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

# 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:

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

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:

$$\begin{cases} y = h(n_A||n_B) \\ K_{AB} = \overleftarrow{T_{128bit}}(y) \end{cases}$$

dove per $\overleftarrow{T_{128bit}}(y)$ si intende il troncamento di $y$ ai suoi 128 *bit* più significativi.

# 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*:

| | **A** | **B** |
|---|---|---|
| **key authentication** | $A \models A \overset{K}{\leftrightarrow} B$ | $B \models A \overset{K}{\leftrightarrow} B$ |
| **key confirmation** | $A \models B \models A \overset{K}{\leftrightarrow} B$ | $B \models A \models A \overset{K}{\leftrightarrow} B$ |
| **key freshness** | $A \models \sharp\left(A \overset{K}{\leftrightarrow} B\right)$ | $B \models \sharp\left(A \overset{K}{\leftrightarrow} B\right)$ |

## 3.2 Protocollo idealizzato

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

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

$$M2 : A \leftarrow B \quad \left\{ \overset{K_B{}^+}{\mapsto} B,\ L_B \right\}_{K_T{}^-}$$

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

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

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

Dove $L_A$ ed $L_B$ rappresentano, rispettivamente, i periodi di validità del certificato di $A$ e di quello di $B$. Dei certificati, nel protocollo idealizzato,

vengono riportati solo i campi strettamente necessari (chiave pubblica e periodo di validità).

## 3.3 Ipotesi

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

| | **A** | **B** |
|---|---|---|
| **public keys** | $A \models \overset{K_A^+}{\mapsto} A$ | $B \models \overset{K_B^+}{\mapsto} B$ |
| **third party** | $A \models \overset{K_T^+}{\mapsto} T$<br>$A \models T \Rightarrow \overset{K_X^+}{\mapsto} X$ | $B \models \overset{K_T^+}{\mapsto} T$<br>$B \models T \Rightarrow \overset{K_X^+}{\mapsto} X$ |
| **freshness** | $A \models \sharp(n_A)$<br>$A \models \sharp(L_B)$ | $B \models \sharp(n_B)$<br>$B \models \sharp(L_A)$ |
| **secrets** | $A \models A \overset{n_A}{\leftrightharpoons} B$ | $B \models A \overset{n_B}{\leftrightharpoons} B$ |

## 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*.

### 3.4.1 Messaggio $M1$

Messaggio $M1$:

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

per la *meaning rule*

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

allora, per la *nonce verification rule*

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

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 $M3$:

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

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], l' ipotesi

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

risulta essere ben fondata.

### 3.4.4 Messaggio $M4$

Messaggio $M4$:

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

L' unica entità in grado di leggere il messaggio $M4$ è $A$[2]. Pertanto, poiché l' ipotesi

$$B \models A \overset{n_B}{\leftrightharpoons} B$$

risulta ben fondata, $B$ può ritenere che

$$B \models A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B \qquad B \textit{ ottiene key authentication}$$

[1] $B$, infatti, è l' unica entità a possedere la chiave $K_B{}^-$ necessaria per decifrare i messaggi cifrati con $K_B{}^+$.

[2] $A$, infatti, è l' unica entità a possedere la chiave $K_A{}^-$ necessaria per decifrare i messaggi cifrati con $K_A{}^+$.

inoltre,

$$\frac{B \models \sharp(n_B)}{B \models \sharp\left(A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B\right)} \qquad B \textit{ ottiene key freshness}$$

per quanto riguarda $A$, invece, otteniamo, per la meaning rule

$$\frac{A \models A \overset{n_A}{\rightleftharpoons} B,\ A \triangleleft \left(n_A,\ n_B,\ A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B\right)}{A \models B \mid\sim \left(n_A,\ n_B,\ A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B\right)}$$

allora, per la *nonce verification rule*

$$\frac{A \models B \mid\sim \left(n_A,\ n_B,\ A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B\right),\ A \models \sharp(n_A)}{A \models B \models \left(n_A,\ n_B,\ A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B\right)}$$

e in particolare

$$A \models B \models A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B \qquad A \textit{ ottiene key confirmation}$$

### 3.4.5 Messaggio $M5$

Messaggio $M5$:

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

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

$$A \models A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B \qquad A \textit{ ottiene key authentication}$$

inoltre,

$$\frac{A \models \sharp(n_A)}{A \models \sharp\left(A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B\right)} \qquad A \textit{ ottiene key freshness}$$

per quanto riguarda $B$, invece, otteniamo

$$\frac{B \models \sharp(n_B)}{B \models \sharp\left(n_B,\ A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B\right)}$$

e poiché, per la *meaning rule*

$$\frac{B \models A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B,\ B \triangleleft \left( n_B,\ A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B \right)}{B \models A \mid\sim \left( n_B,\ A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B \right)}$$

allora, per la *nonce verification rule*

$$\frac{B \models A \mid\sim \left( n_B,\ A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B \right),\ B \models \sharp \left( n_B,\ A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B \right)}{B \models A \models \left( n_B,\ A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B \right)}$$

e in particolare

$$B \models A \models A \overset{\langle n_A \rangle_{n_B}}{\longleftrightarrow} B \qquad B \textit{ ottiene key confirmation}$$

# 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`
- copiare il contenuto della cartella `file di configurazione` nella directory `progetto_sicurezza_sacco_silvestri` appena creata
- eseguire Trent
- eseguire Peer