Introduction to Security Protocols

Ton van Deursen

Goals

This lecture will teach you the . . .

- Basics of security protocol analysis
- Classic examples
  (secrecy protocols, Needham-Schroeder protocol)
- Standard terminology
  (Dolev-Yao adversary, perfect cryptography)
- Currently active research areas

Protocols

A protocol is a series of steps carried out by two or more entities.

Ex: HTTP, TCP, SMTP

Protocols

A protocol is a series of steps carried out by two or more entities.

Ex: HTTP, TCP, SMTP

A security protocol is a protocol that runs in an untrusted environment and tries to achieve a security goal.

Academic examples	Industrial examples
Needham-Schroeder-Lowe	Kerberos
Diffie Hellman key exchange	SSL/TLS
PAKE	IPSec

Protocols

Communication protocols usually assume:

- Trusted channels: No hostile agents have access to the communication medium to interfere with the protocol.
- Honest participants: All agents participate in the protocol cooperate to achieve the protocol goal.

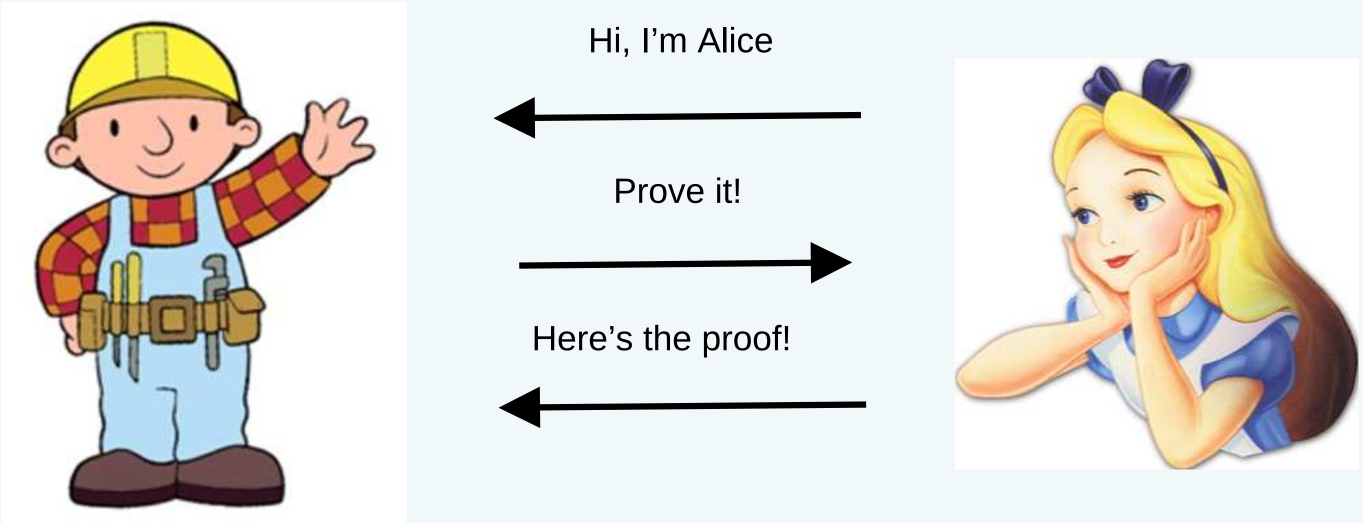
Security protocols

Security protocols usually assume:

- Untrusted channels: Hostile agents that do not participate in the protocol have access to the communication medium.
- Dishonest participants: Hostile agents that claim to be honest, but try to prevent the security goals from being achieved.

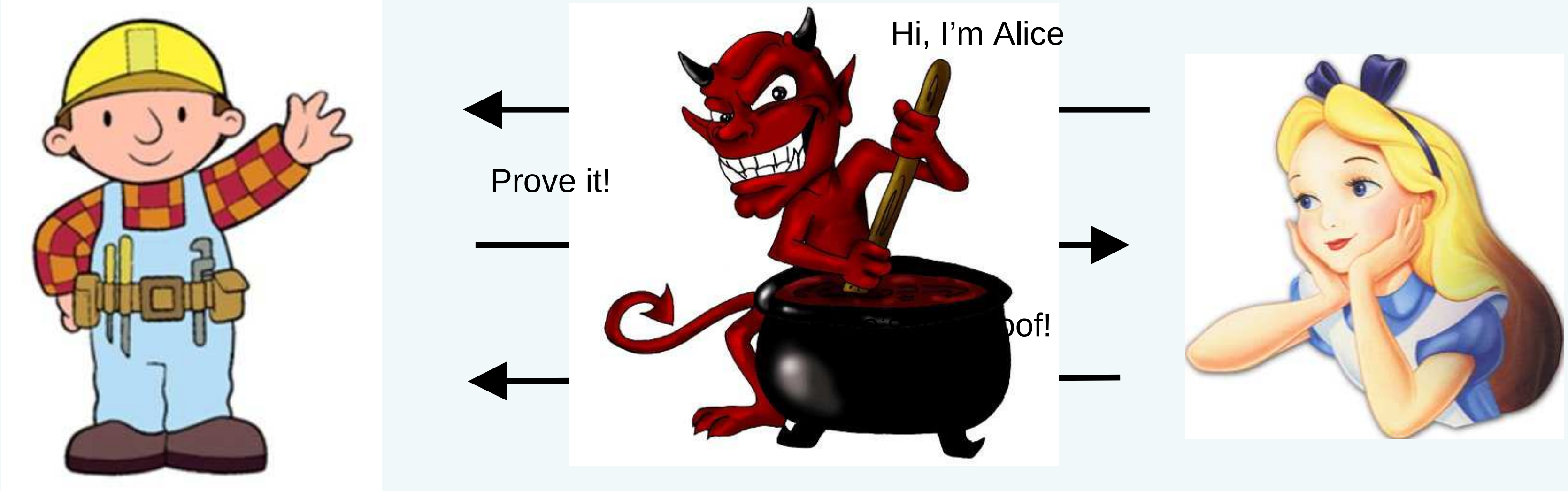
Security protocols are about Alice and Bob.

Security protocols are about Alice and Bob.



Their enemy is called Eve.

Their enemy is called Eve.



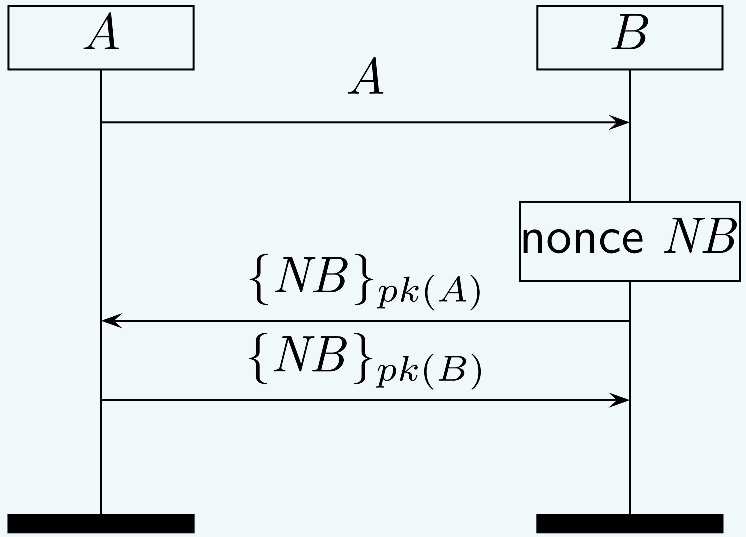
Alice & Bob notation

$A \longrightarrow B$: A

$B \longrightarrow A$: $\{NB\}_{pk(A)}$

$A \longrightarrow B$: $\{NB\}_{pk(B)}$

Message sequence charts



Protocol verification

Computational verification model (simulation based)

- Manual proofs of security properties in an ad-hoc proof model.
- Proof models are often game based.
- Proofs are done by reduction to known hard problems (e.g. discrete log, factorization).

Symbolic verification model (formal-methods based)

- Find attacks by exhibiting all possible behavior (traces) of a protocol.
- Can be combined with other techniques for correctness proofs.

Protocol verification

Computational verification model (simulation based)

- Manual proofs of security properties in an ad-hoc proof model.
- Proof models are often game based.
- Proofs are done by reduction to known hard problems (e.g. discrete log, factorization).

Symbolic verification model (formal-methods based)

- Find attacks by exhibiting all possible behavior (traces) of a protocol.
- Can be combined with other techniques for correctness proofs.

Symbolic verification

Symbolic verification . . .

- abstracts away from cryptographic details: cryptography is perfect.
- does not consider implementation flaws or side-channel attacks.

Symbolic verification

Symbolic verification . . .

- abstracts away from cryptographic details: cryptography is perfect.
- does not consider implementation flaws or side-channel attacks.

Needham-Schroeder protocol:

- Designed in 1978.
- Proven correct/secure in 1989.
- Flaw found in 1996.

Protocol specification

An agent is an entity capable of action, such as a computer or a process.

A role is a specification of the behavior of an agent.

A protocol is a collection of zero or more roles.

Messages

Messages are constructed by a term algebra:

$A \subset Term$			Atomic messages
$x, y \in Term$	$\Rightarrow$	$(x, y) \in Term$	Pairing
$x \in Term$	$\Rightarrow$	$h(x) \in Term$	Hashing
$x, y \in Term$	$\Rightarrow$	$\{x\}_y \in Term$	Encryption
$x \in Term$	$\Rightarrow$	$x^{-1} \in Term$	Inverse

Adversary

Adversary’s powers consist of:

- his ability to manipulate messages
- his ability to control the communication between agents

Inference

$K \vdash m$: "the message m can be derived from the knowledge set K".

$$
\begin{array}{lcl}
t \in K & \Rightarrow & K \vdash t \\
K \vdash t_1 \wedge K \vdash t_2 & \Rightarrow & K \vdash (t_1, t_2) \\
K \vdash (t_1, t_2) & \Rightarrow & K \vdash t_1 \wedge K \vdash t_2 \\
K \vdash t_1 \wedge K \vdash t_2 & \Rightarrow & K \vdash \{t_1\}_{t_2} \\
K \vdash \{t_1\}_{t_2} \wedge K \vdash t_2^{-1} & \Rightarrow & K \vdash t_1 \\
K \vdash t & \Rightarrow & K \vdash h(t)
\end{array}
$$

These rules satisfy the perfect cryptography assumption.

Dolev-Yao intruder

A Dolev-Yao intruder can . . .

- construct messages using the inference rules
- eavesdrop on messages
- delay or block messages
- forge and inject messages
- employ malicious agents in the system

Dolev-Yao intruder

A Dolev-Yao intruder can . . .

- construct messages using the inference rules
- eavesdrop on messages
- delay or block messages
- forge and inject messages
- employ malicious agents in the system

In essence, a Dolev-Yao intruder can do everything except breaking cryptography.

Secrecy protocols (1)

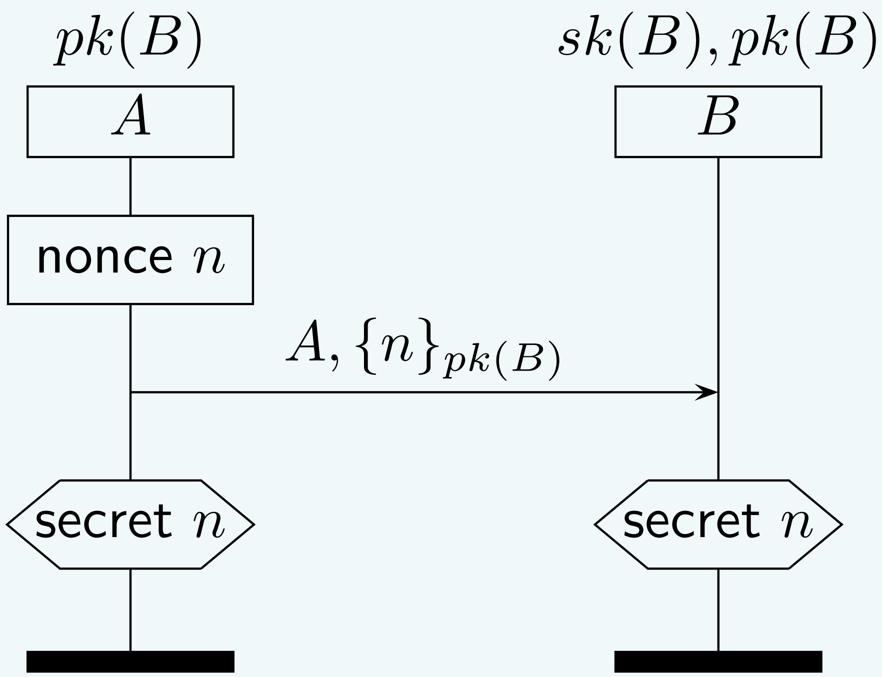
Definition (Secrecy)**.** *A term* t *is secret for an agent* A *in role* R *if and only if whenever* A *executes* R *and believes to be communicating with honest agents,* t *will not be inferable from the adversary's knowledge.*

Secrecy protocols (1)

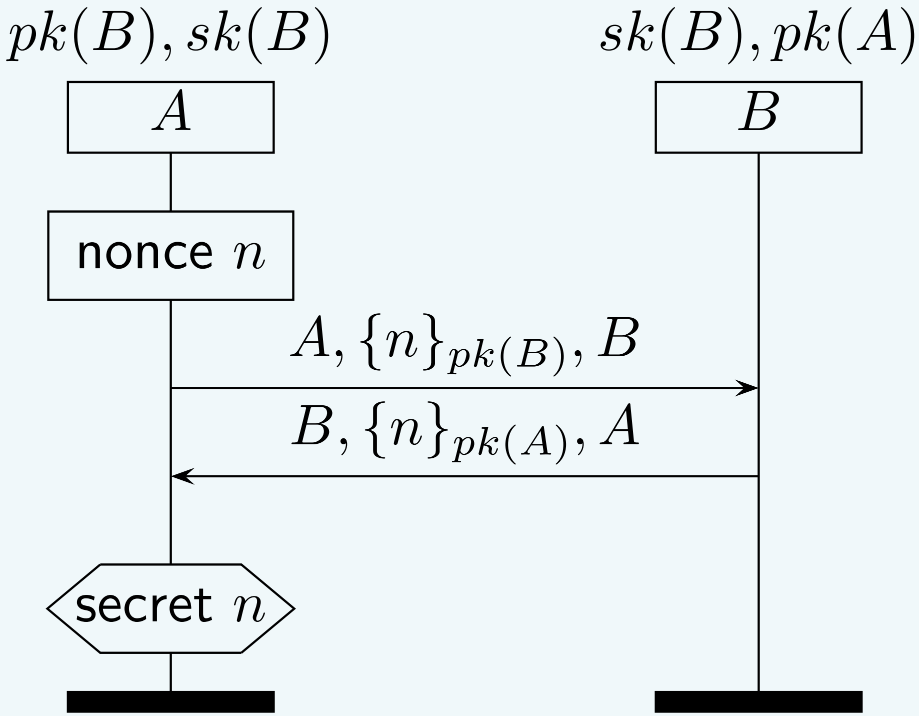
Definition (Secrecy)**.** *A term* t *is secret for an agent* A *in role* R *if and only if whenever* A *executes* R *and believes to be communicating with honest agents,* t *will not be inferable from the adversary's knowledge.*

- Secrecy is a *local* property.
- Secrecy can only be considered in case all *alleged* communication partners are honest.

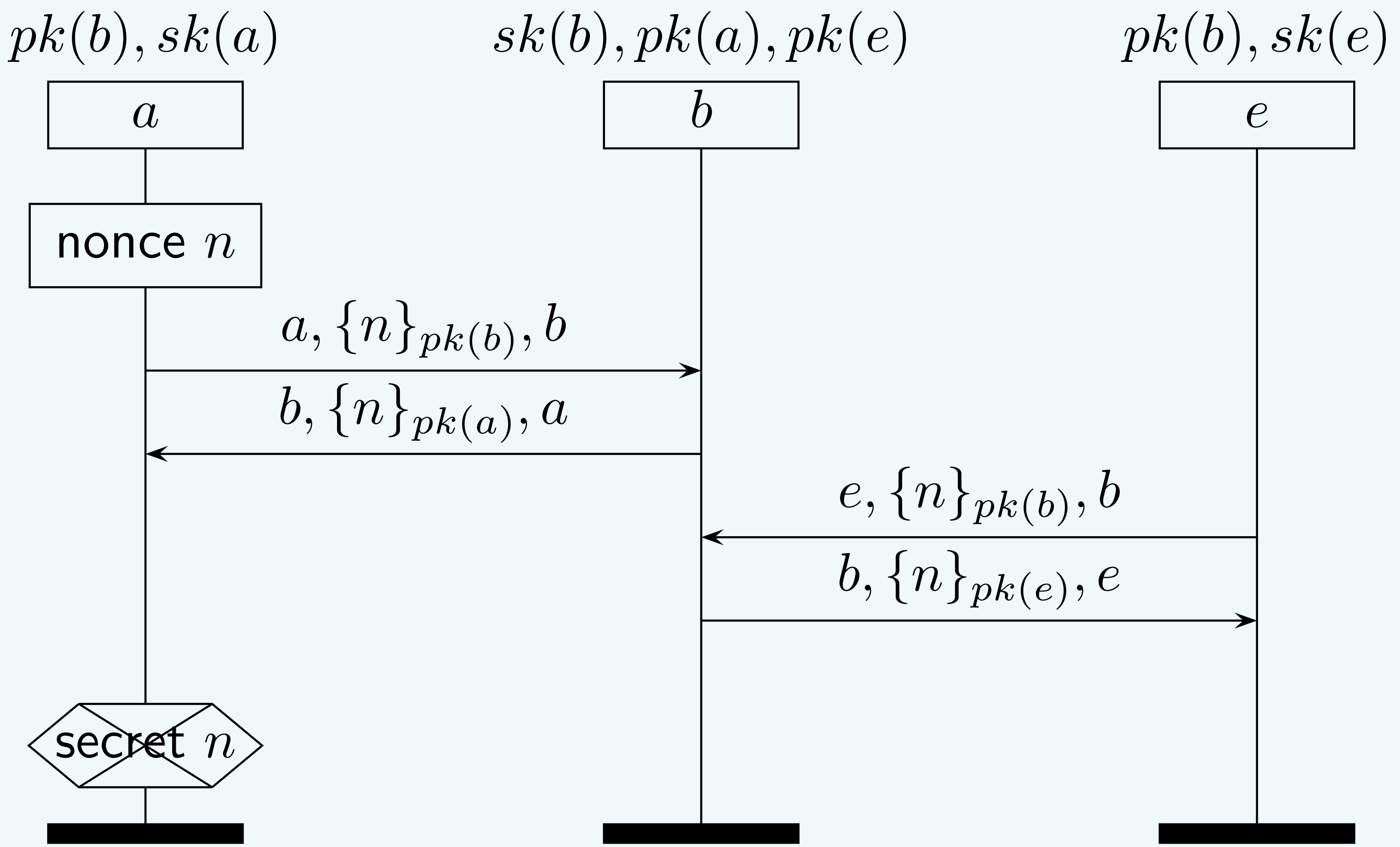
Secrecy protocols (2)



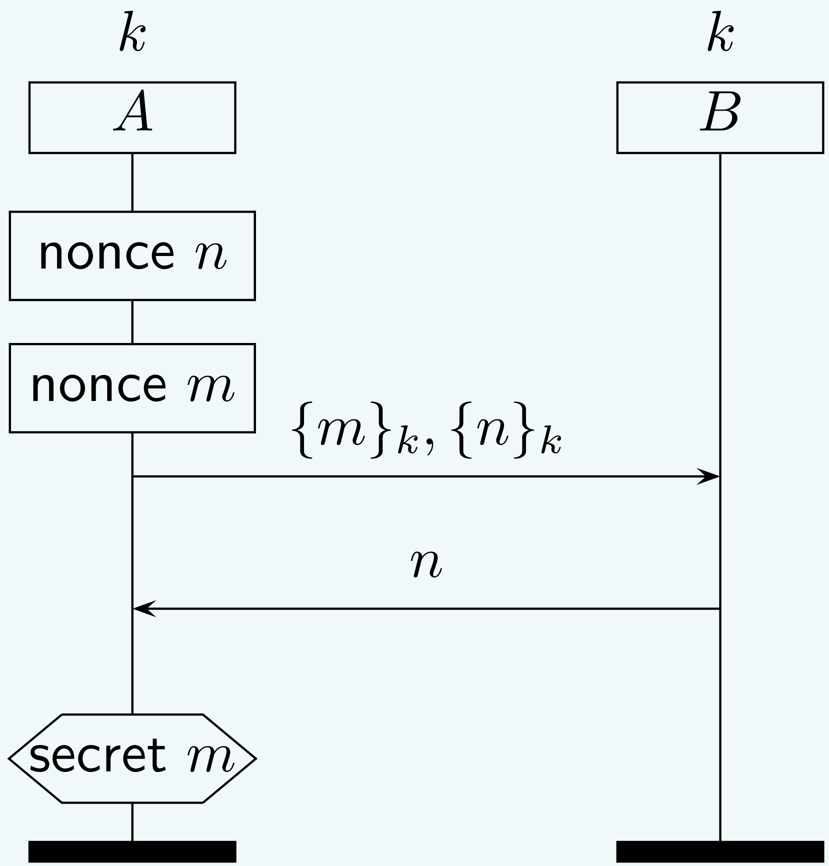
Secrecy protocols (3)



Secrecy protocols (4)

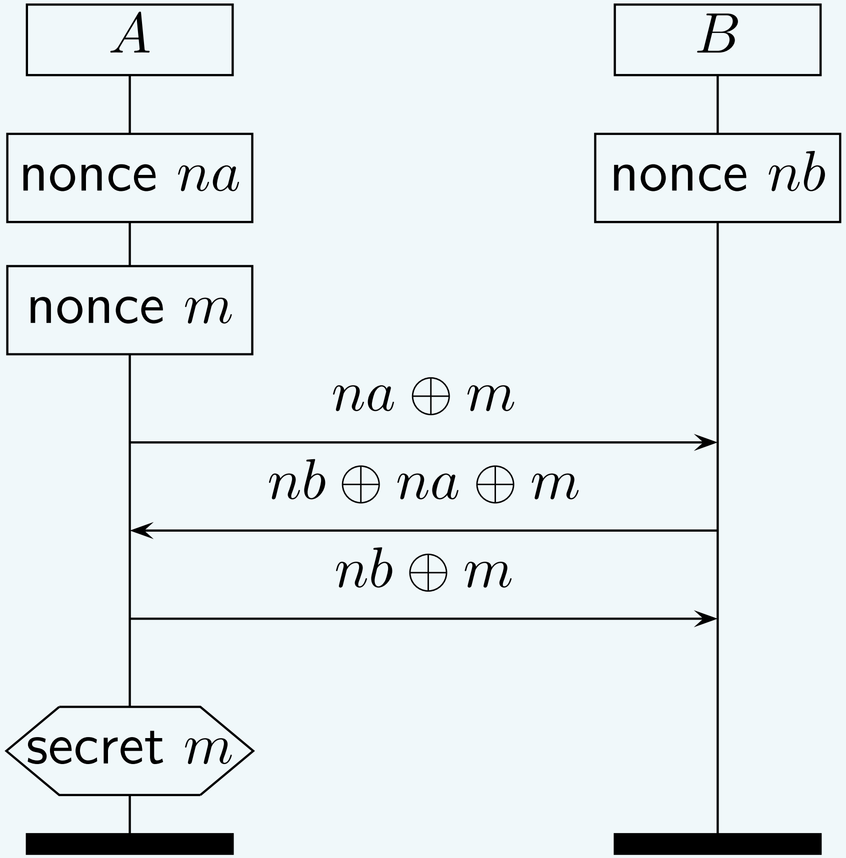


Exercise: find an attack on the secrecy claim in the following protocol:



Secrecy protocols (6)

Exercise: find an attack on the secrecy claim in the following protocol:

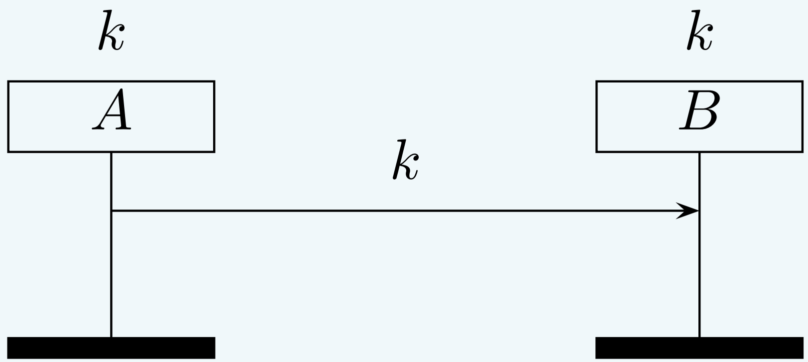


Authentication protocols (1)

An authentication protocol provides assurance of the identity of the communicating party in a protocol.

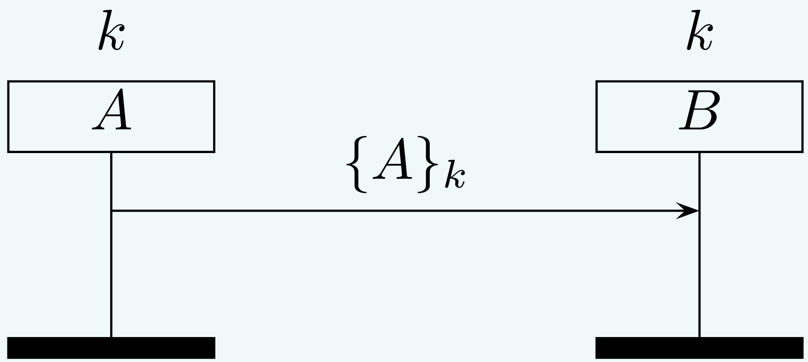
Authentication protocols (1)

An authentication protocol provides assurance of the identity of the communicating party in a protocol.

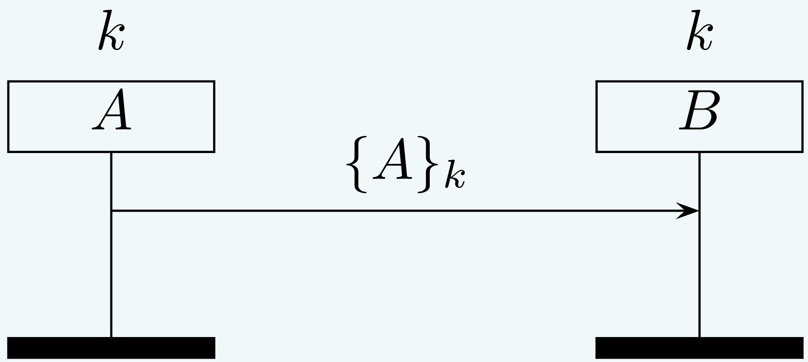


Authentication protocols (2)

An authentication protocol provides assurance of the identity of the communicating party in a protocol.



An authentication protocol provides assurance of the identity of the communicating party in a protocol.



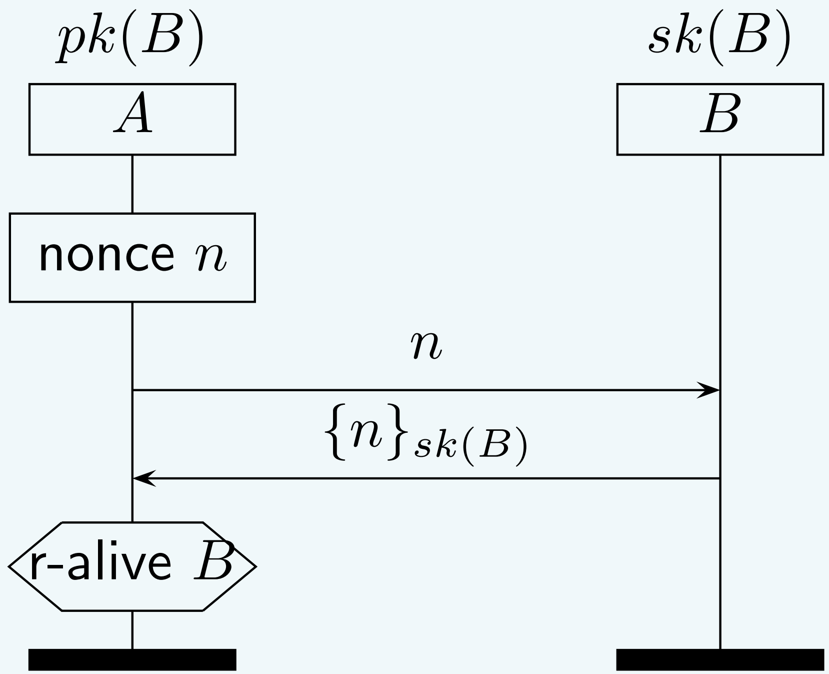
Definition. *(aliveness) A protocol guarantees to an agent a in role A aliveness of an agent b in role B if, whenever a completes a run of role A, believing to be communicating with b, then b has previously been running the protocol.*

Authentication protocols (3)

Protocols satisfying aliveness guarantee that the communicating party has sent a message in the past.

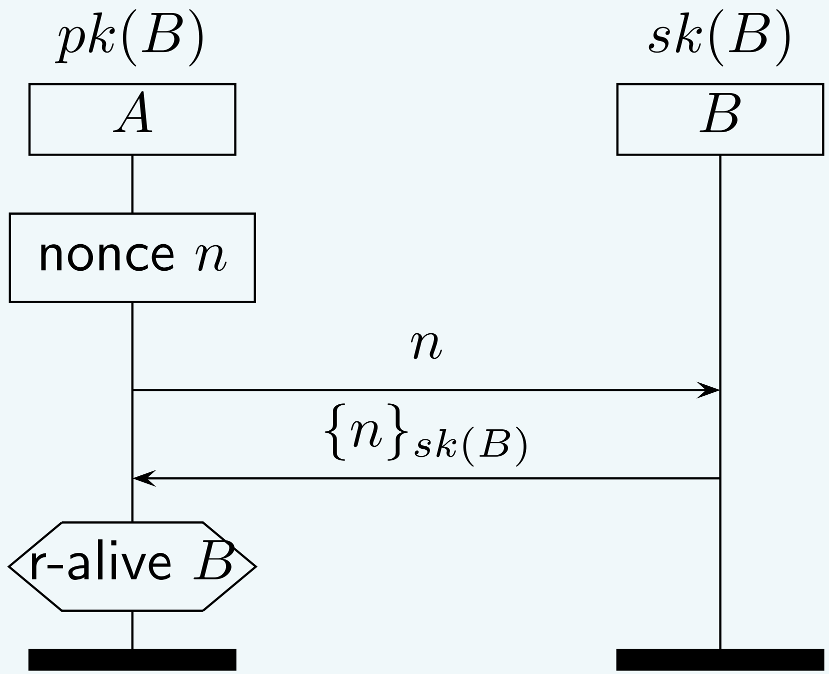
Authentication protocols (3)

Protocols satisfying aliveness guarantee that the communicating party has sent a message in the past.



Authentication protocols (3)

Protocols satisfying aliveness guarantee that the communicating party has sent a message in the past.



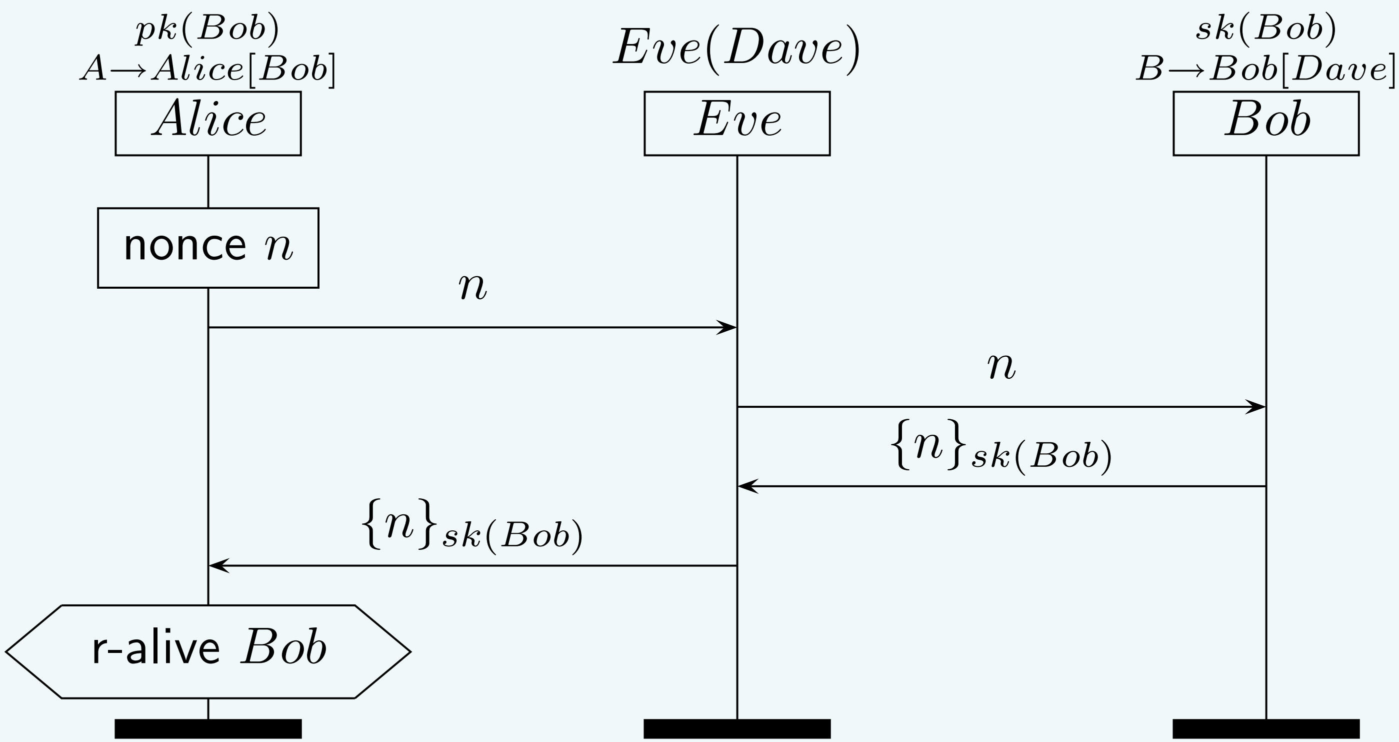
Definition. *(recent aliveness) A protocol guarantees to an agent a in role A recent aliveness of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then b has been running the protocol during a's run.*

Authentication protocols (3)

Protocols satisfying recent aliveness guarantee that the communicating party has recently sent “a message”.

Authentication protocols (3)

Protocols satisfying recent aliveness guarantee that the communicating party has recently sent “a message”.



Authentication protocols (3)

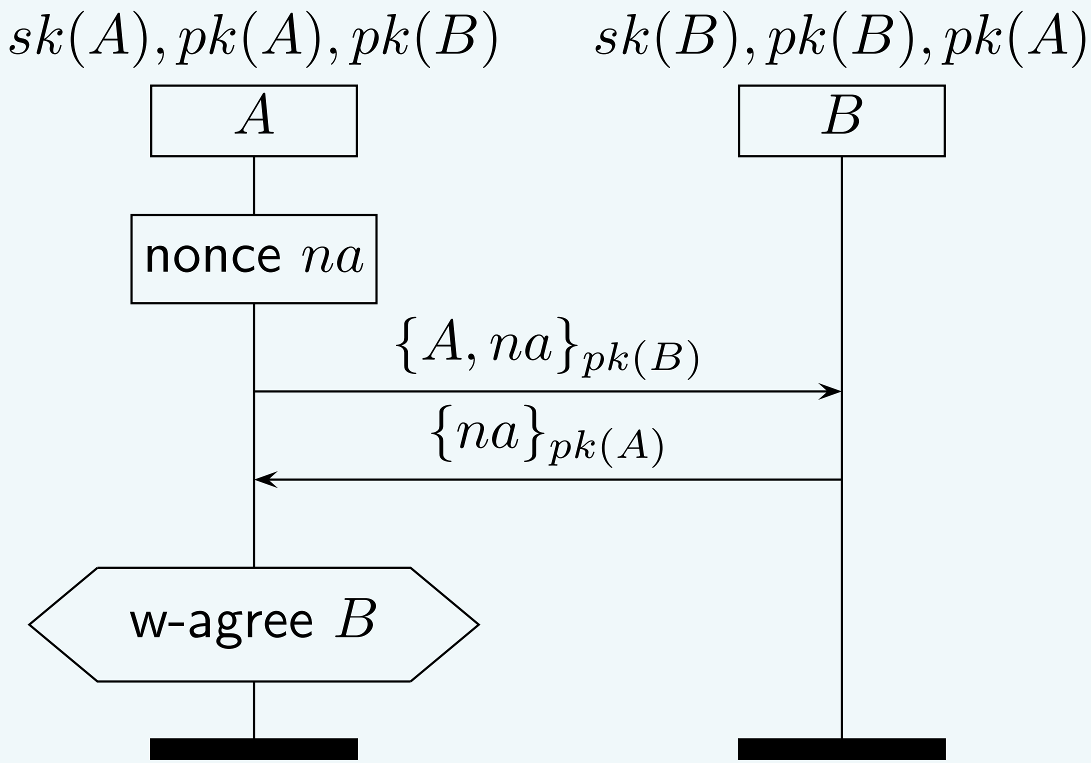
Definition. *(weak agreement) A protocol guarantees to an agent a in role A weak agreement of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then*

- *b has been running the protocol believing to be communicating with a.*

Authentication protocols (3)

Definition. *(weak agreement) A protocol guarantees to an agent a in role A* *weak agreement* *of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then*

- *b has been running the protocol believing to be communicating with a.*



Authentication protocols (3)

Definition. *(non-injective agreement) A protocol guarantees to an agent a in role A non-injective agreement of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then*

- *b has been running the protocol believing to be communicating with a and*
- *a and b agree on the contents of all the messages exchanged*

Authentication protocols (3)

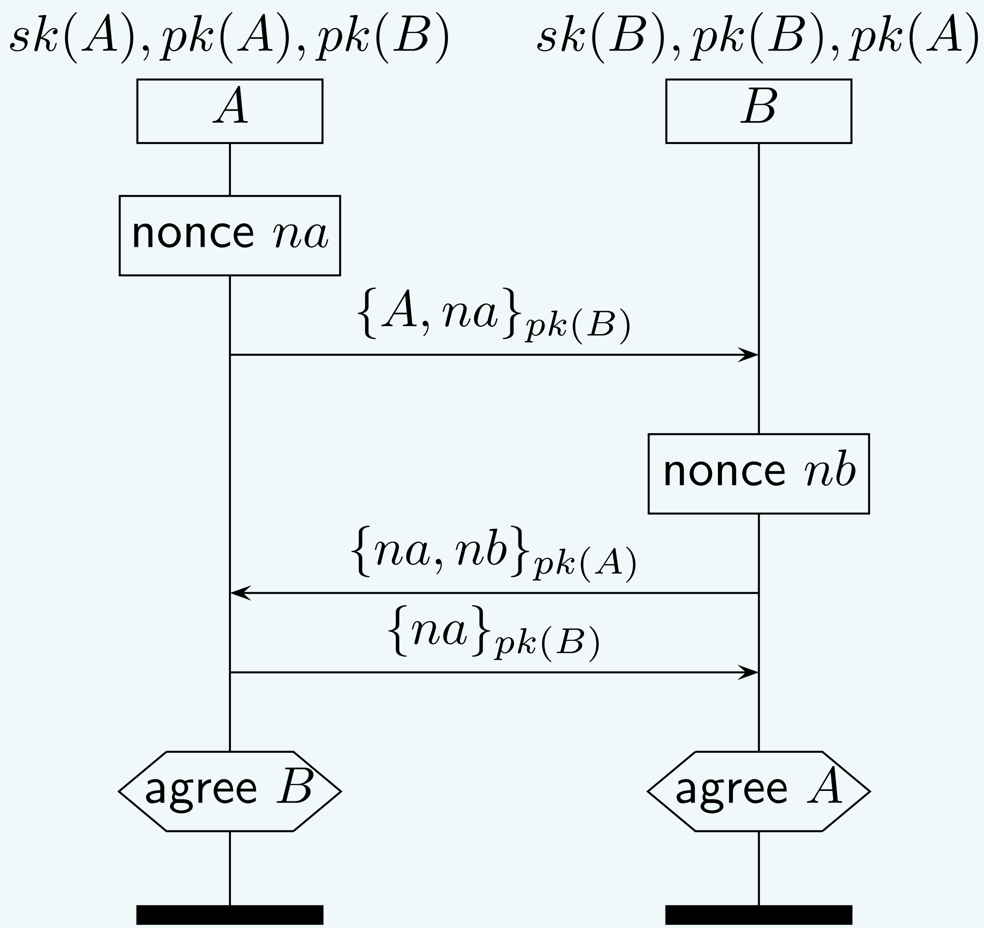
Definition. *(non-injective agreement) A protocol guarantees to an agent a in role A non-injective agreement of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then*

- *b has been running the protocol believing to be communicating with a and*
- *a and b agree on the contents of all the messages exchanged*

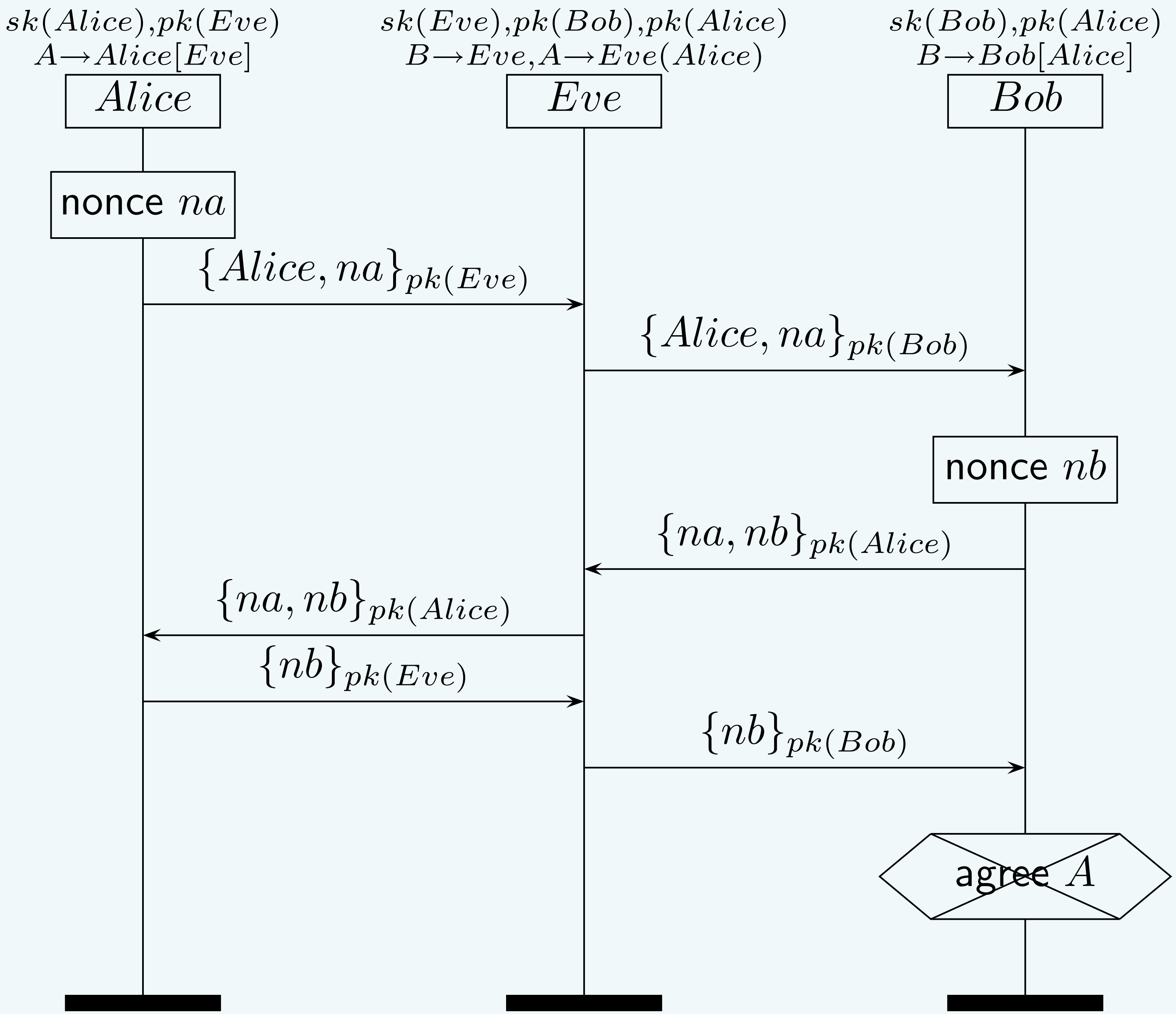
Definition. *(agreement) A protocol guarantees to an agent a in role A agreement of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then*

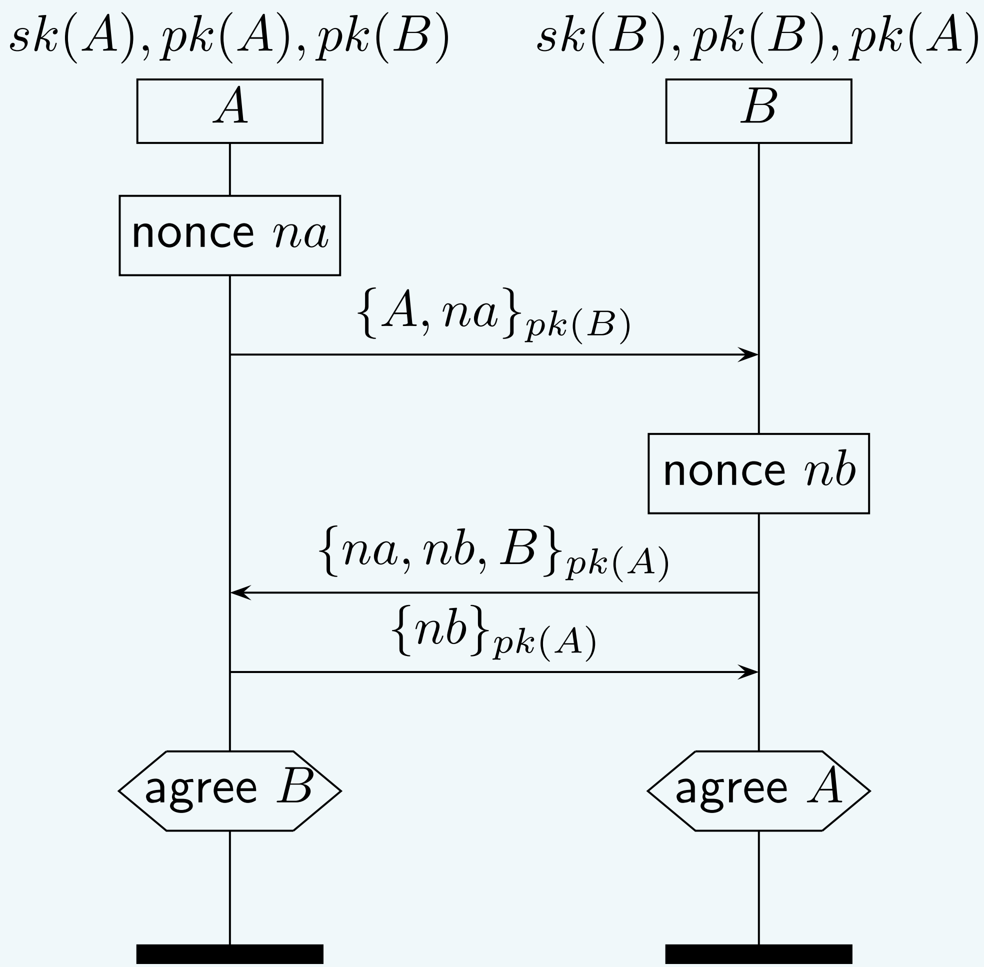
- *b has been running the protocol believing to be communicating with a,*
- *a and b agree on the contents of all the messages exchanged, and*
- *each run of A corresponds to a unique run of B.*

Famous example: Needham-Schroeder



Famous example: Needham-Schroeder




$sk(A), pk(A), pk(B)$
$sk(B), pk(B), pk(A)$
A
B
nonce na
$\{A, na\}_{pk(B)}$
nonce nb
$\{na, nb, B\}_{pk(A)}$
$\{nb\}_{pk(A)}$
agree B
agree A

Symbolic verification methods

Roughly three categories:

- Based on epistemic logic, e.g. BAN logic.
- Based on theorem proving, e.g. Paulson’s inductive approach.
- Based on model checking, e.g. Scyther, Avispa, ProVerif.

Automatic verification

Model checking based approaches build traces to represent all possible behavior of the system.

Security properties are verified on these traces.

Automatic verification

State:

- Active runs: $\mathcal{P}(Agent \times Events)$
- Send buffer: $\mathcal{P}(Term)$
- Read buffer: $\mathcal{P}(Term)$
- Intruder knowledge: $\mathcal{P}(Term)$

Automatic verification

State:

- Active runs: $\mathcal{P}(Agent \times Events)$
- Send buffer: $\mathcal{P}(Term)$
- Read buffer: $\mathcal{P}(Term)$
- Intruder knowledge: $\mathcal{P}(Term)$

Agent rules:

- Create: creates a new active run
- Send: advances a run by adding a message to the send buffer
- Read: advances a run by reading a message from the read buffer

Automatic verification

State:

- Active runs: $\mathcal{P}(Agent \times Events)$
- Send buffer: $\mathcal{P}(Term)$
- Read buffer: $\mathcal{P}(Term)$
- Intruder knowledge: $\mathcal{P}(Term)$

Agent rules:

- Create: creates a new active run
- Send: advances a run by adding a message to the send buffer
- Read: advances a run by reading a message from the read buffer

$$[\text{send}]\,\frac{a = (Ag, send(M) \cdot Ev) \in A}{\langle A, S, R, I\rangle \xrightarrow{send(Ag,M)} \langle A \backslash \{a\} \cup \{(Ag, Ev)\}, S \cup \{M\}, R, I\rangle}$$

Automatic verification

State:

- Active runs: $\mathcal{P}(Agent \times Events)$
- Send buffer: $\mathcal{P}(Term)$
- Read buffer: $\mathcal{P}(Term)$
- Intruder knowledge: $\mathcal{P}(Term)$

Intruder rules:

- Transmit: Moves a message from the S to R.
- Eavesdrop: Moves a message from S to R and adds it to I
- Block: Removes a message from R.
- Inject: Adds a message derivable from I to R.

$$[\text{inject}]\,\frac{I \vdash M}{\langle A, S, R, I\rangle \xrightarrow{inject(M)} \langle A, S, R \cup \{M\}, I\rangle}$$

Automatic verification

Many sources of infiniteness:

- Infinite number of agents
- Infinite number of sessions
- Infinite number of freshly generated messages
- Infinite number of constructable messages

Automatic verification

Many sources of infiniteness:

- Infinite number of agents
- Infinite number of sessions
- Infinite number of freshly generated messages
- Infinite number of constructable messages

State-space reductions:

- For secrecy protocols nonces do not have to be fresh
- For secrecy protocols 2 agents are enough, for authentication 3 agents
- For authentication protocols traces can be combined into trace classes.

Active research areas (1)

Current research in security protocols focuses on:

- protocol compositions,

Active research areas (2)

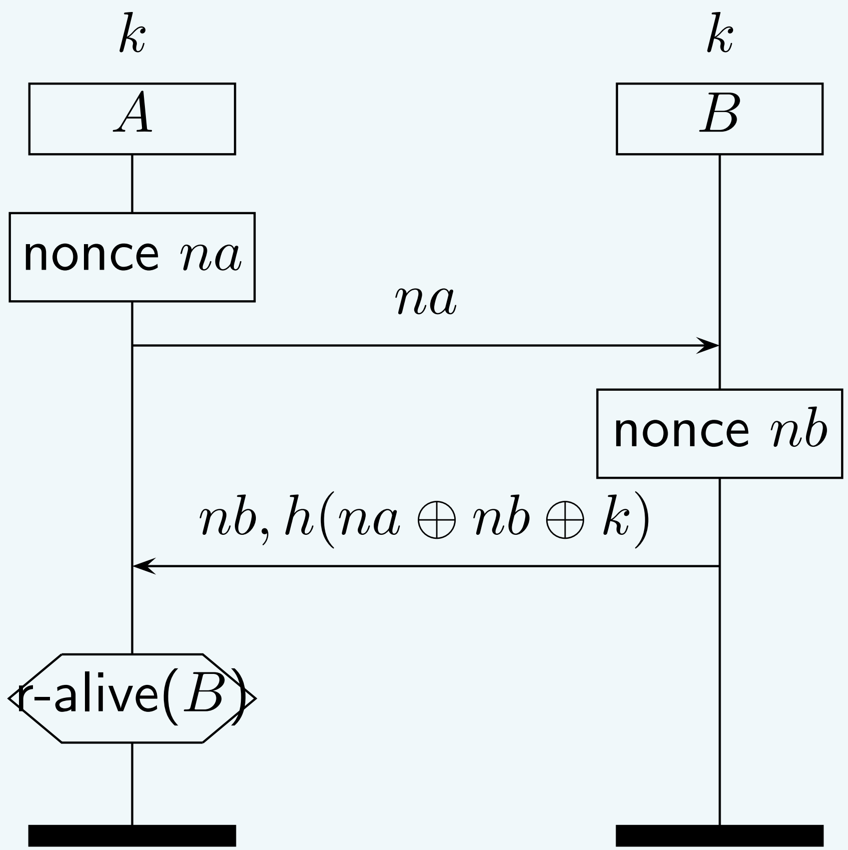
Current research in security protocols focuses on:

- analysis of other types of properties
  - Voting: Coercion resistance/receipt freeness, individual verifiability.
  - Fair exchange/contract signing: Fairness, abuse freeness.
  - Anonymity: Sender anonymity, receiver anonymity, unlinkability.
  - PAKE: Absence of guessing attacks
  - RFID: Untraceability

Active research areas (3)

Current research in security protocols focuses on:

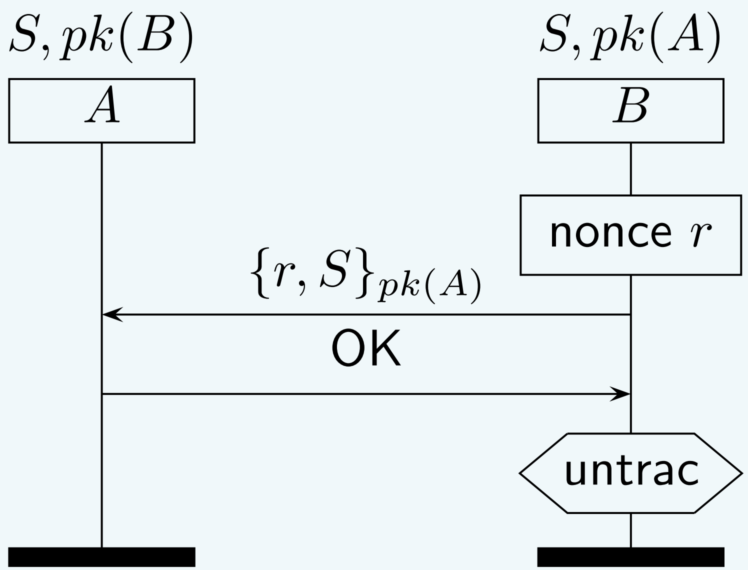
- analysis of protocols with algebraic properties (XOR, DH exponentiation),



Active research areas (4)

Current research in security protocols focuses on:

- bridging the gap between the computational and the symbolic protocol worlds.



Let $yP = pk(A)$. The encryption $\{r, S\}_{pk(A)}$ could be implemented by the ElGamal encryption $(rP, I + ryP)$.