

Esempio di Modellazione di protocolli di sicurezza con CSP

Introduzione (1)

- Per tutti i protocolli sono prescritte **sequenze di interazioni** tra agenti per raggiungere un determinato obiettivo
- I protocolli di comunicazione hanno l'obiettivo di stabilire una **comunicazione** tra gli agenti
 - stabilire il link
 - accordarsi sulla sintassi

Introduzione (2)

- I protocolli di sicurezza hanno lo scopo di stabilire le **opportune sequenze di interazioni** che permettono di offrire determinati **servizi di sicurezza** nella comunicazione tra **agenti distribuiti**
- Richiedono scambio di informazioni tra i nodi comunicanti
 - talvolta è richiesta la presenza di una terza

parte **fidata (trusted)**

Caratteristiche di Sicurezza (1)

- Segretezza:
 - nessun intruso può dedurre le attività che gli utenti legittimi stanno svolgendo
 - nessun intruso può leggere i messaggi che gli utenti legittimi si stanno scambiando
- Autenticazione:
 - dell'origine dei messaggi
 - delle entità coinvolte nella comunicazione

Caratteristiche di Sicurezza (2)

- Integrità dei dati:
 - Garanzia che i messaggi ricevuti non sono stati corrotti
 - È un corollario dell'autenticazione precedente
- Non ripudio:
 - Non è possibile negare di aver partecipato a una comunicazione

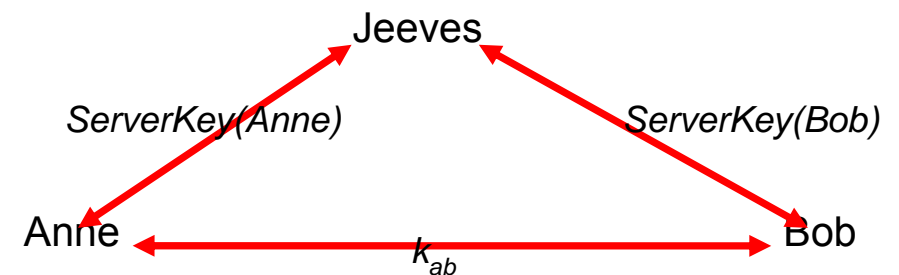
Caratteristiche di Sicurezza (3)

- Equità (Fairness):
 - Evitare che un comunicante possa avvantaggiarsi durante la comunicazione rispetto ad un altro
 - Ad es. interrompendo la comunicazione dopo aver ricevuto dati sensibili E prima di aver trasmesso i propri
- Anonimia:
 - Poter utilizzare determinati servizi senza dover comunicare la propria identità

Caratteristiche di Sicurezza (4)

- Disponibilità
 - Garanzia che il servizio per il quale si è attivata la comunicazione è effettivamente offerto

Schema di Riferimento Classico per Canale Sicuro (1)



Schema di Riferimento Classico per Canale Sicuro (2)

- Protocollo di sicurezza basato su un algoritmo simmetrico di crittografia
- I due agenti Anne (a) e Bob (b) devono comunicare attraverso un canale sicuro con l'aiuto di un server fidato (Jeeves)
- In generale, tutti i gli agenti registrati condividono con Jeeves chiavi di crittografia private e a lunga vita

Schema di Riferimento Classico per Canale Sicuro (3)

- Queste chiavi permettono ad ogni agente di comunicare in modo sicuro con Jeeves
- Non permettono la comunicazione tra Anne e Bob

Possibili comunicazioni sicure(1)

- Una soluzione che permetta a due agenti di comunicare tra di loro è data da:
 - Anne manda a Jeeves il msg destinato a Bob, codificato con il codice $ServerKey(Anne)$
 - Jeeves decrittografa il msg da Anne, lo ricodifica secondo $ServerKey(Bob)$
 - Jeeves invia il msg a Bob
- La comunicazione è sicura ma è troppo onerosa: Jeeves è un collo di bottiglia

Possibili comunicazioni sicure(2)

- Soluzione: Jeeves fornisce chiavi a lunga vita per ogni possibile coppia di agenti comunicanti
 - se il numero N di agenti è elevato servono N^2 chiavi
 - in genere la dimensione della rete potrebbe essere molto dinamica

Possibili comunicazioni sicure(3)

- Soluzione: richiedere l'intervento di Jeeves solo quando necessario per fornire una chiave che permetta la comunicazione sicura tra Anne e Bob

Protocollo Yahalom

Msg1 $a \rightarrow b : a.n_a$
Msg2 $b \rightarrow J : b.\{a.n_a.n_b\}_{\text{ServerKey}(b)}$
Msg3 $J \rightarrow a : \{b.k_{ab}.n_a.n_b\}_{\text{ServerKey}(a)}.\{a.k_{ab}\}_{\text{ServerKey}(b)}$
Msg4 $a \rightarrow b : \{a.k_{ab}\}_{\text{ServerKey}(b)}.\{n_b\}_{k_{ab}}$

Premessa (1)

- Due agenti coinvolti nella comunicazione (a e b) e uno di supporto (J)
- Tra i due agenti a e b
 - uno è l'initiator (poniamo a)
 - uno è il responder (b)
- Si vuole modellizzare con CSP i tre ruoli

Premessa (2)

- Ogni agente ha due domini di comunicazione, verso
 - l'altro agente
 - il proprio utente
- Assumiamo per semplicità che ogni processo sia provvisto dei due canali **send** e **receive** per tutte le comunicazioni con gli altri nodi
 - L'input assume la forma: receive.a.b.m
 - L'output assume la forma: send.a.b.m

Premessa (3)

- Tutti i messaggi del protocollo sono sottoposti a send/receive
- Le minacce alla sicurezza della comunicazione possono provenire da tutte le direzioni

Accettazione messaggi

- Necessità di vincolare un processo ad accettare solo i messaggi che hanno forma che il processo sia in grado di comprendere
 - scelta sui messaggi accettabili
 - se il messaggio ricevuto non è accettabile, deve essere eseguito un processo AbortRun
 - AbortRun non è qui modellato

Initiator (1)

Initiator (a, n_a) =

env?b:Agent -> send.a.b.a.n_a ->

□

$k_{ab} \in \text{Key}$
 $n_b \in \text{Nonce}$
 $m \in T$

$\left(\begin{array}{l} \text{receive.J.a.}\{b.k_{ab}.n_a.n_b\}_{\text{ServerKey}(a)}.m \rightarrow \\ \text{send.a.b.m.}\{n_b\}_{k_{ab}} \rightarrow \text{Session}(a,b,k_{ab},n_a,n_b) \end{array} \right)$

dove T è l'insieme degli oggetti che il nodo può accettare

Initiator (2)

- Per i nostri scopi non è necessario dettagliare ulteriormente lo stato Session
 - Possiamo assumere che sia lo stato della sessione di comunicazione in cui avvengono gli scambi di informazione tra i comunicanti
- La chiave ServerKey(a) è la chiave che a condivide con J
- La comunicazione iniziale env?b:Agent è la richiesta che l'ambiente di a invia ad a per cominciare la comunicazione con b

Initiator (3)

- Il pacchetto che a riceve dal server e passa a b secondo il protocollo originale è

$$\{a.k_{ab}\}_{\text{ServerKey}(b)}$$

- Nella modellizzazione con CSP diventa semplicemente l'input m
 - a non svolge alcuna operazione su di esso, solo verifica che appartenga all'insieme delle azioni accettabili

Responder (1)

- Responder (b, n_b) =

$$\square$$

$$k_{ab} \in \text{Key}$$

$$n_a \in \text{Nonce}$$

$$a \in \text{Agent}$$

$$\left(\begin{array}{l} \text{receive.a.b.a.n}_a \rightarrow \\ \text{send.b.J.b.}\{a.n_a.n_b\}_{\text{ServerKey}(b)} \rightarrow \\ \text{receive.a.b.}\{a.k_{ab}\}_{\text{ServerKey}(b)} \cdot \{n_b\}_{k_{ab}} \rightarrow \end{array} \right)$$

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

dove Agent è l'insieme degli agenti con cui b può

comunicare

Responder (2)

- Il protocollo è attivato dalla ricezione di un messaggio da parte di a
 - Non più da parte dell'ambiente

Server

- Serv (J, k_{ab}) =

$$\square$$

$$n_a, n_b \in \text{Nonce}$$

$$a, b \in \text{Agent}$$

$$\left(\begin{array}{l} \text{receive.b.J.b.}\{a.n_a.n_b\}_{\text{ServerKey}(b)} \rightarrow \\ \text{send.J.a.}\{b.k_{ab}.n_a.n_b\}_{\text{ServerKey}(a)} \\ \{a.k_{ab}\}_{\text{ServerKey}(b)} \rightarrow \text{Server}(J,ks) \end{array} \right)$$