

UNIVERSITÀ DEGLI STUDI DI PISA

FACOLTÀ DI INGEGNERIA

Laurea Magistrale in

INGEGNERIA INFORMATICA

SICUREZZA NEI SISTEMI SOFTWARE

**Progetto di una applicazione peer-to-peer  
con comunicazione sicura**

Implementazione in Java

a cura di

*Sacco Cosimo e Silvestri Davide*

Anno Accademico 2010/2011



# Indice

<b>1</b>	<b>Protocollo</b>	<b>1</b>
<b>2</b>	<b>Certification Authority</b>	<b>3</b>
<b>3</b>	<b>Analisi del protocollo di scambio chiavi</b>	<b>5</b>
3.1	<i>Beliefs</i> da ottenere . . . . .	5
3.2	Protocollo idealizzato . . . . .	5
3.3	Ipotesi . . . . .	6
3.4	Analisi dei <i>beliefs</i> . . . . .	6
3.4.1	Messaggio <i>M1</i> . . . . .	6
3.4.2	Messaggio <i>M2</i> . . . . .	7
3.4.3	Messaggio <i>M3</i> . . . . .	7
3.4.4	Messaggio <i>M4</i> . . . . .	7
3.4.5	Messaggio <i>M5</i> . . . . .	8
<b>4</b>	<b>Installazione</b>	<b>11</b>



# Capitolo 1

## Protocollo

Il sistema è stato pensato per garantire confidenzialità nella comunicazione in un sistema peer-to-peer in cui ciascun membro dispone di una coppia di chiavi pubblica e privata opportunamente certificata. In particolare, la comunicazione tra i peer è preceduta dall'esecuzione del protocollo per stabilire una chiave di sessione. Tale protocollo garantisce *key authentication* (ovvero, ogni peer crede che la chiave stabilita sia la chiave di sessione), *key confirmation* (ovvero, ogni peer crede che la controparte sia convinta sulla key authentication relativamente alla chiave di sessione) e infine *key freshness* (si garantisce che entrambe le parti siano convinte della freschezza della chiave di sessione generata). Viene illustrato, qui di seguito, il protocollo:

$$\begin{aligned}M1 : A &\rightarrow B & C_A \\M2 : A &\leftarrow B & C_B \\M3 : A &\rightarrow B & \{n_A\}_{K_B^+} \\M4 : A &\leftarrow B & \{n_A, n_B\}_{K_A^+} \\M5 : A &\rightarrow B & \{n_B\}_{K_B^+}\end{aligned}$$

Significato dei messaggi:

- M1** :  $A$  invia a  $B$  il proprio certificato  $C_A$ ;
- M2** : avendo verificato la validità del certificato di  $A$ ,  $B$  invia a  $A$  il proprio certificato  $C_B$ ;
- M3** : avendo verificato la validità del certificato di  $B$ ,  $A$  genera un nonce random  $n_A$  e lo invia cifrandolo con la chiave pubblica di  $B$ . In questo modo, solamente  $B$  potrà leggere  $M3$ ;
- M4** :  $B$  decifra il nonce con la sua chiave privata, a sua volta crea un nonce random  $n_B$  ed invia ad  $A$   $n_A$  ed  $n_B$  cifrando il messaggio con la chiave pubblica di  $A$ . In questo modo, solo  $A$  potrà leggere  $M4$ ;

**M5** :  $A$  decifra il nonce  $n_A$  e controlla che corrisponda a quello che ha inviato. In caso affermativo, reinvia a  $B$  il nonce  $n_B$ , dopo averlo decifrato con la sua chiave privata. Infine,  $A$  genera la chiave di sessione.  $B$  verifica che il nonce  $n_B$  ricevuto corrisponda a quello precedente inviato. In tal caso, procede a generare la chiave di sessione.

La chiave di sessione viene generata a partire dai nonce  $n_A$  ed  $n_B$ . In questo modo entrambe le parti partecipano alla creazione della chiave, quindi non è necessario che i peer facciano assunzioni sulla capacità della controparte di generare nonce realmente freschi. L'algoritmo di generazione della chiave di sessione è il seguente:

$$\begin{cases} y = h(n_A || n_B) \\ K_{AB} = T(y) \end{cases}$$

dove per  $T(y)$  si intendone il troncamento di  $y$  ai suoi primi 128 bit.

## Capitolo 2

# Certification Authority

Per realizzare la creazione dei certificati e delle chiavi pubbliche e private, abbiamo implementato una piccola *Certification Authority*. La Certification Authority si autocertifica, dopodiché genera i certificati e le chiavi per i peer. I certificati e le chiavi pubbliche e private vengono salvati su file in modo da poter essere recuperati dai peer corrispondenti.





## Capitolo 3

# Analisi del protocollo di scambio chiavi

### 3.1 *Beliefs* da ottenere

Procediamo ad analizzare il protocollo esposto nel capitolo 1. Si vuole provare che il protocollo produce, in ciascuna delle parti, i seguenti *beliefs*:

	<b>A</b>	<b>B</b>
<b>key authentication</b>	$A \models A \xleftrightarrow{K} B$	$B \models A \xleftrightarrow{K} B$
<b>key confirmation</b>	$A \models B \models A \xleftrightarrow{K} B$	$B \models A \models A \xleftrightarrow{K} B$
<b>key freshness</b>	$A \models \# \left( A \xleftrightarrow{K} B \right)$	$B \models \# \left( A \xleftrightarrow{K} B \right)$

### 3.2 Protocollo idealizzato

Viene riportato, qui di seguito, il *protocollo idealizzato* relativo al protocollo di scambio delle chiavi esposto nel capitolo 1.

$$\begin{aligned}
 M1 : A \rightarrow B & \quad \left\{ \overset{K_A^+}{\mapsto} A, L_A \right\}_{K_T^-} \\
 M2 : A \leftarrow B & \quad \left\{ \overset{K_B^+}{\mapsto} B, L_B \right\}_{K_T^-} \\
 M3 : A \rightarrow B & \quad \left\{ n_A, A \xleftrightarrow{n_A} B \right\}_{K_B^+} \\
 M4 : A \leftarrow B & \quad \left\{ n_A, n_B, A \xleftrightarrow{\langle n_A \rangle n_B} B \right\}_{K_A^+} \\
 M5 : A \rightarrow B & \quad \left\{ n_B, A \xleftrightarrow{\langle n_A \rangle n_B} B \right\}_{K_B^+}
 \end{aligned}$$

### 3.3 Ipotesi

Vengono esplicitate, qui di seguito, le ipotesi sotto le quali il protocollo viene eseguito.

	<b>A</b>	<b>B</b>
<b>public keys</b>	$A \models \overset{K_A^+}{\mapsto} A$	$B \models \overset{K_B^+}{\mapsto} B$
<b>third party</b>	$A \models \overset{K_T^+}{\mapsto} T$	$B \models \overset{K_T^+}{\mapsto} T$
	$A \models T \Rightarrow \overset{K_X^+}{\mapsto} X$	$B \models T \Rightarrow \overset{K_X^+}{\mapsto} X$
<b>freshness</b>	$A \models \#(n_A)$	$B \models \#(n_B)$
	$A \models \#(L_B)$	$B \models \#(L_A)$

### 3.4 Analisi dei *beliefs*

Procediamo, ora, con l'analisi dei singoli messaggi. Partendo dalle ipotesi esposte nella sezione 3.3 e applicando le *regole di inferenza* della logica BAN, ciascuna parte può ampliare l'insieme dei propri *beliefs*. Se, tra i beliefs finali, compaiono quelli elencati nella sezione 3.1, allora possiamo affermare che il protocollo esposto è corretto.

#### 3.4.1 Messaggio *M1*

Messaggio *M1*:

$$M1 : A \rightarrow B \quad \left\{ \overset{K_A^+}{\mapsto} A, L_A \right\}_{K_T^-}$$

per la *meaning rule*

$$\frac{B \models \overset{K_T^+}{\mapsto} T, B \triangleleft \left\{ \overset{K_A^+}{\mapsto} A, L_A \right\}_{K_T^-}}{B \models T \sim \left( \overset{K_A^+}{\mapsto} A, L_A \right)}$$

e poiché

$$\frac{B \models \#(L_A)}{B \models \# \left( \overset{K_A^+}{\mapsto} A, L_A \right)}$$

allora, per la *nonce verification rule*

$$\frac{B \models T \sim \left( \overset{K_A^+}{\mapsto} A, L_A \right), B \models \# \left( \overset{K_A^+}{\mapsto} A, L_A \right)}{B \models T \models \left( \overset{K_A^+}{\mapsto} A, L_A \right)}$$

e, in particolare,

$$B \models T \models \overset{K_A^+}{\mapsto} A$$

infine, per la *jurisdiction rule*

$$\frac{B \models T \models \overset{K_A^+}{\mapsto} A, B \models T \Rightarrow \overset{K_A^+}{\mapsto} A}{B \models \overset{K_A^+}{\mapsto} A}$$

### 3.4.2 Messaggio M2

In maniera del tutto analoga a quanto visto per il messaggio M1, il *belief* ottenuto da A dopo aver ricevuto il messaggio M2 è

$$A \models \overset{K_B^+}{\mapsto} B$$

### 3.4.3 Messaggio M3

Messaggio M3:

$$M3 : A \rightarrow B \quad \left\{ n_A, A \overset{n_A}{\rightleftharpoons} B \right\}_{K_B^+}$$

L' applicazione delle regole di inferenza non porta, su B, alla realizzazione di alcun nuovo belief. Tuttavia, poiché l' unica entità in grado di leggere il nonce  $n_A$  è  $B^1$ , A può ritenere che

$$A \models A \overset{n_A}{\rightleftharpoons} B$$

### 3.4.4 Messaggio M4

Messaggio M4:

$$M4 : A \leftarrow B \quad \left\{ n_A, n_B, A \overset{\langle n_A \rangle_{n_B}}{\longleftarrow} B \right\}_{K_A^+}$$

L' unica entità in grado di leggere il messaggio M4 è  $A^2$ . Pertanto, B può ritenere che

$$B \models A \overset{\langle n_A \rangle_{n_B}}{\longleftarrow} B \quad B \text{ ottiene key authentication}$$

inoltre,

$$\frac{B \models \#(n_B)}{B \models \# \left( A \overset{\langle n_A \rangle_{n_B}}{\longleftarrow} B \right)} \quad B \text{ ottiene key freshness}$$

---

<sup>1</sup>B, infatti, è l' unica entità a possedere la chiave  $K_B^-$  necessaria per decifrare i messaggi cifrati con  $K_B^+$ .

<sup>2</sup>A, infatti, è l' unica entità a possedere la chiave  $K_A^-$  necessaria per decifrare i messaggi cifrati con  $K_A^+$ .

### 8 CAPITOLO 3. ANALISI DEL PROTOCOLLO DI SCAMBIO CHIAVI

per quanto riguarda  $A$ , invece, otteniamo

$$\frac{A \models \#(n_A)}{A \models \# \left( n_A, n_B, A \xleftarrow{\langle n_A \rangle_{n_B}} B \right)}$$

e poiché, per la *meaning rule*

$$\frac{A \models A \xrightarrow{n_A} B, A \triangleleft \left( n_A, n_B, A \xleftarrow{\langle n_A \rangle_{n_B}} B \right)}{A \models B \sim \left( n_A, n_B, A \xleftarrow{\langle n_A \rangle_{n_B}} B \right)}$$

allora, per la *nonce verification rule*

$$\frac{A \models B \sim \left( n_A, n_B, A \xleftarrow{\langle n_A \rangle_{n_B}} B \right), A \models \# \left( n_A, n_B, A \xleftarrow{\langle n_A \rangle_{n_B}} B \right)}{A \models B \models \left( n_A, n_B, A \xleftarrow{\langle n_A \rangle_{n_B}} B \right)}$$

e in particolare

$$A \models B \models A \xleftarrow{\langle n_A \rangle_{n_B}} B \quad A \text{ ottiene key confirmation}$$

#### 3.4.5 Messaggio $M5$

Messaggio  $M5$ :

$$M5 : A \rightarrow B \quad \left\{ n_B, A \xleftarrow{\langle n_A \rangle_{n_B}} B \right\}_{K_B^+}$$

L' unica entità in grado di leggere il messaggio  $M5$  è  $B$ . Pertanto,  $A$  può ritenere che

$$A \models A \xleftarrow{\langle n_A \rangle_{n_B}} B \quad A \text{ ottiene key authentication}$$

inoltre,

$$\frac{A \models \#(n_A)}{A \models \# \left( A \xleftarrow{\langle n_A \rangle_{n_B}} B \right)} \quad A \text{ ottiene key freshness}$$

per quanto riguarda  $B$ , invece, otteniamo

$$\frac{B \models \#(n_B)}{B \models \# \left( n_B, A \xleftarrow{\langle n_A \rangle_{n_B}} B \right)}$$

e poiché, per la *meaning rule*

$$\frac{B \models A \xrightarrow{\langle n_A \rangle_{n_B}} B, \quad B \triangleleft \left( n_B, A \xrightarrow{\langle n_A \rangle_{n_B}} B \right)}{B \models A \sim \left( n_B, A \xrightarrow{\langle n_A \rangle_{n_B}} B \right)}$$

allora, per la *nonce verification rule*

$$\frac{B \models A \sim \left( n_B, A \xrightarrow{\langle n_A \rangle_{n_B}} B \right), \quad B \models \# \left( n_B, A \xrightarrow{\langle n_A \rangle_{n_B}} B \right)}{B \models A \models \left( n_B, A \xrightarrow{\langle n_A \rangle_{n_B}} B \right)}$$

e in particolare

$$B \models A \models A \xrightarrow{\langle n_A \rangle_{n_B}} B \quad B \text{ ottiene key confirmation}$$



## Capitolo 4

# Installazione

Installazione ed esecuzione:

- Installare il provider Bouncy Castle secondo la procedura seguente:

- scaricare il package BC dal sito di riferimento

`http://www.bouncycastle.org/latest\_releases.html`);

- copiare il package nella directory `<JAVA_HOME>/jre/lib/ext`;
- modificare il file `<JAVA_HOME>/jre/lib/security/java.security` aggiungendo, come ultimo elemento della lista che ha il formato

`security.provider.n=provider`

la riga

`security.provider.n+1=org.bouncycastle.jce.provider.BouncyCastleProvider`

- scompattare l'archivio `progetto_sicurezza_sacco_silvestri.jar`
- eseguire Trent
- eseguire Peer

