

# *Protocols*

Chaire Informatique et sciences numériques  
Collège de France, cours du 11 mai 2011

# Security protocols

- Security protocols are concerned with properties such as authenticity and secrecy.
  - Primary examples are protocols (like SSL) that establish communication channels.
  - Other examples include protocols for electronic voting and commerce.
- In distributed systems, security protocols invariably rely on cryptography.

*Authentication protocols  
(or channel-establishment protocols)*

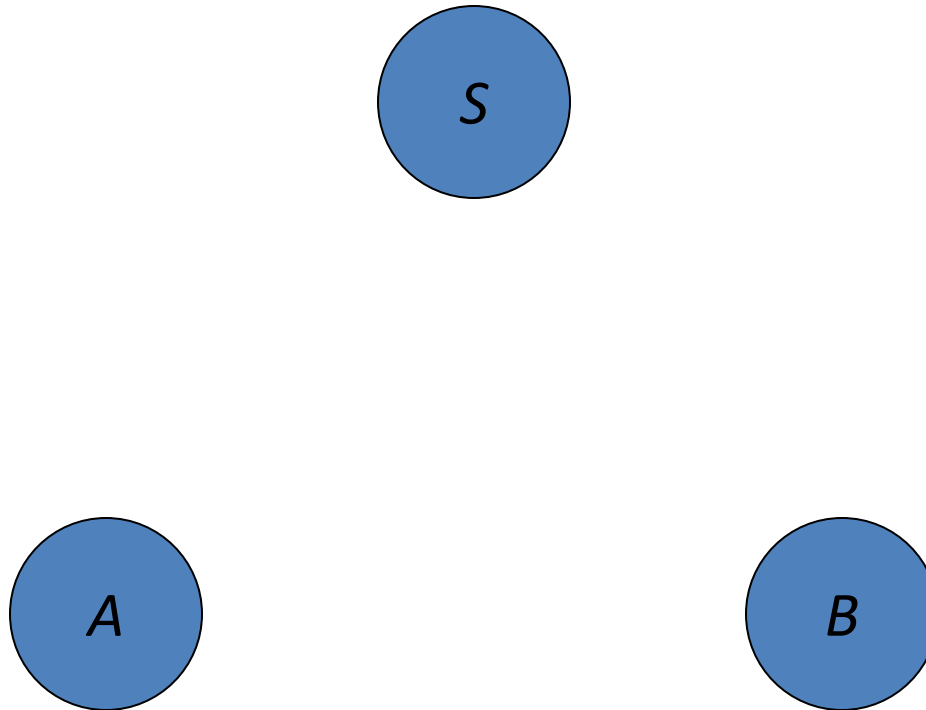
# Authentication protocols

There are many authentication protocols.

They typically involve:

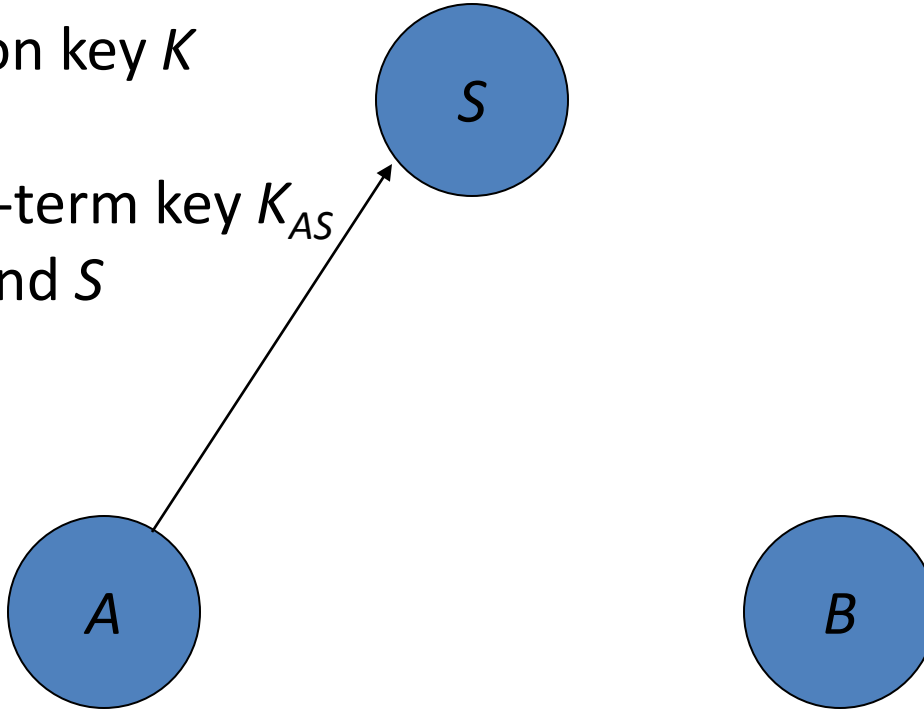
- two principals (hosts, users, ...),
- secrets (possibly shared, usually keys),
- cryptography (shared-key or public-key),
- trusted servers,
- proofs of timeliness (nonces, timestamps).

# A typical authentication protocol



# A typical authentication protocol

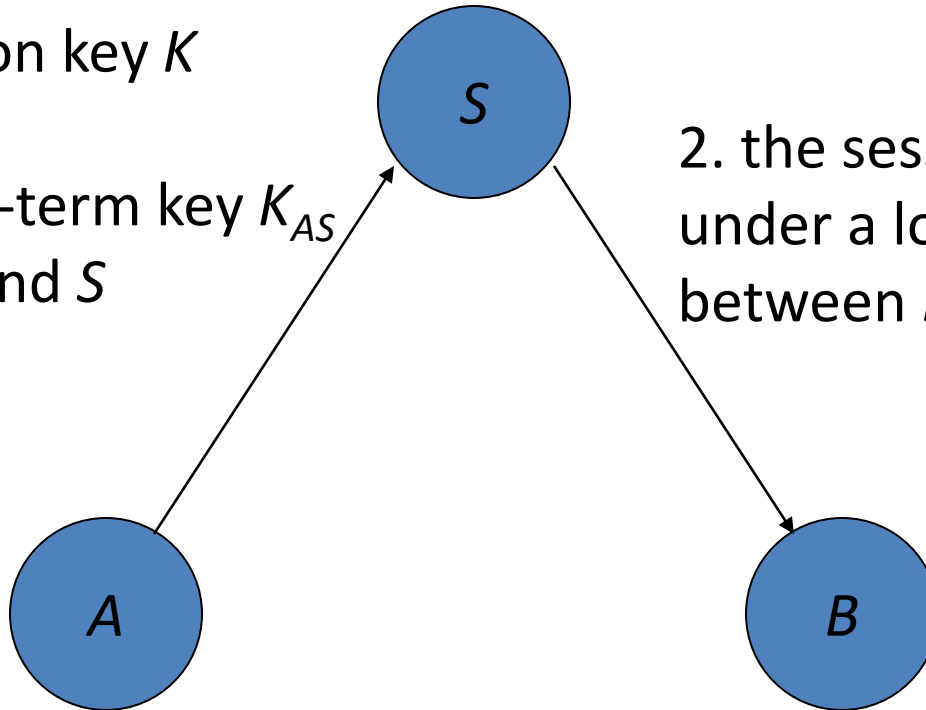
1. new session key  $K$   
for  $A$  and  $B$ ,  
under a long-term key  $K_{AS}$   
between  $A$  and  $S$



# A typical authentication protocol

1. new session key  $K$   
for  $A$  and  $B$ ,  
under a long-term key  $K_{AS}$   
between  $A$  and  $S$

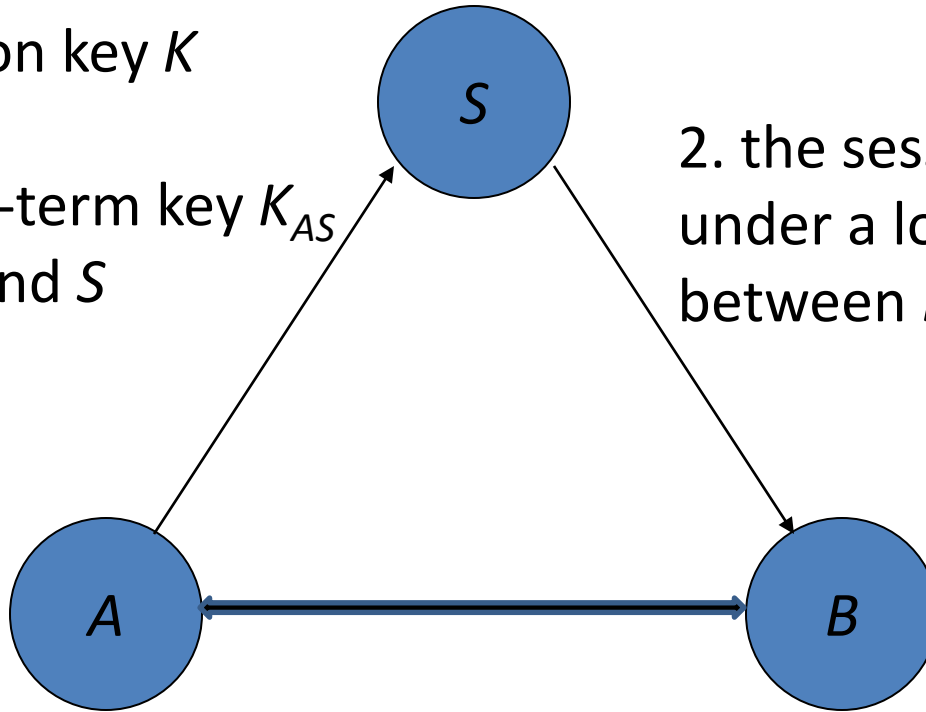
2. the session key  $K$   
under a long-term key  $K_{BS}$   
between  $B$  and  $S$



# A typical authentication protocol

1. new session key  $K$   
for  $A$  and  $B$ ,  
under a long-term key  $K_{AS}$   
between  $A$  and  $S$

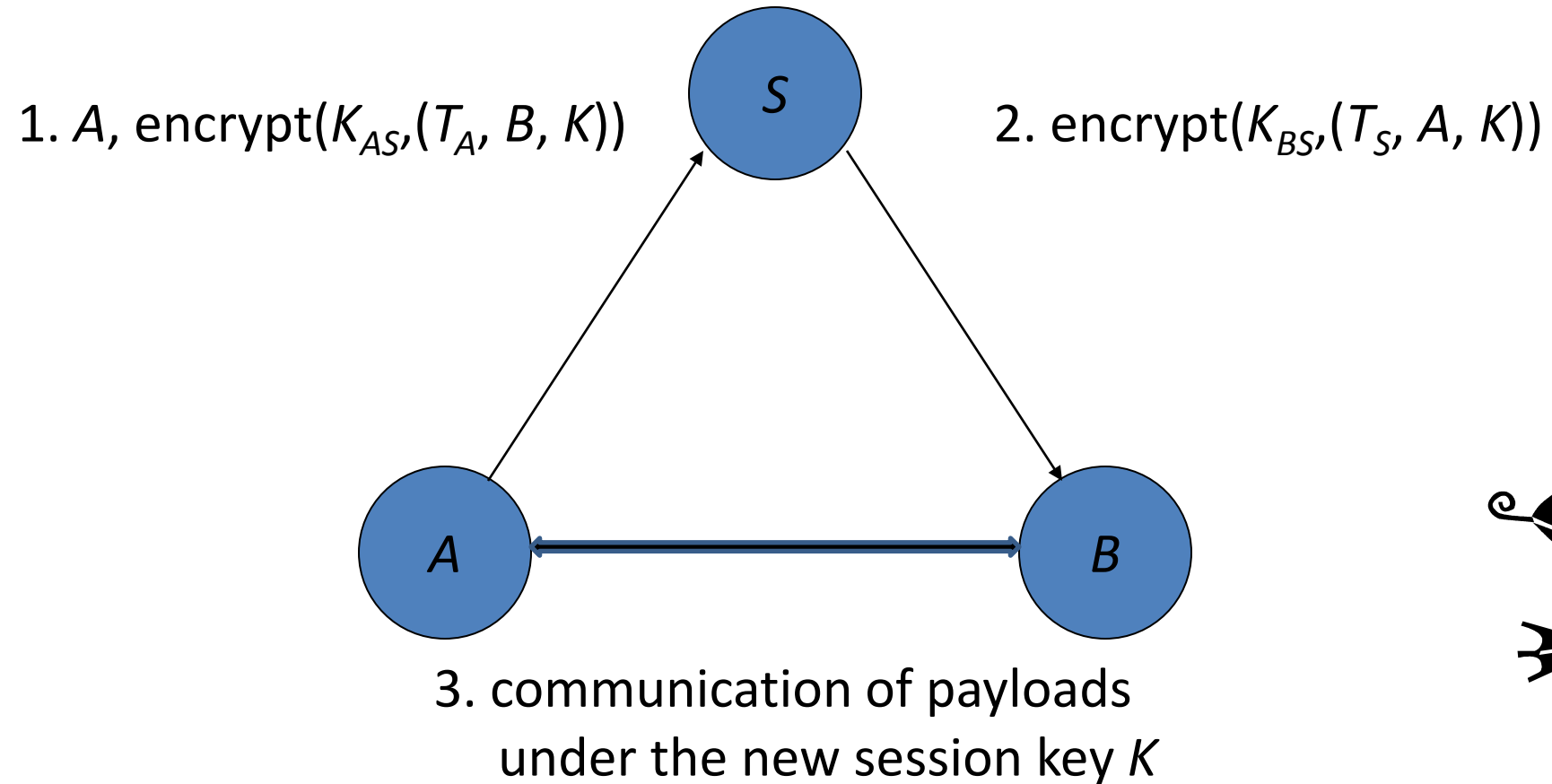
2. the session key  $K$   
under a long-term key  $K_{BS}$   
between  $B$  and  $S$



3. communication of payloads  
under the session key  $K$



# A closer look: the WMF Protocol



$T_A, T_S$  are timestamps.

Here **encrypt** is symmetric encryption. *It may include authentication.*

# Questions

- What assumptions are we making?
- Does the protocol work?
- Can we do better?



# Assumptions: communication

*We assume that an intruder can interpose a computer in all communication paths, and thus can alter or copy parts of messages, replay messages, or emit false material. While this may seem an extreme view, it is the only safe one when designing authentication protocols.*

(Needham and Schroeder, 1978)

*This view is sadly realistic, at least for the Internet.  
It partly explains the use of timestamps in protocols.*

# Assumptions: end-point security

*We also assume that each principal has a secure environment in which to compute, such as is provided by a personal computer or would be by a secure shared operating system.* (Needham and Schroeder, 1978)

*In fact, end-points are not secure monoliths.*

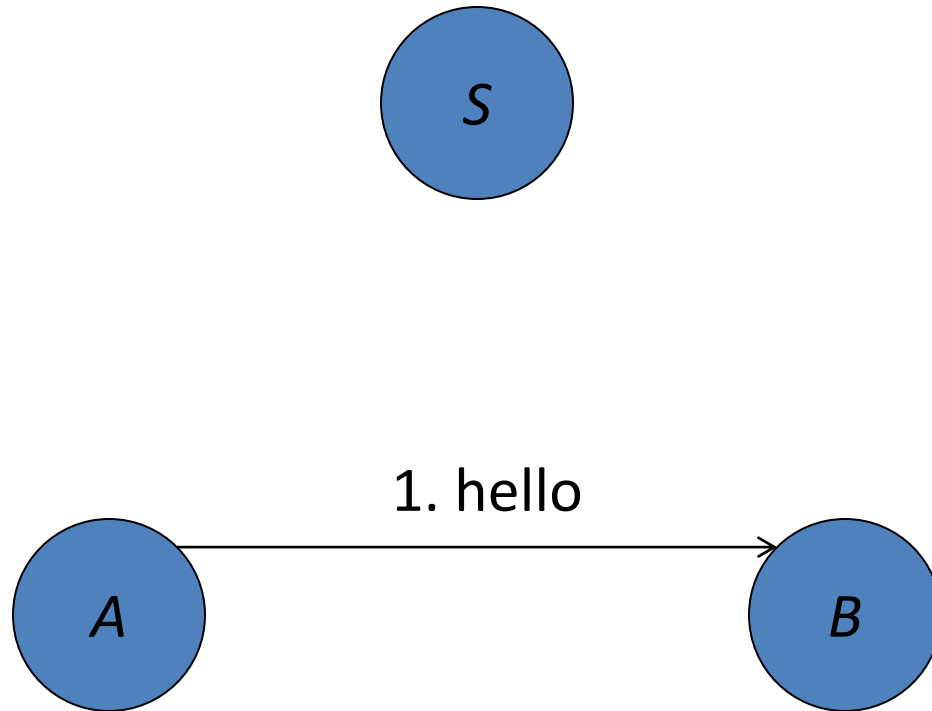
*E.g., Web 2.0 gadgets in mashups may not be trusted, and communicate with small protocols.*

# No single protocol will do...

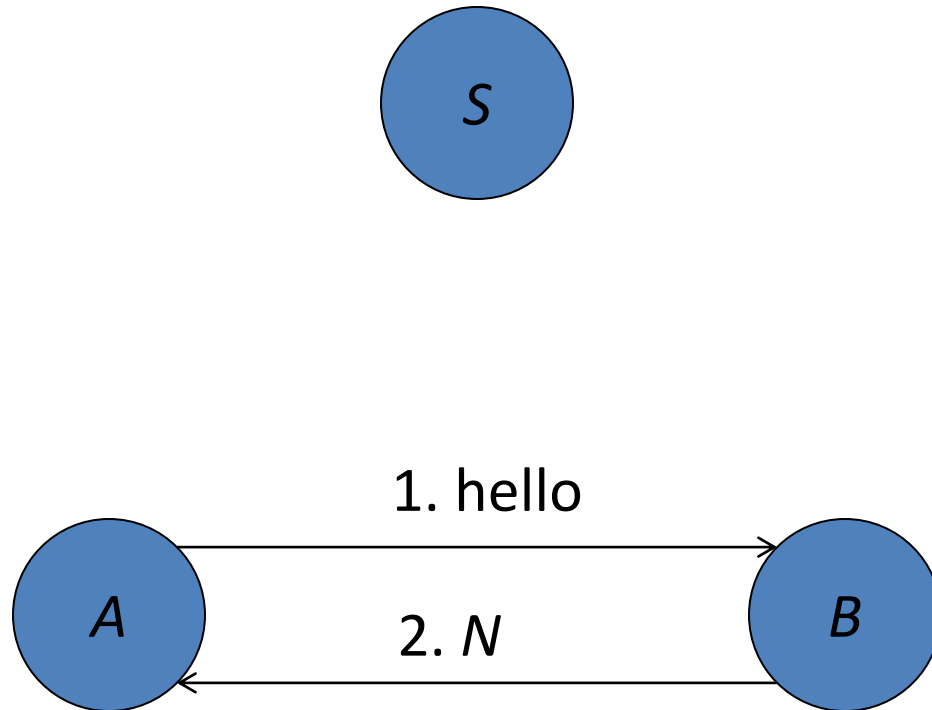
We may want:

- few messages,
- little encryption,
- little trust of other machines,
- human users (with limited memory or smart-cards),
- asynchronous checking (for storage and e-mail),
- different cryptosystems,
- little server state,
- one-way or two-way authentication,
- client anonymity.

# A WMF variant with a nonce

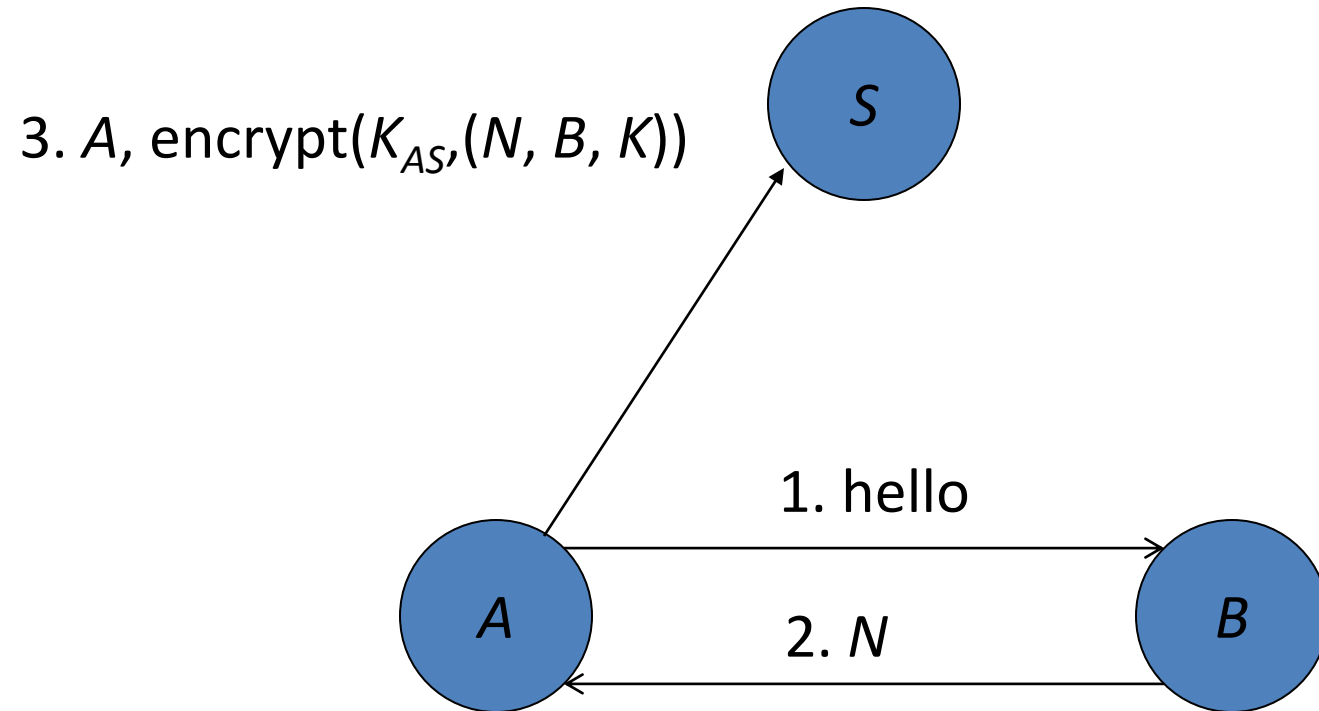


# A WMF variant with a nonce



*N* are is a **nonce**: a quantity generated for the purpose of being fresh.

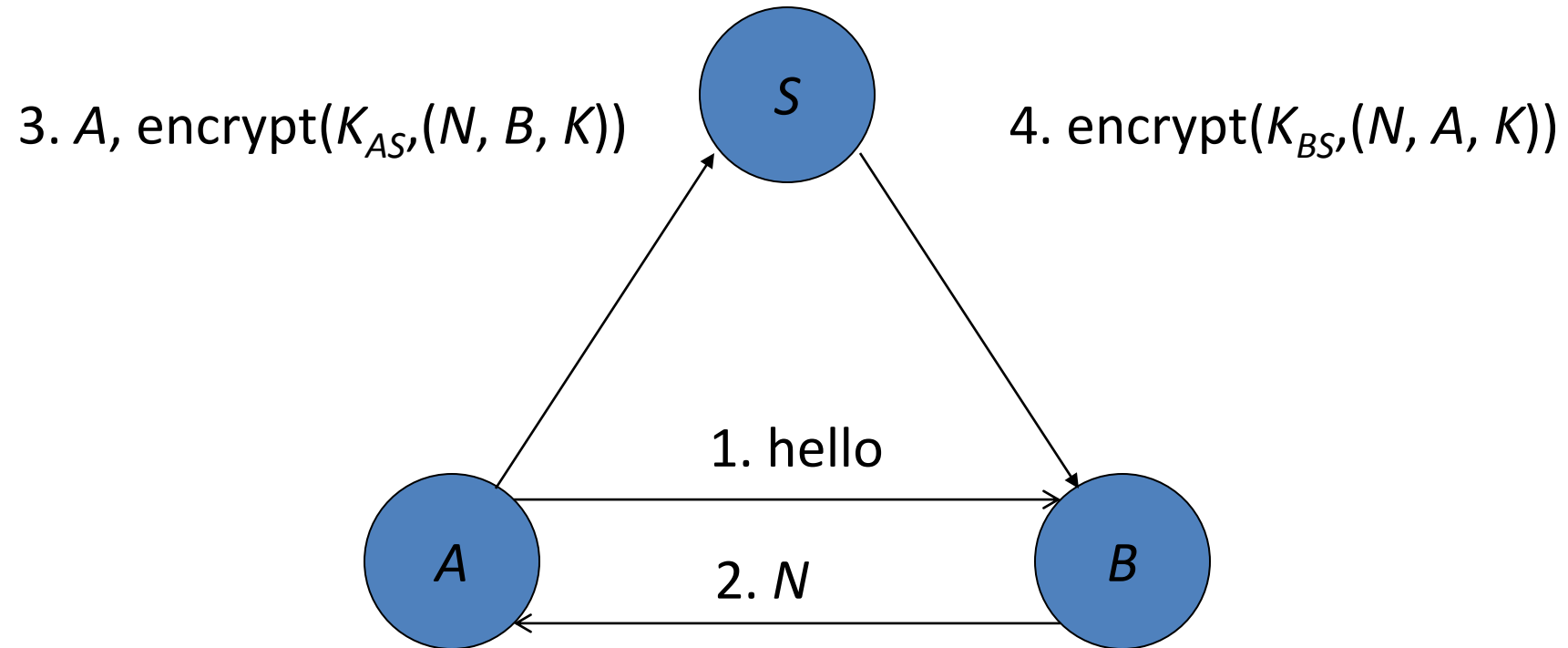
# A WMF variant with a nonce



$N$  are is a **nonce**: a quantity generated for the purpose of being fresh.

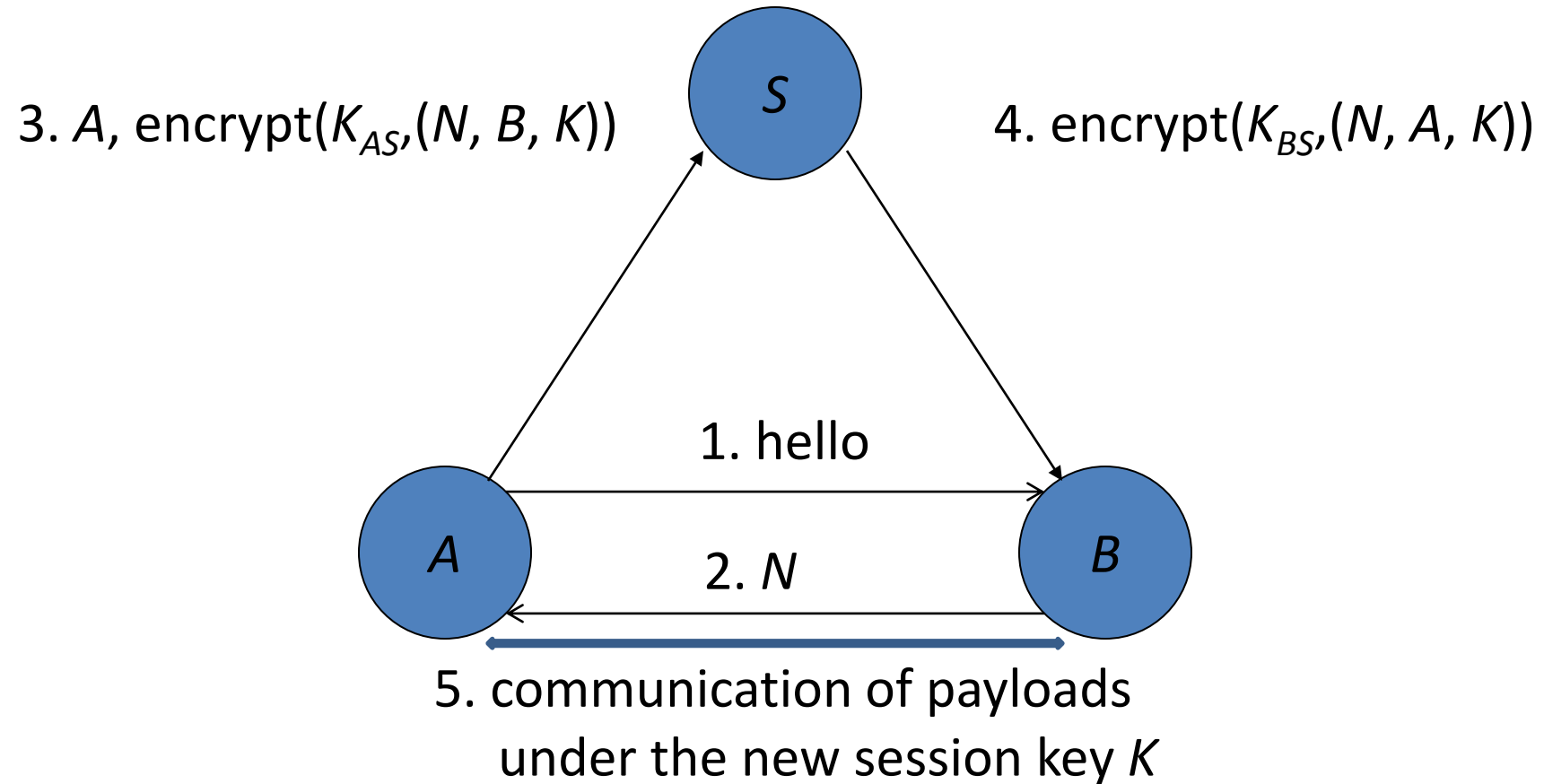


# A WMF variant with a nonce

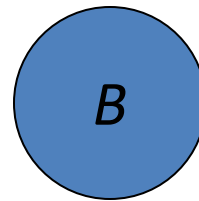
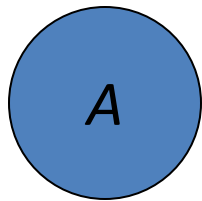
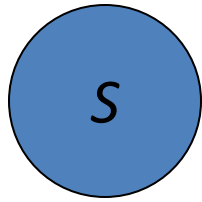


$N$  are is a **nonce**: a quantity generated for the purpose of being fresh.

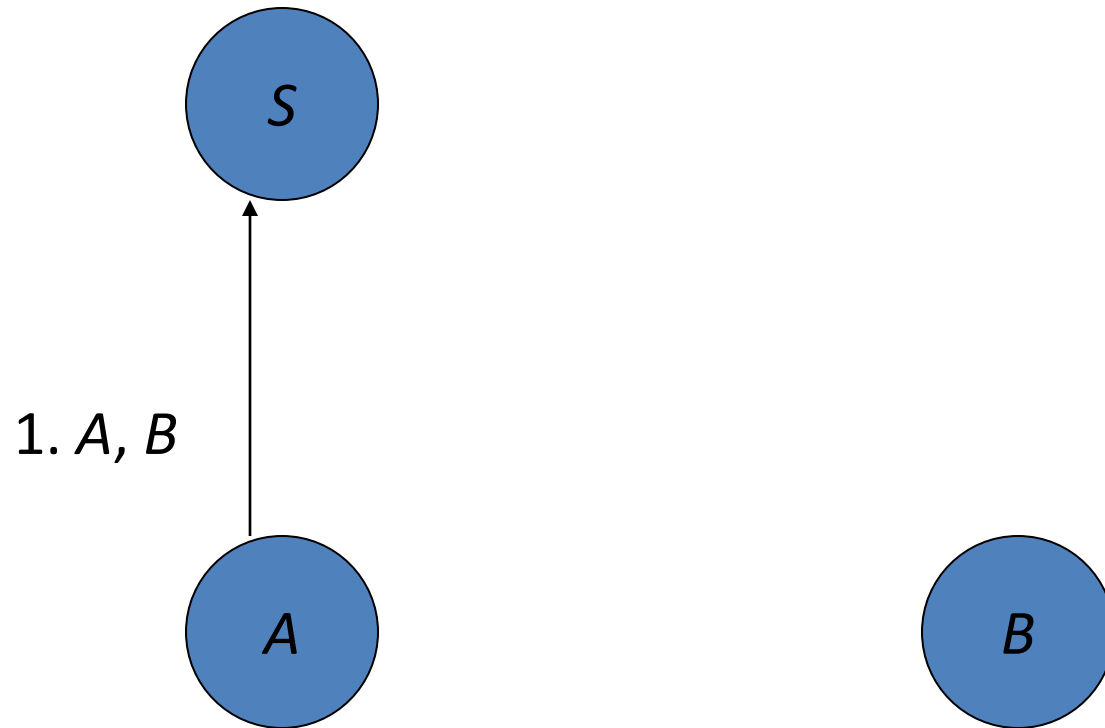
# A WMF variant with a nonce



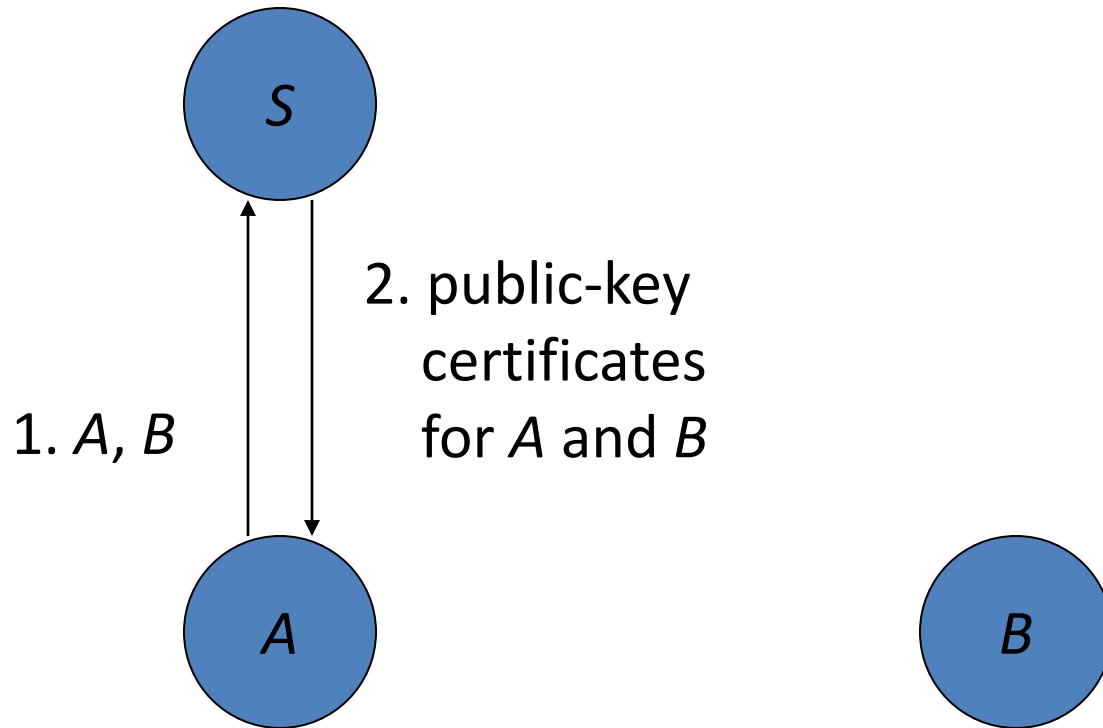
# Another authentication protocol



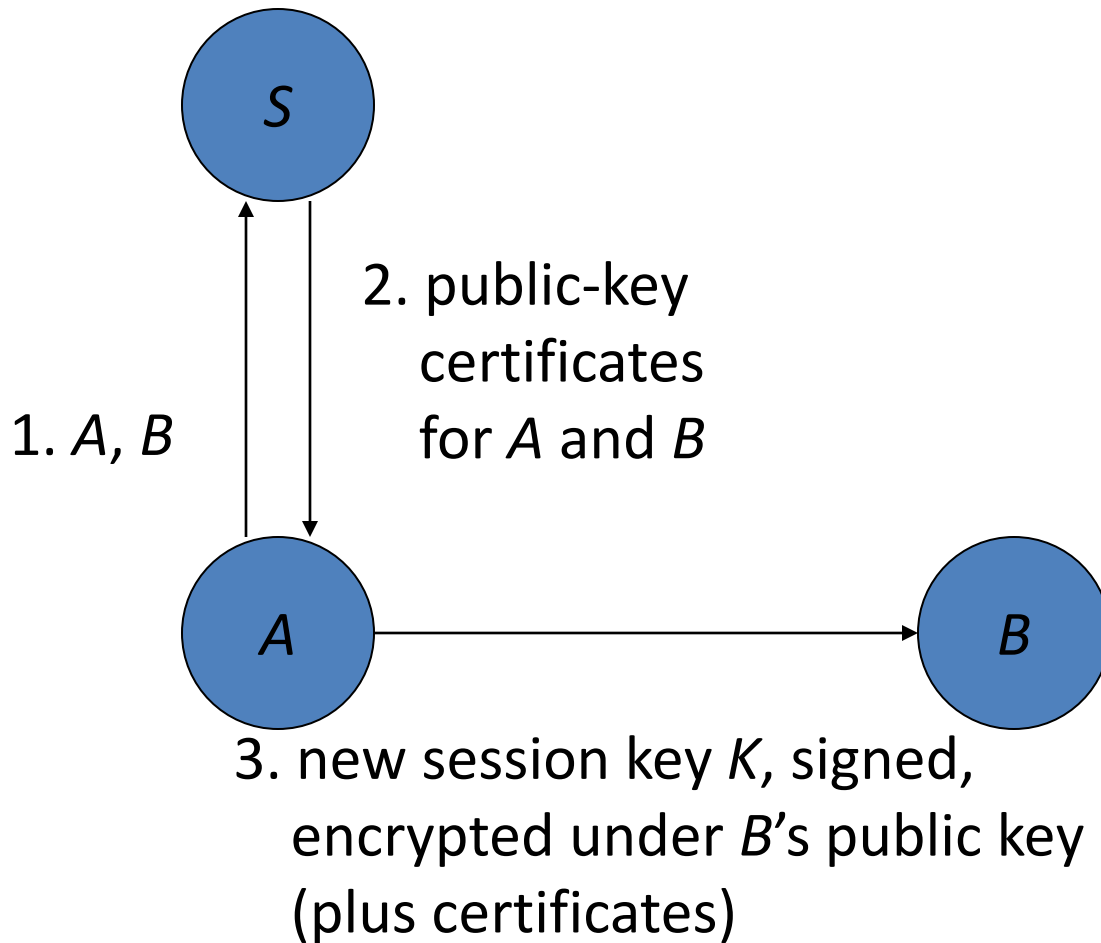
# Another authentication protocol



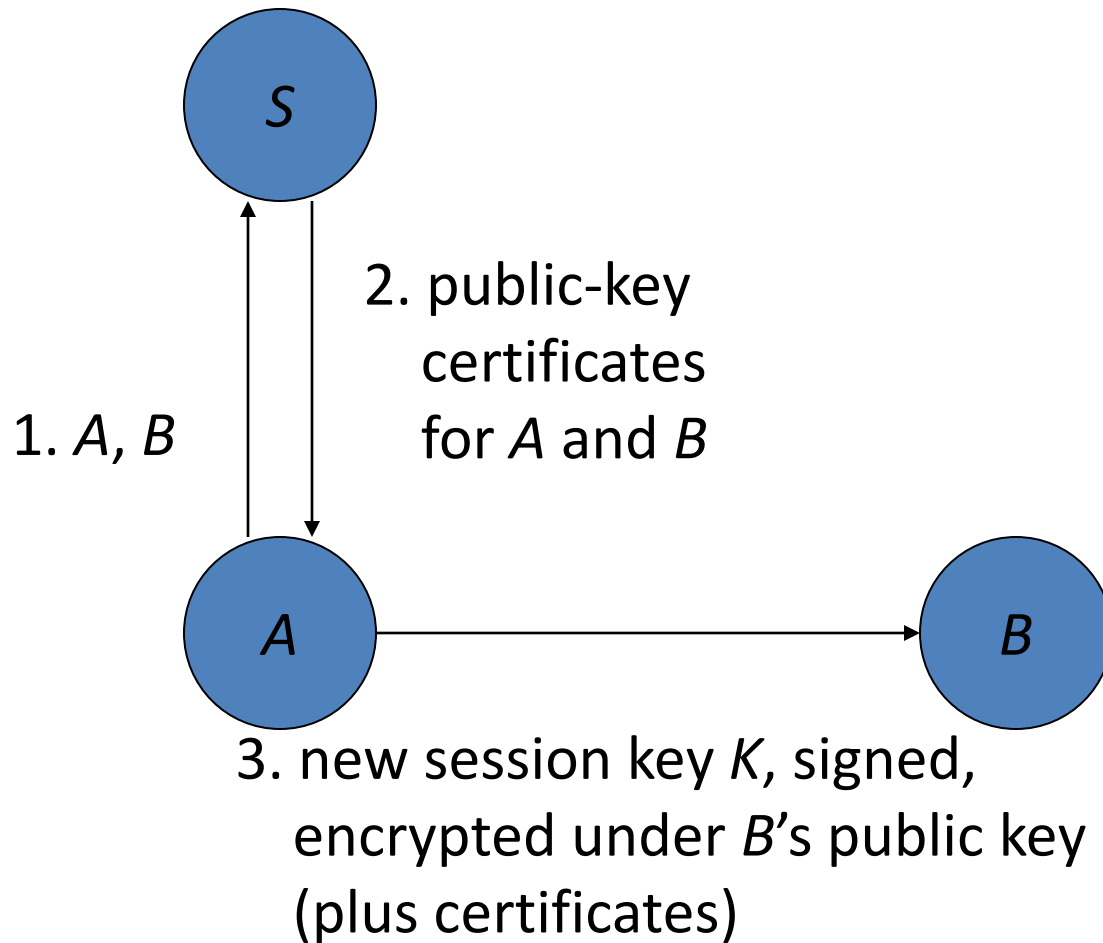
# Another authentication protocol



# Another authentication protocol

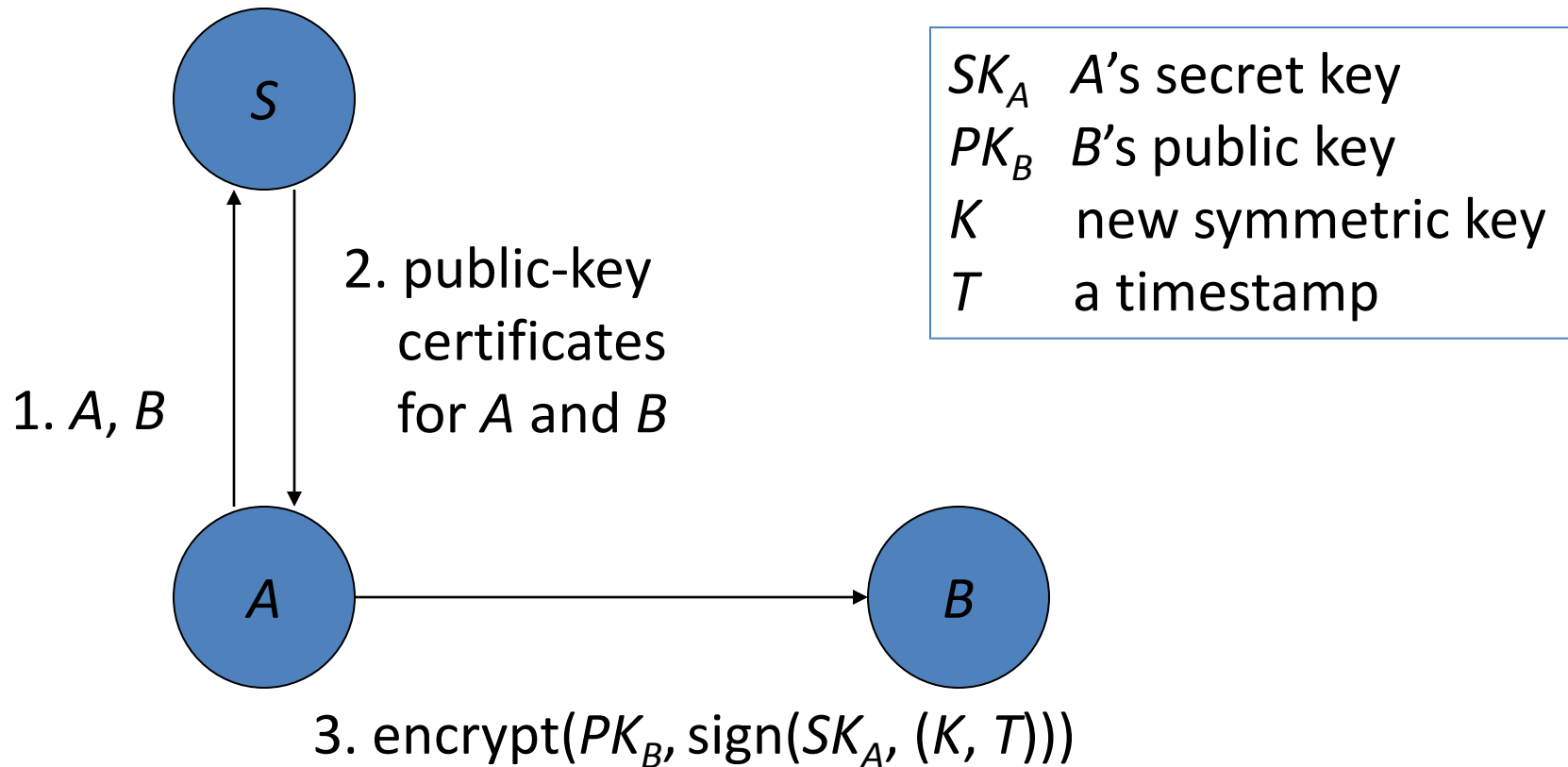


# Another authentication protocol



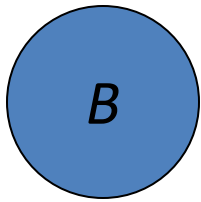
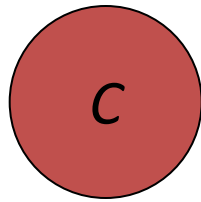
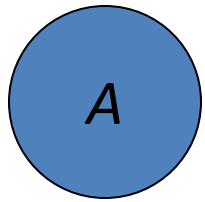
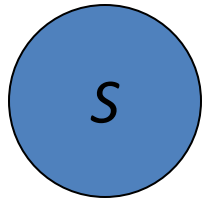
As usual, *K* can then be used for encrypting and MACing payloads.

# A closer look: the Denning-Sacco public-key protocol

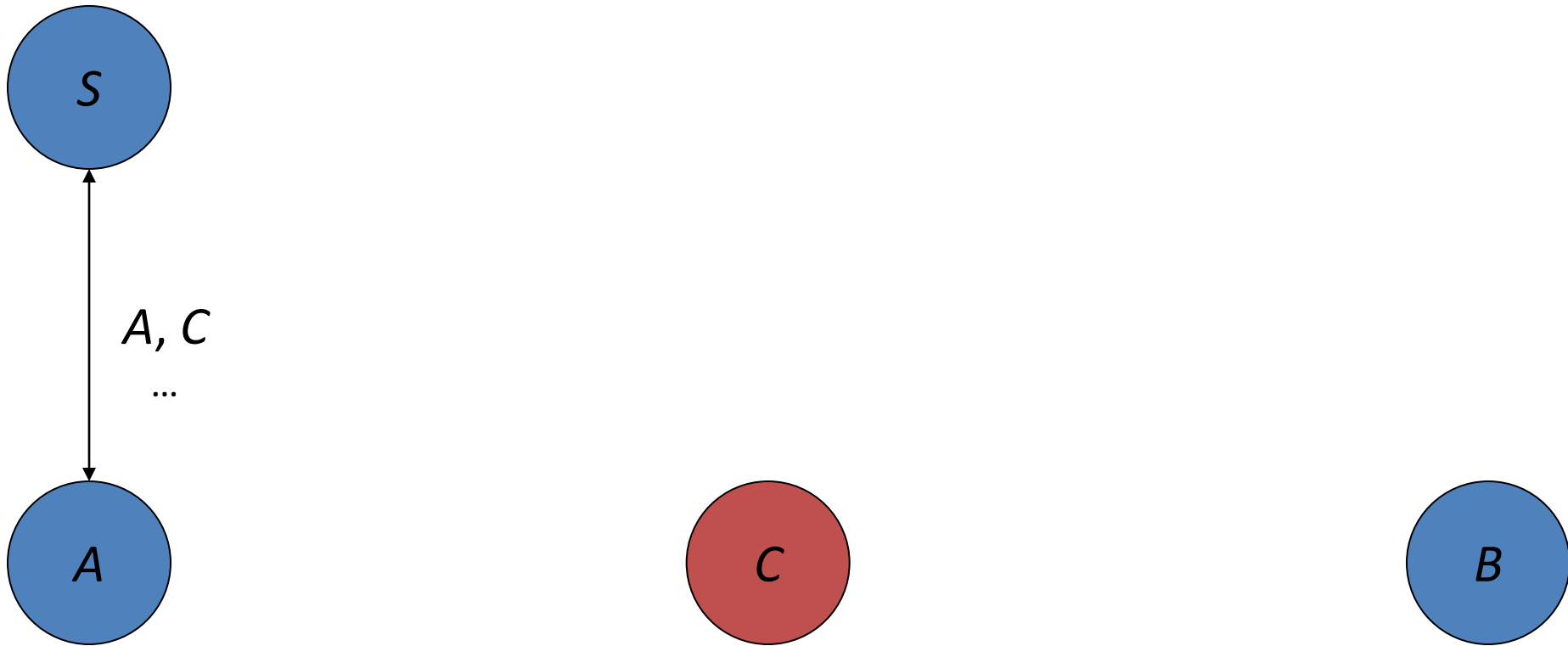




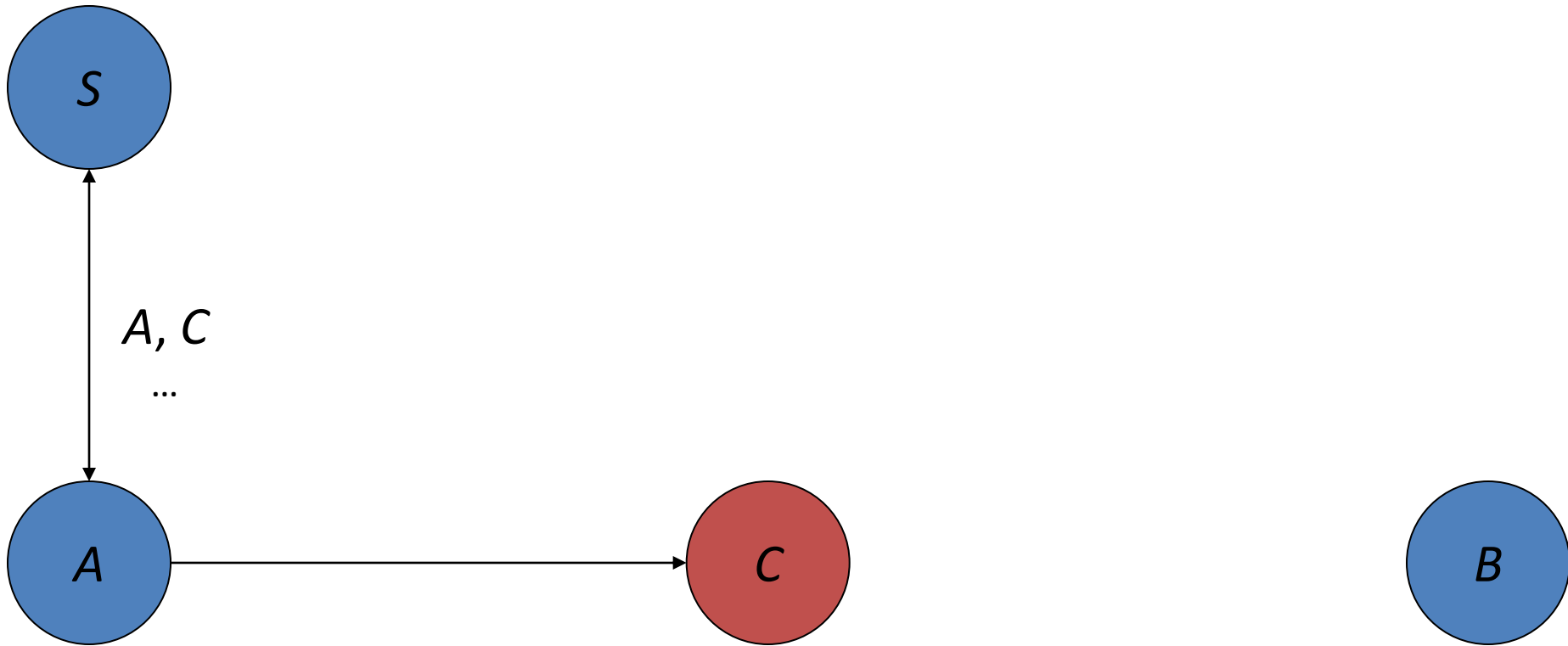
# An attack



# An attack

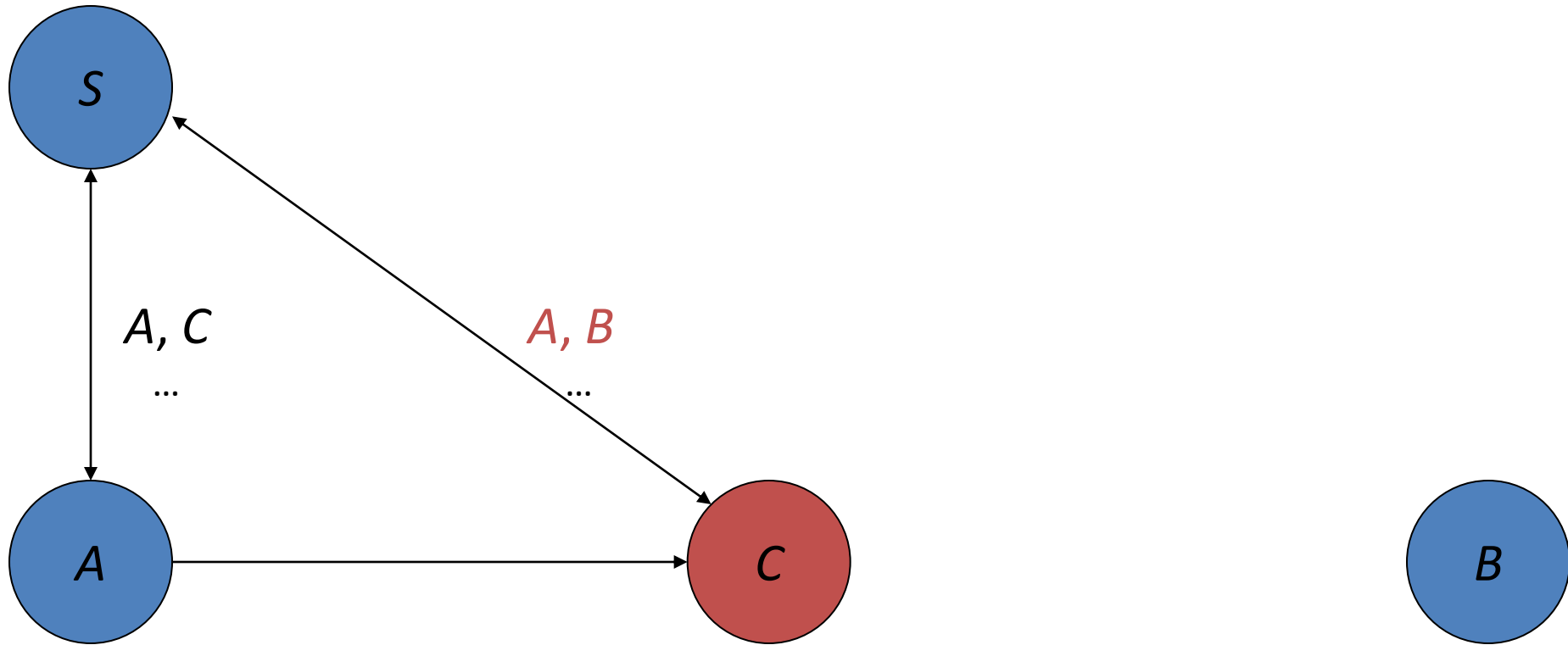


# An attack



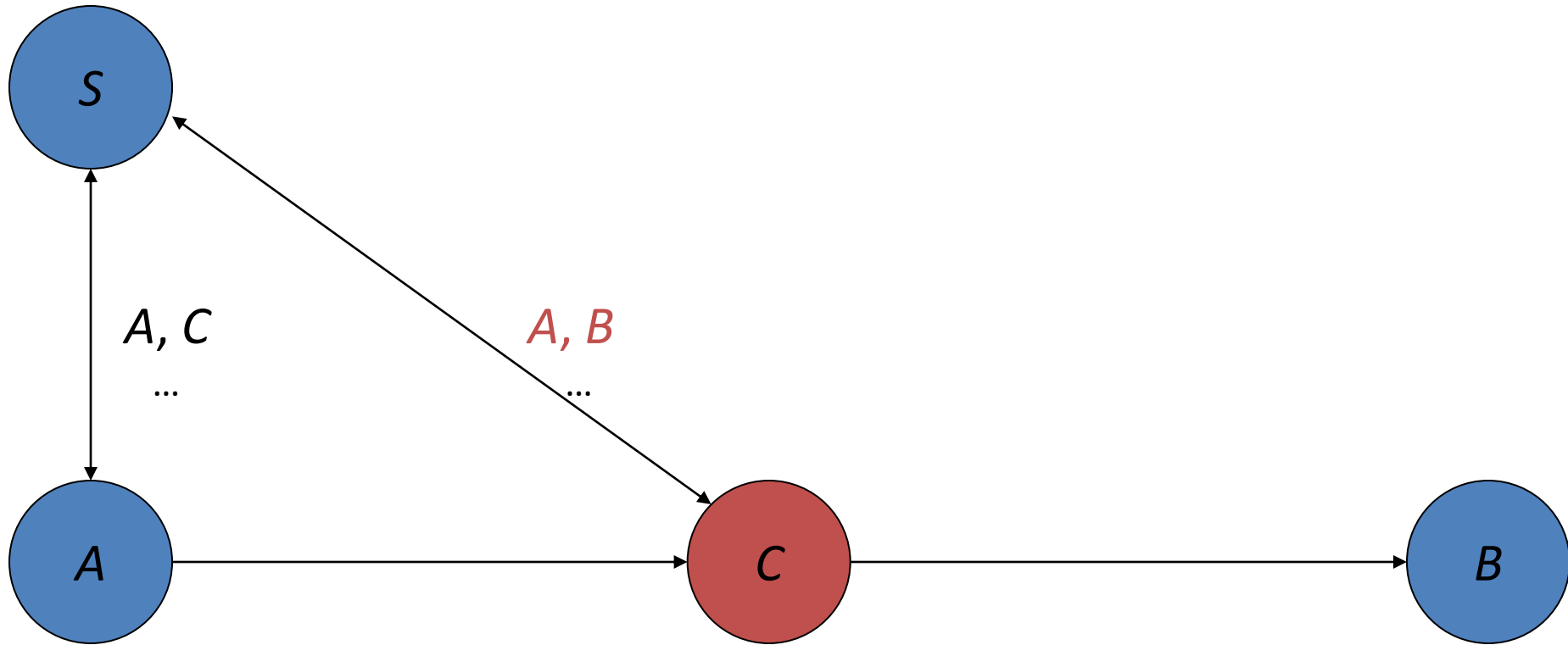
3.  $\text{encrypt}(PK_C, \text{sign}(SK_A, (K, T)))$

# An attack



3.  $\text{encrypt}(PK_C, \text{sign}(SK_A, (K, T)))$

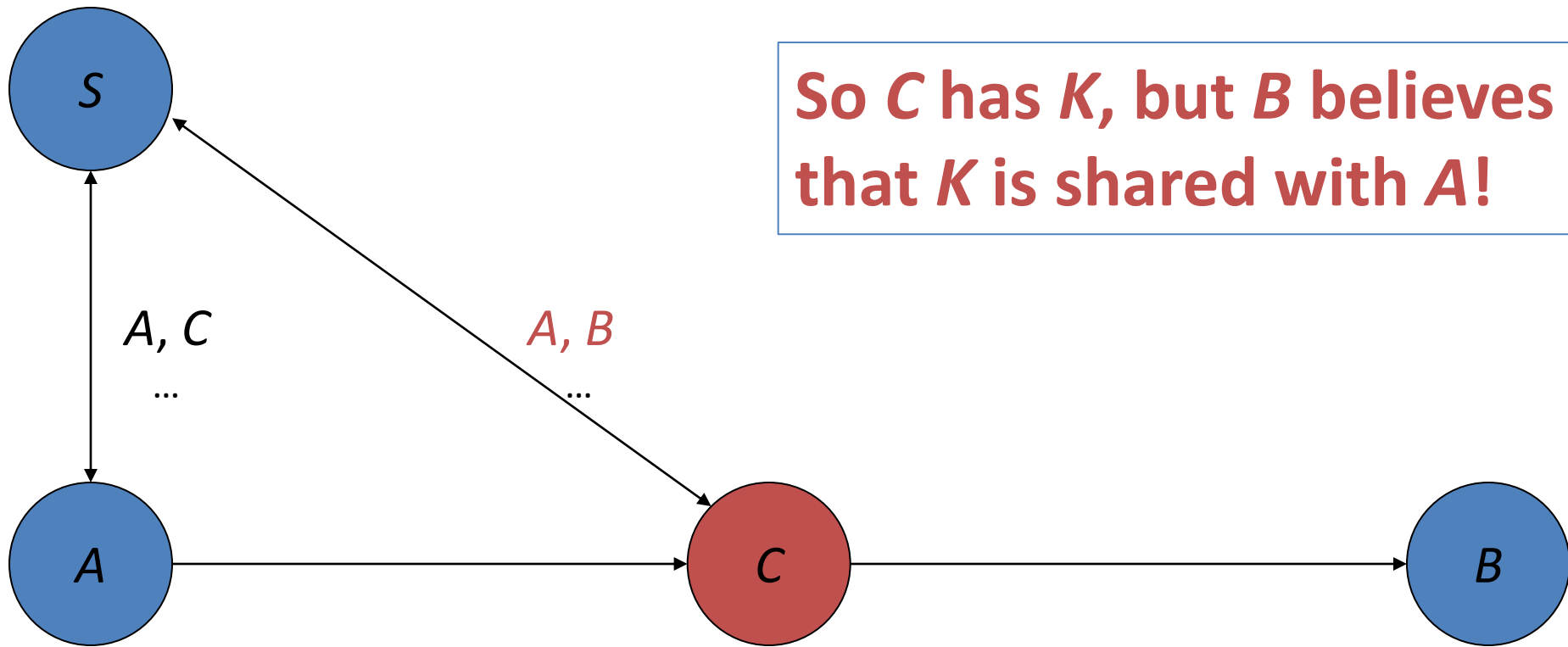
# An attack



3.  $\text{encrypt}(PK_C, \text{sign}(SK_A, (K, T)))$

3.  $\text{encrypt}(PK_B, \text{sign}(SK_A, (K, T)))$

# An attack



3.  $\text{encrypt}(PK_C, \text{sign}(SK_A, (K, T)))$

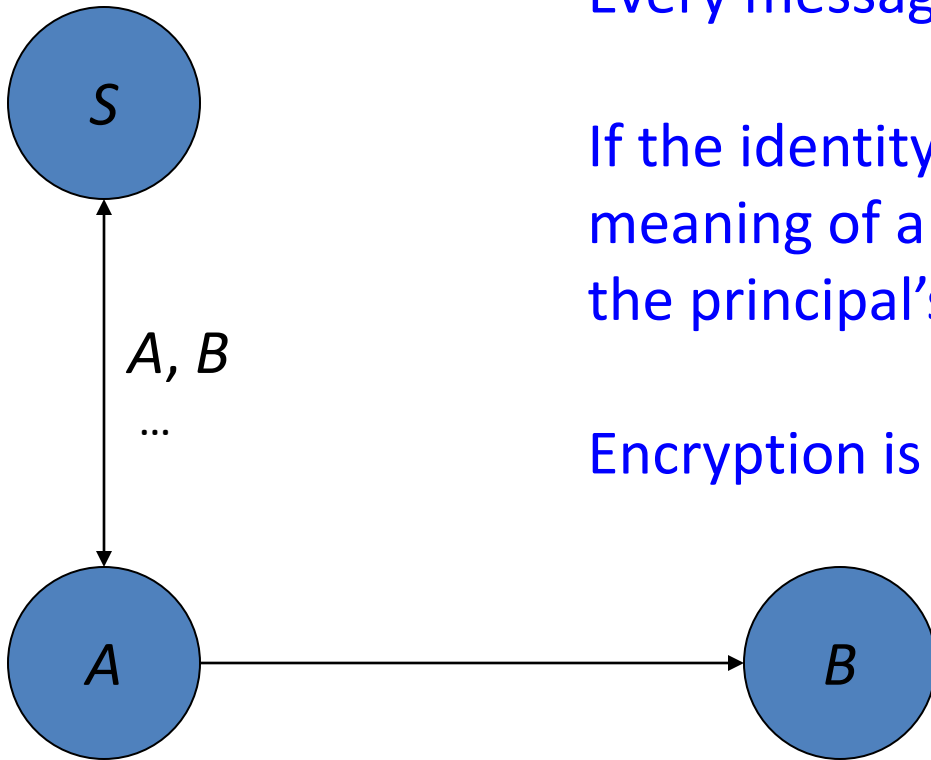
3.  $\text{encrypt}(PK_B, \text{sign}(SK_A, (K, T)))$

# Correction [with Needham]

Every message should say what it means.

If the identity of a principal is important for the meaning of a message, it is prudent to mention the principal's name explicitly in the message.

Encryption is not synonymous with security.




3.  $\text{encrypt}(PK_B, \text{sign}(SK_A, (\text{"K is a good key for A and B at time T"})))$

# Other subtleties and flaws


*There are many!*

- even in recent years,
- in both design and implementation.



**US-CERT**  
UNITED STATES COMPUTER EMERGENCY READINESS TEAM

[Vulnerability Notes Database](#) **Vulnerability Note VU#612636**  
**Google SAML Single Sign on vulnerability**

Tech titans meet in secret to plug SSL hole  
**Web authentication busted on Apache, IIS**  
By [Dan Goodin in San Francisco](#)   
Posted in [Security](#), 5th November 2009 07:51 GMT

[TechNet Home](#) > [TechNet Security](#) > [Bulletins](#)

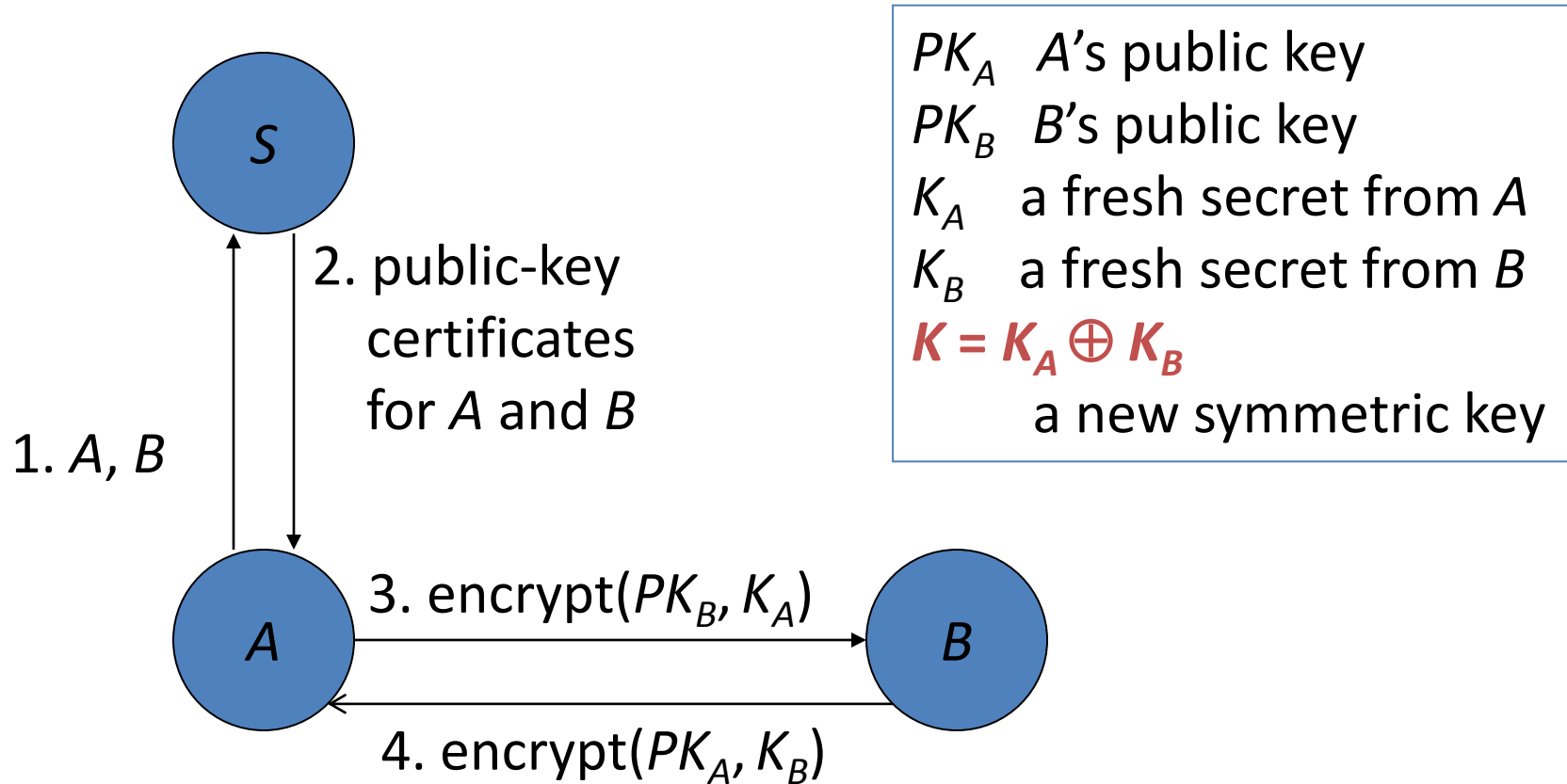
## Microsoft Security Bulletin MS10-070 - Important

Vulnerability in ASP.NET Could Allow Information Disclosure (2418042)

Published: September 28, 2010 | Updated: November 03, 2010

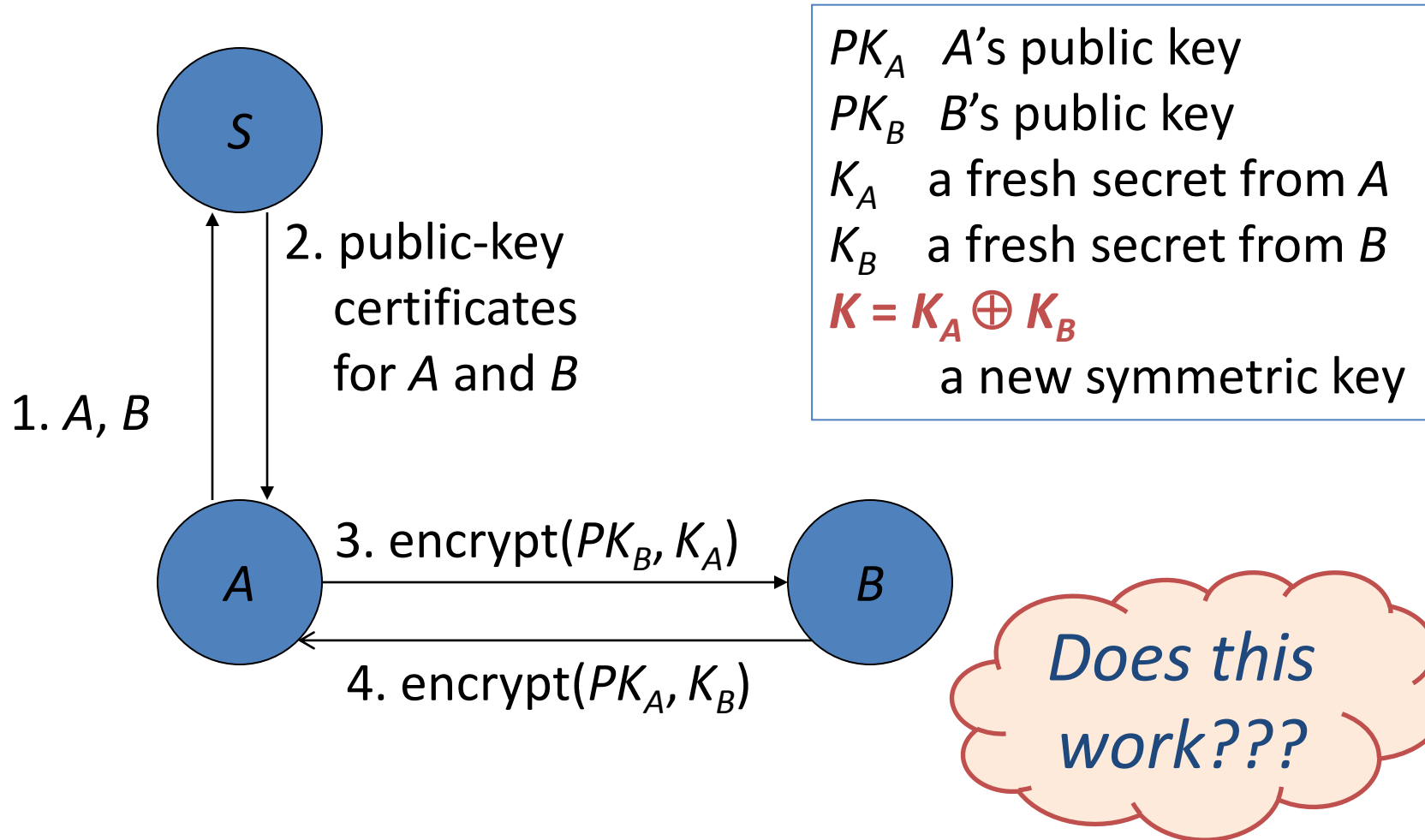


# Yet another protocol



As usual,  $K$  can then be used for encrypting and MACing payloads.

# Yet another protocol



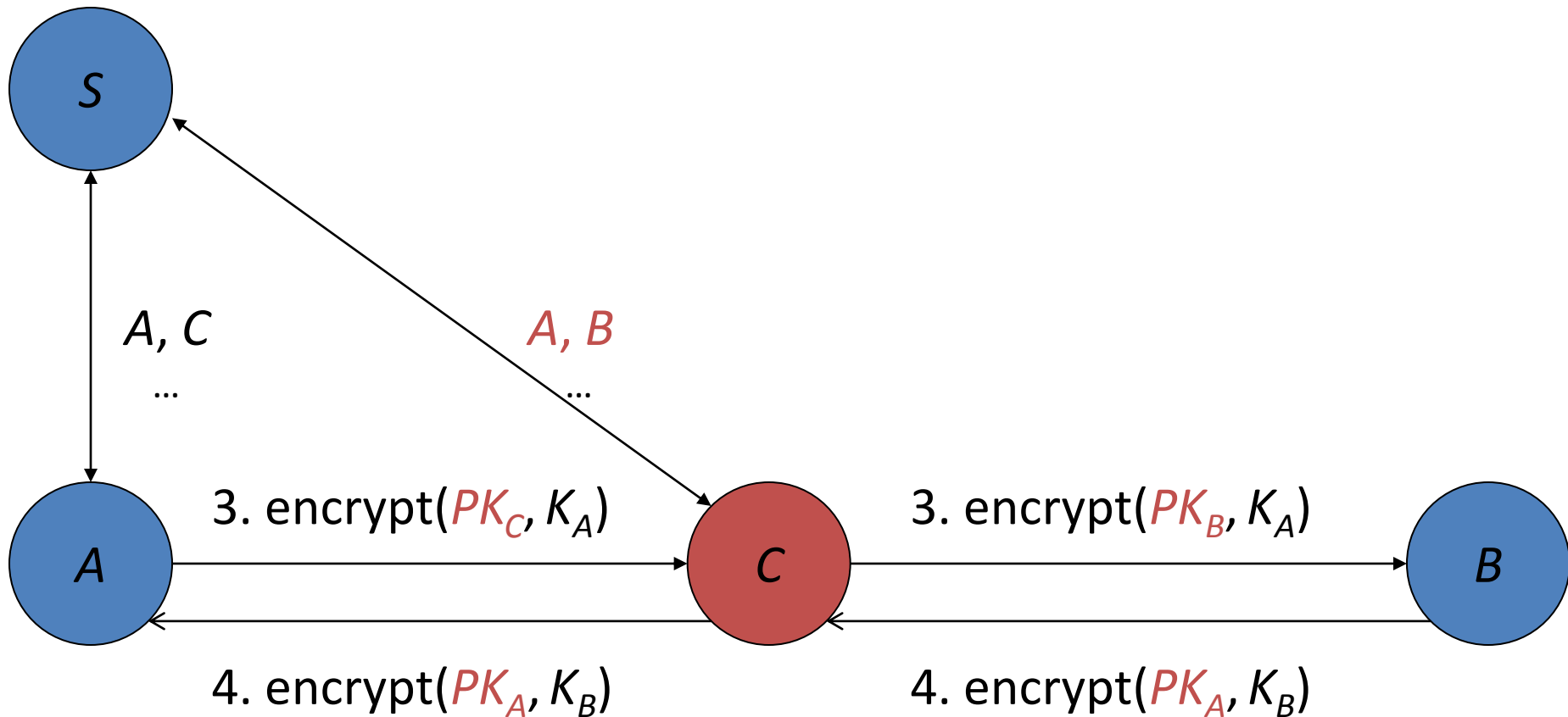
As usual,  $K$  can then be used for encrypting and MACing payloads.

# An informal analysis

*If A follows the protocol then she is assured that the shared key [. . . ] is not know to anyone except B (though A does not have the assurance that B knows the key). And analogously for B.*

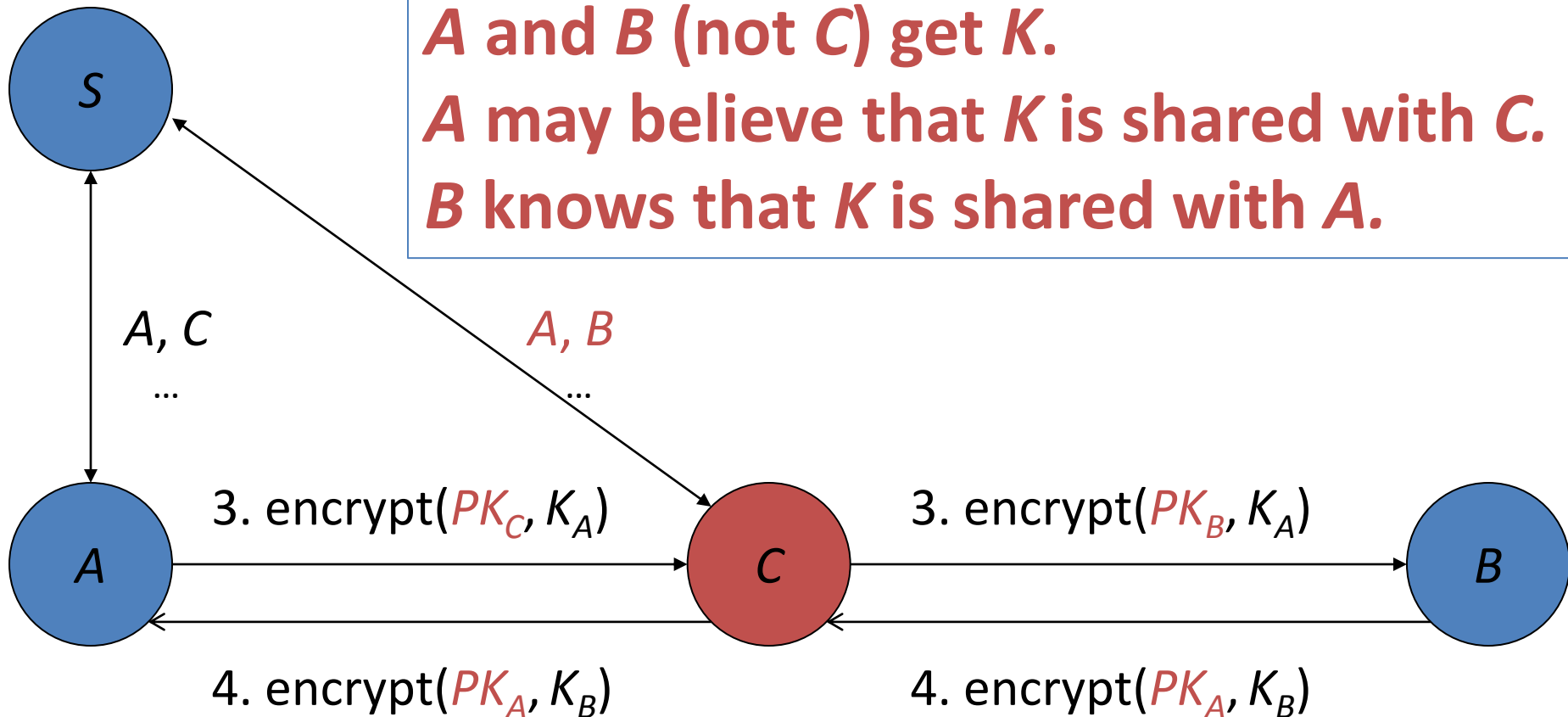
(H. Krawczyk)

# An attack?



# An attack?

*A and B (not C) get K.  
A may believe that K is shared with C.  
B knows that K is shared with A.*



## An attack? (cont.)

***A and B (not C) get K.***

***A may believe that K is shared with C.***

***B knows that K is shared with A.***

So:

- A should not give *credit* to C for messages received under K.
- C has *responsibility* for messages to A under K.
- If A sends confidential data to C under K, then B will see it, but C could divulge it to B anyway.

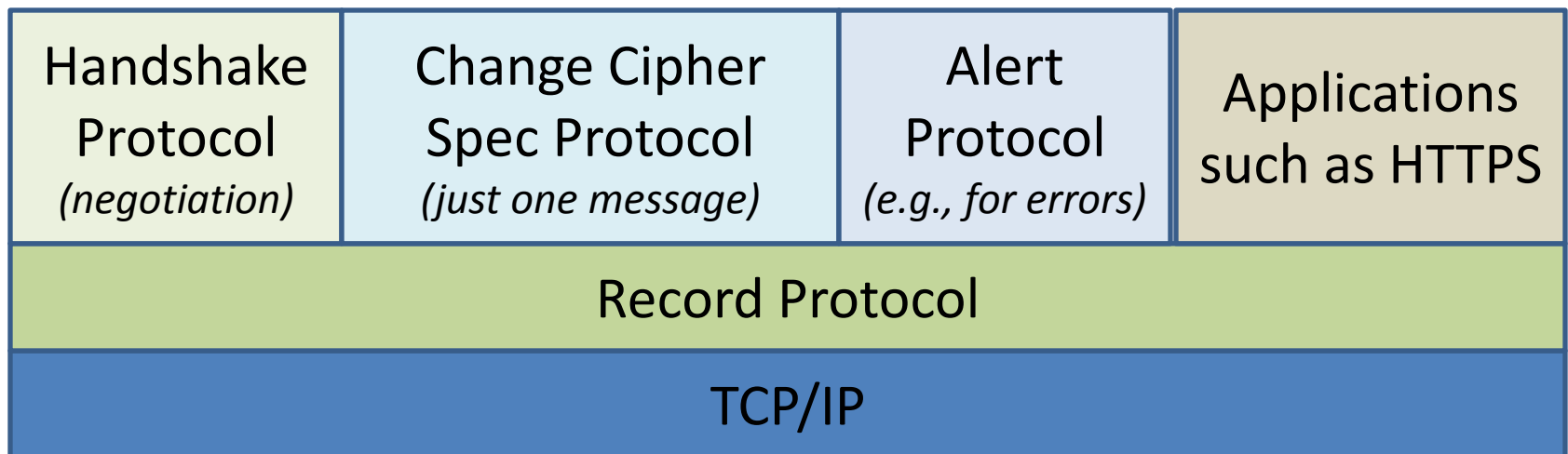
# Perspectives

Authentication may yield responsibility, credit, or both.

- *Responsibility is essential.*
  - It is the basis of access control.
  - It is compatible with delegation. If *C* says that *B* speaks for *C*, then *B* does speak for *C*.
- *Credit may be optional.*
  - It can be left for higher-level communication.
  - But establishing credit may contribute to robustness.     *(Exercise: strengthen the protocol!)*

# An important modern example: Secure Socket Layer (SSL)

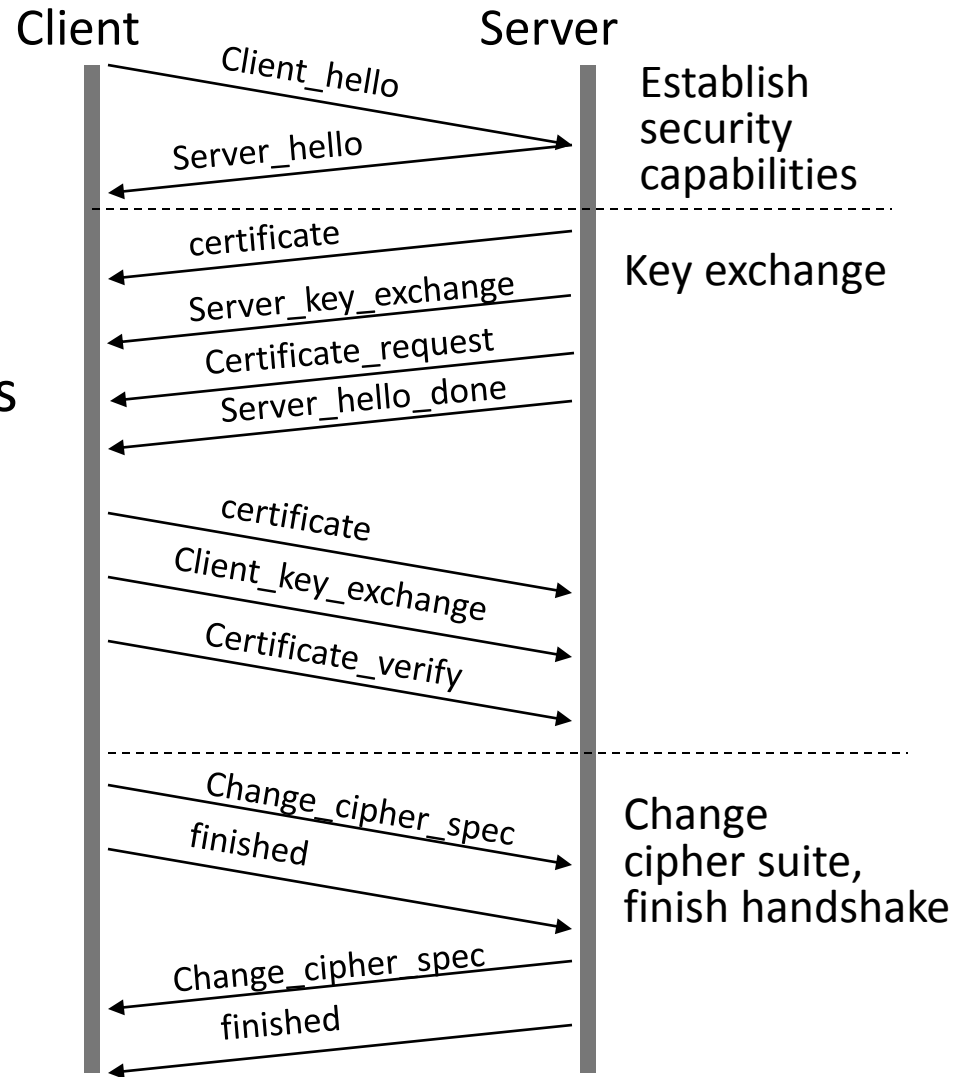
- SSL relies on TCP/IP and aims to provide secure end-to-end communication.
- SSL actually includes two layers of protocols:
  - SSL Record Protocol for transport,
  - higher protocols for negotiation and alerts.



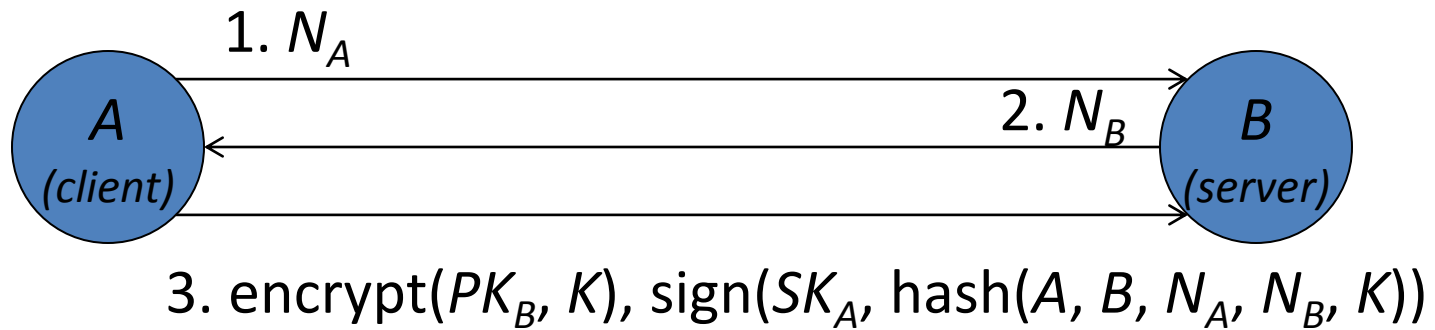


# The handshake

1. Establish security capabilities
2. Authenticate server
3. Authenticate client
4. Finish



# A piece of the handshake (simplified)



*K: the **premaster secret** from which a **master secret** is derived, and later encryption and MAC keys are also derived.*

***Exercise:** What happens if various fields (e.g., A, B, ...) are omitted in the hash of the third message? (as they were originally!)*

# Sessions and connections

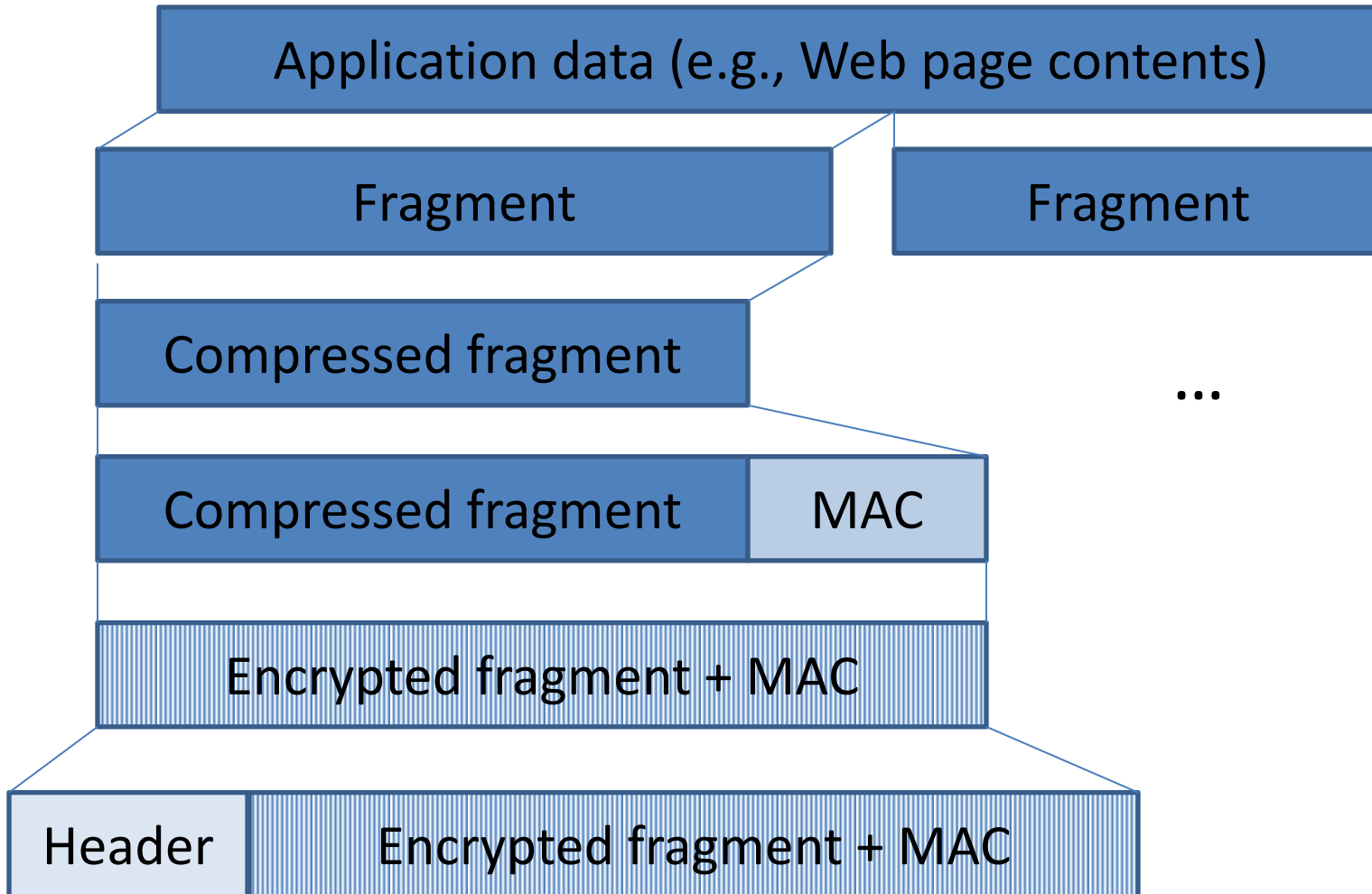
***SSL session:*** a client-server association.

Peer certificate	X509.v3 certificate of the peer (may be null)
Compression method	Compression algorithm
Cipher spec	Cryptographic algorithms and parameters
Master secret	48-byte secret shared by client and server
Is resumable?	Can new connections start with session?

***SSL connection:*** a stream within a session.

Server and client random	Byte sequences chosen by server and client
Server and client write MAC secrets	Keys used in MAC operations
Server and client write keys	Encryption keys
Initialization vectors	Initialization vectors for CBC encryption
Sequence numbers	Maintained by each party for each direction

# SSL Record Protocol



# SSL status

- Evolution and deployment:
  - various improvements in the TLS protocol,
  - widespread deployment,
  - but typically without client authentication
    - ⇒ so users still rely on passwords, etc.,
  - and with users checking server certificates
    - ⇒ so users are victims of phishing attacks.
  - ⇒ *Still a challenge:* strong mutual authentication in practice.

# SSL status (cont.)

- Analysis:
  - several informal analyses;
  - automated formal proofs of secrecy and integrity properties (for big fragments);
  - some vulnerabilities found over time.
- ⇒ *Still a challenge: a complete analysis, of actual implementations, from solid cryptographic assumptions.*

# Other concerns

- SSL slows down servers.
- SSL breaks caching and complicates virtual hosting (multiple identities for the same host).
- SSL protects data in transit, but not at rest.

# From SSL to SSL Double Layer

*tous les fichiers étaient encryptés – un code sérieux, SSL Double Layer, 128 bits. Bref, j'ai rien pu faire, je l'ai envoyé à la BEFTI. C'était quoi le type, un parano ?*

*(de La carte et le territoire de M. Houellebecq)*

There is no SSL Double Layer (not yet), and it would probably not be a storage technique.

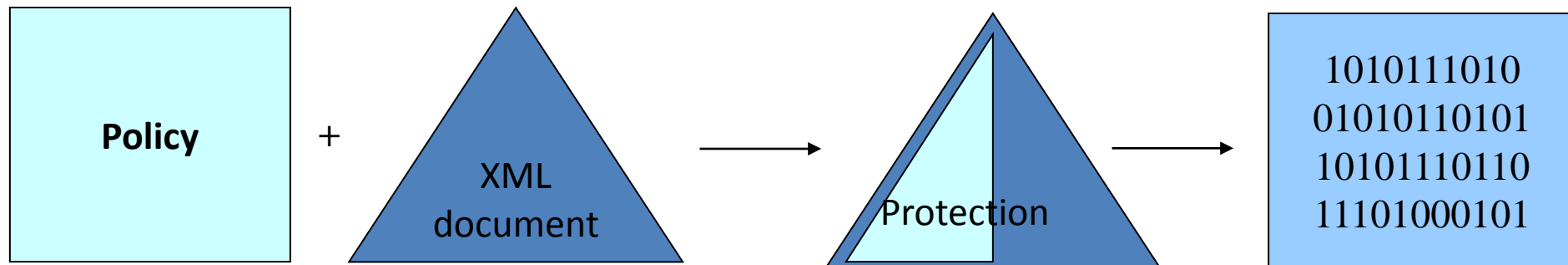
However, secure storage and publication do rely on some of the same ideas as security protocols.



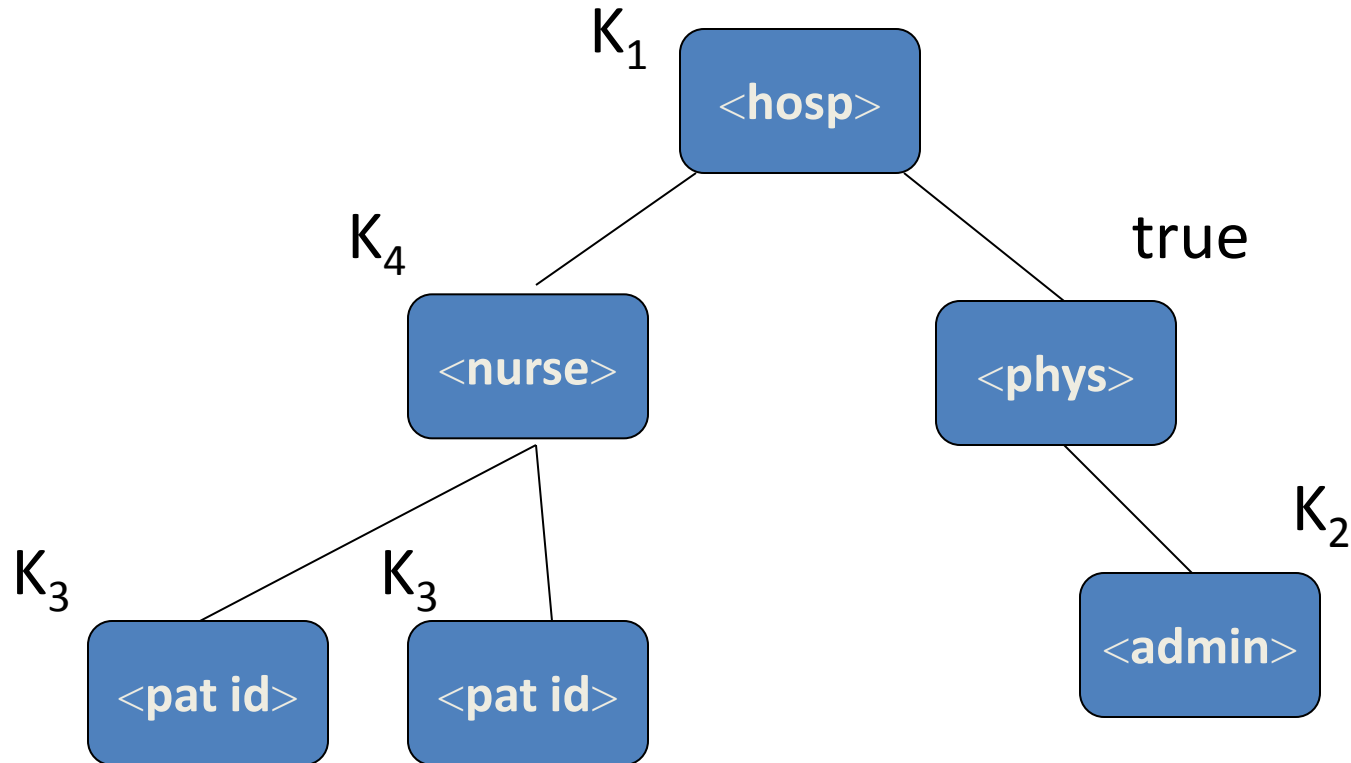
# An example: XML access control

[Miklau and Suciu]

- A policy language
- A mapping of policies to annotated XML documents called *protections*
- A cryptographic implementation of protections in terms of bitstrings



# A protection

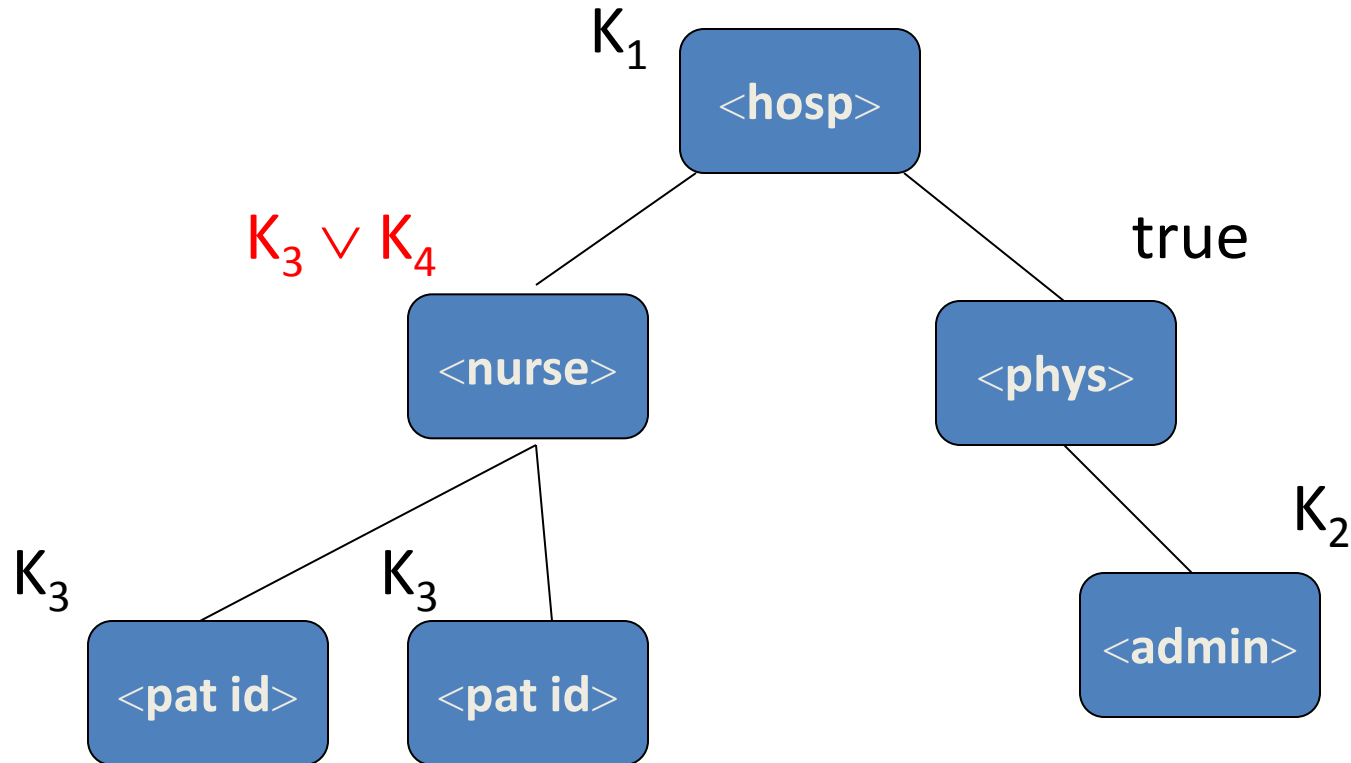


**Intended semantics:** Access to the information in a node requires possession of the keys that guard the node.

(Here the label “true” means no key.)

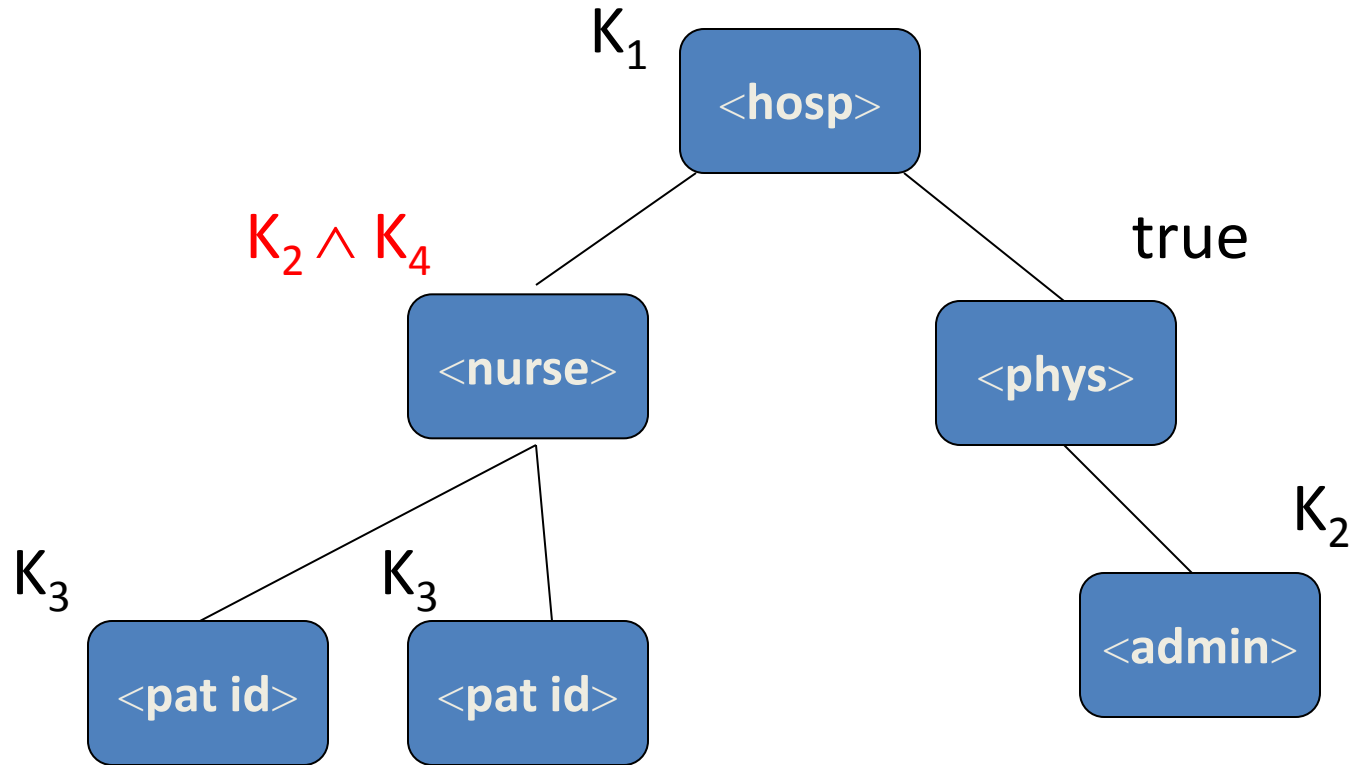
**Implementation:** By symmetric encryptions.

# A protection with disjunction ( $\vee$ )



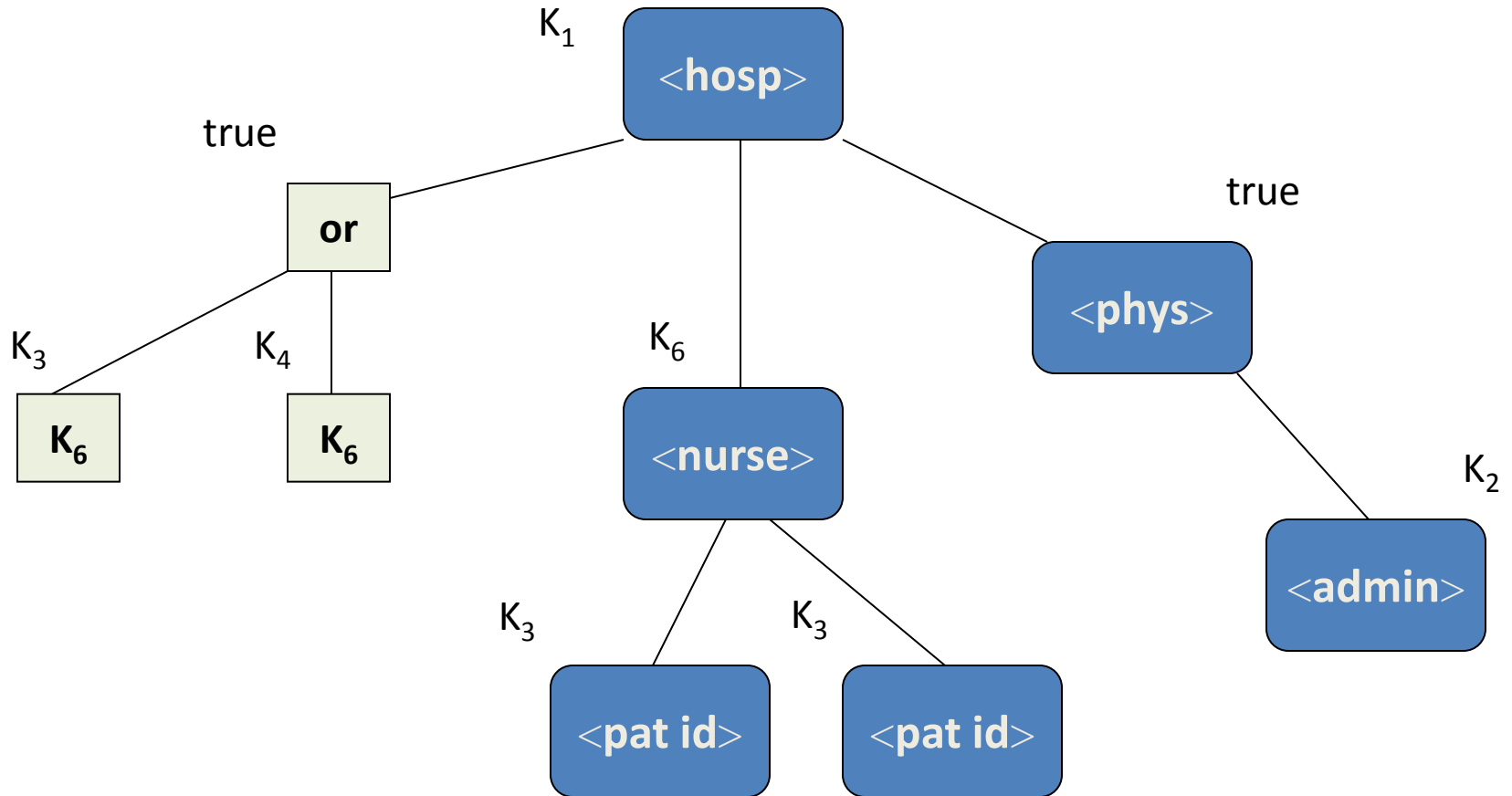
**Intended semantics:** Access to the information in a node requires possession of a combination of keys that satisfy the formulas that guard the node.

# A protection with conjunction ( $\wedge$ )



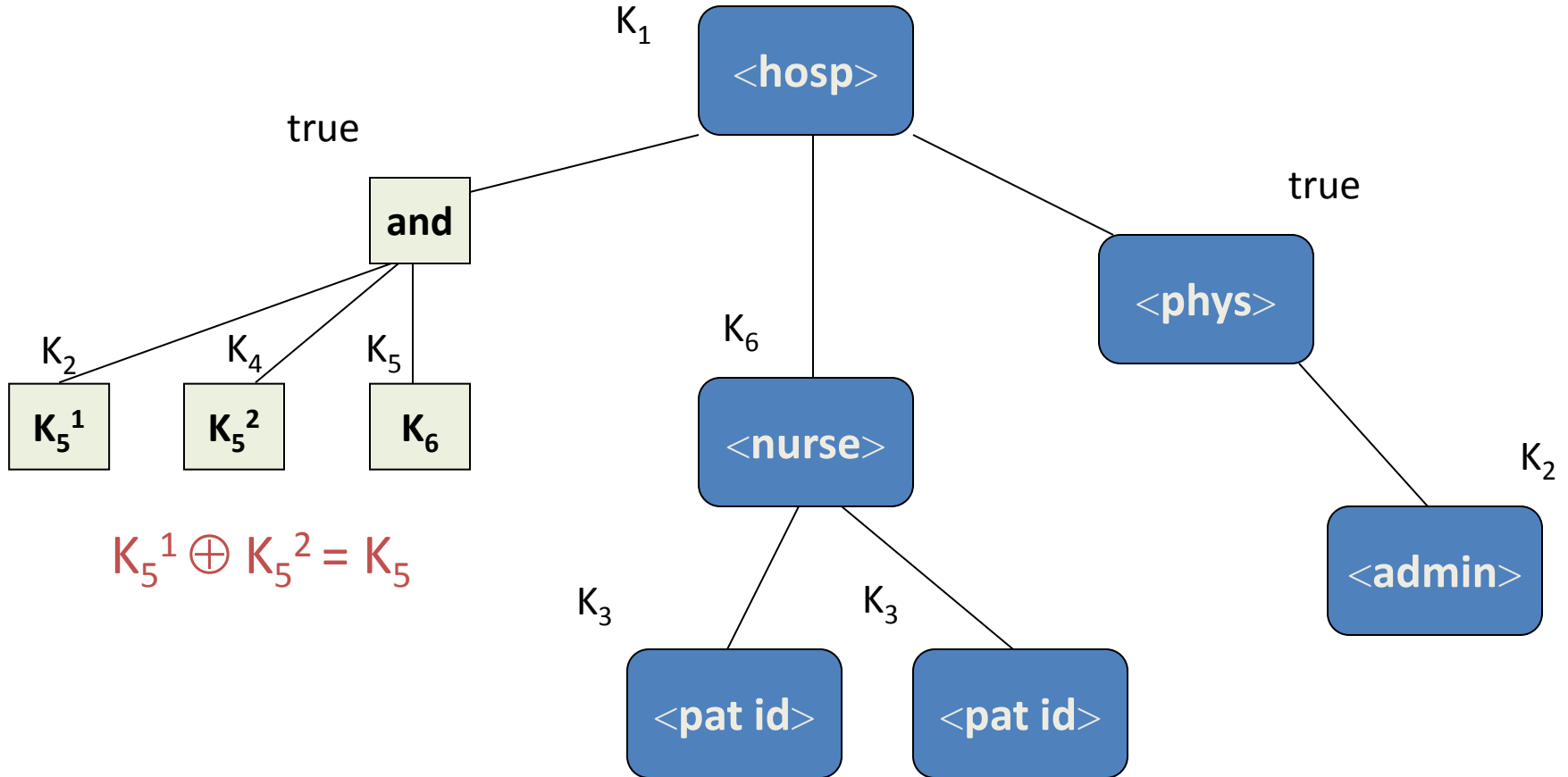
**Intended semantics:** Access to the information in a node requires possession of a combination of keys that satisfy the formulas that guard the node.

# Implementing disjunctions



∨ is implemented using auxiliary keys,  
so that only atomic formulas guard each node.

# Implementing conjunctions



$\wedge$  is implemented using secret sharing,  
so that only atomic formulas guard each node.

*Protocols for  
secure multiparty computation*

# Other security protocols

- Payment
- Voting
- Multi-party computation
- ...



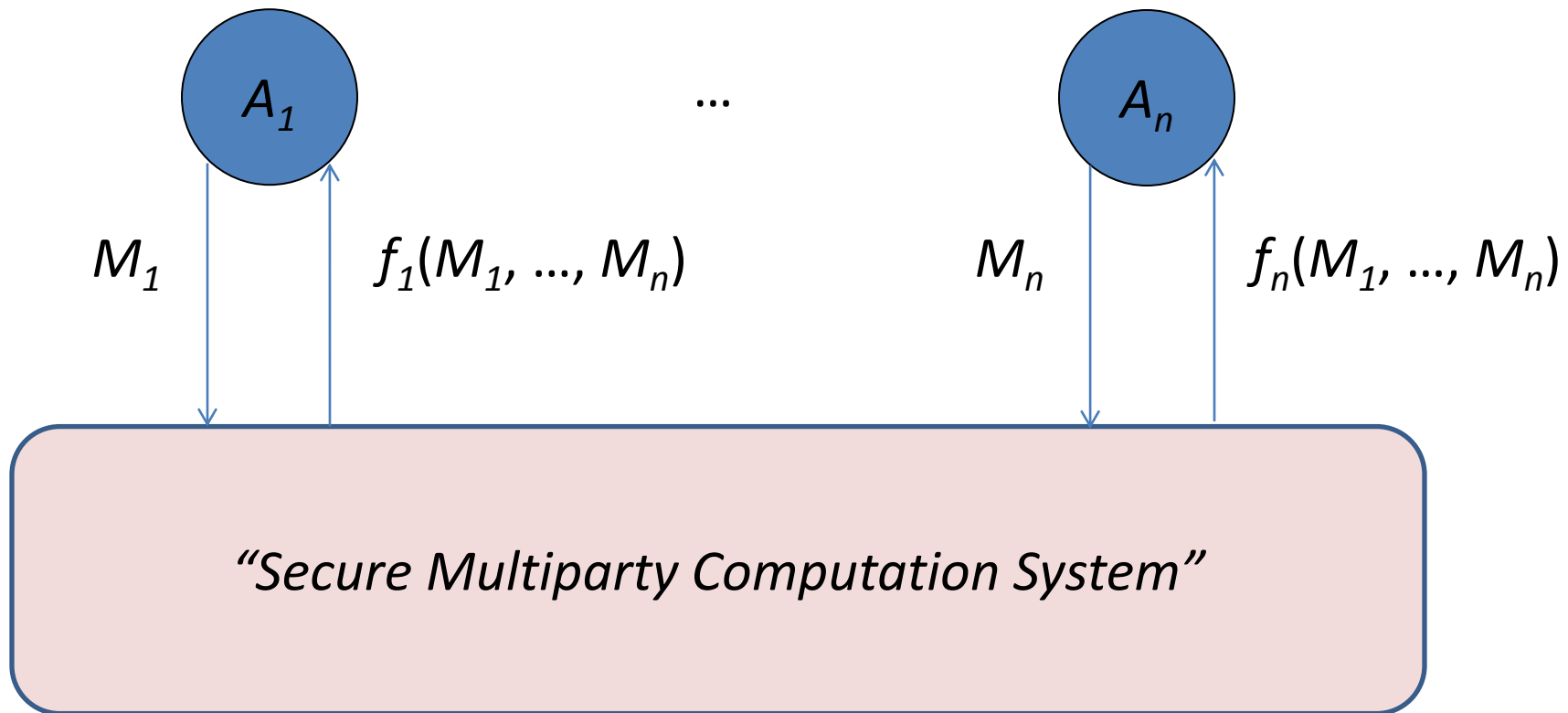
# Secure multiparty computation

## The problem:

How can  $A_1, \dots, A_n$ ,  
who know  $M_1, \dots, M_n$  respectively,  
cooperate to compute functions of  $M_1, \dots, M_n$   
and share the results  
without revealing their inputs or anything else?

*We assume that  $A_1, \dots, A_n$  don't lie on  $M_1, \dots, M_n$ .*

In general, each participant may learn the output of a different function  $f_1, \dots, f_n$ :



*The problem is **trivial with a trusted third party.**  
The difficulty is to solve it otherwise!*

# Examples

- $A$  and  $B$  are two millionaires who each know their own wealth.
- They want to know who is richer.

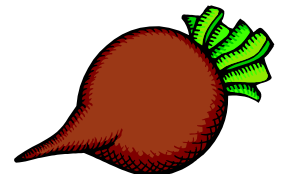
[Yao, 1982]



# Examples (cont.)

- Each of  $A_1, \dots, A_n$  wants to buy or sell sugar beets, and knows how much it is willing to pay/charge for various quantities.
- Collectively,  $A_1, \dots, A_n$  want to compute the “market clearing price”.

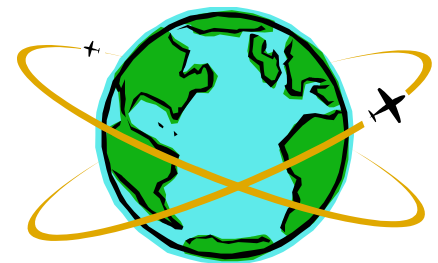
[Bogetoft et al., 2008]



# Examples (cont.)

- *A* and *B* each knows their own location.
- They want to know if they are in the same location (or how far they are).

[Narayanan et al., 2011]



A physical, in-person protocol for checking equality in a small set [Ajtai]

A physical, in-person protocol for checking equality in a small set [Ajtai]



# A physical, in-person protocol for checking equality in a small set [Ajtai]

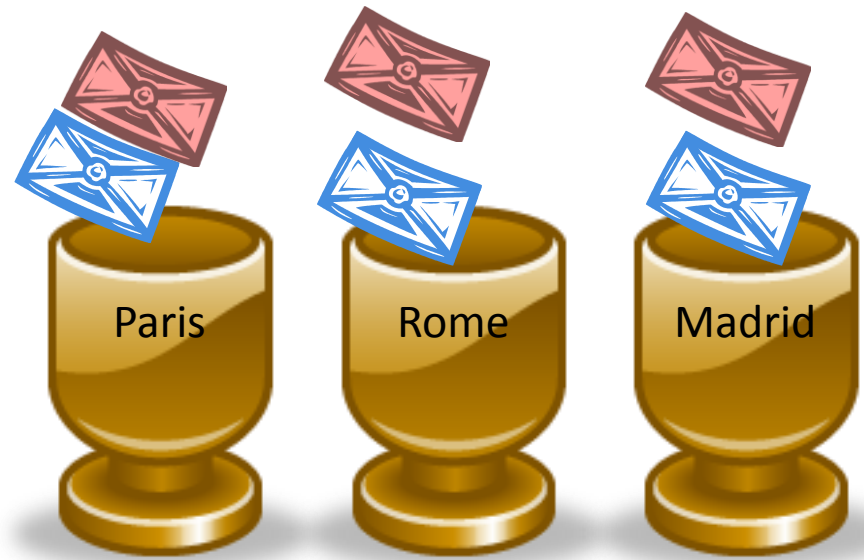
1. Label cups with the names of the locations, say Paris, Rome, and Madrid.





# A physical, in-person protocol for checking equality in a small set [Ajtai]

1. Label cups with the names of the locations, say Paris, Rome, and Madrid.
2. *A* and *B* each put an envelope with a piece of paper that says “yes” or “no” in each cup.



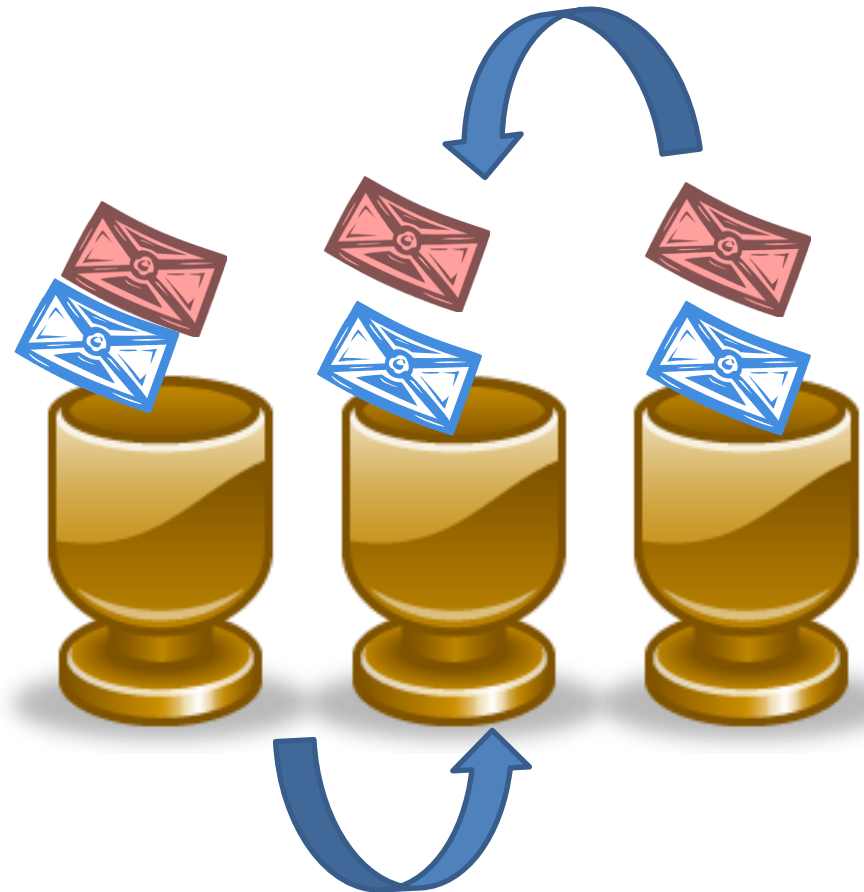
# A physical, in-person protocol for checking equality in a small set [Ajtai]

1. Label cups with the names of the locations, say Paris, Rome, and Madrid.
2. *A* and *B* each put an envelope with a piece of paper that says “yes” or “no” in each cup.
3. They remove the labels.



# A physical, in-person protocol for checking equality in a small set [Ajtai]

1. Label cups with the names of the locations, say Paris, Rome, and Madrid.
2. *A* and *B* each put an envelope with a piece of paper that says “yes” or “no” in each cup.
3. They remove the labels.
4. They shuffle the cups.



# A physical, in-person protocol for checking equality in a small set [Ajtai]

1. Label cups with the names of the locations, say Paris, Rome, and Madrid.
2. *A* and *B* each put an envelope with a piece of paper that says “yes” or “no” in each cup.
3. They remove the labels.
4. They shuffle the cups.
5. They open the envelopes.



# A physical, in-person protocol for checking equality in a small set [Ajtai]

1. Label cups with the names of the locations, say Paris, Rome, and Madrid.
2. A and B each put an envelope with a piece of paper that says “yes” or “no” in each cup.
3. They remove the labels.
4. They shuffle the cups.
5. They open the envelopes.

They are in the same location if and only if one of the cups contains two “yeses”.



# A cryptographic protocol

[Narayanan et al.]

Recall Diffie-Hellman:

- Let  $p$  be a prime and  $g$  a generator of  $\mathbf{Z}_p^*$  (chosen with a little care).
- $A$  invents  $x$  and publishes  $g^x$ .  
 $B$  invents  $y$  and publishes  $g^y$ .
  - $x$  and  $y$  serve as secret keys.
  - $g^x$  and  $g^y$  serve as public keys.

All this is “mod  $p$ ” implicitly.

# A cryptographic protocol (cont.)

[Narayanan et al.]

- $A$  (at location  $a$ ) invents  $r$  and publishes  $(g^r, g^{x(a+r)})$ .
- $B$  (at location  $b$ ) invents  $s$  and  $t$  and publishes  $(g^{rs}g^t, g^{x(a+r)s}g^{x(t-sb)})$ , i.e.,  $(g^{rs+t}, g^{x(s(a-b) + (rs+t))})$
- From this and  $x$ ,  $A$  can compute  $g^{x(rs+t)}$  and then  $g^{xs(a-b)}$ .
- The locations are equal iff  $g^{xs(a-b)} = 1$ .

(Cf. “ElGamal encryption”.)

# Perspectives

- There are many clever, surprising protocols!
- Some of them explore trade-offs between generality and efficiency.
- There are (still) few actual applications for cryptographic multi-party computation.



# Some reading

- “Using Encryption for Authentication in Large Networks of Computers”, by Needham and Schroeder.
- The TLS protocol, as an example, from <http://datatracker.ietf.org/wg/tls/>.
- “Comparing Information Without Leaking It”, by Fagin, Naor, and Winkler.