



# Expressing Security Properties in CSP

- Security properties: the goals that a protocol is meant to satisfy, relatively to specific kinds and levels of threat – the intruders and their capabilities
- We will consider the following security properties:
  - **Secrecy**
    - No information leakage
  - **Authentication**
    - No falsification of identity
  - **Non-repudiation**
    - Evidence of the involvement of the other party
  - ✓ **Anonymity**
    - Protecting the identity of agents wrt particular events

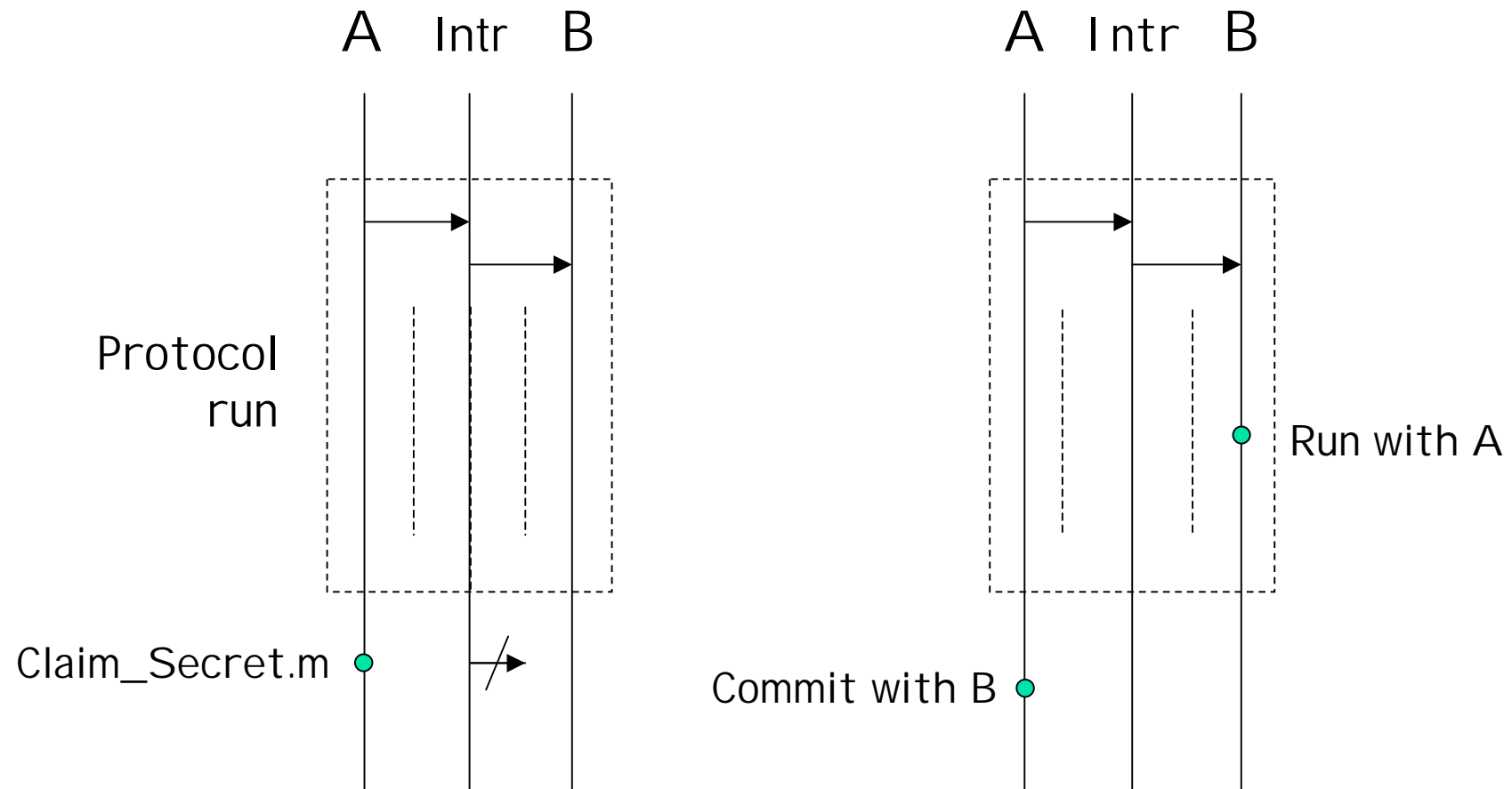


# Secrecy and authentication

---

- They are both safety properties: *a certain bad thing should not happen*
- **Explicit annotations:** In the CSP approach, these properties are defined by “enhancing” the code of the processes with explicit signal claiming the success of the protocol wrt the intended property
- **Secrecy:** `Claim_secret. m`
  - Information `m` has not become known to the intruder
- **Authentication:** `Run with A , Commit with B`
  - The matching of these two events guarantees the identities of `A` and `B`

# Secrecy and authentication





## Example: The Yahalom Protocol

- The protocol

Message 1    $a \rightarrow b : a.n_a$

Message 2    $b \rightarrow s : b.\{a.n_a.n_b\}_{\text{ServerKey}(b)}$

Message 3    $s \rightarrow a : \{b.k_{ab}.n_a.n_b\}_{\text{ServerKey}(a)} \{a.k_{ab}\}_{\text{ServerKey}(b)}$

Message 4    $a \rightarrow b : \{a.k_{ab}\}_{\text{ServerKey}(b)} . \{n_b\}_{k_{ab}}$

- Authentication of the participants
- $K_{ab}$  should remain secret
- We may require secrecy also on  $n_b$



## Exm: Secrecy in the Yahalom protocol

- CSP description of the two parties - Original

Initiator( $a, n_a$ ) =

env?b: Agent

→ send.a.b.a. $n_a$

→ [] (receive.J.a{b.  $k_{ab} \cdot n_a \cdot n_b$ }<sub>ServerKey(a)</sub> .m

$k_{ab} \in \text{Key}$  → send.a.b.m.{ $n_b$ } <sub>$k_{ab}$</sub>

$n_b \in \text{Nonce}$  → Session(a,b, $k_{ab}, n_a, n_b$ ) )

$m \in T$

Responder( $b, n_b$ ) =

[] (receive.a.b.a. $n_a$  → send.b.J.b.{a. $n_a \cdot n_b$ }<sub>ServerKey(b)</sub>

$k_{ab} \in \text{Key}$  → receive.a.b.{a.  $k_{ab}$ }<sub>ServerKey(b)</sub> .{ $n_b$ } <sub>$k_{ab}$</sub>

$n_b \in \text{Nonce}$  → Session(b,a, $k_{ab}, n_a, n_b$ ) )

$m \in T$



# Exm: Secrecy in the Yahalom protocol

- CSP description of the two parties - Enhanced

Initiator'(a, n<sub>a</sub>) =

env?b: Agent

→ send.a.b.a.n<sub>a</sub>

→ [] (receive.J.a{b. k<sub>ab</sub>.n<sub>a</sub>.n<sub>b</sub>}<sub>ServerKey(a)</sub> .m

k<sub>ab</sub> ∈ Key → send.a.b.m.{n<sub>b</sub>}<sub>k<sub>ab</sub></sub>

n<sub>b</sub> ∈ Nonce → signal.Claim\_Secret.a.b. k<sub>ab</sub>

m ∈ T → Session(a,b,k<sub>ab</sub>,n<sub>a</sub>,n<sub>b</sub>) )

Responder'(b, n<sub>b</sub>) =

[] (receive.a.b.a.n<sub>a</sub> → send.b.J.b.{a.n<sub>a</sub>.n<sub>b</sub>}<sub>ServerKey(b)</sub>

k<sub>ab</sub> ∈ Key → receive.a.b.{a. k<sub>ab</sub>}<sub>ServerKey(b)</sub> .{n<sub>b</sub>}<sub>k<sub>ab</sub></sub>

n<sub>b</sub> ∈ Nonce → signal.Claim\_Secret.a.b. k<sub>ab</sub>

m ∈ T → Session(b,a,k<sub>ab</sub>,n<sub>a</sub>,n<sub>b</sub>) )



# Exm: Secrecy in the Yahalom protocol

- CSP description of the server

$\text{Server}(J, k_{ab}) =$

$[] \text{ (receive.b.J.b} \cdot \{a.n_a.n_b\}_{\text{ServerKey}(b)}$

$A, B \in \text{Agent} \quad \rightarrow \text{send.J.a. } \{b.k_{ab}.n_a.n_b\}_{\text{ServerKey}(a)} \cdot \{a.k_{ab}\}_{\text{ServerKey}(b)}$

$N_b, n_b \in \text{Nonce} \quad \rightarrow \text{Server}(J, k_s)$

$\text{Server}(J) = ||| \text{Server}(J, k_{ab})$

$k_{ab} \in \text{Keys}_{\text{Server}}$



# Exm: Secrecy in the Yahalom protocol

- CSP description of the intruder

$\text{Intruder}(X) = \text{learn?m: messages} \rightarrow \text{Intruder}(\text{close}(X \cup \{m\}))$

$[]$

$\text{say!m: } X \wedge \text{messages} \rightarrow \text{Intruder}(X)$

- $\text{Close}(X)$  represents all the possible information that the attacker can infer from  $X$ . Typically we assume

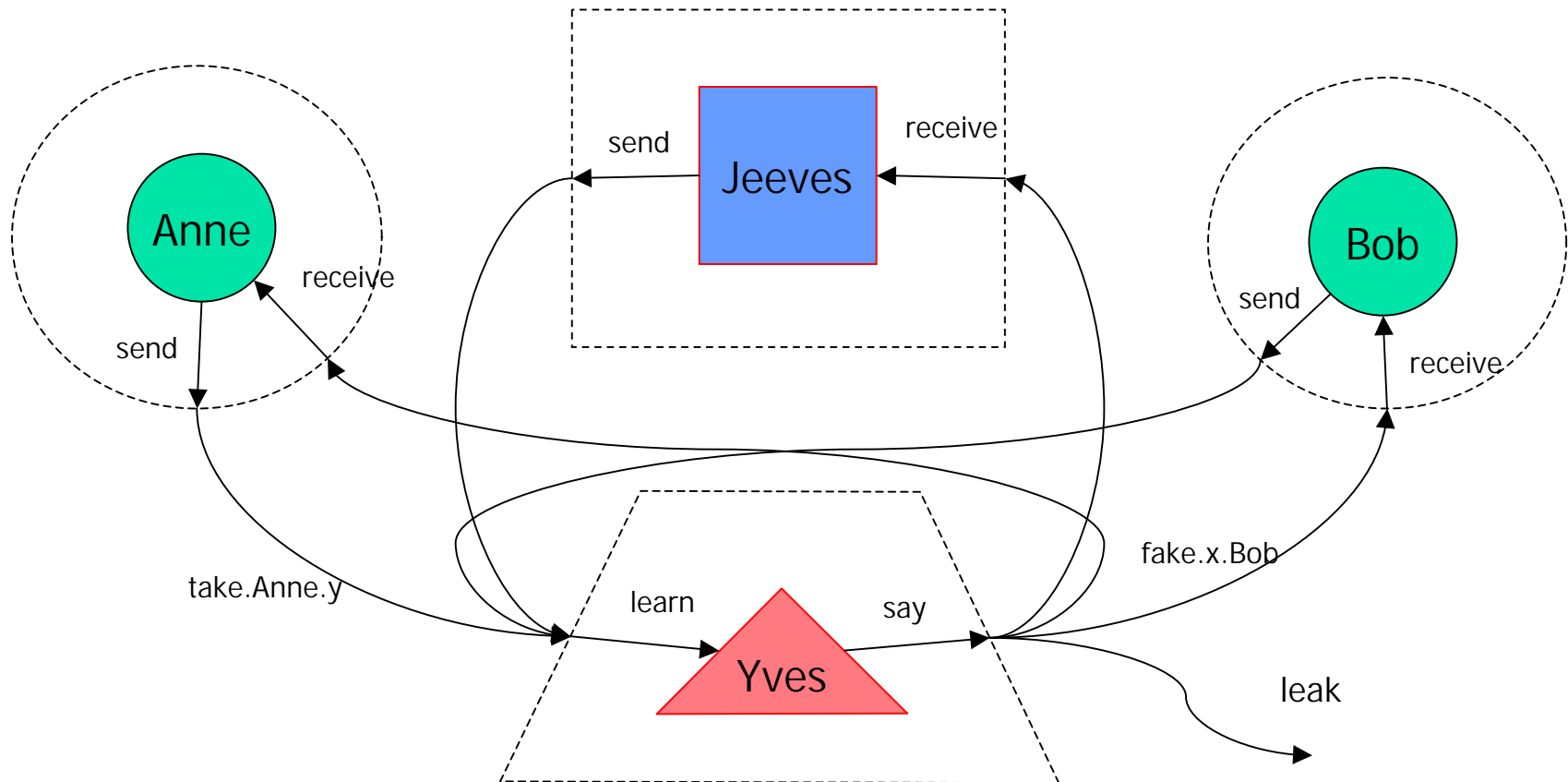
- $\{k, m\} \mid\text{-} \text{encrypt}(k, m)$
- $\{\text{encrypt}(k, m), k^{-1}\} \mid\text{-} m$
- $\{\text{Sq}\langle x_1, \dots, x_n \rangle\} \mid\text{-} x_i$
- $\{x_1, \dots, x_n\} \mid\text{-} \text{Sq}\langle x_1, \dots, x_n \rangle$



# Exm: Secrecy in the Yahalom protocol

Initiator'(Anne,  $n_A$ )S ||| Responder(Bob,  $n_B$ )S ||| Server(Jeeves)S ||| Intruder'( $\phi$ )S'

$S = [\text{fake}, \text{take}/\text{receive}, \text{send}]$   
 $S' = [\text{take.x.y}/\text{learn}][\text{fake.x.y}, \text{leak}/\text{say}]$





## Exm: Secrecy in the Yahalom protocol

---

- The property to be verified:

$\text{Signal.Claim\_Secret.a.b.m} \in \text{Traces}(\text{System})$



$\text{not}(\text{leak.m} \in \text{Traces}(\text{System}) )$

- As usual, this property can be verified automatically by checking the traces



# Authentication

---

- The CSP approach is based on inserting signals:
  - **Running.a.b** (in **a**'s protocol)
    - Agent **a** is executing a protocol run apparently with **b**
  - **Commit.b.a** (in **b**'s protocol)
    - Agent **b** has completed a protocol run apparently with **a**
- Authentication is achieved if **Running.a.b** always precedes **Commit.b.a** in the traces of the system
  - Weaker or stronger forms of authentication can be achieved by variations of the parameters of these signals and the constraints on them



## Authentication in the Yahalom Pr.

---

- The Yahalom Protocol aims at providing **authentication of both parties** :  
authentication of the initiator to the responder, and viceversa
- We will analyze the two authentication properties separately
- This requires two separate enhancements of the protocol

# Yahalom: authentication of initiator

- CSP description of the two parties - Enhanced

Initiator'(a, n<sub>a</sub>) =

env?b: Agent

→ send.a.b.a.n<sub>a</sub>

→ [] (receive.J.a{b. k<sub>ab</sub>.n<sub>a</sub>.n<sub>b</sub>}<sub>ServerKey(a)</sub> .m

kab ∈ Key → signal.Running\_Initiator.a.b.n<sub>a</sub>.n<sub>b</sub>.k<sub>ab</sub>

nb ∈ Nonce → send.a.b.m.{n<sub>b</sub>}<sub>kab</sub>

m ∈ T → Session(a, b, k<sub>ab</sub>, n<sub>a</sub>, n<sub>b</sub>) )

Responder'(b, n<sub>b</sub>) =

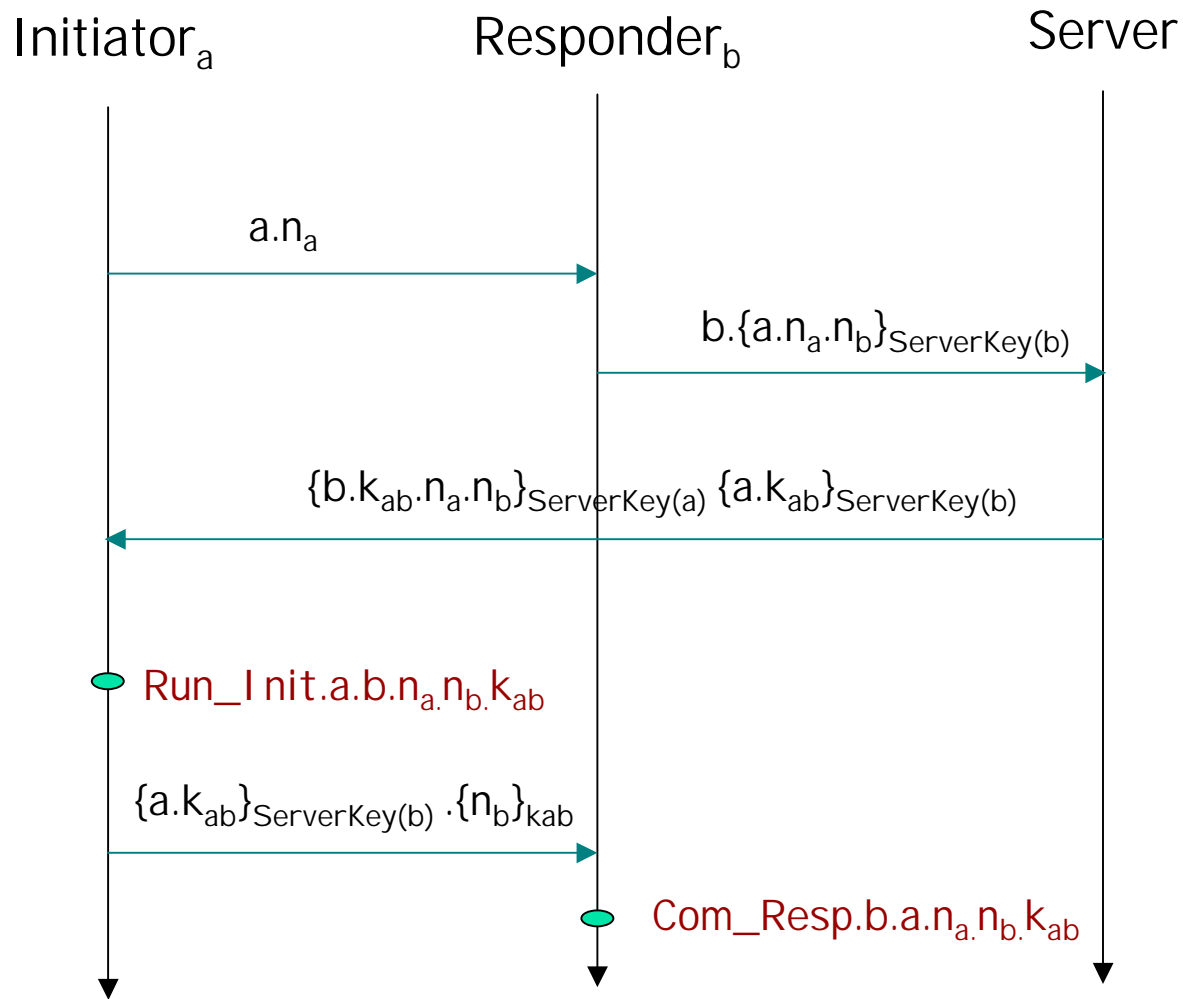
[] (receive.a.b.a.n<sub>a</sub> → send.b.J.b.{a.n<sub>a</sub>.n<sub>b</sub>}<sub>ServerKey(b)</sub>

kab ∈ Key → receive.a.b.{a. k<sub>ab</sub>}<sub>ServerKey(b)</sub> .{n<sub>b</sub>}<sub>kab</sub>

nb ∈ Nonce → signal.Commit\_Responder.b.a.n<sub>a</sub>.n<sub>b</sub>.k<sub>ab</sub>

m ∈ T → Session(b, a, k<sub>ab</sub>, n<sub>a</sub>, n<sub>b</sub>) )

# Yahalom: authentication of initiator





## Yahalom: authentication of initiator

---

- The property to be verified:

signal. Running\_Initiator.a.b. $n_a$ . $n_b$ . $k_{ab}$   
precedes

signal.Commit\_Responder.b.a. $n_a$ . $n_b$ . $k_{ab}$   
in all the Traces(System)

- Again, this property can be verified automatically by checking the traces

# Yahalom: authentication of responder

- CSP description of the two parties - Enhanced

Initiator'(a, n<sub>a</sub>) =

env?b: Agent

→ send.a.b.a.n<sub>a</sub>

→ [] (receive.J.a{b. k<sub>ab</sub>.n<sub>a</sub>.n<sub>b</sub>}<sub>ServerKey(a)</sub> .m

kab ∈ Key → send.a.b.m.{n<sub>b</sub>}<sub>kab</sub>

nb ∈ Nonce → signal.Commit\_Initiator.a.b.n<sub>a</sub>.n<sub>b</sub>.k<sub>ab</sub>

m ∈ T → Session(a, b, k<sub>ab</sub>, n<sub>a</sub>, n<sub>b</sub>) )

Responder'(b, n<sub>b</sub>) =

[] (receive.a.b.a.n<sub>a</sub> → send.b.J.b.{a.n<sub>a</sub>.n<sub>b</sub>}<sub>ServerKey(b)</sub>

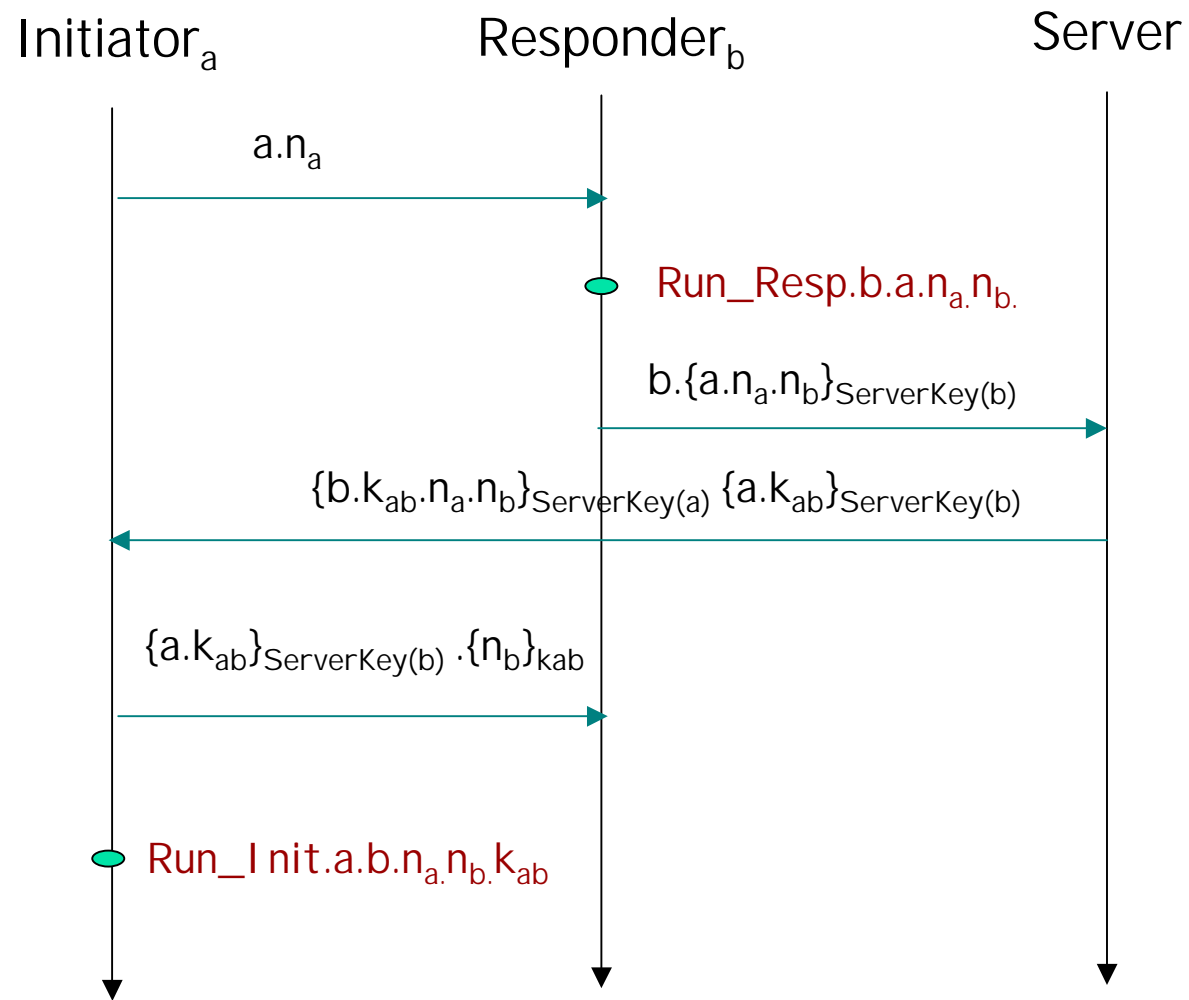
kab ∈ Key → signal.Running\_Responder.b.a.n<sub>a</sub>.n<sub>b</sub>

nb ∈ Nonce → receive.a.b.{a. k<sub>ab</sub>}<sub>ServerKey(b)</sub> .{n<sub>b</sub>}<sub>kab</sub>

m ∈ T → Session(b, a, k<sub>ab</sub>, n<sub>a</sub>, n<sub>b</sub>) )



# Yahalom: authentication of responder





## Yahalom: authentication of responder

---

- The property to be verified:

signal. Running\_Responder.b.a. $n_a$ . $n_b$

precedes

signal.Commit\_Initiator.a.b. $n_a$ . $n_b$ . $k_{ab}$

in all the Traces(System)

- Again, this property can be verified automatically by checking the traces



# Non-repudiation

- **Goal:** provide the parties of an interaction with evidence so that later they cannot deny having participated
- **Example:** The Zhou-Gollmann protocol

Message 1  $a \rightarrow b : \{f_{NRO} . b.l.c\}_{Sk_a}$

Message 2  $b \rightarrow a : \{f_{NRR} . a.l.c\}_{Sk_b}$

Message 3  $a \rightarrow j : \{f_{SUB} . b.l.k\}_{Sk_a}$

Message 4  $b \leftrightarrow j : \{f_{CON} . a.b.l.k\}_{Sk_j}$

Message 5  $a \leftrightarrow j : \{f_{CON} . a.b.l.k\}_{Sk_j}$

- $c = k(m)$  where  $m$  is the message to be transmitted
  - $a$  and  $b$  are the parties,  $j$  is the trusted server
  - $f_{NRO}$ ,  $f_{NRR}$ , etc. are flags identifying the steps.  $l$  is a nonce
  - $Sk_a$ ,  $Sk_b$ , etc. are signature keys known only to their owners
- 
- $a$  can prove that  $b$  has got the message by presenting  $\{f_{NRR} . a.l.c\}_{Sk_b}$  and  $\{f_{CON} . a.b.l.k\}_{Sk_j}$



# The Zhou-Gollmann protocol

---

- **Non-Repudiation of Recipient:**  $a$  can prove that  $b$  has got the message by presenting

$$\{f_{\text{NRR}} .a.l.c\}_{\text{Sk}_b} \text{ and } \{f_{\text{CON}} .a.b.l.k\}_{\text{Sk}_j}$$

- **Non-Repudiation of Origin:**  $b$  can prove that  $a$  has sent the message by presenting

$$\{f_{\text{NRO}} .b.l.c\}_{\text{Sk}_a} \text{ and } \{f_{\text{CON}} .a.b.l.k\}_{\text{Sk}_j}$$



# CSP analysis of Non-Repudiation

- Specification of the Zhou-Gollmann protocol in CSP

$\text{Agent}_a(S) =$

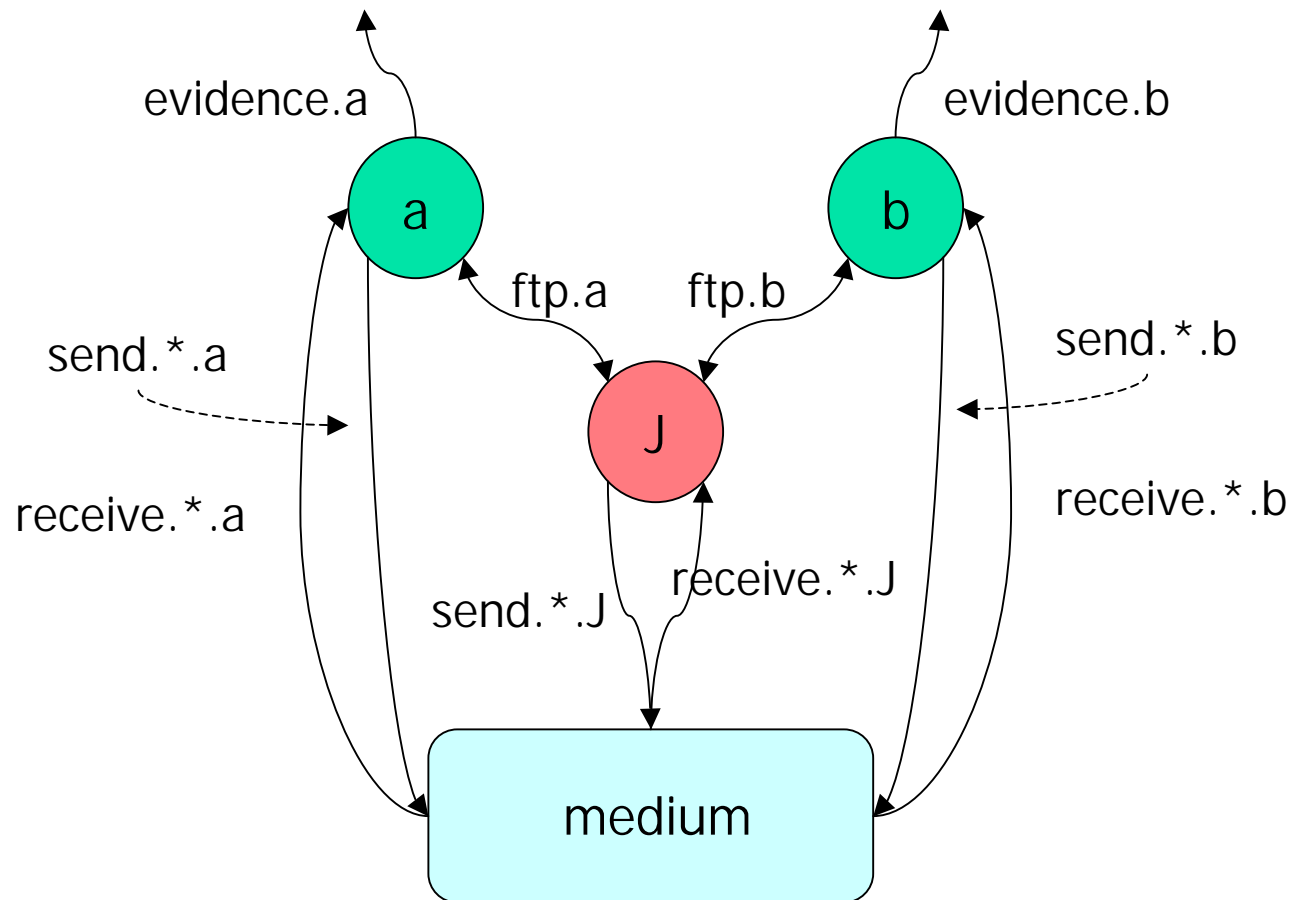
$$\begin{aligned} & [] \text{ } b \in \text{Agent}, m \in S \text{ send.a.b.m} \rightarrow \text{Agent}_i(S) \\ & [] \text{ receive.a.b?m} \rightarrow \text{Agent}_a(\text{close}(S \cup \{m\})) \\ & [] \text{ ftp.a.Jeeves?m} \rightarrow \text{Agent}_a(\text{close}(S \cup \{m\})) \\ & [] \text{ } m \in S \text{ evidence.a.m} \rightarrow \text{Agent}_i(S) \end{aligned}$$

- $\text{Close}(S)$  represent the capability of inferring new information

$\text{Server}(S) =$

$$\begin{aligned} & \text{receive.a.Jeeves?. } \{f_{\text{SUB}} . \text{b.l.k}\}_{\text{Sk}_a} \\ & \quad \rightarrow \text{Server}(S \cup \{f_{\text{CON}} . \text{a.b.l.k}\}_{\text{Sk}_j}) \\ & [] \text{ } b \in \text{Agent}, m \in S \text{ ftp.a.Jeeves.m} \rightarrow \text{Server}(S) \end{aligned}$$

# The Zhou-Gollmann protocol in CSP





# Analysis of the Zhou-Gollmann protocol

- Non-Repudiation of Recipient:

$\text{evidence.a.}\{f_{\text{NRR}}.\text{a.l.c}\}_{\text{Sk}_b} \text{ in Tr} \Leftrightarrow b \text{ sent } (f_{\text{NRR}}.\text{a.l.c})$

$\text{evidence.a.}\{f_{\text{CON}}.\text{a.b.l.k}\}_{\text{Sk}_j} \text{ in Tr} \Leftrightarrow \text{receive.a.j.}\{f_{\text{CON}}.\text{a.b.l.k}\}_{\text{Sk}_j} \text{ in Tr}$

- Non-Repudiation of Origin:

$\text{evidence.b.}\{f_{\text{NRO}}.\text{b.l.c}\}_{\text{Sk}_a} \text{ in Tr} \Leftrightarrow a \text{ sent } (f_{\text{NRO}}.\text{b.l.c})$

$\text{evidence.b.}\{f_{\text{CON}}.\text{a.b.l.k}\}_{\text{Sk}_j} \text{ in Tr} \Leftrightarrow a \text{ sent } (f_{\text{SUB}}.\text{b.l.k})$

- Again, these properties on traces can be proven automatically