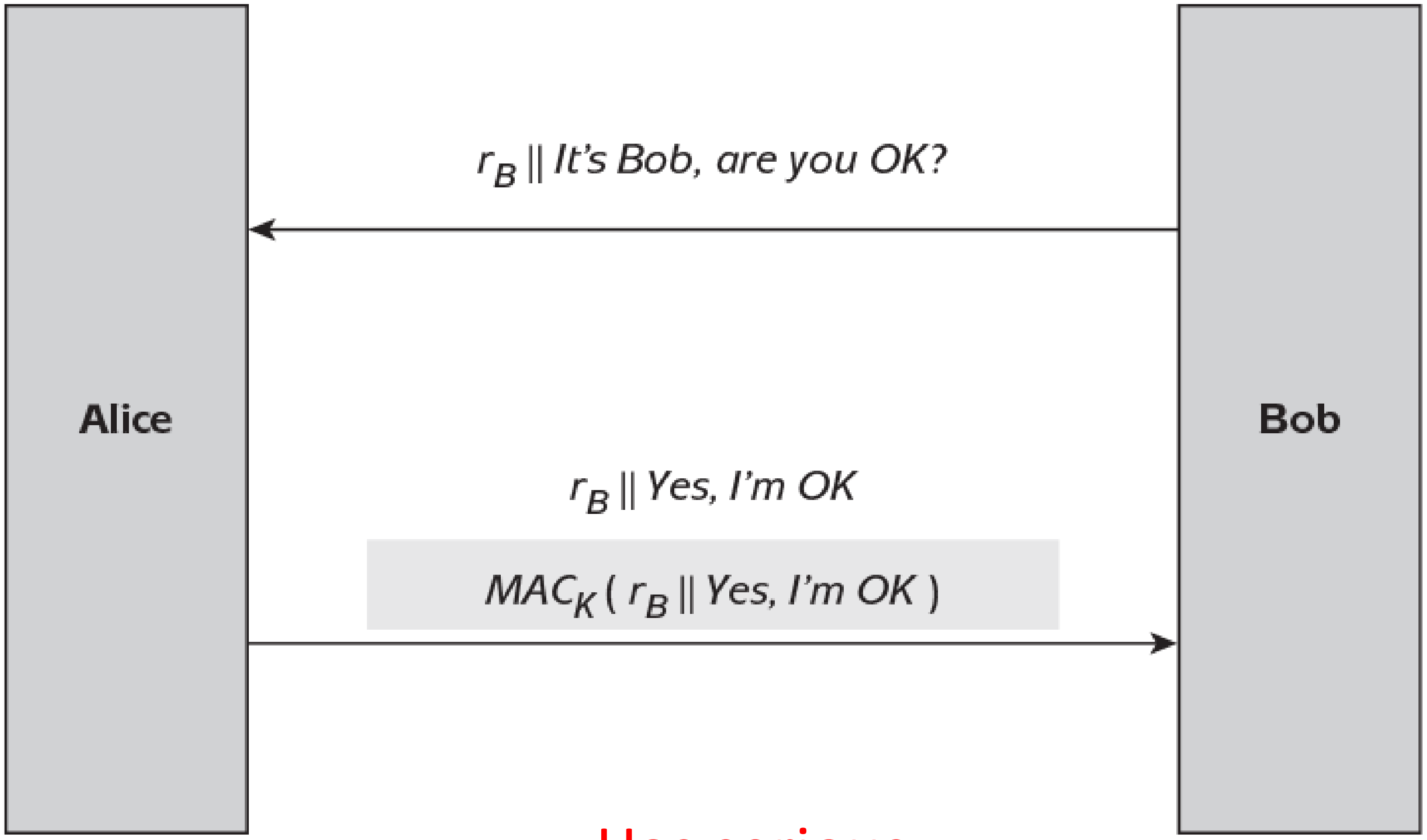
r_B || It's Bob, are you OK?
Alice
Bob
r_B || Bob || Yes, I'm OK
Sig_A (r_B || Bob || Yes, I'm OK)

Protocol 2 analysis

- The analysis of Protocol 2 is exactly as for Protocol 1, except for:
  - **Data origin authentication of Alice's reply:** Digital signature
- **Protocol Assumptions**
  1. Bob has access to a source of randomness. As for Protocol 1.
  2. Alice has been issued with a signature key, and Bob has access to a verification key corresponding to Alice's signature key. This is the digital signature scheme equivalent of the second assumption for Protocol 1.
  3. Alice and Bob agree on the use of a strong digital signature scheme.

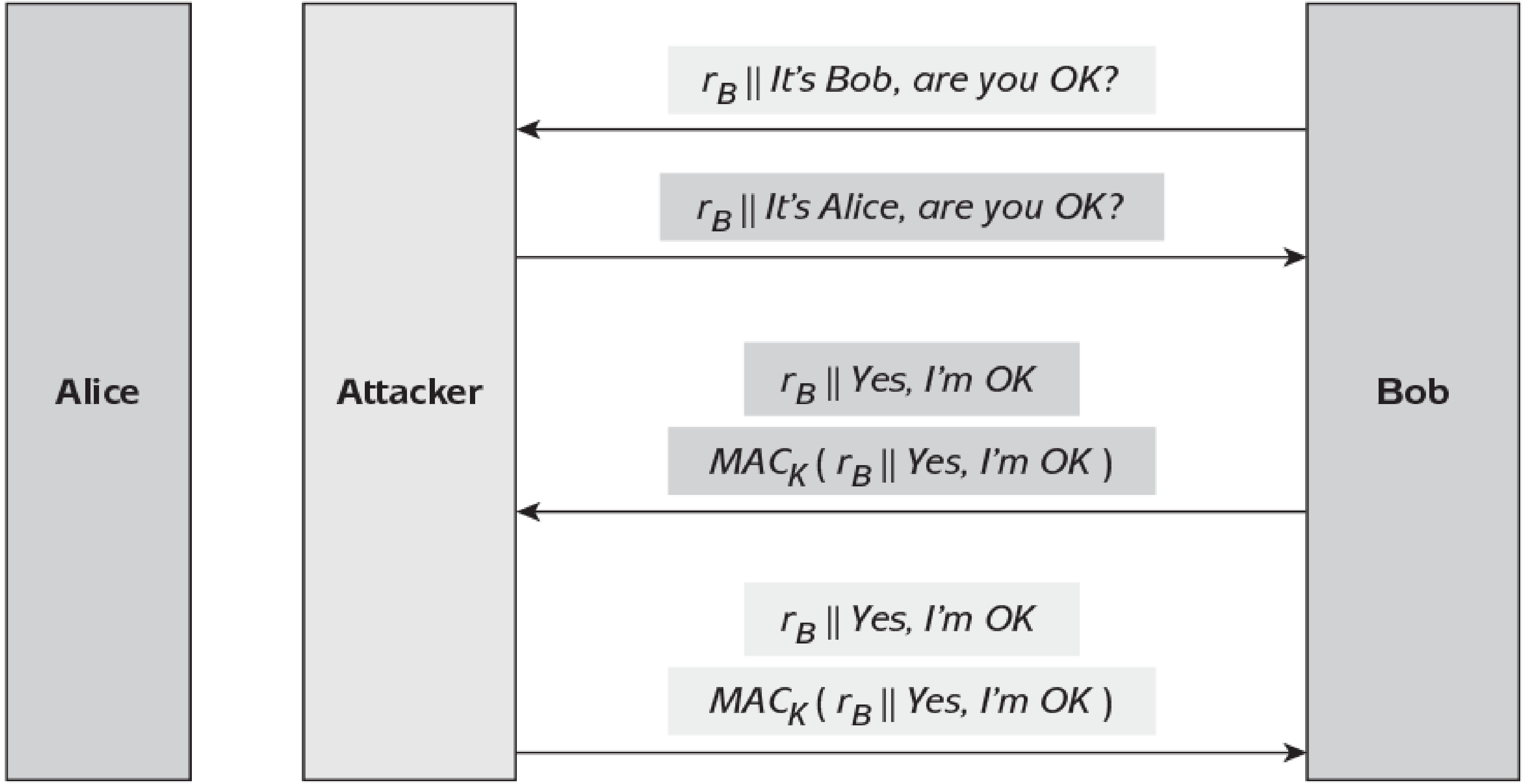
- Protocol 2 also meets the three security goals.

Protocol 3



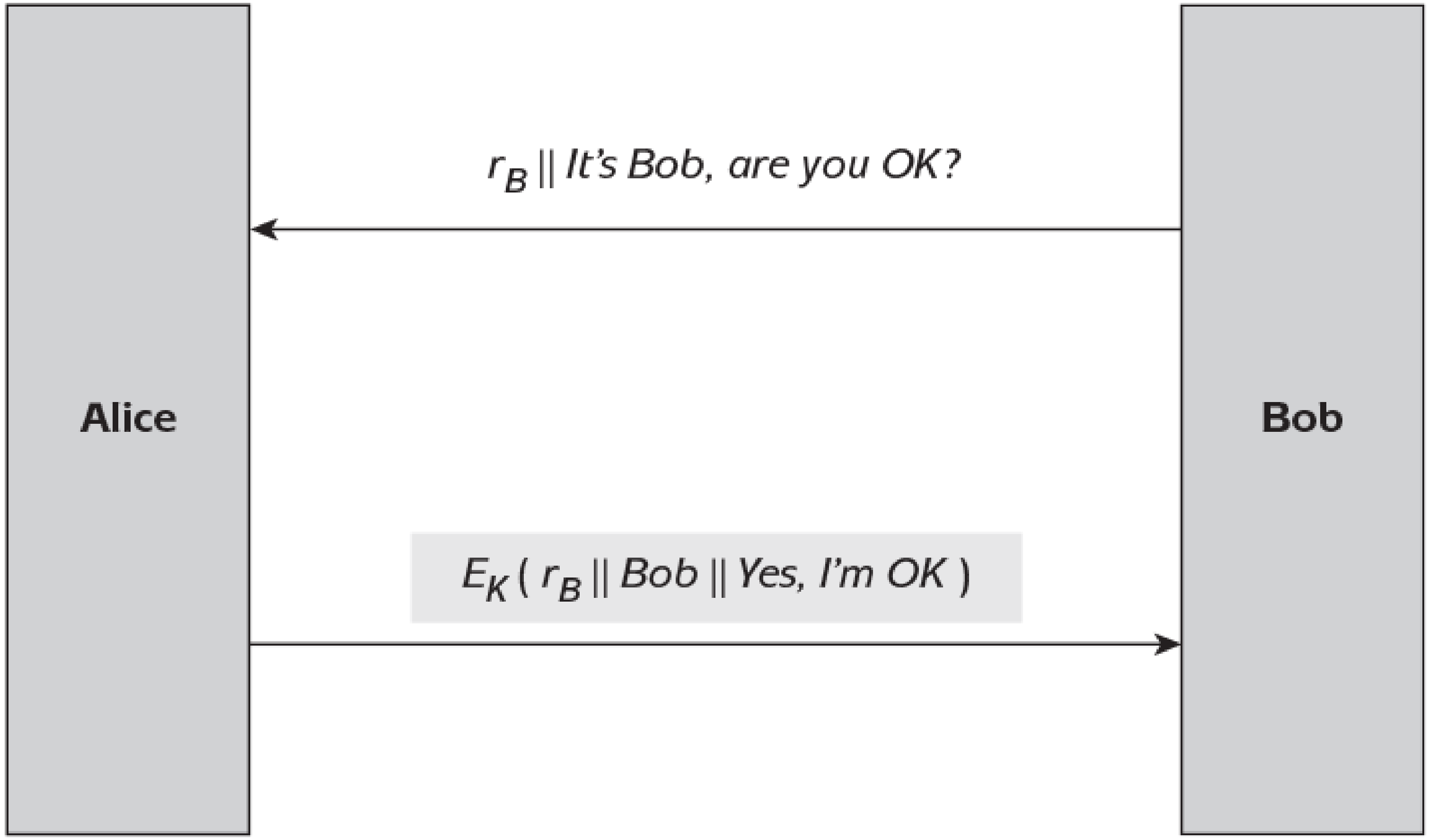
Has serious problems! →

Reflection attack



It is generally good practice in the design of cryptographic protocols to **include the identifiers** of recipients in protocol messages to prevent reflection attacks of this type.

Protocol 4



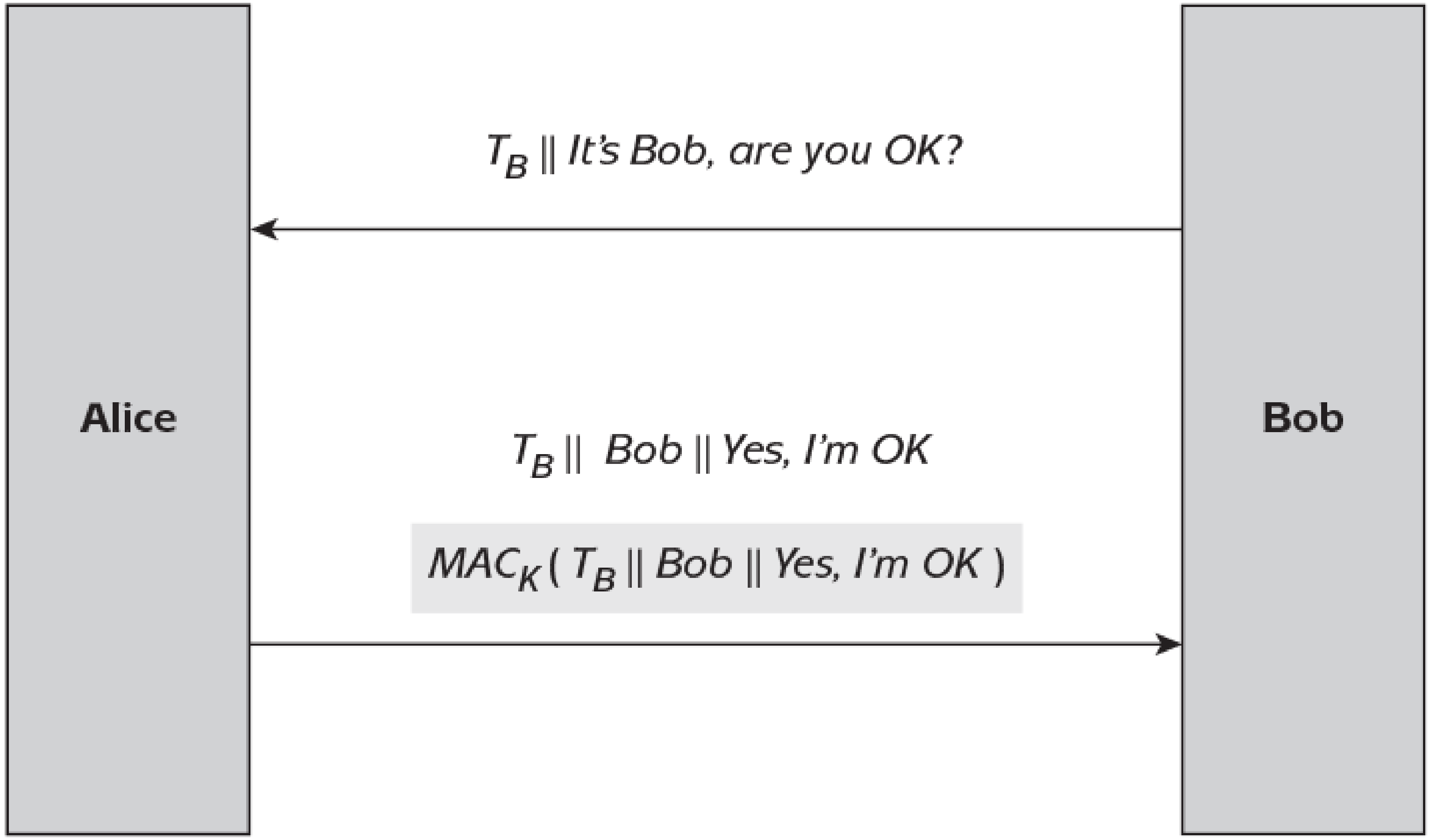
Protocol 4 analysis

Protocol Assumptions: identical to Protocol 1, except that we assume Alice and Bob have agreed on the use of a strong symmetric encryption algorithm *E* (rather than a MAC).

Issues:

- Encryption does not, in general, provide data origin authentication.
  - key separation
  - Types of encryption mechanism. Stream cipher?
- Encryption tends only to be used in this way if confidentiality of the message data is also required.

Protocol 5


Alice
Bob
T_B || It's Bob, are you OK?
T_B || Bob || Yes, I'm OK
MAC_K (T_B || Bob || Yes, I'm OK)

Protocol 5 Analysis

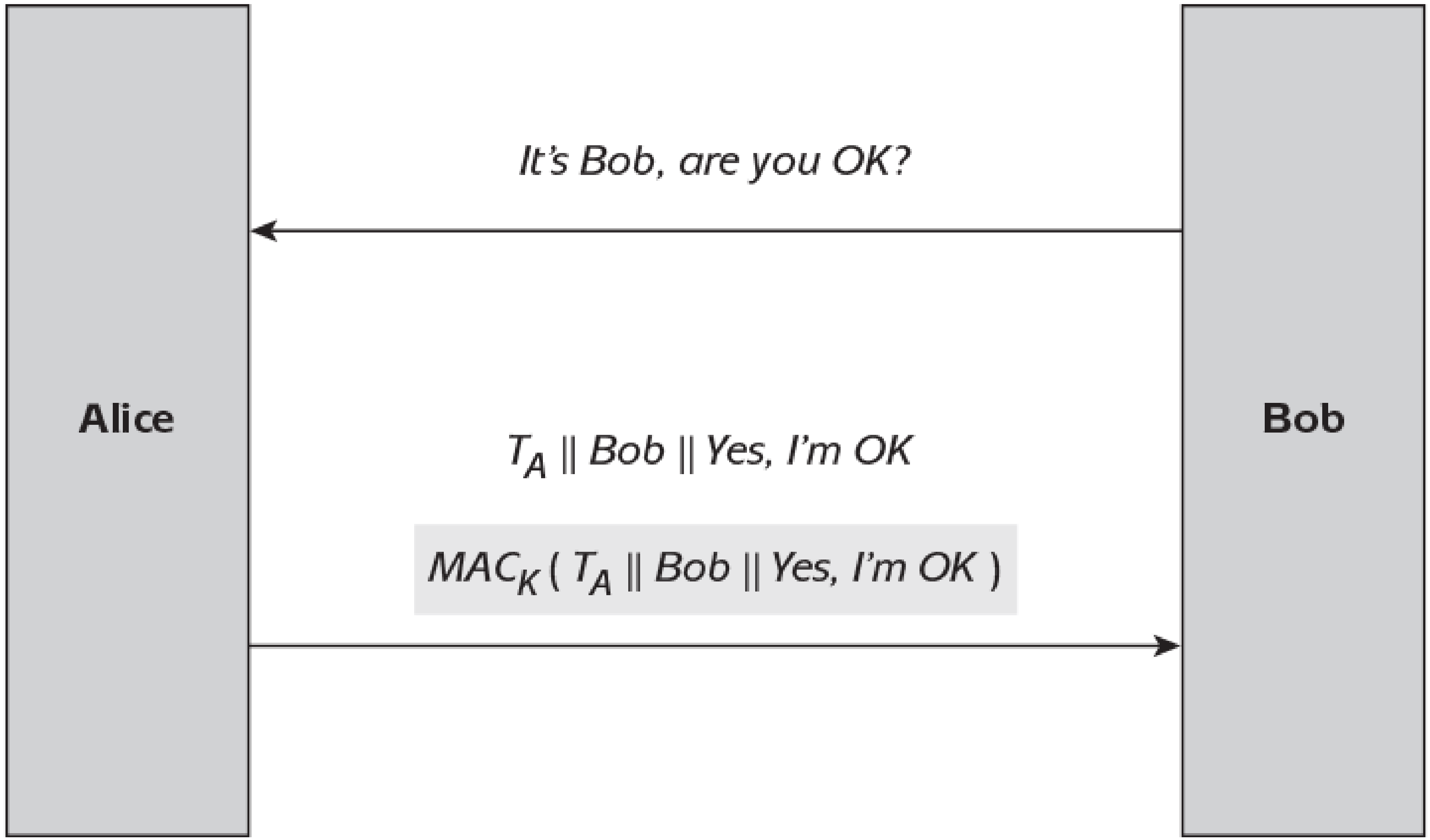
Protocol Assumptions: same as protocol 1, except for the source of randomness being replace by Bob can generate and verify integrity-protected timestamps.

Protocol Analysis

- Data origin authentication of Alice's reply. As for Protocol 1.
- Freshness of Alice's reply. The reply text includes the timestamp T_B which Bob generated at the start of the protocol.
- Assurance that Alice's reply corresponds to Bob's request: timestamp, identifier "Bob".

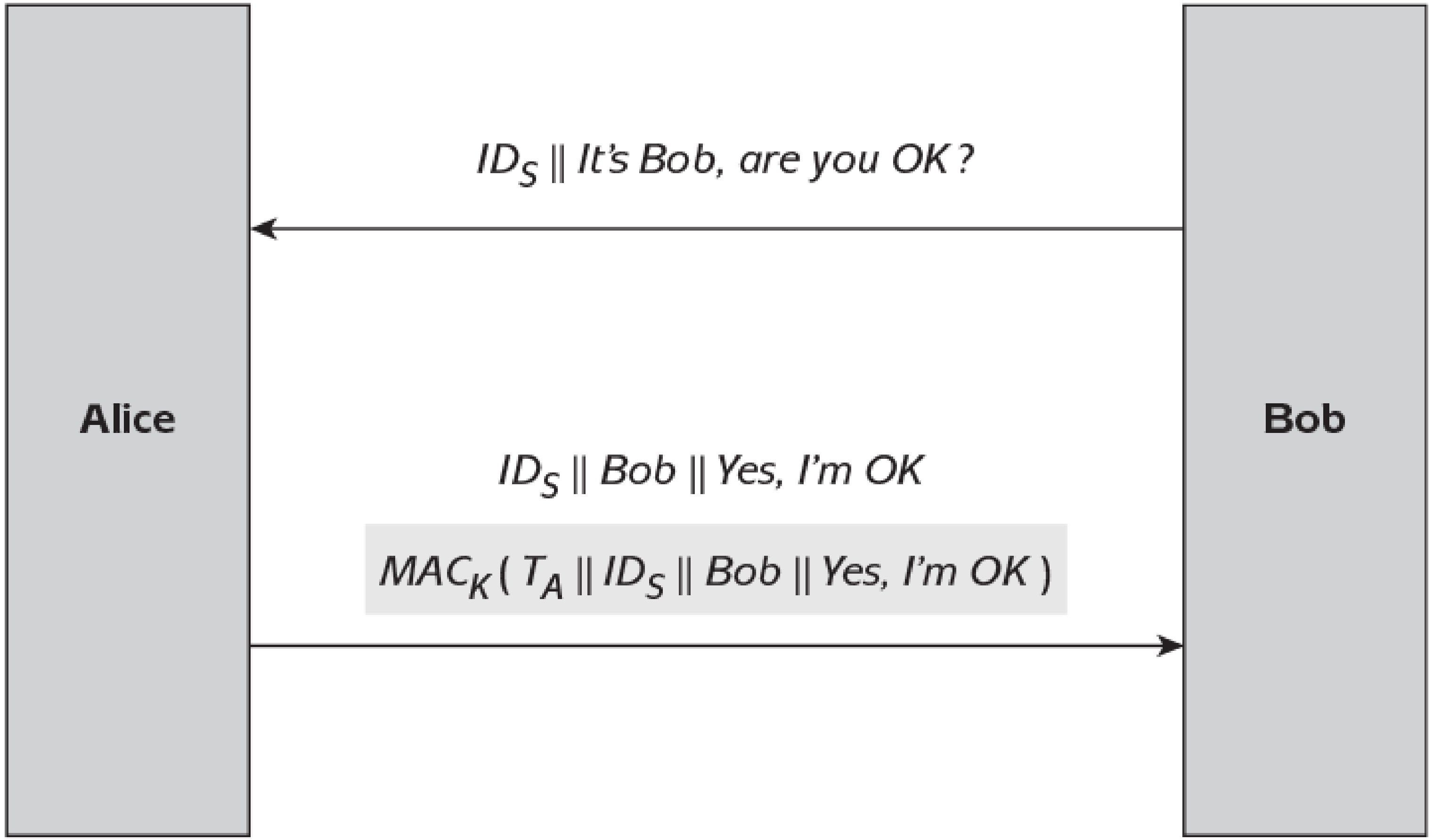
- Protocol 5 meets the three security goals.

Protocol 6



Alice
Bob
It's Bob, are you OK?
T_A || Bob || Yes, I'm OK
MAC_K (T_A || Bob || Yes, I'm OK)

Protocol 7



Alice
Bob
ID_S || It's Bob, are you OK ?
ID_S || Bob || Yes, I'm OK
$MAC_K(T_A \| ID_S \| Bob \| Yes, I'm OK)$

Simple protocol summary

- **There is no one correct way to design a cryptographic protocol**.
  - Three protocols provide all three security goals.
  - The choice of the most suitable protocol design thus depends on what assumptions are most suitable for a given application environment.
- **Designing cryptographic protocols is hard**.
  - The deficiencies of several of these protocol variants are very subtle. Given that this application is artificially simple, the complexity of designing protocols for more intricate applications should be clear.

Authentication and Key Establishment (AKE) protocol

Objectives:

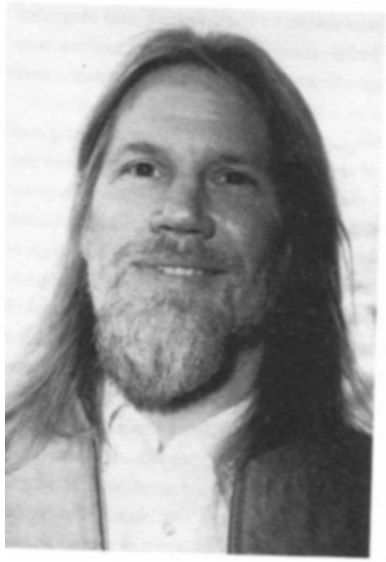
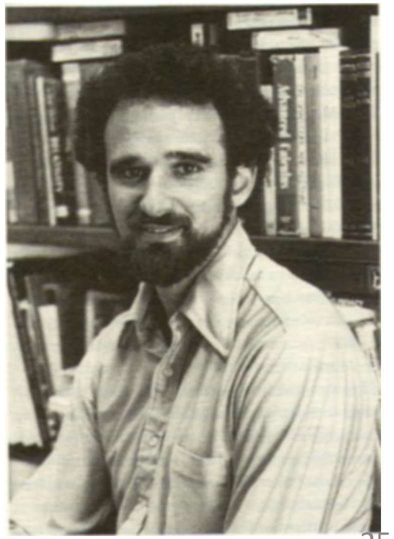
- Mutual entity authentication
- Establishment of a common symmetric key

AKE protocol goals:

- Mutual entity authentication.
- Mutual data origin authentication.
- Mutual key establishment.
- Key confidentiality.
- Key freshness.
- Mutual key confirmation.
- Unbiased key control.

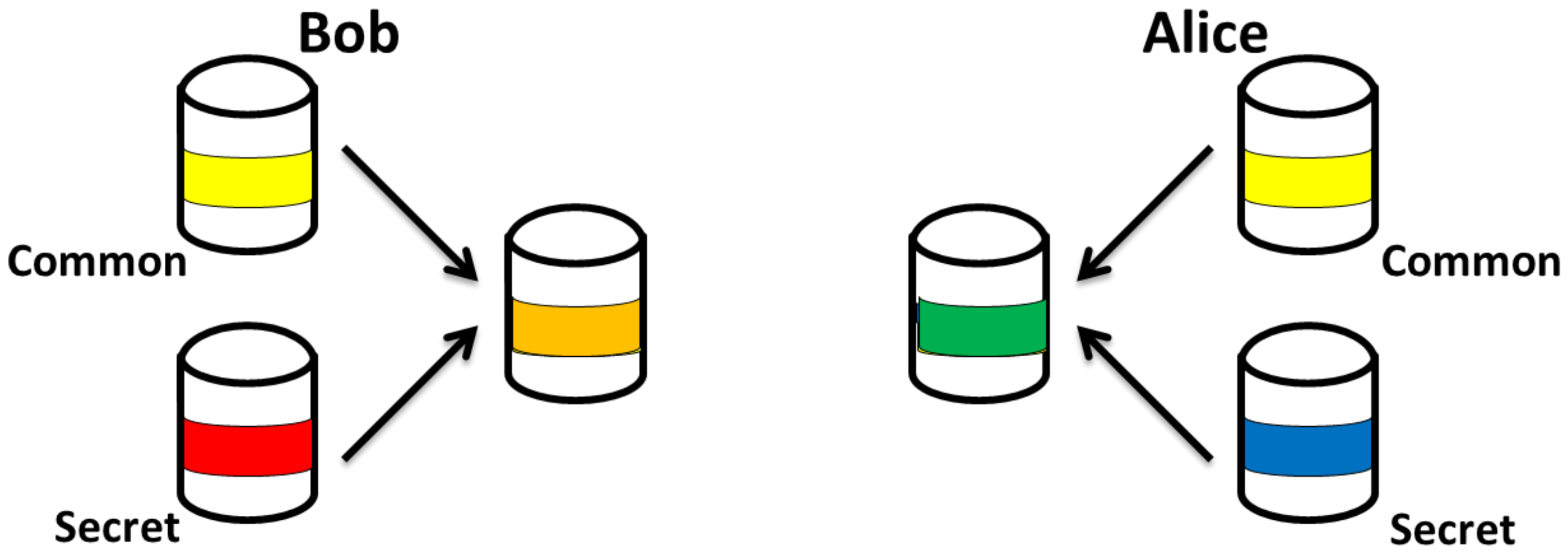
Diffie–Hellman key agreement protocol

- One of the most influential cryptographic protocols
- The basis for majority of modern AKE protocols based on key agreement
- Designed for environments where secure channels do not yet exist
- Based on the difficulty of discrete logarithm.

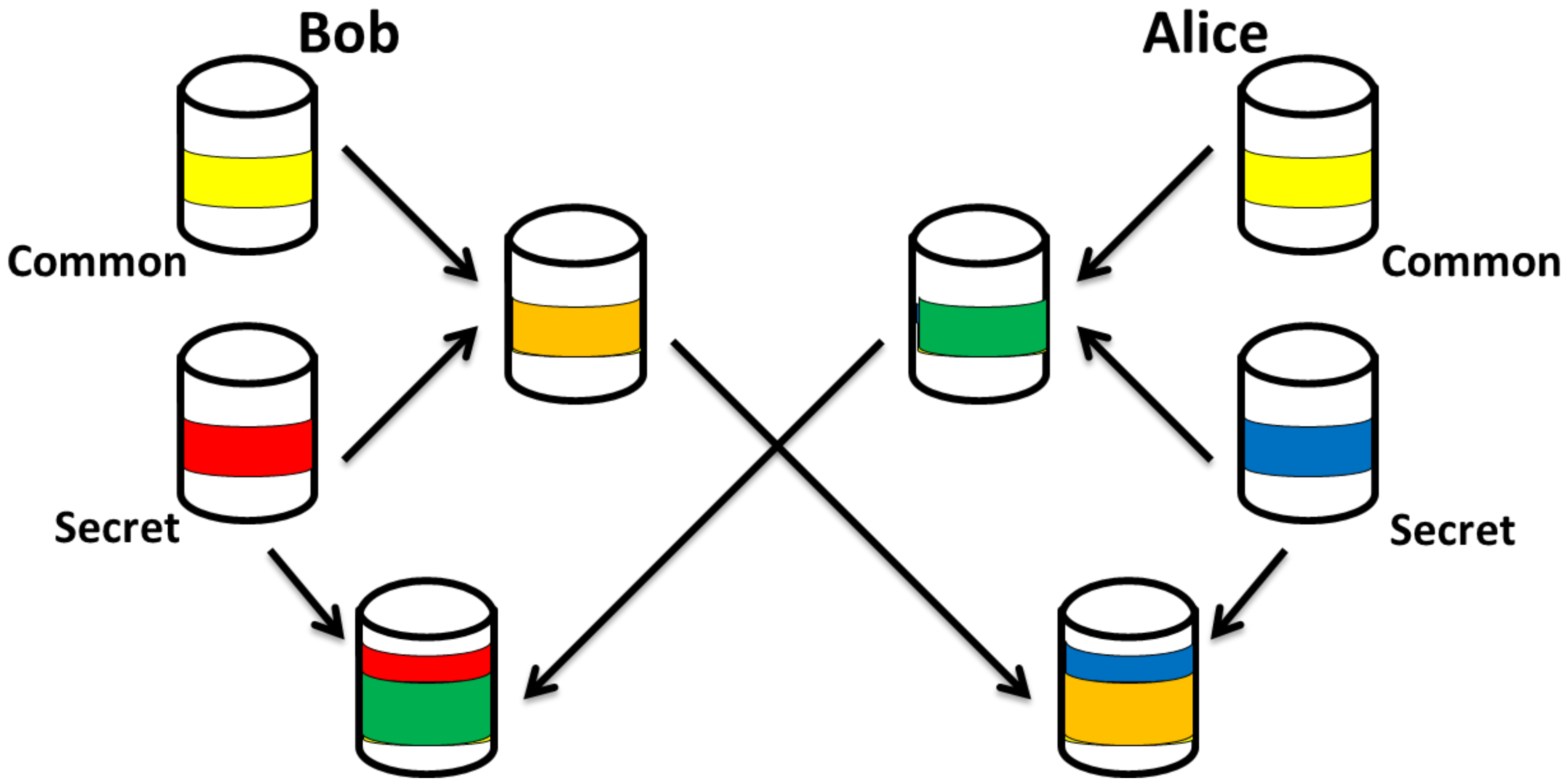



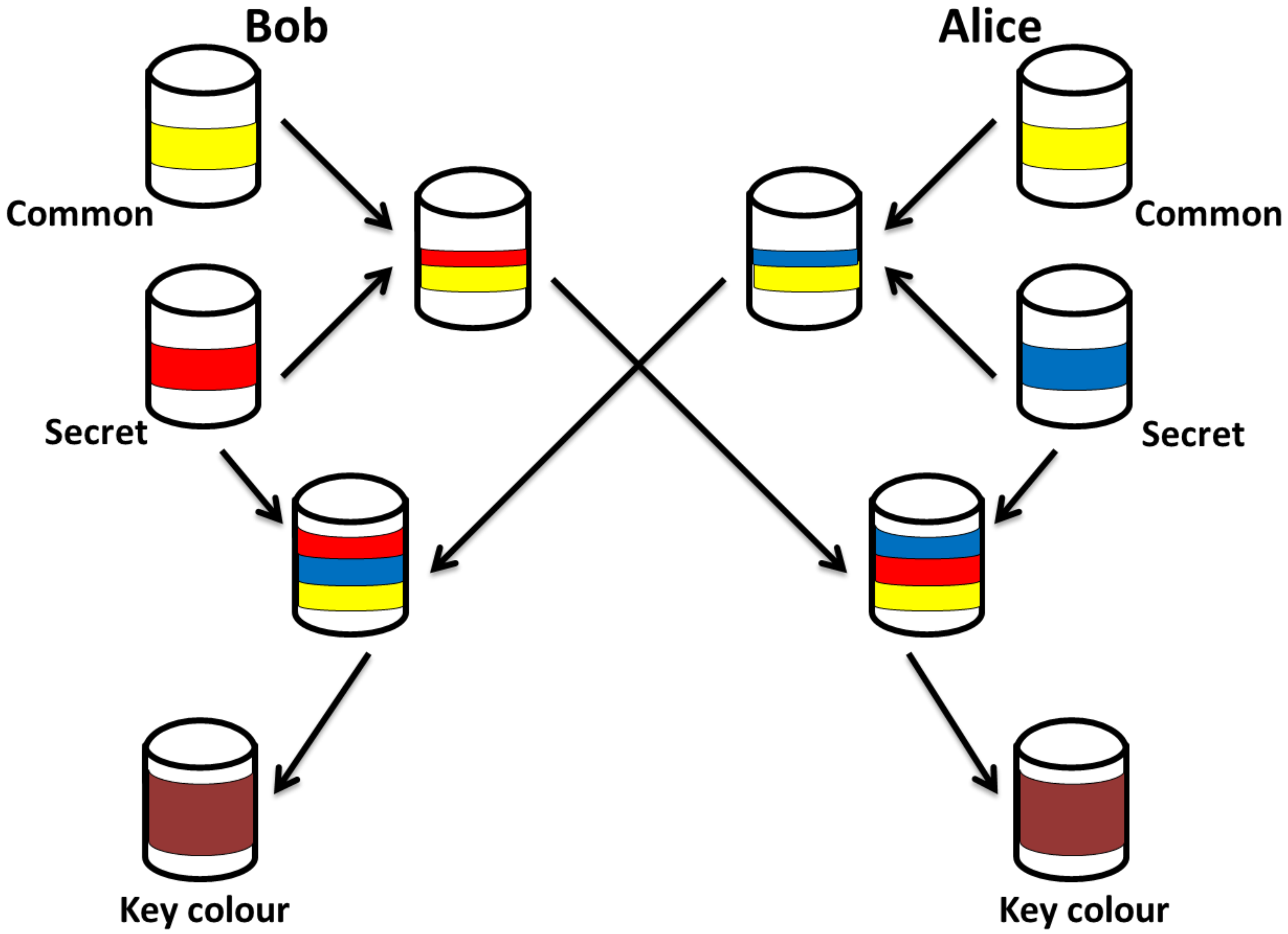
Diffie **and** ***Hellman***

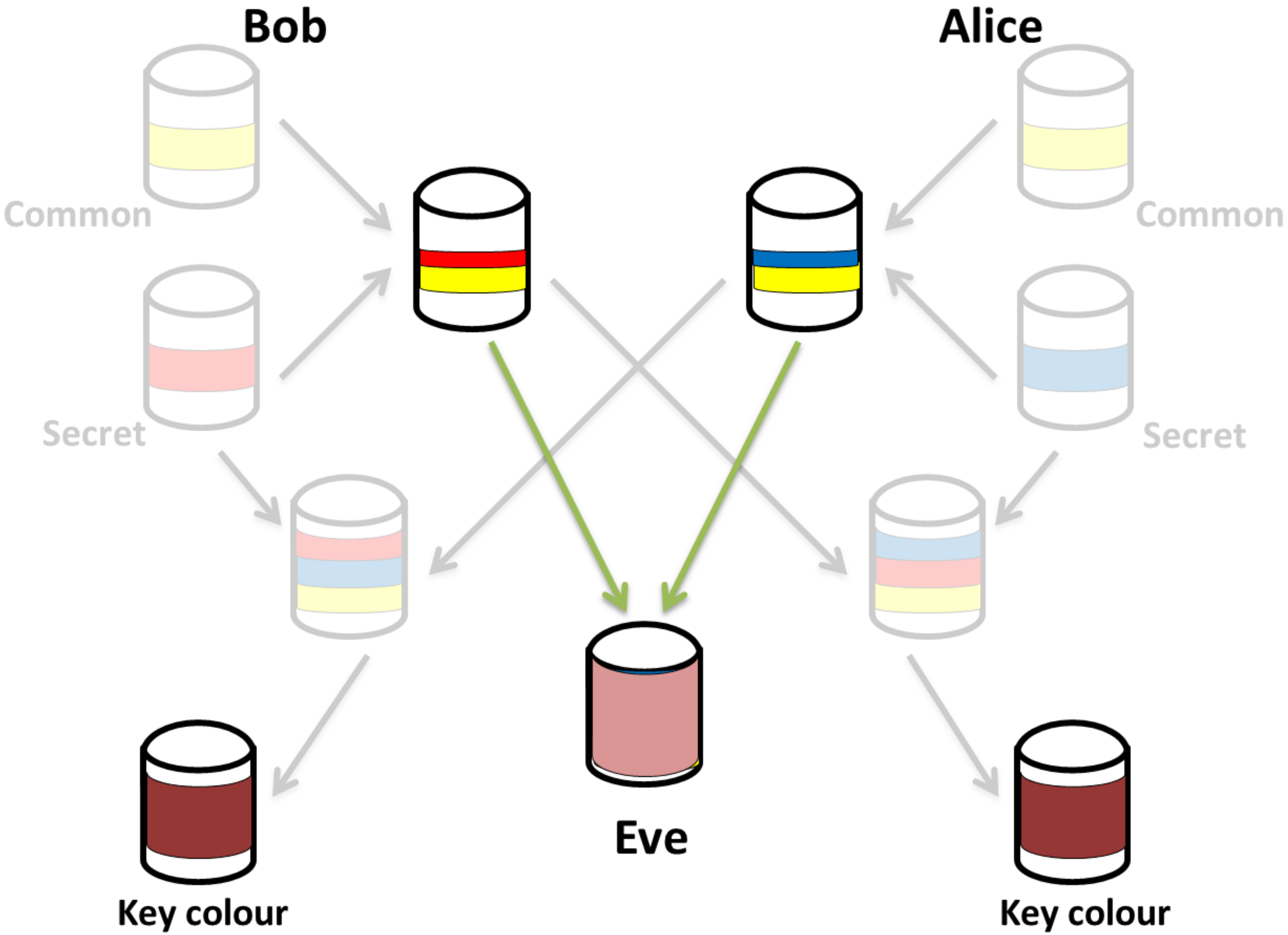
Analogy: The secret colour



Analogy: The secret colour




Bob
Alice
Common
Common
Secret
Secret
Key colour
Key colour


Bob
Alice
Common
Common
Secret
Secret
Eve
Key colour
Key colour

Basic idea of Diffie-Hellman

1. Alice sends her public key P_A to Bob.
2. Bob sends his public key P_B to Alice.
3. Alice computes $F(S_A,P_B)$. Note that only Alice can conduct this computation, since it involves her private key S_A.
4. Bob computes $F(S_B,P_A)$. Note that only Bob can conduct this computation, since it involves his private key S_B.

The special property for the public-key cryptosystem and the combination function F is that $F(S_A,P_B)=F(S_B,P_A)$.

Diffie-Hellman algorithm

Global Public Elements	
q	Prime number
α	$\alpha < q$, α a primitive root of q

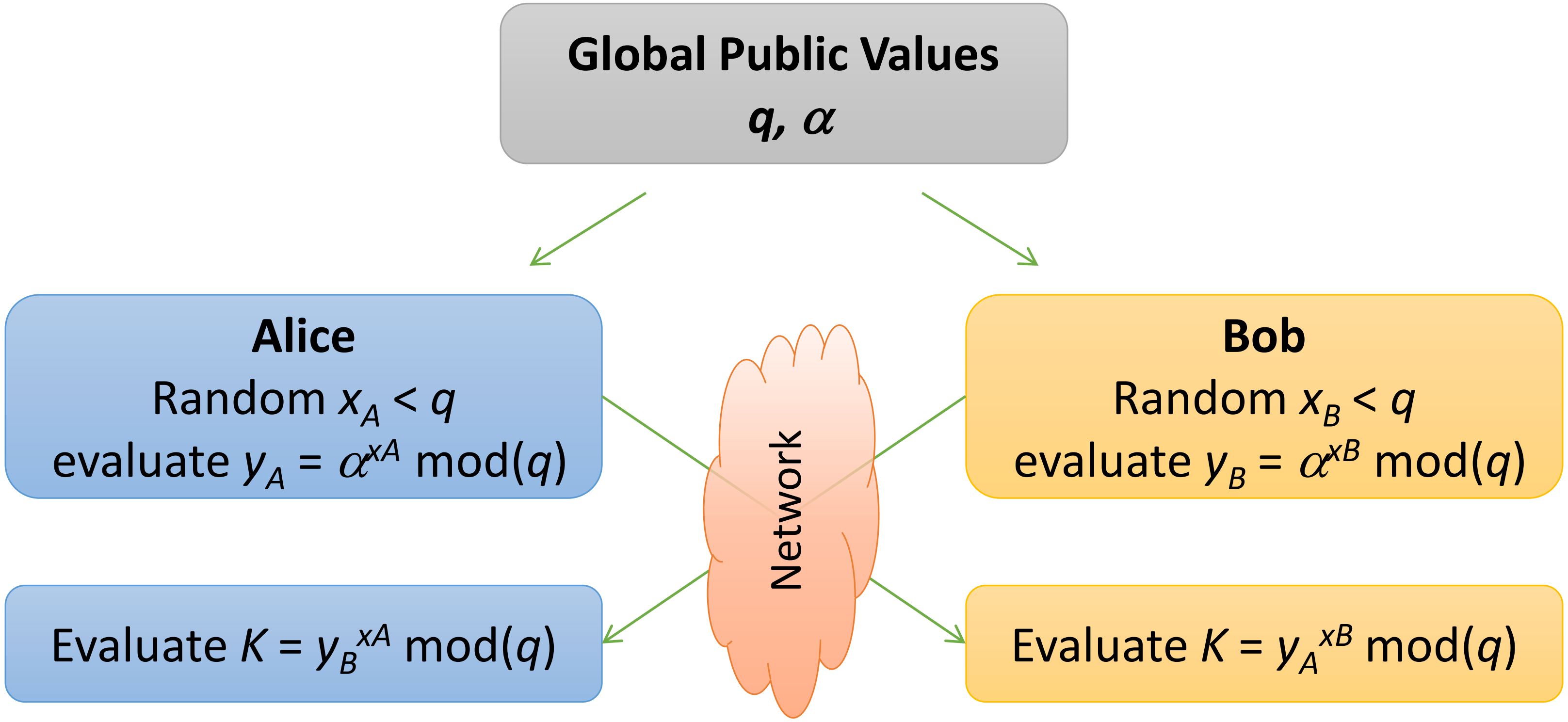
User Key generation

Alice	
Select a private x_A	$x_A < q$
Calculate public y_A	$y_A = \alpha^{xA} \bmod q$

Bob	
Select a private x_B	$x_B < q$
Calculate public y_B	$y_B = \alpha^{xB} \bmod q$

Details of Diffie-Hellman algorithm are not part of evaluation.

Diffie-Hellman algorithm



Man-in-the-middle attack

1. Eve prepares for the attack by generating two random private keys X_{E1} and X_{E2} and then computing the corresponding public keys Y_{E1} and Y_{E2}.
2. Alice transmits Y_A to Bob.
3. Eve intercepts Y_A and transmits Y_{E1} to Bob. Eve also calculates $K_2 = y_A^{XE2} \bmod(q)$.
4. Bob receives Y_{E1} and calculates $K_1 = y_{E1}^{XB} \bmod(q)$.
5. Bob transmits Y_B to Alice.
6. Eve intercepts Y_B and transmits Y_{E2} to Alice. Eve calculates $K_1 = y_B^{XE1} \bmod(q)$.
7. Alice receives Y_{E2} and calculates $K_2 = y_{E2}^{XA} \bmod(q)$.

At this point, Bob and Alice think that they share a secret key, but instead Bob and Eve share secret key and Alice and Eve share secret key.

Man-in-the-middle attack

Global Public Values

$\boldsymbol{q, \alpha}$

Alice

Random $x_A < q$

Evaluate $y_A = \alpha^{XA} \bmod(q)$

Y_A

Bob

Random $x_B < q$

Evaluate $y_B = \alpha^{XB} \bmod(q)$

Y_B

Eve

Random $x_{E1}<q$ & $x_{E2}<q$

Evaluate

$\boldsymbol{Y_{E1}} = \alpha^{XE1} \bmod(q)$

&

$\boldsymbol{Y_{E2}} = \alpha^{XE2} \bmod(q)$

Y_{E2}

Evaluate $K = y_B^{XA} \bmod(q)$

Evaluate $K_2 = y_{E2}^{XA} \bmod(q)$

Y_{E1}

Evaluate $K = y_A^{XB} \bmod(q)$

Evaluate $K_1 = y_{E1}^{XB} \bmod(q)$

Eve also has

$\boldsymbol{K_2} = y_A^{XE2} \bmod(q)$ & $\boldsymbol{K_1} = y_B^{XE1} \bmod(q)$

- This man-in-the middle attack was only able to succeed because there is no **data origin authentication**.
- Solution: *Public-key certificates*

One AKE protocol using Diffie-Hellman

1. Alice randomly generates a positive integer X_A and calculates Y_A. Alice sends Y_A to Bob, along with the certificate $Cert_A$ for her verification key.
2. Bob verifies $Cert_A$. If he is satisfied with the result, then Bob randomly generates a positive integer X_B and calculates Y_B. Next, Bob signs a message consisting of Alice's name, Y_A and Y_B. Bob then sends Y_B to Alice, along with the certificate $Cert_B$ for his verification key and the signed message.
3. Alice verifies $Cert_B$. If she is satisfied with the result, then she uses Bob's verification key to verify the signed message. If she is satisfied with this, she signs a message consisting of Bob's name, Y_A and Y_B, which she then sends back to Bob. Finally, Alice uses Y_B and her private key X_A to compute symmetric key.
4. Bob uses Alice's verification key to verify the signed message he has just received. If he is satisfied with the result, then Bob uses Y_A and his private key X_B to compute symmetric key.

Key Management

Key management

- Crucial to the security of any cryptosystem.
- Key management: *secure administration of cryptographic keys.*
- Key lifecycle:
  - **Key generation**: the creation of keys.
  - **Key establishment**: the process of making sure keys reach the end points where they will be used. the most difficult phase of the key lifecycle to implement.
  - **Key storage**: the safekeeping of keys.
  - **Key usage**: how keys are used.

Key lifetimes

- A key can only be used for a specified period of time, during which it is regarded as being *live*.
- Finite key lifetimes mitigate against key compromise, key management failures, future attacks.
- Finite key lifetimes provide flexibility to suit application requirements. E.g. short data keys that expire quickly.

Key lengths

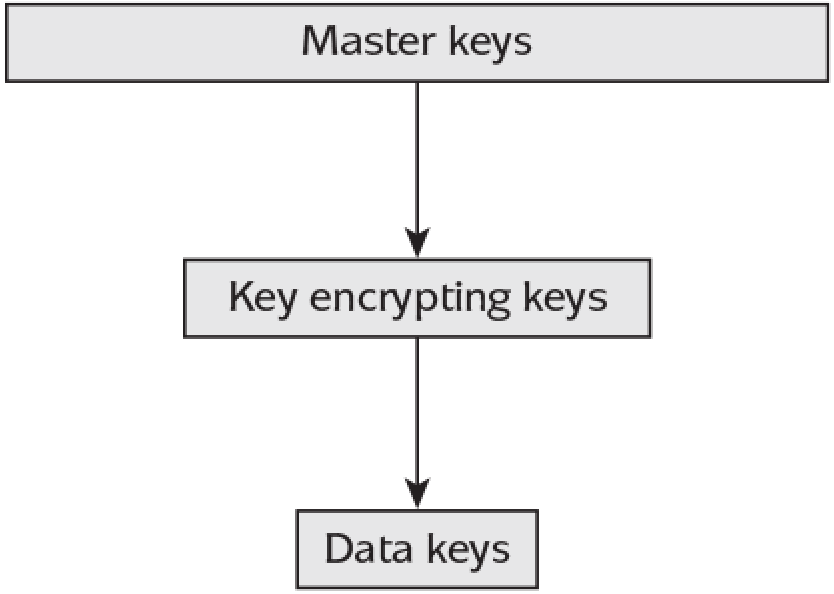
- Key length recommendations for symmetric cryptography tend to be **algorithm-independent**.
- Key length recommendations for public-key cryptography tend to be **algorithm-specific**.

Key lengths example

Protection	Notes	Key length
Vulnerable to attacks in ‘real time’ by individuals	Limited use	32
Very short-term protection against small organisations	Not for new applications	64
Short-term protection against medium organisations; medium-term protection against small organisations		72
Very short-term protection against agencies; long-term protection against small organisations	Protection to 2012.	80
Legacy standard level	Protection to 2020.	96
Medium-term protection	Protection to 2030.	112
Long-term protection	Protection to 2040.	128
‘Foreseeable future’	Good protection against quantum computers	256

Key hierarchy

- Ranking of keys, with high-level keys being more ‘important’ than low-level keys. Keys at one level are used to encrypt keys at the level beneath.

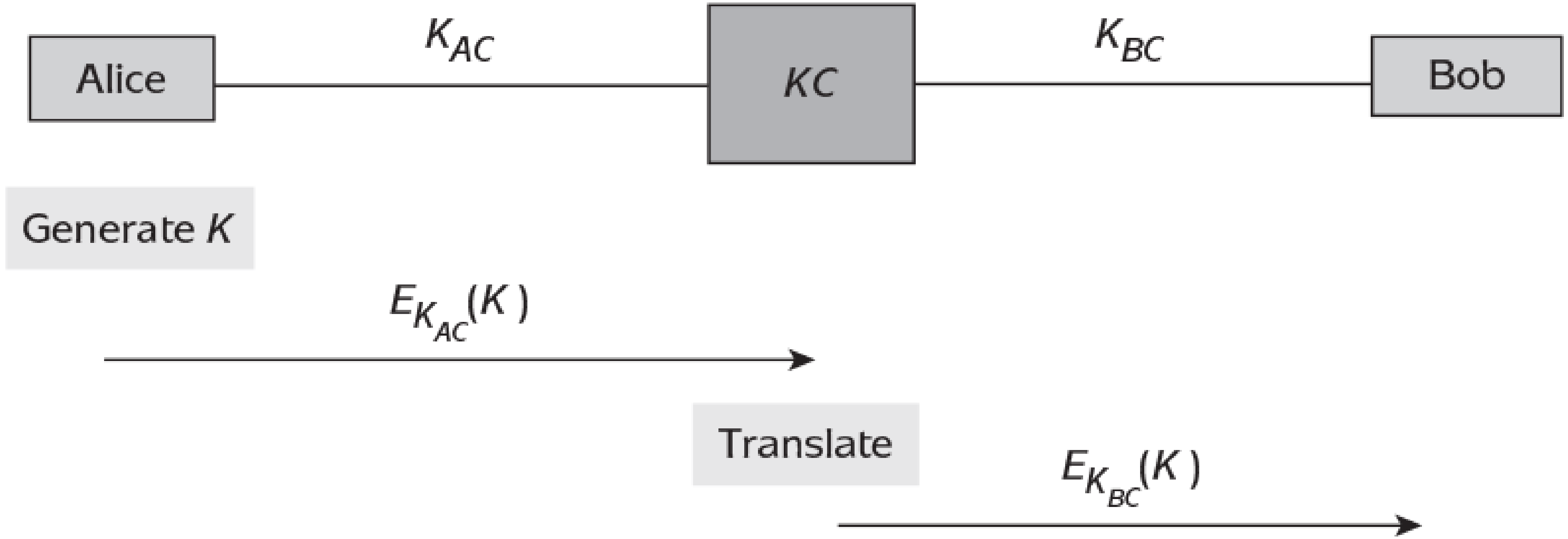


A key distribution scenario

- Consider a simple two-level hierarchy consisting of only master and data keys.
- If we have a network of n users, then the number of possible pairs of users is $n(n-1)/2$, which is the number of shared master keys. This is not practical for a network with many users.
- *Key Centre* (Key Distribution Centre) – a trusted third party.
- Each user in the network shares a master key with the KC.

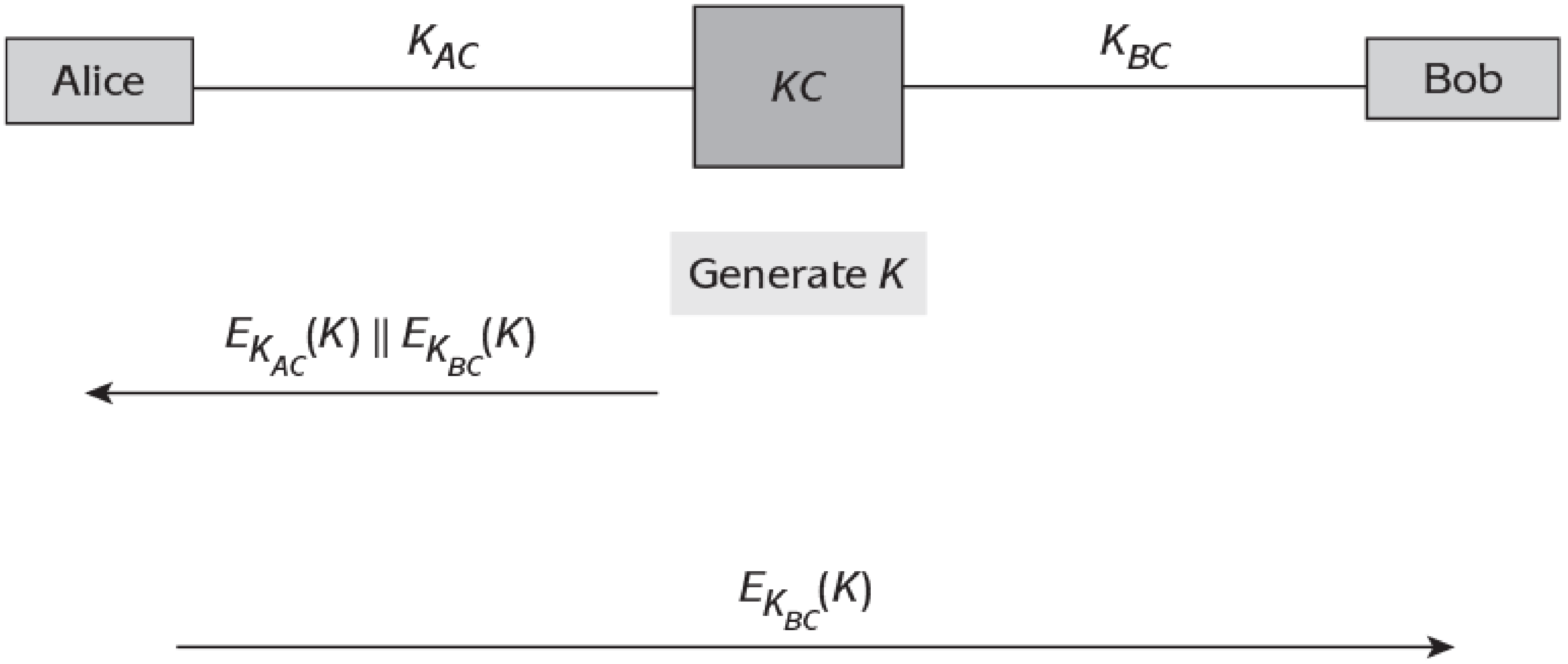
Key distribution using symmetric encryption

- Key translation



Key distribution using symmetric encryption

- Key despatch



Key distribution using public-key encryption

- Hybrid encryption can be used for key distribution.
- The big question is how we can be sure the identity of another party, i.e, the public key a party claims to belong to is actually that party’s public key?
- Solution: public-key certificate.

Public-key certificate

- A *public-key certificate* is data binding a public key to data relating to the assurance of purpose of this public key. It can be thought of as a trusted directory entry in a sort of distributed database.
- Contents of a Public-Key Certificate
  - ***Name of owner***. The name of the owner of the public key. This owner could be a person, a device, or even a role within an organisation.
  - ***Public-key value***. The public key itself.
  - ***Validity time period***. This identifies the date and time from which the public key is valid and, more importantly, the date and time of its expiry.
  - ***Signature***. The creator of the public-key certificate digitally signs all the data that forms the public-key certificate, including the name of owner, public-key value, and validity time period.
  - And more… (X.509: public-key certificate standard)

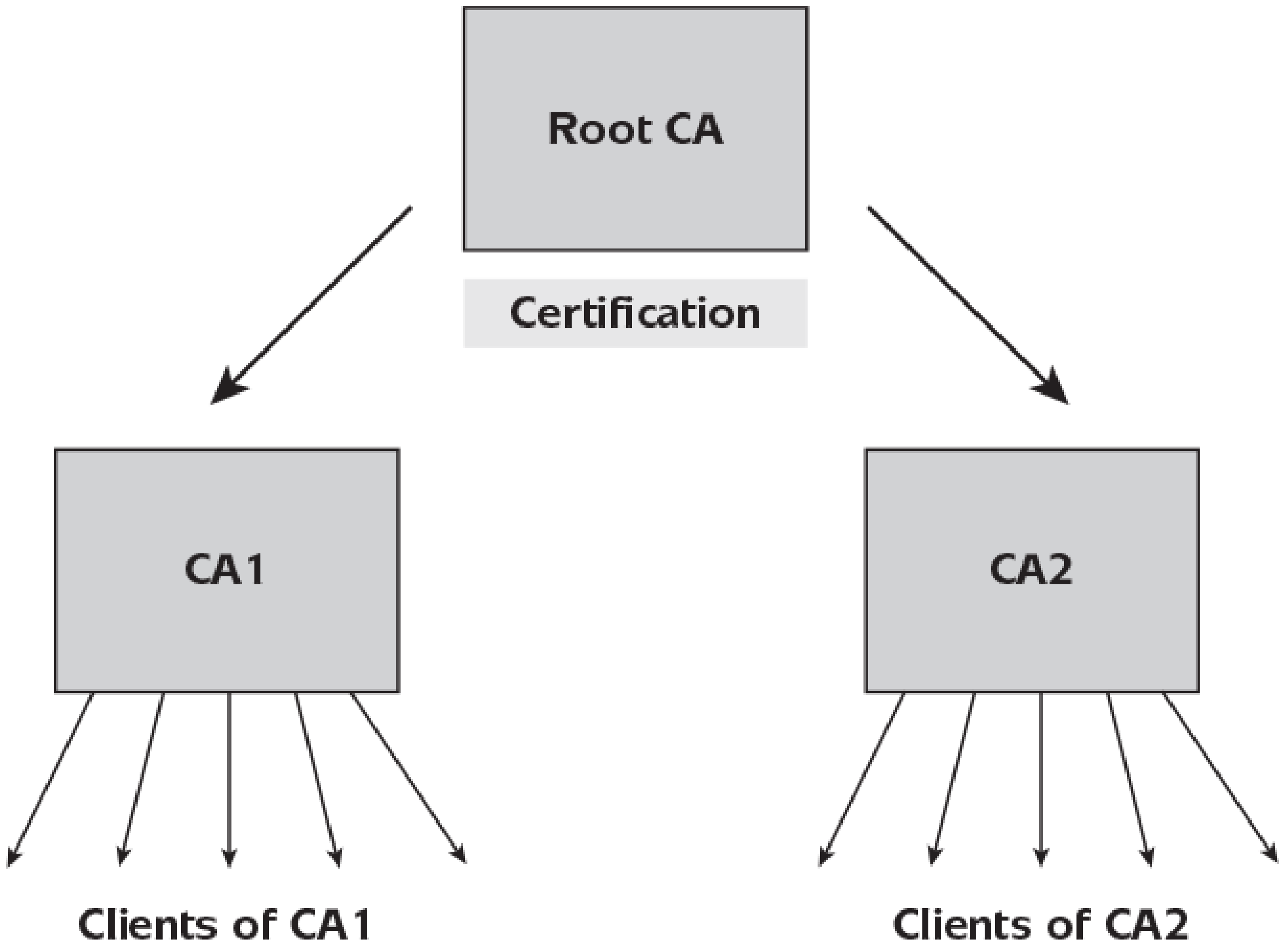
Certificate authority

- *Certificate authority:* creator of a public-key certificate.
  - Certificate **creation**: creating and signing the public-key certificate, and then issuing it to the owner.
  - Certificate **revocation**. The CA is responsible for revoking the certificate in the event that it becomes invalid.
  - Certificate **trust anchor**. The CA acts as the point of trust for any party relying on the correctness of the information contained in the public-key certificate.

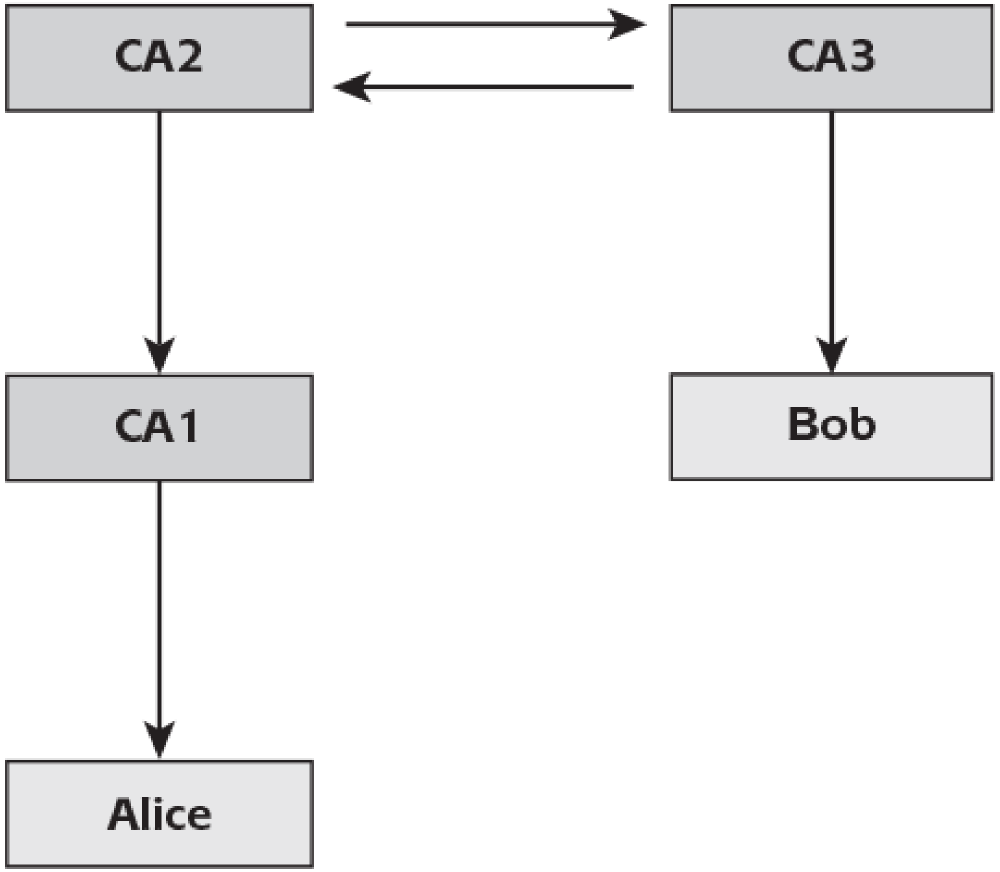
Relying on a Public-Key Certificate

- **Trust the CA**. The relying party needs to be able to trust the CA to have performed its job correctly when creating the certificate.
- **Verify the signature on the certificate**. The relying party needs to have access to the verification key of
- **Check the fields**. The relying party needs to check all the fields in the public-key certificate. In particular, they must check the name of the owner and that the public-key certificate is valid.

Certification hierarchies



Certificate chains



Web of trust

- Alternate approach to certificate-based approach.
- Suppose Alice wishes to directly provide relying parties with her public key.
- The idea of a web of trust involves other public-key certificate owner’s acting as ‘lightweight CAs’ by digitally signing Alice’s public key.
- Alice gradually develops a key ring, which consists of her public key plus a series of digital signatures by other owners attesting to the fact that the public-key value is indeed Alice’s.
- Used in PGP (Pretty Good Privacy)

Summary

- Cryptographic Protocols
  - Components, stages
- Authentication and Key Establishment (AKE) protocol
- Diffie–Hellman key agreement
- Key management
  - Lifetime, lengths, hierarchy
  - Key distribution
- Public-key certificate
  - Certificate Authority (CA)
- Web of trust