Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Studio ed Implementazione di Sistemi di Autenticazione Biometrici in Piattaforme Multi-Agente

S. Vitabile, G. Pilato, G. Gioè, V. Conti, F. Sorbello

RT-ICAR-PA-07-04 **Aprile 2004**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sezione di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Studio ed Implementazione di Sistemi di Autenticazione Biometrici in Piattaforme Multi-Agente

S. Vitabile[1], G. Pilato[1], G. Gioè[2], V. Conti[2], F. Sorbello [1- 2]

Rapporto Tecnico N.4: **RT-ICAR-PA-07-04**

Data: **Aprile 2004**

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sezione di Palermo
[2] Università degli Studi di Palermo Dipartimento di Ingegneria Informatica

Indice

CAPITOLO 1 ..... **3**

LA PIATTAFORMA JADE **3**

1.1 Introduzione *3*
1.2 Caratteristiche di JADE *4*
1.3. JADE-S 6
1.3.1 Il modello di sicurezza della piattaforma JADE-S *7*
1.3.2 Autorità per la certificazione *9*
1.3.2.1 Autenticazione *9*
1.3.2.2 Autorizzazione *10*
1.3.2.3 Integrità dei messaggi e confidenzialità *10*

CAPITOLO 2 ..... **11**

USO DELLE CARATTERISTICHE BIOMETRICHE **11**

NEI SISTEMI MULTI - AGENTE **11**

2.1 INTRODUZIONE 11
2.2 SISTEMI DI AUTENTICAZIONE BIOMETRICI 13
2.2.1 I componenti di un'infrastruttura biometrica *13*
2.2.2 Parametri di Valutazione dei sistemi biometrici *13*
2.2.3. Esempi di utilizzo del riconoscimento biometrico *14*
2.2.4. Principali sistemi di riconoscimento biometrico *14*
2.2.4.1 Riconoscimento tramite retina 14
2.2.4.2 Riconoscimento tramite impronta digitale 14
2.2.4.3 Riconoscimento tramite geometria della mano 15
2.2.4.4 Riconoscimento tramite iride 15
2.3 IL SISTEMA DI AUTENTICAZIONE DELLA PIATTAFORMA IMPLEMENTATA 15
2.4 MODULI AGGIUNTIVI SVILUPPATI E IMPLEMENTATI 18
2.4.1 IL LOGIN MODULE 18
2.4.2 USO DEI CERTIFICATI X-SECURITY PER L'AUTENTICAZIONE 23
2.4.2.1 Struttura dei certificati *25*

CAPITOLO 3 ..... **27**

VALIDAZIONE SPERIMENTALE **27**

3.1 INTRODUZIONE 27
3.2 L'AMBIENTE DI SVILUPPO 27
3.3 IL SENSORE 27
3.4 LA PROCEDURA DI ACCESSO 28
3.5 AVVIO DELLA PIATTAFORMA 28
3.6 AVVIO DI UN AGENTE IN UNA PIATTAFORMA 30
3.7 COMUNICAZIONE TRA GLI AGENTI 31
3.8 CONSIDERAZIONI SULL'USO DELLE IMPRONTE DIGITALI 34

CONCLUSIONI **35**

BIBLIOGRAFIA **37**

Capitolo 1

La Piattaforma JADE

1.1 Introduzione

JADE (Java Agent DEvelopement framework) [10][15] è stato sviluppato da TILab S.p.A in collaborazione con l'Università di Parma. L'ultima versione è del 26 luglio 2004. È basato sulla versione 1.4 di Java.

Non sono presenti strumenti di sviluppo, mentre il supporto runtime è molto buono: dall'interfaccia grafica di Jade (che è essa stessa un agente, chiamato RMA, Remote Management Agent) è possibile gestire e monitorare anche gli agenti che risiedono su altre piattaforme. Uno strumento estremamente utile per il collaudo di un sistema è l'agente Sniffer, che analizza le conversazioni che avvengono all'interno del sistema: l'utente sceglie gli agenti che desidera monitorare, e lo Sniffer traccia in tempo reale un diagramma di sequenza UML comprendente tutti i messaggi scambiati da quegli agenti.

La documentazione della libreria di classi è molto buona. Il framework JADE è FIPA-compliant [10].
Per mettere il luce le potenzialità di JADE riportiamo alcune tra le principali caratteristiche che lo contraddistinguono:

- Piattaforma per agenti distribuita. La piattaforma può essere ripartita tra numerosi host, purché essi siano raggiungibili tramite RMI (Remote Method Invocation). Su ciascun host viene eseguita una sola applicazione Java, e quindi è sufficiente una sola macchina virtuale. Gli agenti sono implementati come thread Java e risiedono in *container* di agenti, i quali forniscono l'infrastruttura di servizi necessaria all'esecuzione degli agenti.
- Interfaccia grafica per gestire molteplici agenti e container anche attraverso un collegamento da un host remoto.
- Strumenti di debugging per aiutare nello sviluppo di applicazioni multi-agente basate su JADE.
- Mobilità intra-piattaforma degli agenti. Gli agenti possono spostarsi da un container ad un altro all'interno della stessa piattaforma. È supportata la mobilità del codice e dello stato interno degli agenti.
- Supporto per la gestione di numerose attività concorrenti degli agenti, eseguite in parallelo secondo il modello a *comportamenti* o *behaviour*. Lo scheduling tra i comportamenti viene effettuato trasparentemente da JADE in maniera non-preemptive.
- Piattaforma ad agenti conforme agli standard FIPA. Sono inclusi un agente *AMS* (*Agent Management System*), uno o più agenti *DF* (*Directory Facilitator*), ed una implementazione di *ACC* (*Agent Communication Channel*). Queste tre componenti sono tutte attivate automaticamente durante la fase di avvio della piattaforma.
- Sistema di trasferimento dei messaggi conforme agli standard FIPA. Per connettere differenti piattaforme ad agenti viene utilizzato il protocollo IIOP.
- Per implementare applicazioni multi-dominio possono essere istanziati numerosi DF. Un dominio è inteso come un insieme logico di agenti, i cui servizi sono

pubblicizzati attraverso un servizio condiviso. Ogni DF eredita una interfaccia grafica e tutte le normali caratteristiche definite da FIPA, quindi la possibilità di registrare, cancellare, modificare e cercare descrizioni di agenti; inoltre è offerta la possibilità di creare una rete di DF tra loro federati.

- Trasporto efficiente di messaggi ACL all'interno della stessa piattaforma. Infatti i messaggi durante il trasferimento sono codificati come oggetti Java, piuttosto che come stringhe. Solo quando risulta realmente necessario, ad esempio quando un messaggio viene inviato al di fuori della piattaforma, allora il messaggio viene automaticamente convertito in sintassi, codifica e protocollo di trasporto conformi agli standard FIPA. Lo stesso accede quando viene ricevuto un messaggio dall'esterno. Entrambe queste conversioni sono trasparenti agli sviluppatori di agenti, che devono occuparsi solo di oggetti Java.
- Implementazione completa di una libreria di protocolli di interazione FIPA.
- Registrazione e cancellazione automatica degli agenti presso l'AMS.
- Servizio di naming conforme alle specifiche FIPA: all'avvio gli agenti ottengono dalla piattaforma il proprio identificatore *GUID* (*Global Inique Identifier*).
- Supporto alle applicazioni per la definizione e l'uso di nuovi linguaggi e ontologie per le comunicazioni.

L'ambiente di JADE si avvia tramite la classe *Boot*, e un agente può essere creato attraverso l'interfaccia grafica RMA, oppure da un altro agente: un particolare interessante è che un agente che ne crea un altro non riceve mai indietro un riferimento esplicito all'oggetto-agente creato (questo garantisce una maggiore indipendenza tra gli agenti). JADE supporta bene la creazione di più agenti nella stessa Virtual Machine. Un agente può creare un proprio log tramite il metodo *write*().
Ad ogni agente è assegnato un thread (Agent implementa Runnable), ai behaviour no: l'esecuzione concorrente dei behaviour è gestita da uno scheduler interno all'agente (nascosto all'utente). Il codice principale di un behaviour è contenuto nel metodo *action*().
In JADE individuiamo due parti:

- Una piattaforma FIPA compatibile
- Un pacchetto rappresentato da una gerarchia di classi per lo sviluppo degli agenti.

1.2 Caratteristiche di JADE

La piattaforma di JADE è completamente compatibile con le specifiche FIPA quindi in essa sono presenti i componenti obbligatori di cui si è discusso nella parte relativa alle specifiche FIPA; quindi avremo [10][15]:

- L'agent Management System (AMS) che si occupa della supervisione e del controllo degli accessi alla piattaforma.
- Il Directory Facilitator (DF) che offre un servizio di "pagine gialle", in cui gli agenti si registrano.
- Agent Communication Channel (ACC) che si occupa di visionare il traffico di messaggi di una piattaforma, sia intra-platform che inter-platform.

All’avvio di una piattaforma vengono creati automaticamente Figura 1.1 i componenti AMS e DF e viene impostato l’ACC per la trasmissione di messaggi.

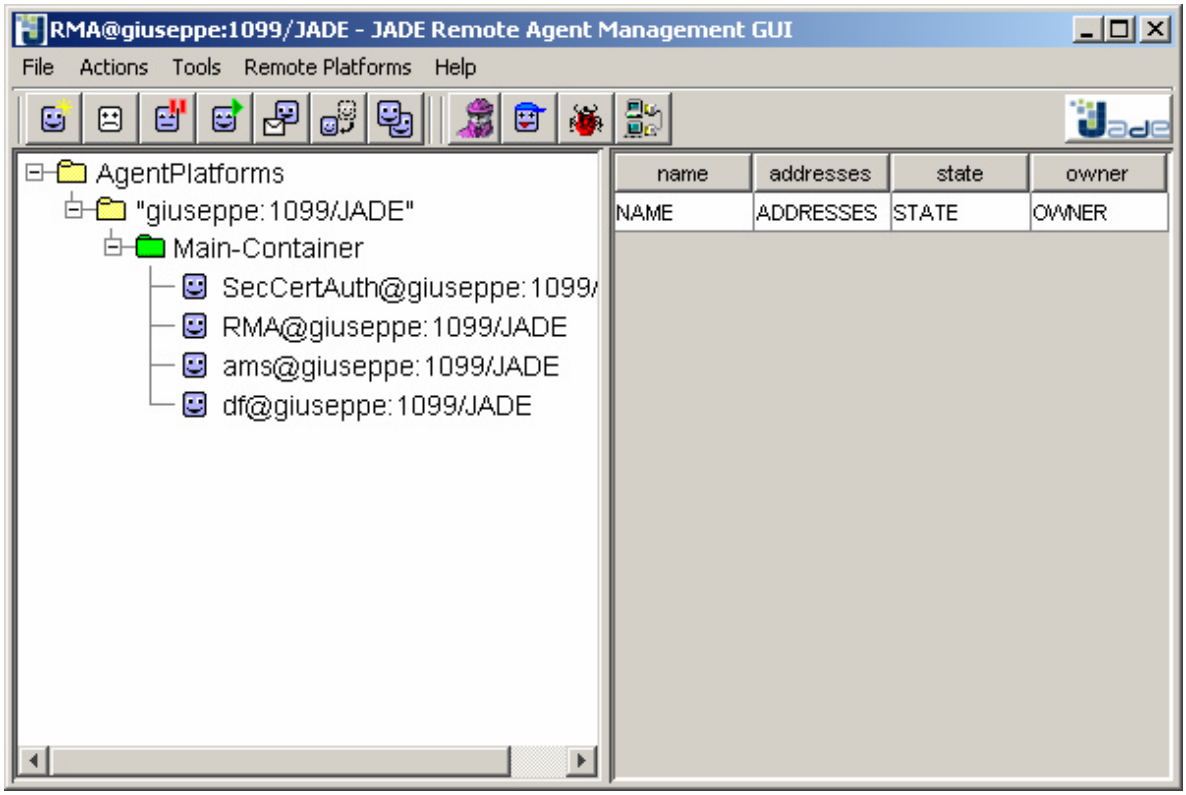


Figure 1.1: Piattaforma JADE aperta, con gli agenti di default, ams, df, RMA

.
La piattaforma JADE può essere suddivisa su diversi host ed in esso possono essere eseguite più di una JVM (Java Virtual Machine) ma per le numerose problematiche che ciò comporta, è consigliabile eseguire una sola JVM su ogni host. Tra le problematiche che comporta l’esecuzione di più JVM su uno stesso host annoveriamo:

- La CPU dell’Host va in sovraccarico.
- La memoria non e’ condivisa.
- Le comunicazioni sono lente.
- I vari agenti non si possono indirizzare con il solo nickname, ma occorre fornire anche il platform ID ed almeno un indirizzo di trasporto.
- Se il server ha un crash, muoiono tutti gli agenti perché comunque girano sulla stessa macchina

Possiamo pensare la piattaforma JADE come costituita da uno o più contenitori di agenti, i cosiddetti Agent Container, a cui corrisponde una JVM..
Esistono differenti Agent Container, uno per ogni host; questi sono legati ad un unico main container chiamato FRONT-END, uno per singola piattaforma, che si trova nell’host dove risiedono DF, AMS, ACC.

1.3. JADE-S

I sistemi distribuiti multi-agente, in ambienti aperti, richiedono un elevato livello di sicurezza sia a livello di infrastruttura che a livello di applicazione. I sistemi multi-agente fanno forza sull'autonomia e sulla mobilità degli agenti, richiedendo maggiore attenzione sul problema sicurezza [11][12][14]. In funzione dello scenario per cui un sistema multi-agente è sviluppato, ci sono un certo numero di minacce di sicurezza di cui bisogna tenere conto; come ad esempio: un agente maligno potrebbe uccidere un agente e prenderne il posto, accettando così ordini e pagamenti da utenti ignari, che pensano di trattare con l'agente vero; oppure potrebbe introdursi in un contenitore remoto della piattaforma e leggere o cancellare tutti i file.
La piattaforma JADE-S v2 [13][17] utilizzata in questo lavoro assicura all'utente un certo livello di sicurezza, introducendo caratteristiche come:

- autenticazione di utente/agente,
- autorizzazione
- comunicazione sicura tra agenti di piattaforme diverse.

In JADE-S ogni componente, ad esempio un agente o un contenitore, appartiene ad un utente che è responsabile delle sue azioni, ogni utente può possedere diversi agenti in differenti container come mostrato in Figura 1.2. Ogni utente della piattaforma deve essere abilitato fornendo nome utente e password, dopo l'avvenuta identificazione può prendere possesso di un componente.

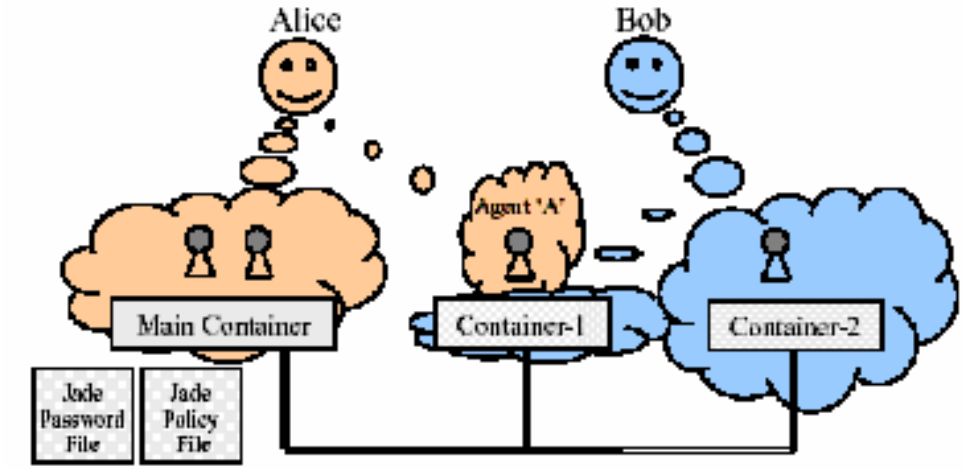


Figure 1.2: Uno scenario multi-utente. L'utente 'Alice' possiede il Main-Container, due agenti su di esso e l'agente 'A'. L'utente 'Bob' possiede il Container-1, Container2 e gli agenti su di esso.

Non tutti gli utenti possono eseguire tutte le azioni disponibili sulla piattaforma, le informazioni su quali azioni privilegiate possono essere eseguite da utenti autorizzati sono contenute sul file di policy [13].
La versione 2 rilasciata il 16 luglio 2004 ha le seguenti limitazioni:

- Non è prevista la mobilità dei permessi. Come conseguenza, per essere sicura una piattaforma basata su JADE-S V2 non dovrebbe supportare la mobilità. Per

impedire la mobilità è sufficiente non avviare l'Agent Mobility Service quando viene avviata la piattaforma;

- Pezzi di informazioni scambiati tra contenitori (chiamati comandi orizzontali) sono trasferiti su un canale sicuro (SSL) ma non sono firmati. Un agente malizioso o un hacker potrebbe generare un falso comando orizzontale per forzare un container remoto a comportarsi in un modo differente dall'intento originale;
- Molte caratteristiche di sicurezza non sono accessibili dagli agenti eseguiti sui dispositivi MIDP.

Il rilascio della versione 3 con l'aggiunta delle caratteristiche prima citate è previsto entro la fine dell'anno 2004.

1.3.1 Il modello di sicurezza della piattaforma JADE-S

IL modello di sicurezza di Jade è basato sui seguenti concetti [13]:

- **Principal**

Nel modello di sicurezza previsto per JADE un *principal* rappresenta una qualsiasi entità la cui identità può essere autenticata. Nella maggior parte delle volte sono associati a singole persone, anche se possono anche rappresentare entità come dipartimenti, intere compagnie o ogni altra unità autenticabile. Inoltre in JADE anche i singoli agenti vengono associati ad un *principal*, il cui nome è quello assegnato dal sistema all'agente stesso; rispetto ai propri agenti, l'utente costituisce un gruppo di appartenenza, rendendo possibile concedere particolari permessi a tutti gli agenti lanciati da uno stesso utente.
D'altra parte anche gli utenti possono essere strutturati in gruppi, realizzando gerarchie anche complesse e rendendo più flessibile la gestione delle politiche di sicurezza.

- **Risorse**

Tra le *risorse* annoveriamo il file system locale, gli accessi di rete, le variabili d'ambiente, gli accessi alle basi di dati; inoltre esistono delle risorse proprie dei sistemi ad agenti che devono essere protette da accessi non autorizzati quali: gli agenti stessi ed i container.
È possibile associare a queste *risorse* particolari permessi che rappresentano operazioni come creare un nuovo agente, clonarne uno già esistente, terminarlo o inviarlo in una locazione diversa. Sia le *risorse* dalla sicurezza di Java che quelle introdotte da JADE sono organizzate gerarchicamente in modo da poter facilmente essere raggruppate nella concessione dei permessi.
Tra le *risorse* disponibili non compaiono il tempo di CPU, lo spazio occupato in memoria e lo spazio occupato su disco. La protezione di queste *risorse* infatti non risulta possibile nell'ambiente di esecuzione di una applicazione Java se non modificando le specifiche e l'implementazione della macchina virtuale, il che ovviamente esula dagli obbiettivi prefissati in questo contesto.

- **Permessi**

Un *permesso* è un oggetto che attribuisce la capacità di eseguire azioni in un sistema.
In particolare i *permessi* di JADE-S, ereditati dal modello della sicurezza di Java, rappresentano le risorse del sistema. Tutti i *permessi* hanno un nome e molti includono anche un elenco delle azioni che sono possibili sull'oggetto.
Per prendere una decisione durante il tentativo di accesso ad una *risorsa*, le funzioni di controllo di accesso confrontano i *permessi* concessi al *principal* con i *permessi* richiesti per eseguire l'operazione; l'accesso viene consentito se si è in possesso di tutti i *permessi* richiesti.

- **Amministrazione**

Uno strumento utile per facilitare *l'amministrazione* della sicurezza di una organizzazione è la delega di diritti di *amministrazione* applicabili solo a precisi sottoinsiemi dell'intero dominio dell'organizzazione. Il requisito importante è poter concedere il diritto di gestire un piccolo insieme di utenti o gruppi all'interno della propria area di responsabilità e, allo stesso tempo, non dare il *permesso* di gestire oggetti in altre aree dell'organizzazione.
Questo è possibile in quanto la presenza di *risorse* organizzate in una gerarchia rende facile assegnare ad un amministratore la responsabilità della gestione di particolari gruppi o insiemi di *risorse*. Inoltre il fatto che non sia possibile in alcun modo creare oggetti o utenti con un insieme di diritti d'accesso più grande di quello posseduto, garantisce che le eventuali ulteriori deleghe di *amministrazione* restino confinate al dominio di competenza assegnato.

- **Ereditarietà**

L'ereditarietà dei diritti di accesso specifica come l'informazione di controllo di accesso definita ai livelli più alti della gerarchia si propaghi verso il basso fino ai singoli agenti. Questo riguarda in particolare il concetto di *ruolo*, visto come insieme di *permessi* che possono essere assegnati unitariamente ad un gruppo di utenti o agenti. Ci sono generalmente due modelli per implementare *l'ereditarietà* dei diritti d'accesso: *ereditarietà statica* e *dinamica*.
L'ereditarietà dinamica determina gli effettivi diritti d'accesso di un oggetto valutando i *permessi* definiti esplicitamente sull'oggetto e quelli definiti per tutti gli oggetti superiori nella gerarchia. Questo garantisce la flessibilità di modificare i diritti d'accesso su porzioni dell'albero di gerarchia facendo cambi su uno specifico nodo che automaticamente influenzano tutti i nodi e gli oggetti sottostanti. Il compromesso per questa flessibilità è il costo in termini di prestazioni richiesto per calcolare i diritti di accesso effettivi ogni volta che un agente richieda un servizio su uno specifico oggetto della gerarchia.
Il modello della sicurezza di JADE prevede una forma *statica* di *ereditarietà* dei diritti di accesso definita al momento della creazione degli oggetti. Ogni cambio ai diritti di accesso fatto successivamente nei livelli più alti della gerarchia deve essere esplicitamente applicato a tutti gli oggetti che devono essere influenzati.

1.3.2 Autorità per la certificazione

La sicurezza di JADE si fonda in gran parte sulla disponibilità di un certo numero di *certificati*, i quali possono attestare l'identità di una entità oppure i permessi ad essa concessi. Affinché sia possibile verificare l'autenticità di tali *certificati* è necessario che questi vengano firmati da una autorità pubblicamente riconosciuta.

1.3.2.1 Autenticazione

L'autenticazione fornisce una garanzia che un utente che avvia una piattaforma JADE, e con ciò genera container e agenti all'interno della piattaforma, è considerato legittimo in relazione alla politica di sicurezza del sistema di calcolo ospitante il contenitore principale della piattaforma. Legittimo nel caso del processo di autenticazione JADE implica che l'utente è noto al sistema avendo almeno una identità valida con password associata.
In generale, un sistema di autenticazione è composto da due elementi: un "CallbackHandler" che consente agli utenti di fornire la propria username e password e un "LoginModule" che controlla se la username e la password sono valide.
Il sistema di autenticazione di JADE-S è basato su JAAS (Java Autentication and Authorization Service), API che abilitano ad applicare differenti sistemi di controllo sui sistemi utenti. Le API di JAAS forniscono un insieme di base di modalità di autenticazione, basati su un domain controller Unix, su un domain controller NT e su Kerberos. I metodi di autenticazione legati a Unix e NT sono dipendenti dal sistema operativo e sono stati sviluppati per utilizzare l'identità dell'utente prelevandola direttamente dalla sua sessione attuale aperta nel sistema operativo. Il metodo di autenticazione che fa uso di Kerberos è, invece, indipendente dal sistema operativo, ma richiede specifiche configurazioni prima di potere essere utilizzato. Oltre ai metodi prima citati è previsto un LoginModule che fa uso solamente di username e password memorizzati in un database. In questo lavoro, il modulo di autenticazione sfrutta il package CSAI, realizzato presso il Laboratorio CSAI, che utilizza le impronte digitali per l'autenticazione degli utenti.
La coppia di credenziali username/password può essere introdotta in diversi modi di seguito riportati:

- **Cmdline** – username e password sono specificate nei parametri di configurazione di JADE, entrambi come parametri a linea di comando (-owner user:pass), oppure nel file di configurazione (owner=user:pass).
- **Text** – durante l'avvio il container richiede l'inserimento dello username e della password;
- **Dialog** – all'avvio del container compare una finestra di dialogo dove l'utente può inserire lo username e la password.

Se si verifica un problema durante l'autenticazione o l'utente non supera correttamente l'autenticazione allora il sistema esce e genera un appropriato messaggio di errore.

1.3.2.2 Autorizzazione

Grazie ai meccanismi di autenticazione descritti nella sezione precedente, una piattaforma basata su JADE-S diventa un sistema multi-utente dove tutti i componenti (contenitori e agenti) sono legati ad un utente autenticato. Tutte le azioni che gli agenti possono eseguire nella piattaforma sono concesse o negate in accordo ad un insieme di regole. In questo modo è possibile accordare accessi ai servizi della piattaforma in maniera selettiva. Questo insieme di regole è di solito descritto in un file denominato "policy.txt", il quale segue la sintassi di JAVA/JAAS [16], ma è usato per estendere il modello di sicurezza di Java al fine di ottenere flessibilità in una scenario distribuito basato su agenti.
Analogamente all'architettura JADE, che include un contenitore principale e diversi contenitori secondari, possono essere usati due tipi di file di policy per concedere permessi agli agenti.

1. Il file della policy relativo al contenitore principale che specifica i permessi relativi alla piattaforma come "gli agenti legati all'utente Bob possono uccidere gli agenti legati all'utente Alice".
2. Il file della policy per i contenitori periferici (uno per contenitore) che specificano permessi specifici sui contenitori (come "agenti legati all'utente Bob possono uccidere gli agenti legati all'utente Alice nel contenitore locale").

I file delle policy relativi ai contenitori inoltre regolano l'accesso alle risorse locali (JVM, file system, network, etc..).

1.3.2.3 Integrità dei messaggi e confidenzialità

La firma e la crittografia dei messaggi garantiscono un certo livello di sicurezza quando inviamo un messaggio ACL ad un agente in esecuzione nella stessa o in una diversa piattaforma. La firma dei messaggi assicura l'integrità dei messaggi e l'identità di colui che ha spedito il messaggio. La crittografia, inoltre, assicura la riservatezza del messaggio proteggendo il contenuto dello stesso da sguardi indiscreti. Un messaggio ACL è composto da due parti: la busta che contiene informazioni relative al trasporto e l'informazione vera e propria che contiene i dati sensibili.
In JADE-S "Firma" e "Crittografia" sono applicate all'intera informazione utile per proteggere le informazioni contenute nei messaggi ACL. Le informazioni relative alla sicurezza (come la firma, l'algoritmo o la chiave) sono posti nella busta.

Capitolo 2

Uso delle caratteristiche biometriche nei sistemi multi - agente

2.1 Introduzione

Il sistema di sicurezza proposto si pone come obiettivo quello di garantire un elevato livello di sicurezza alla piattaforma ad agenti in modo che essa sia ritenuta in grado di ospitare agenti software che svolgono funzioni delicate in termini di privacy o riservatezza, o che comportino transazioni economiche.
Il raggiungimento di questo obiettivo passa inevitabilmente dalla necessità di evitare che una piattaforma multi-agente venga lanciata da utenti non autorizzati oppure non autenticati. Stesso discorso vale per gli agente all'interno di una piattaforma.
Il primo passo da seguire, ancor prima di parlare di protezione di piattaforme o di agenti in una piattaforma, nella costruzione di una piattaforma ad agenti mobili sicura è appunto quello di associare ad un utente autorizzato e autenticato gli agenti di base della piattaforma stessa; dobbiamo quindi essere sicuri che la piattaforma sia stata avviata da un utente autorizzato. Il livello di sicurezza già presente in JADE-S prevede che la piattaforma non sia aperta da chiunque ma solo da utenti autorizzati; l'autorizzazione è identificata da uno username e una password, che vengono richiesti all'avvio della piattaforma. Inoltre, una caratteristica molto interessante presente nella versione 2 della piattaforma JADE-S è che i permessi e le autorizzazione non sono legate al codice ma all'utente che esegue quel codice. Possiamo avere lo stesso agente software che lanciato dall'amministratore della piattaforma è abilitato ad uccidere altri agenti oppure a inviare e ricevere messaggi da determinati agenti, mentre, lanciato da un altro utente, con credenziali diverse, non è in grado di compiere le stesse operazioni. Questo meccanismo è implementato integrando il modello di sicurezza JAAS (Java Authentication and Authorization Service) di JAVA, presente nell'sdk 1.4.0.1 e successive, nella piattaforma JADE.
JAAS consente di legare i permessi del codice anche alle referenze che l'utente ha nella rete, sia windows che unix. Cioè è possibile utilizzare e riferirsi alle credenziali di autenticazione sul dominio di un utente per concedere o negare privilegi. Ad esempio, è possibile specificare, nel modello di sicurezza, che tutti gli agenti lanciati dagli utenti, appartenenti ad un gruppo particolare della rete, abbiano accesso al file system della macchina che ospita la piattaforma. In questa maniera, se un amministratore della piattaforma, amministratore anche della rete, volesse associare tale permesso ad un utente particolare potrebbe anche inserire il profilo dell'utente da utilizzare nel particolare gruppo di utenti di rete, senza apportare alcuna modifica alla piattaforma o all'agente.
In JADE-S versione 2 è stato implementato solo l'autenticazione tramite username e password. In questo lavoro, al fine di applicare un paradigma di autenticazione degli utenti basato su "ciò che sono" e non su "ciò che sanno", è stata presa in considerazione una caratteristica biometria: le impronte digitali [5][8].

Username e Password rappresentano il più diffuso livello di sicurezza, la maggioranza dei sistemi di sicurezza prevede che un utente sia identificato da uno username e una password, senza le quali esso non può svolgere le operazioni dovute.
Esse restano un buon livello di sicurezza fintanto che restano rigorosamente nascoste da occhi indiscreti, nel momento in cui un estraneo ne viene in possesso, da quel istante in poi diventa un clone del vero proprietario infatti davanti una finestra che chiede username e password, chiunque le possieda può avere libero accesso al sistema. Allora benché valido, il sistema di sicurezza con username e password non assicura l'identità dell'utente.
L'impronta digitale è legata all'utente che la possiede, non è possibile che due utenti abbiano la stessa impronta, quindi un soggetto che viene in possesso di username e password, senza l'impronta digitale del proprietario non può entrare [1][2][3][4][5][6][7]. La piattaforma non viene avviata fino a quando l'utente non viene autenticato e autorizzato a lanciare la piattaforma. Una volta aperta la piattaforma, gli agenti AMS, DF, RMA, avranno come proprietario l'utente che ha lanciato la piattaforma. La possibilità di conoscere sempre con certezza l'identità del soggetto che manda in esecuzione un software è molto importante anche per le implicazioni legali derivanti da azioni illecite. Stesso discorso vale per il lancio di un agente all'interno di una piattaforma.
Quanto finora visto garantisce l'autenticazione dell'utente/agente nella piattaforma da dove viene lanciato. Per risolvere il problema derivante dalla necessità degli agenti di provare la propria identità durante la migrazione e durante lo scambio di messaggi è stato utilizzato il meccanismo di sicurezza "X-Security" della Gesterner Laboratori [24]. Tale sistema di sicurezza non prevede l'autenticazione durante il lancio degli agenti, ma fa uso di un agente speciale, denominato "Security Certification Agent", che associa un certificato conforme allo standard X.509 ad ogni agente registrato. L'agente SCA ha una interfaccia di registrazione degli utenti, attraverso la quale è possibile registrare nuovi utenti e quindi generare i relativi certificati. In questo lavoro, i due plug-in sono stati integrati in modo da avere un sistema completo di sicurezza. X-Security inoltre, consente un meccanismo di comunicazione sicuro tra gli agenti utilizzando tecniche di firma elettronica con chiave pubblica e privata e tecniche di crittografia basate sull'algoritmo DSA. In questo lavoro sono stati utilizzati certificati conformi alle specifiche X509 v3. Attraverso il campo delle estensione viene riportato la funzione di matching delle impronte digitali da utilizzare e il relativo "score" da raggiungere durante la fase di autenticazione.
In questo lavoro il progetto X-security è stato modificato in modo da potere generare il certificato degli agenti contestualmente alla fase di registrazione dell'identità dell'owner nella SCA. Il certificato generato è firmato con la chiave privata della SCA. All'utente viene fornito il certificato firmato e la chiave pubblica della SCA.
All'atto del lancio di un agente, prima della fase di verifica dell'impronta e dopo avere fornito una coppia di credenziali username /password valide, è obbligatorio dimostrare il possesso del certificato. Il sistema di autenticazione verifica l'integrità del certificato e successivamente consente il passaggio alla fase di verifica dell'impronta digitale.
Il certificato rappresenta un token a tutela del dato biometrico che ne garantisce l'uso lecito durante una sessione di autenticazione. Questo serve a rilevare e a bloccare i cosiddetti "replay attack" che sono portati intercettando il dato biometrico e riproducendo la sessione di autenticazione.
Vediamo adesso come si integra X-Security modificato con l'architettura JADE-S.

2.2 Sistemi di autenticazione biometrici

Con il termine “informazione biometria” si intende un modello di un tratto univoco fisiologico o comportamentale, che può essere utilizzato per autenticare gli accessi a determinate risorse critiche. In primo luogo va rammentato che esistono varie tipologie di riconoscimento biometrico e che nessuna di esse può definirsi a priori adatta a tutte le applicazioni. Tuttavia, esistono dei fattori da analizzare in ordine al raggiungimento della migliore soluzione disponibile, che sono i seguenti [5]:

- valore dei dati da proteggere: è normale che quanto più saranno strategiche le informazioni da mettere in sicurezza, tanto più si dovranno mettere in conto dei costi sostenuti, e, naturalmente, si dovrà “scendere” a compromessi soprattutto sotto il punto di vista della privacy e delle prestazioni.
- livelli di accesso da garantire e tipologia di utenti: ogni applicazione ha degli utenti con differenti autorizzazioni di accesso. Secondo questi livelli si provvederà a utilizzare una metodica più o meno sicura, anche in funzione della semplicità di utilizzo e delle prestazioni di cui si è appena parlato.
- tolleranza di errori o ritardi: questi possono avvenire sia in fase di acquisizione del dato biometrico sia durante l'autenticazione.
- piattaforma esistente hardware e software: questo è un fattore che riguarda prettamente l'integrazione con la base installata dei dispositivi biometrici.
- costi: si tratta di un parametro da correlare ai precedenti, in particolar modo al primo. I costi, infatti, sono direttamente proporzionali al tipo di sicurezza richiesta.

2.2.1 I componenti di un'infrastruttura biometrica

Una soluzione basata sulla biometria si compone generalmente di tre parti. La prima è costituita dal repository dei dati biometrici. Si tratta di un database ove sono contenute le informazioni fisiologiche o comportamentali dei soggetti da autenticare. Le informazioni ivi memorizzate vengono comparate con quelle fornite dal cosiddetto "wanna be user" al momento dell'autenticazione.
Altro componente importante è rappresentato dall'insieme delle procedure e dei dispositivi di input. Si tratta dei sistemi (lettori biometrici, procedure di caricamento delle informazioni e così via) che connettono l'utente con il sistema di validazione. L'ultimo anello è costituito dalle procedure di output e dalle interfacce grafiche. Si tratta in realtà del front end di tutta la struttura, cioè dell'interfaccia utente.

2.2.2 Parametri di Valutazione dei sistemi biometrici

Nella valutazione dei sistemi biometrici esistono, oltre ai precedenti fattori, una serie di coefficienti che vanno presi in considerazione per le operazioni di disamina.
Il False Accept Rate (FAR), conosciuto anche come False Matching Rate (FMR), è la percentuale di riconoscimento concesso a persone non autorizzate.
Il False Reject Rate (FRR), noto anche come False Non Matching Rate, è costituito dalla percentuale di errato riconoscimento negato a utenti effettivamente autorizzati.

In un ambiente ideale, i due fattori dovrebbero essere uguali a zero. Naturalmente, in realtà, questo non è realizzabile. L'obiettivo è quello di raggiungere l'equilibrio migliore a seconda dell'applicazione impiegata.
La velocità di registrazione/verifica è un altro fattore critico. Esso dipende dalla banda passante e dai media dove le informazioni sono memorizzate.
L'opinione comune è quella di suddividere le informazioni tra più media (per esempio smart card, database distribuiti e così via) in modo tale da frazionare i possibili punti di intervento da parte dei "malicious hacker" e di distribuire la banda richiesta. Naturalmente uno degli obiettivi è quello di ottimizzare le risorse e la velocità operativa. La velocità di autenticazione è decisamente importante, soprattutto nelle applicazioni di tipo bancario.

2.2.3. Esempi di utilizzo del riconoscimento biometrico

Trattandosi di tecnologie ad alta precisione, le applicazioni biometriche sono fondamentalmente destinate a impieghi di carattere strategico. Ecco alcuni esempi.

- Accesso ad aree privilegiate: quando si ha l'esigenza di entrare, per esempio, in determinate porzioni di edifici o strutture, come un laboratorio di ricerca e sviluppo;
- Corporate and remote banking: si può aumentare la sicurezza durante l'uso di servizi quali Bancomat e remote banking;
- Military application, sanità e servizi pubblici, quali certificati e voto elettronico. Network and computer security, come accesso a database e applicazioni critiche [5].

2.2.4. Principali sistemi di riconoscimento biometrico

2.2.4.1 Riconoscimento tramite retina

È basato sull'acquisizione e la verifica della mappa vascolare della retina dell'occhio umano. Il pattern formato dalle vene è definito stabile e univoco. La sua acquisizione è possibile proiettando sull'occhio un fascio a bassa intensità di luce infrarossa e catturando l'immagine con un dispositivo simile a un “retinascopio”. I problemi di questa metodica sono relativi al fatto che è richiesta una cooperazione molto ampia dell’utente e che i costi sono decisamente alti. Il retina scanning è quindi un'applicazione adatta a segmenti ad alte richieste di sicurezza, come per esempio prigioni.

2.2.4.2 Riconoscimento tramite impronta digitale

È una tecnica sempre più diffusa grazie allo sviluppo di sensori “solid state” che hanno diminuito di molto i costi di produzione. I problemi sono correlati alla titubanza di alcuni utenti (che si vedono in questo modo "paragonati" ai criminali) e, dal punto di vista tecnico/pratico, al grosso volume di risorse di calcolo richieste. Inoltre, vi

sarebbero dei problemi nell'acquisizione dei dati e nella loro verifica attraverso il tempo, in presenza di cause genetiche, età avanzata o ragioni ambientali.

2.2.4.3 Riconoscimento tramite geometria della mano

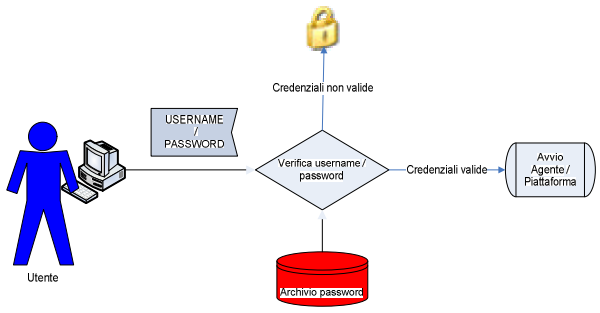
Consiste nel riconoscimento delle persone mediante la verifica delle misure e della conformazione della mano e consente un discreto coefficiente di univocità. Tuttavia, va tenuto presente che detta geometria può mutare a causa dell'età avanzata o a seguito di patologie fisiche, quali, per esempio, l'artrite. Tutti questi fattori fanno della "hand geometry" un'applicazione adatta a scopi con esigenze di sicurezza minime.

2.2.4.4 Riconoscimento tramite iride

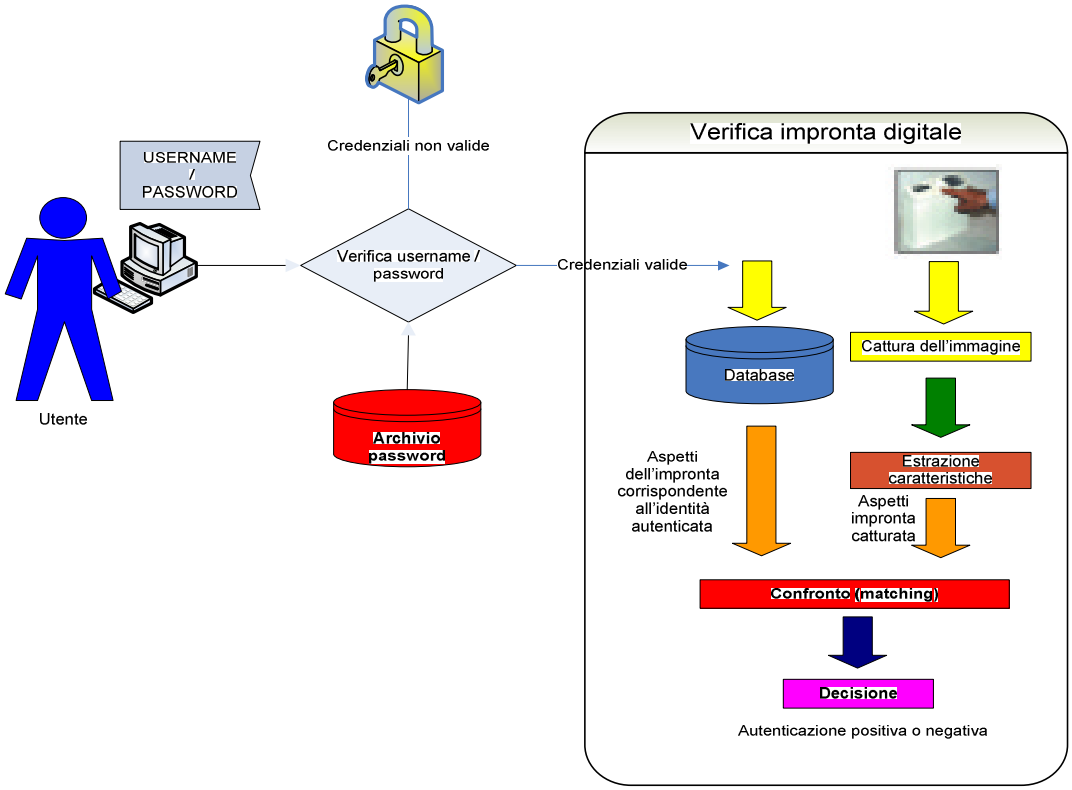
È, allo stato attuale, uno dei più precisi in circolazione. Non fornisce problemi esterni né subisce condizionamenti a seguito di fattori esterni o interni. L'utente, infatti, che fa uso di droghe, lenti a contatto correttive o colorate, o è affetto da patologie (glaucoma e simili) può senza problemi effettuare le operazioni di identificazione. Il processo di acquisizione dell'immagine viene effettuato in bianco e nero, a mezzo di una camera stereo ad alta precisione. La dimensione della camera sta diminuendo sempre più, la qual cosa consentirebbe in poco tempo un'ottima diffusione del metodo anche nelle applicazioni distribuite.
Alcuni ritengono che un limite sia costituito dall'iridiologia. Si tratta di una forma di medicina alternativa (non riconosciuta da quella ufficiale) che consente, mediante il controllo delle caratteristiche dell'iride, di risalire a determinate patologie dell'utente, nonché di prevederne proattivamente le evoluzioni. Questo è un fattore negativo se si pensa alla possibilità di utilizzare impropriamente le informazioni, comunque facenti parte di un database di record biometrici. L'iridiologia non è un vero pericolo per la privacy degli utenti, sempre che se ne faccia un uso corretto, ovviamente.

2.3 Il sistema di autenticazione della piattaforma implementata

Il sistema di autenticazione della piattaforma JADE-S è schematizzato nella figura seguente [2][3][4][6]:



La conoscenza di una coppia di credenziali username /password è sufficiente per aprire la piattaforma.
Il sistema di autenticazione della piattaforma proposta aggiunge al livello di sicurezza già presente in JADE-S, un secondo livello di sicurezza sull'identificazione personale basato sul riconoscimento biometrico di "aspetti statici" e sul trasporto di dati sicuro.
Quindi per aprire la piattaforma JADE, o per lanciare un agente in una già aperta, un utente deve prima essere stato registrato nel database degli utenti autorizzati; dopo la registrazione l'utente può utilizzare la piattaforma.
Per ottenere questo obiettivo si è aggiunto al sistema esistente il blocco "Verifica Impronta Digitale" (vedi fig. 2.2) che effettua la verifica dell'impronta digitale con quella presente in archivio.
Dalla (Fig. 2.2) si vede come il sistema proposto interagisce con l'utente: il primo livello di sicurezza da superare è quello costituito dalla conoscenza di username e password; il sistema verifica se l'utente è presente e poi ne verifica la password, se uno dei due passi non ha esito positivo l'accesso viene negato. Se, invece, entrambi danno un esito positivo si passa al blocco "Verifica Impronta Digitale" che conduce al livello di sicurezza più alto. Viene prima effettuato un controllo, tramite lo username introdotto, per verificare se l'impronta relativa all'utente si trova in archivio, se la verifica da esito negativo la procedura viene arrestata altrimenti si procede con la verifica dell'integrità dell'impronta. Passata questa ulteriore verifica si procede con l'immissione dell'impronta e il confronto con quella in archivio.



La verifica delle impronte digitali da sola non basta per essere certi dell'identità dell'uso. Gli utenti quando si registrano ricevono un certificato che memorizzano su di un floppy firmato con la chiave privata della Security Certification Authority.
Prima di iniziare il processo di verifica delle impronte, il sistema di autenticazione della piattaforma cerca sul floppy (ad esempio) il certificato. Se lo trova, ne verifica l'integrità e l'autenticità ed estrae da esso le informazioni sull'identità del soggetto da autenticare. Per prima cosa verifica che lo username e la password inserite corrispondano all'identità del soggetto riportato nel certificato. In caso positivo legge dal certificato le estensioni in cui è riportata la funzione di matching da utilizzare e lo score da raggiungere nella fase di verifica delle impronte digitali [1].
Superata la verifica delle impronte digitali l'agente ha accesso alla piattaforma.
Il certificato è utilizzato sia come token per evitare attacchi di tipo "replay attack" portati intercettando il dato biometrico che in casi di riconoscimento dubbio delle impronte.
Supponiamo che un utente non raggiunga lo score indicato nella funzione di matching, ma si avvicina molto. In questo caso il sistema potrebbe ritenere che il possesso del certificato sia sufficiente a fare passare l'utente anche se non ha raggiunto lo score.
Il possesso del certificato è utile anche nel caso in cui l'utente supera lo score perché il possesso del certificato assicura che l'impronta sottoposta al sensore sia proprio quella della persona che dice di essere.
Se un utente perde contemporaneamente le credenziali username/password, il floppy contenente il certificato e si fa prelevare in maniera non lecita le impronte ha sempre la possibilità di farsi revocare l'account nella piattaforma in modo da stare sicuro che nessun altro agisca utilizzando la sua identità.
Altro aspetto di sicurezza preso in considerazione nella piattaforma realizzata è l'utilizzo di un canale di comunicazione sicuro.
Gli agenti che usano il trasposto dei dati basato su X-Security utilizzano una struttura di messaggio cosi composto [24]:

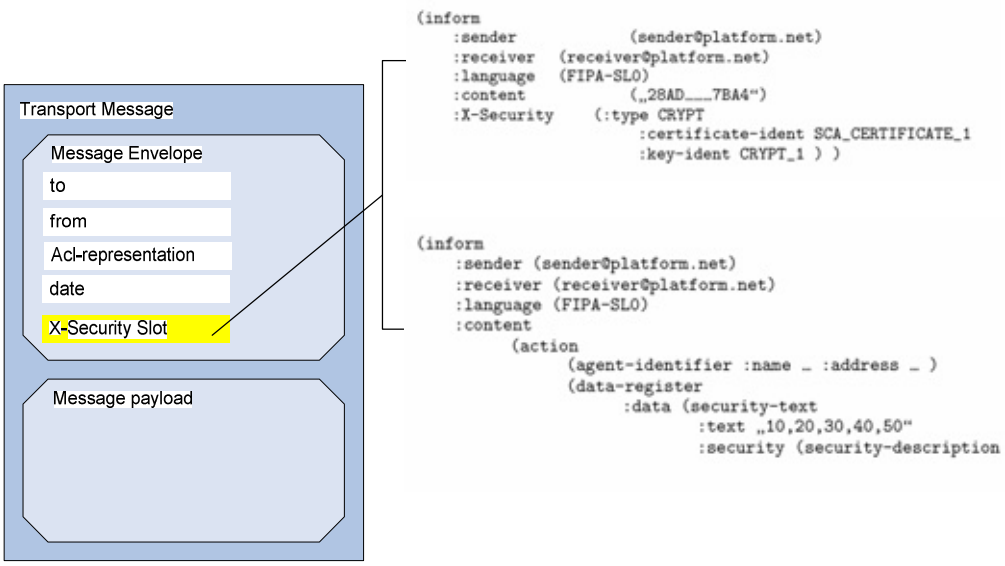


Figure 2.3: Struttura modificata del messaggio ACL

Come si vede dalla figura precedente, nella struttura di base dei messaggio ACL è stato inserito un nuovo slot denominato “X-Security” utilizzato per includere nei messaggi ACL le informazioni relative alla chiave di cifratura utilizzata e alla chiave pubblica da utilizzare per ricostruire il testo originale del messaggio.

2.4 Moduli aggiuntivi sviluppati e implementati

Nel presente lavoro è stato sviluppato un modulo che gestisce la richiesta di accesso di un agente nella piattaforma ed è stato esteso l’uso dei certificato digitali all’autenticazione [5].

2.4.1 Il Login Module

Il modulo di login realizzato, gestisce le richieste di accesso alla piattaforma. Esso è una implementazione del Java Authentication and Authorization Service (JAAS), il quale è stato introdotto come optional nel pacchetto Java 2 SDK, versione standard (J2SDK) 1.3. Questo optional è conosciuto con il nome JAAS 1.0. Con la versione J2SDK 1.4, JAAS è stato integrato nella versione standard di JAVA.
JAAS permette alle applicazioni di restare indipendenti dalla tecnologia di autenticazione sottostante. JAAS permette l’uso di una nuova tecnologia per l’autenticazione. Il login module realizzato fornisce un particolare tipo di autenticazione, basato sull’uso di caratteristiche biometriche e sull’uso dei certificati.
L’autenticazione implica l’uso di un modulo di login che costituisce il meccanismo con il quale i visitatori provano che essi operano per conto di un utente o di un sistema. Nel presente lavoro è stato utilizzato JAAS per realizzare un modulo di login che consente di avere fiducia delle credenziali dichiarate da un utente attraverso l’uso di un LoginContext. (vedi figura 2.4). Dopo aver provato con successo l’identità, il LoginContext genera un oggetto di sessione, il quale permette ad un utente o ad un sistema di essere autenticato da un’altra identità senza ripercorre il processo di autenticazione.

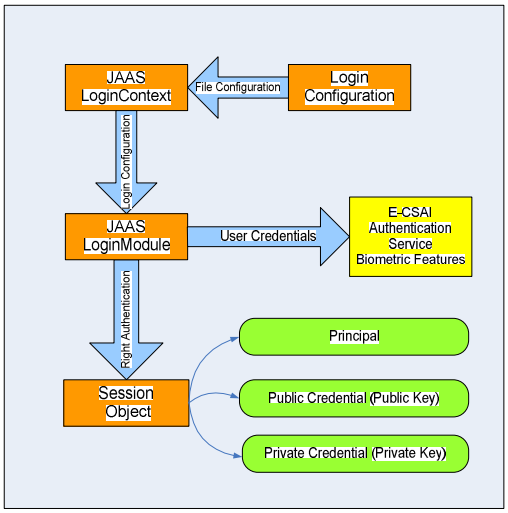


Figure 2.4 Flusso del login module realizzato

Il modulo di login è costituito da tre componenti. Una interfaccia attraverso la quale il sistema di autenticazione interagisce con l'utente richiedendo le credenziali. Un file di configurazione che specifica il metodo di autenticazione da utilizzare e il modulo che esegue il controllo delle credenziali.
Il loginContext legge il file di configurazione e istanzia uno specificato login module. Il login module è istanziato con un soggetto, un Callbackhandler, e uno stato condiviso. Un soggetto è un contenitore che mantiene le informazioni di autenticazione dell'utente o del servizio che è stato autenticato. Esso include un relativo Principal e le credenziali.
Un Principal è una entità come un utente individuale, un login id, o un gruppo al quale l'utente appartiene. Un utente in genere rappresenta una persona. Un gruppo è una categoria di utenti classificati per mezzo di caratteristiche comuni per facilitare l'amministrazione.
Dopo l'inizializzazione il LoginContext invoca il Login Module. Nel presente lavoro è stato scritto un CallBackHandler che in sequenza chiede all'utente di inserire la username e la password, di acconsentire il rilevamento delle impronte digitali ed in infine di eseguire la verifica del certificato digitale.
Per potere utilizzare JAAS è necessario compiere le seguenti attività preliminari:

1. Predisporre un file, di solito denominato JAAS.config, utilizzato dall'ambiente Java, usato per specificare quale è il modulo di login da utilizzare;
2. Predisporre un file di configurazione in cui è specificato quale è il protocollo di trasporto utilizzato, quale è il file dei permessi dei singoli agenti, quale è il metodo di autenticazione e autorizzazione da usare.

Si può vedere più nel dettaglio, partendo dal diagramma delle classi, come avviene il processo di autenticazione e di autorizzazione.

DIAGRAMMA DELLE CLASSI DEL LOGIN MODULE

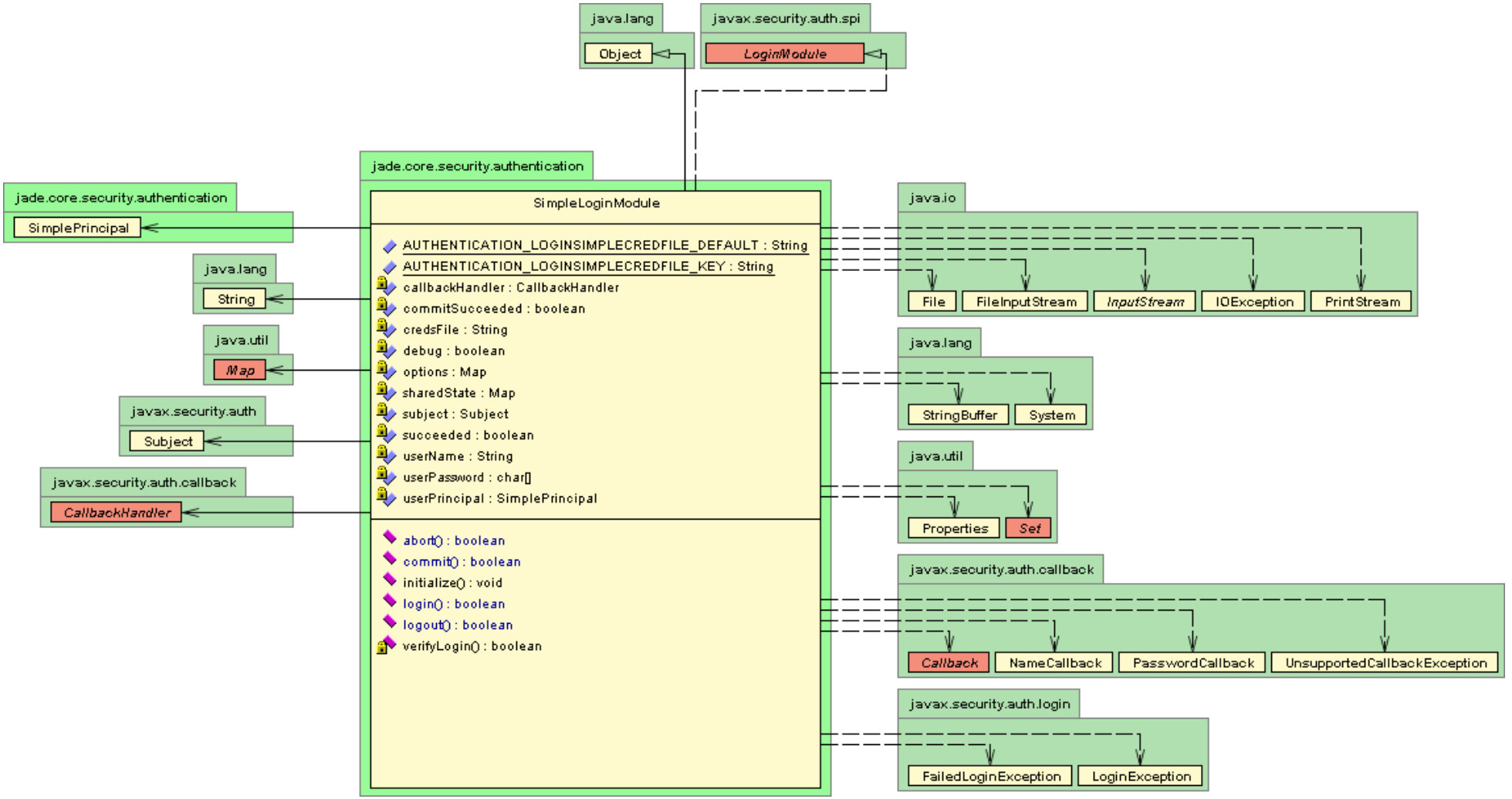


Figure 2.5 Diagramma della classe Login Module

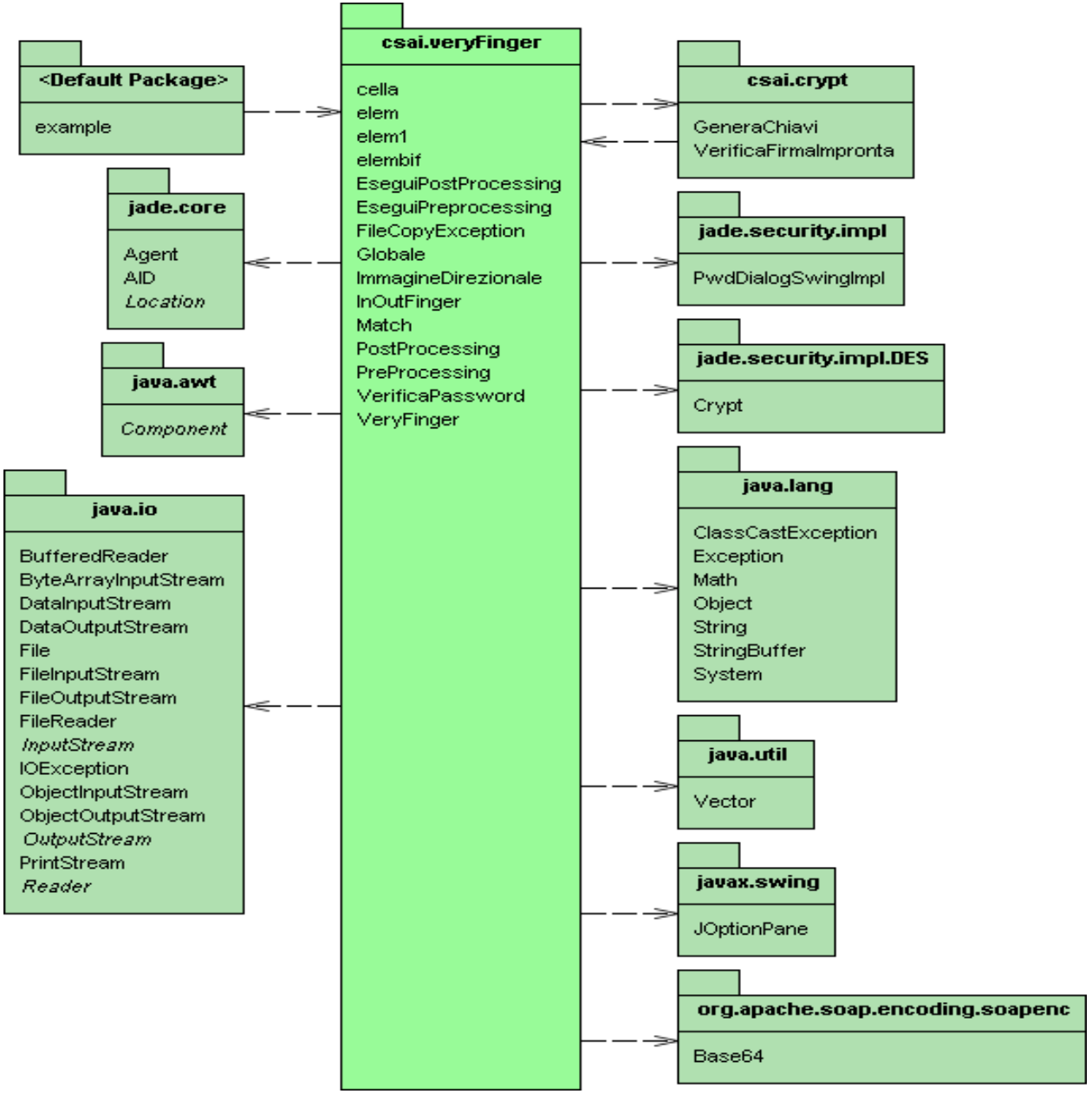


Figure 2.6 Diagramma delle classi package CSAI

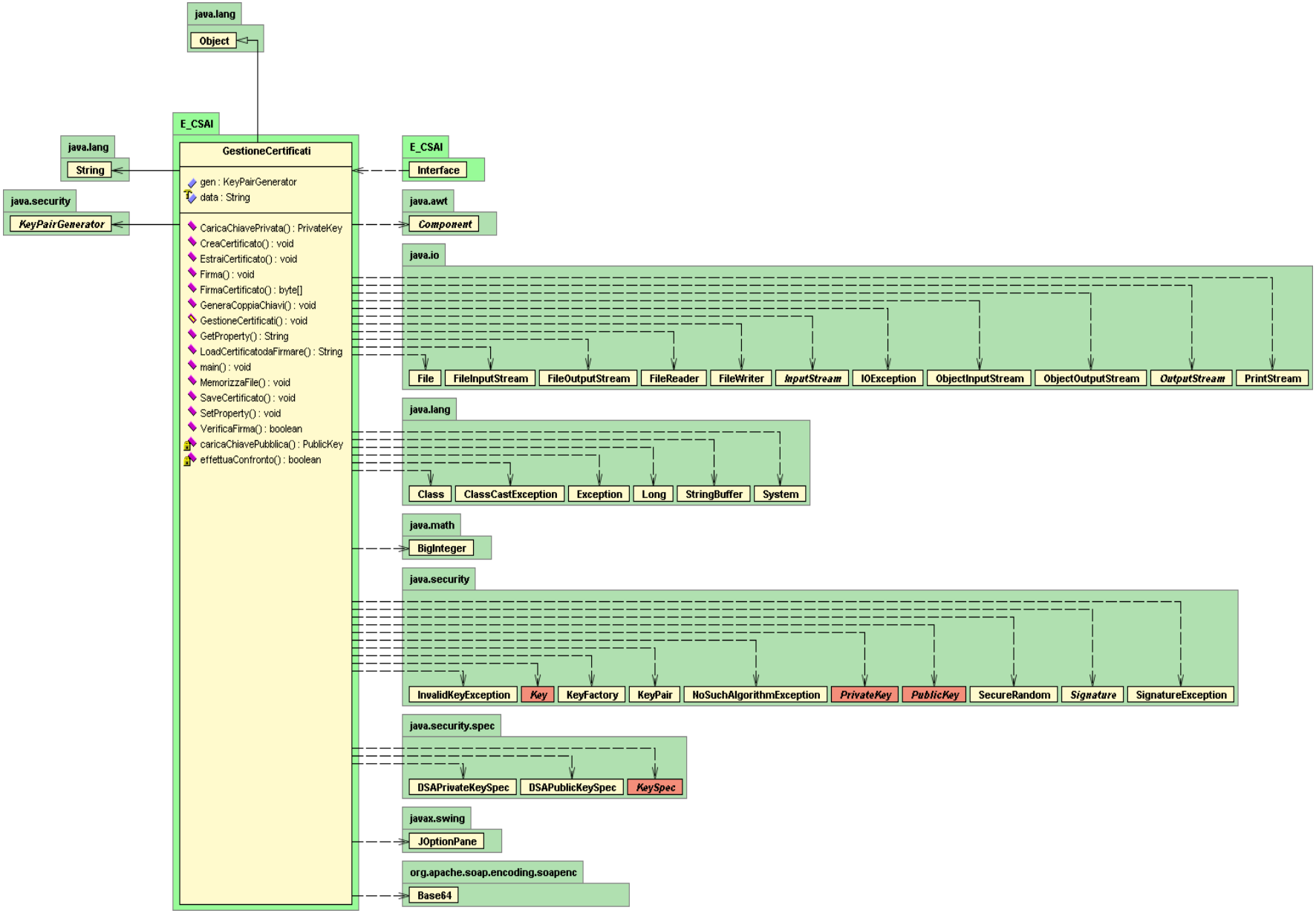


Figure 2.7 Diagramma delle classi del package E_CSAI

Il processo di autenticazione inizia eseguendo il metodo “login()” della classe LoginModule che genera il Principal che sarà associato all’utente che chiede di autenticarsi. Il metodo login(), legge il file jaas.config e imposta il metodo di autenticazione da usare. Successivamente genera un’istanza della classe CallBackHandler() che interagisce con l’utente. Questo ultimo oggetto si occupa dell’acquisizione dello username e della password dell’utente e della relativa verifica. Superata questa prima fase, all’oggetto Principal viene associato l’utente autenticato, fino a questo punto tramite la user-id e password. In seguito viene chiamato il metodo “TestOwner” del package E_CSAI sviluppato. Questo metodo chiede all’utente di inserire il floppy con il certificato X509 v3. In seguito viene verificata la correttezza formale e l’integrità del certificato. Se la verifica ha esito positivo vengono estratte dal certificato i dati del proprietario e i dati relativi alla funzione di matching da usare. Se l’owner del certificato coincide con i dati dell’utente inseriti precedentemente si passa alla verifica delle impronte digitali.
La verifica delle impronte digitali viene fatta utilizzando il package CSAI sviluppato presso il laboratorio CSAI. In particolare viene chiamato il metodo “VeryFinger” che si occupa appunto dell’acquisizione dell’impronta digitale e della verifica della corrispondenza con quella rilevata in fase di creazione del nuovo utente.
Se il metodo Veryfinger ritorna esito favorevole, il metodo TestOwner consente l’accesso nella piattaforma all’agente che si sta avviando. A questo punto a tale agente è associato in maniera ineluttabile un utente, responsabile delle sue azioni. Durante il ciclo di vita dell’agente ad esso è associato sempre il Principal istanziato dal LoginModule. Se qualche cosa non funziona l’agente non ha accesso alla piattaforma e viene visualizzato il relativo messaggio di blocco.

2.4.2 Uso dei certificati X-Security per l’autenticazione

Un elemento determinante per ogni sistema distribuito, basato su codice mobile, è costituito dai meccanismi di sicurezza disponibili per proteggere sia l’host, da azioni potenzialmente dannose prodotte da un frammento di codice in esecuzione, sia il codice, da tentativi di alterazione da parte di host in esecuzione. Nel presente lavoro è stata implementata, a partire da un lavoro sviluppato presso il laboratorio Gestner [24], una infrastruttura che usa i certificati digitali formato X509 V3 sia per la comunicazione tra gli agenti che per l’autenticazione. Vale la pena subito dire che anche JAAS ha un meccanismo di comunicazione sicuro tra gli agenti basato sul protocollo SSL. Nel presente lavoro si è ritenuto vantaggioso utilizzare, invece di SSL, l’infrastruttura X-Security modificata essenzialmente per due motivi:

- SSL fa uso di chiavi di sessioni simmetrici e quindi il livello di sicurezza raggiunto è inferiore a quello ottenuto con l’uso di una infrastruttura di cifratura asimmetrica come quella presente in X-Security;
- Se si utilizza SSL come protocollo di trasporto deve essere utilizzato da tutti gli agenti senza alcuna distinzione e deve essere disponibile nella rete utilitizzata.

Quindi non si può avere una piattaforma che ospita agenti che scambiano messaggi cifrati e agenti che scambiano messaggi in chiaro, perché la scelta è fatta con riferimento alla piattaforma e non ai singoli agenti; X-Security, invece, è indipendente dalla piattaforma utilizzata e non richiede la disponibilità di SSL. Inoltre, consente di

avere una piattaforma in grado di ospitare agenti che comunicano in maniera sicura e agenti che non usano questi meccanismi. Gli agenti che non usano X-Security hanno una struttura dell'ACL message standard senza lo slot aggiuntivo "X-Security".
Il lavoro da cui si è partiti utilizza i certificati X509 V1 per garantire l'integrità e la confidenzialità delle informazioni scambiate tra gli agenti. La soluzione a cui si è arrivati in questo lavoro, come già detto, utilizza i certificati anche per l'autenticazione perché, in questo modo, si riesce a controbattere eventuali attacchi alla piattaforma portati attraverso il possesso non lecito delle informazioni biometriche. Catturando il dato biometrico che viaggia sulla rete oppure dall'host dedicato all'autenticazione è possibile ripetere il processo di autenticazione camuffando la propria identità. Il fatto che il possesso del dato biometrico da solo non garantisca il superamento del processo di autenticazione è a vantaggio della sicurezza della piattaforma. Prima di essere creduti sull'identità dichiarata è necessario provare il possesso di un token (il certificato digitale) ottenuto all'atto della registrazione dell'utente nella piattaforma. Nella piattaforma realizzata si utilizza un paradigma di autenticazione basato su "ciò che si è". Si utilizzano certificati X509 v3 perché prevedono i campi estensione attraverso i quali viene specificata la funzione di matching da utilizzare per il confronto delle impronte digitali e il grado di somiglianza da raggiungere. Una volta che il processo di autenticazione è stato superato, a ciascun principal, e quindi a ciascun agente, sono associati dei permessi. La specifica delle azioni che ciascun agente può compiere è espressa per mezzo di un file di policy, di cui si riporta un frammento come esempio:

```
grant principal jade.security.Name "bob" {
        permission jade.security.ContainerPermission "", "create";
        permission jade.security.AgentPermission "agent-owner:bob", "create, kill";
        permission jade.security.MessagePermission "agent-class=*,agent-owner:bob", "send-to";
        permission jade.security.AMSPermission   "agent-class=*", "register,deregister,modify";

        permission jade.security.MessagePermission "agent-name=*", "send-to";
};

grant principal jade.security.Name "alice" {
        permission jade.security.ContainerPermission "", "create";
        permission jade.security.AgentPermission "agent-owner:bob", "create, kill";
        permission jade.security.MessagePermission "agent-class=*,agent-owner:bob", "send-to";
        permission jade.security.AMSPermission   "agent-class=*", "register,deregister,modify";
        permission jade.security.MessagePermission "agent-name=*", "send-to";
};

grant principal jade.security.Name "SecCertAhth" {
        permission jade.security.ContainerPermission "", "create";
        permission jade.security.AgentPermission "agent-owner:petr", "create, kill";
        permission jade.security.MessagePermission "agent-class=*,agent-owner:petr", "send-to";
        permission jade.security.AMSPermission   "agent-class=*", "register,deregister,modify";
        permission jade.security.MessagePermission "agent-name=*", "send-to";
};
grant principal jade.security.Name "*" {
        permission jade.security.MessagePermission "agent-name=*", "send-to";
};

grant principal jade.security.Name "*" {
        permission jade.security.AgentPermission "agent-owner=bob", "create";
        permission jade.security.AgentPermission "agent-name=bob-rma*", "create,kill";
        permission jade.security.AgentPermission "agent-owner=bob", "kill";
        permission jade.security.AgentPermission "*", "suspend,resume";
        permission jade.security.AMSPermission   "agent-class=*", "register,deregister,modify";
        permission jade.security.MessagePermission "agent-name=*", "send-to";
};
```

Come si vede dal frammento di codice sopra riportato, è possibile associare permessi diversi al codice in funzione del soggetto ad esso associato. Bisogna tenere conto che la

modifica di questi file consente di autorizzare gli utenti a compiere determinate azioni, come ad esempio, chiudere la piattaforma. L'integrità di questo file deve essere garantita in maniera indipendente dalla piattaforma, perché riguarda la sicurezza della stessa da attacchi classificati come "Altri – Sistema".

2.4.2.1 Struttura dei certificati

Come detto in precedenza i certificati utilizzati in questa piattaforma sono memorizzati in file xml. Ciò consente una rapida verifica dell'esattezza formale del certificato derivante dalle elevate prestazioni e caratteristiche dei parser xml disponibili. E' possibile utilizzare anche un browser (Internet Explorer ver. 5.0 o successive e Netscape 4.7 o successive) per la verifica formale del certificato. La correttezza semantica è affidata ovviamente al package E-CSAI sviluppato.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Certificate>
  <Version>V3</Version>
  <SerialNumber>0000000002</SerialNumber>
- <Issuer>
  <Name>SecCertAuth@giuseppe:1099/JADE</Name>
  </Issuer>
- <Subject>
  <Name>Agent1@giuseppe:1099/JADE</Name>
  </Subject>
- <Owner>
  <Organisation>Unipa</Organisation>
  <Coutry>Italy</Coutry>
  <Email>giuseppe.gioe@edilab.it</Email>
  <Name>Gioe</Name>
  </Owner>
- <Validity>
  <NotBefore>20040101000000Z</NotBefore>
  <NotAfter>20041231235959Z</NotAfter>
  </Validity>
- <PublicKeys>
  <DefaultCryptKeyIdent>CRYPT_1</DefaultCryptKeyIdent>
- <PublicKey>
  <Ident>SIGN_1</Ident>
- <Parameters>
  <Algorithm>DSA</Algorithm>
  <Format>X.509</Format>
  </Parameters>
- <Values>
  <Coded>AsciiHex</Coded>

<Modulus>308201B83082012C06072A8648CE3804013082011F02818100FD7F5381
1D75122952DF4A9C2EECE4E7F611B7523CEF4400C31E3F80B6512669455D402
251FB593D8D58FABFC55BA30F6CB9B556CD7813B801D346FF26660B76B9950
A5A49F9FE8047B1022C24FBBA9D7FEB7C61BF83B57E7C6A8A6150F04FB83F
6D3C51EC3023554135A169132F675F3AE2B61D72AEFF22203199DD14801C702
15009760508F15230BCCB292B982A2EB840BF0581CF502818100F7E1A085D69B
3DDECBBCAB5C36B857B97994AFBBFA3AEA82F9574C0B3D07
82675159578EBAD4594FE67107108180B449167123E84C281613B7CF09328CC8A
```

**6E13C167A8B547C8D28E0A3AE1E2BB3A675916EA37F0BFA213562F1FB627A
01243BCCA4F1BEA8519089A883DFE15AE59F06928B665E807B552564014C3BF
ECF492A038185000281810 0E8A4B1792FFA7B80574ABB18086C364F995D11F2
ACFC77AA493F696819EAF973798C3AF2E868442412E52049FC39CE32B518901
C78F498E45CA69CCD18C2F9F6B51F57CC3C1FA354AB08B0E6B60439929F1D
7C2549EE29AB055566AEE54F5DAAD4AEDA82250493D8
C7CA610F6977A916E6A4E7B97C04DE07FDC2C11CC4839860**
</Values>
</PublicKey>
- <PublicKey>
<Ident>**CRYPT_1**</Ident>
- <Parameters>
<Algorithm>**RSA**</Algorithm>
<Format>**X509**</Format>
</Parameters>
- <Values>
<Coded>**AsciiHex**</Coded>

<Modulus>**30819F300D06092A864886F70D010101050003818D0030818902818100
CAF14E06AD41C373ABFB30B31B1973AC0EDA8444391DE676672C5C064CB57
F7644D951EC7A7CF51B6F96E4F71E16C076E08F2F73B8E7EDEAE9DB541255
B95B6EA92A564725432E028832DEFEAB5F287AEF70BDD8D52C6E44C46C555
1006DE901A4BD3F0BA5E6076938F77DE18ACD03A22CC3D19223243AB24DD8
6F4B0E1E7E7B0203010001**</Modulus>
</Values>
</PublicKey>
</PublicKeys>
- <Extensions>
- <Extension>
<Name>**Matching**</Name>
<Value>**CSAI**</Value>
<Condition>**user**</Condition>
</Extension>
- <Extension>
<Name>**security_level**</Name>
<Value>**CLIENT**</Value>
<Condition>**critical**</Condition>
</Extension>
</Extensions>
- <Signature>
<CertIdent>**1**</CertIdent>
<KeyIdent>**SIGN_1**</KeyIdent>
<Author>**SecCertAuth@giuseppe:1099/JADE**</Author>
<DateTime>**20040928213459Z**</DateTime>

<Value>**302D0214605C64CF05AB9BBEF603AD221069360F298DA0330215008887
AA1926F3029C53DC5C6DC12629020B712B1E**</Value>
</Signature>
</Certificate>

Capitolo 3

Validazione Sperimentale

3.1 Introduzione

In questo capitolo descriveremo il funzionamento della piattaforma realizzata partendo dal lancio della piattaforma stessa fino all'avvio di un agente in essa.
Per illustrare meglio il sistema di comunicazione sicuro disponibile nella piattaforma è stato sviluppato un agente che invia un messaggio di test ad un altro agente registrato sfruttando sia la crittografia che la firma dei messaggi inviati.
Come già detto il primo passo da compiere è il lancio della piattaforma. In questo stadio viene lanciato anche l'agente SCA che si occupa della gestione della sicurezza. Anche per questo agente si utilizza lo stesso modulo di login. Sia gli agenti di base che l'agente SCA sono associati ad un owner.
I permessi ai vari agenti vengono specificati attraverso un file di policy per mezzo del quale è possibile specificare anche permessi diversi allo stesso codice (agente) a seconda dell'utente che lancia l'agente stesso.

3.2 L'ambiente di sviluppo

E' stata utilizzata una delle piattaforme Multi-Agente più diffuse: JADE [15]. Per il prelevamento delle impronte digitali un sensore della Secugen.

3.3 Il sensore

Nella piattaforma realizzata la caratteristica biometrica presa in considerazione è le impronte digitali. In questo lavoro è stato utilizzato il package denominato" CSAI" implementato nel laboratorio CSAI del Dipartimento di Ingegneria Inormatica. Tali impronte vengono prelevate da un sensore, il sensore utilizzato è il SecuGen Hamster, tramite collegamento alla porta parallela del pc è stato possibile effettuare il prelevamento delle impronte digitali.
Le caratteristiche di tale sensore sono:

- temperatura di utilizzo tra i –40°C e i 60°C;
- resistente ad impatti sul prisma;
- affidabile il prelevamento anche in presenza di elementi contaminanti in fase di prelevamento, quali oli o sporcizia;
- funzionante anche in ambienti aventi un livello di umidità superiore al 90%.

Questo sensore restituisce in uscita un'immagine raw di 300x260 pixel con una profondità di colore a 8 bit, quindi su una scala di 256 livelli di grigio.

3.4 La procedura di accesso

Nel momento in cui l'utente "Petr" decide di lanciare la piattaforma JADE-S con l'interfaccia grafica, digita in una shell Dos il comando "jade.Boot -conf main.conf". Questo comando fa si che venga avviata la piattaforma con una configurazione particolare specificata nel file "main.conf". Avviata la richiesta di apertura, inizia il processo di autenticazione. Per l'avvio della piattaforma (e per i test che si sono effettuati) si è scelto di specificare username e password direttamente nel file delle policy in modo da evitare di digitare ogni volta le credenziali. Per l'amministratore della piattaforma non vengono fatti controlli di impronte e di certificati perché si ritiene superfluo. L'host che ospita la piattaforma si ritiene sicuro, non ci sono agenti da proteggere prima di lanciare la piattaforma, quindi si ritiene sufficiente utilizzare una coppia di username e password.
La procedura di avvio della piattaforma si articola nelle seguenti fasi:

- richiesta di immissione e verifica di username e password
- apertura della piattaforma

La procedura di avvio di un agente nella piattaforma si articola nelle seguenti fasi:

- richiesta di immissione e verifica di username e password
- lettura del certificate e verifica integrità
- estrazione dell'identità dell'owner dal certificato
- richiesta di immissione e verifica delle impronte digitali
- verifica della disponibilità dell'impronta in archivio
- accesso alla piattaforma

Esamineremo singolarmente i vari passi.

3.5 Avvio della piattaforma

Quando l'utente decide di lanciare la piattaforma deve digitare il comando:

jade.Boot -conf main.conf

Il file main.conf in cui sono descritti tutti i permessi assegnati agli agenti della piattaforma è stato precedentemente predisposto dall'amministratore. Nel file main.conf è specificato che la verifica dello username e della password avviene tramite comando in linea. Non compare nessuna richiesta di inserire username e password.
Comunque la sicurezza è garantita perché l'accesso alla piattaforma host è protetto dalle policy di accesso del dominio dove l'amministratore opera.

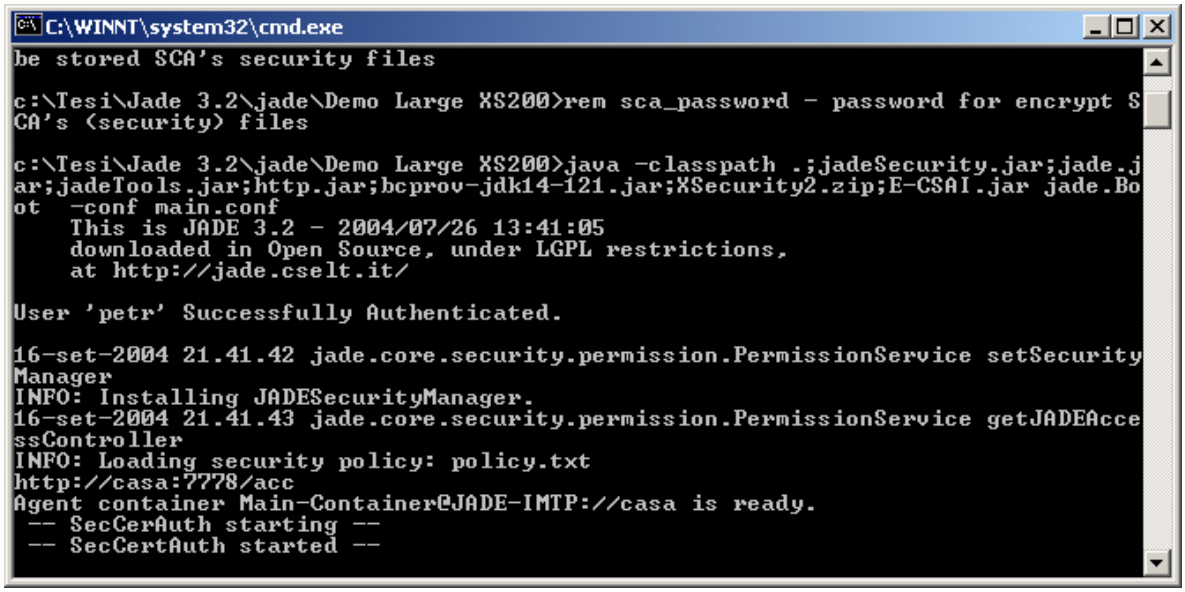


Figura 3.1: Avvio della piattaforma

Come si vede dal log della sessione dos riportato nella figura 3.1 l'utente 'petr' è stato autenticato e considerato l'owner degli agenti di base della piattaforma e dell'agente SCA.

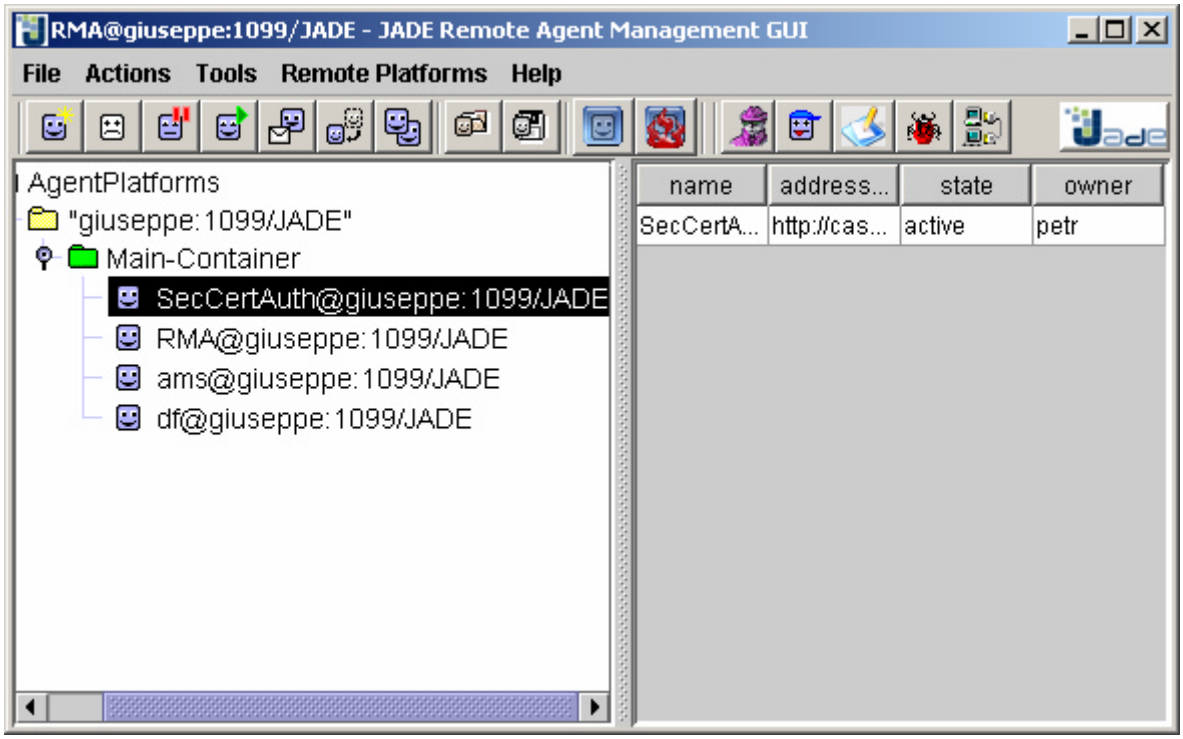


Figure 3.2: Piattaforma JADE con Agenti di bae e Agente SCA

Ovviamente se la coppia di credenziali utilizzate non è sufficiente per avviare la piattaforma o sono errate viene visualizzata la mancata autenticazione e la piattaforma non si avvia.

Nella figura seguente è riportato il messaggio visualizzato all'utente nel caso in cui l'autenticazione non ha avuto esito favorevole.

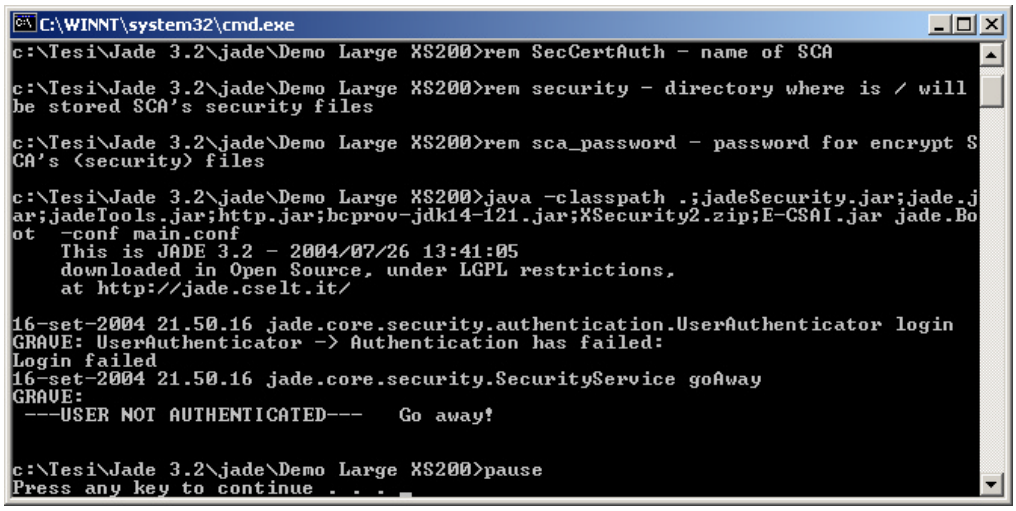


Figure 3.3: Autenticazione non riuscita

3.6 Avvio di un agente in una piattaforma

Analizziamo la procedura di lancio di un agente nella piattaforma. Il primo passo è appunto quello di eseguire in una shell del dos il comando:

jade.Boot -conf cont-1.conf -container -host 127.0.0.1

Con questo comando vogliamo lanciare un agente in accordo con le policy specificati nel file "cont-1.conf" in un nuovo container dell'host locale. E' possibile lanciare l'agente anche in un host remoto cambiando semplicemente l'indirizzo IP specificato nel comando di avvio. Per gli agenti la username e password vengono forniti attraverso una finestra di interfaccia.

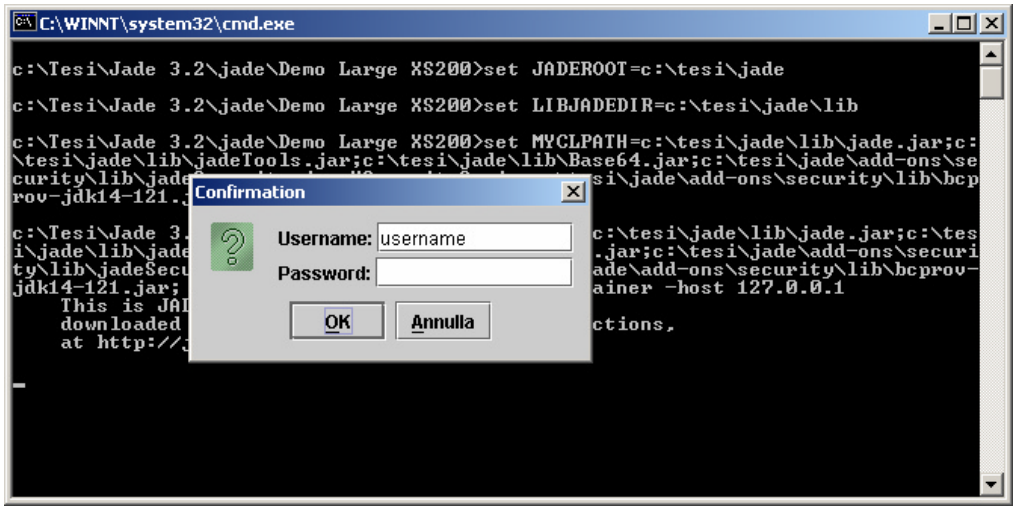


Figure 3.4: Richiesta di username e password per il lancio di un agente

Se la verifica di username e password ha dato esito positivo si passa all'estrazione delle informazione dal certificato firmato.
Se l'utente ha inserito il floppy contenente il certificato il sistema procede senza interagire con l'utente. In caso contrario il sistema chiede di inserire il floppy per continuare oppure abbondare.

Figure 3.5: Richiesta inserimento floppy

Prima di utilizzare le informazioni contenute nel certificato il sistema verifica la sua integrità e correttezza. Questo viene fatto utilizzando la chiave pubblica della SCA per ricostruire il contenuto del certificato in chiaro. Se questa operazione fallisce o si ha il sospetto che il certificato sia stato manomesso l'accesso viene negato.

Figure 3.6: Esito negativo verifica certificato

Superate queste due verifiche, si passa alla verifica delle impronte digitali che come detto in precedenza viene fatto utilizzando il package "CSAI" sviluppato nel laboratorio CSAI.

3.7 Comunicazione tra gli agenti

Per verificare il corretto funzionamento del sistema di comunicazione della piattaforma è stata sviluppata una semplice applicazione che consiste in un agente che invia un messaggio ad un altro agente.

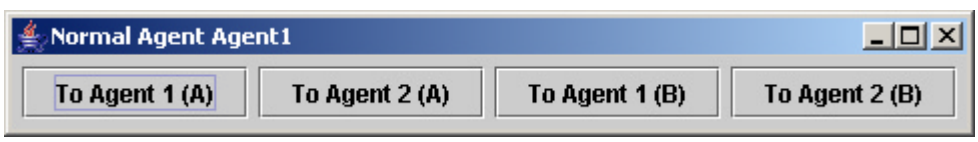


Figure 3.7: Interfaccia dell'agente Agent1

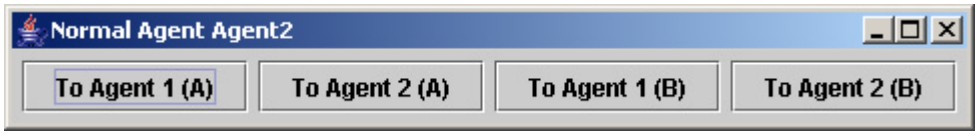


Figure 3.8: Interfaccia dell'agente Agent2

Per eseguire il test sono stati lanciati due agenti uguali ma associati ad utenti diversi.

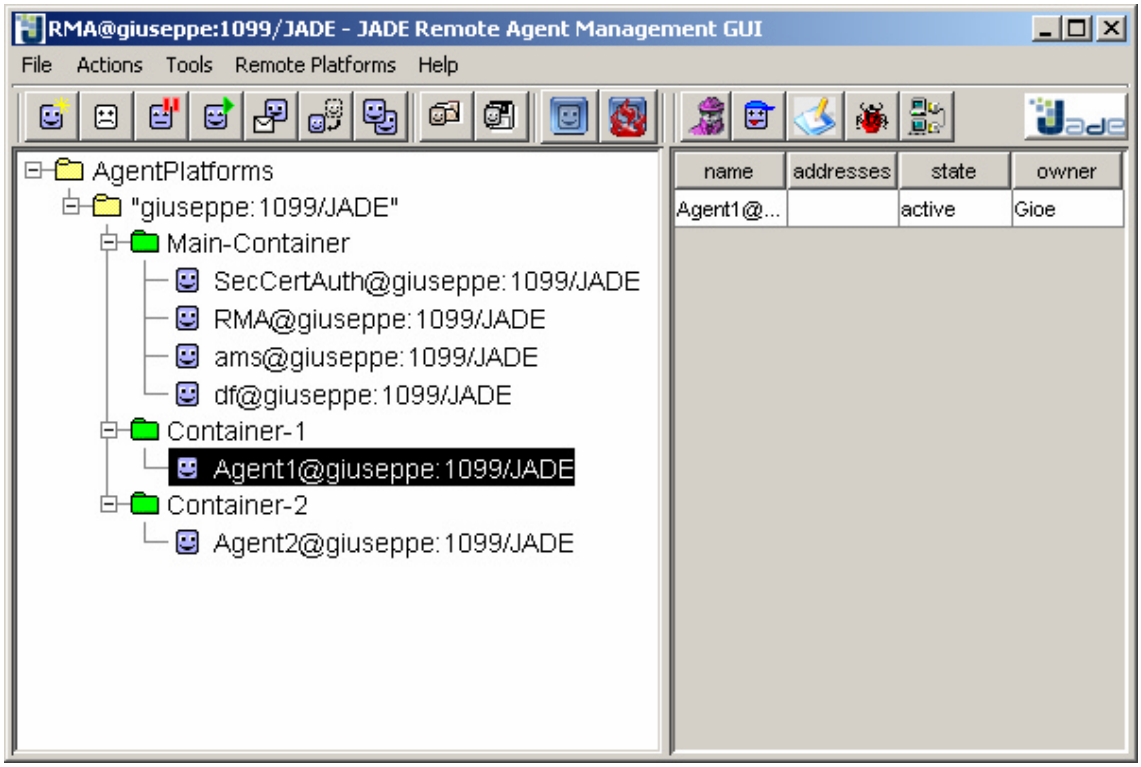


Figure 3.9: Stato piattaforma JADE

Gli agenti lanciati sono localizzati ciascuno in un container separato. Premendo il tasto "To Agent2 (A)" nell'interfaccia dell'agente Agent1 parte un messaggio firmato verso l'agente Agent2. L'Agent2 per prima cosa recupera il certificato dell'Agent1 e successivamente utilizza la chiave pubblica riportata nel certificato per la firma delle informazioni da inviare.
Per analizzare il contenuto dei messaggi inviati viene stampato nella sessione dos dell'agente il contenuto del messaggio ACL inviato. In questo modo è possibile pure vedere l'esistenza dello slot "X-Security" e del suo contenuto

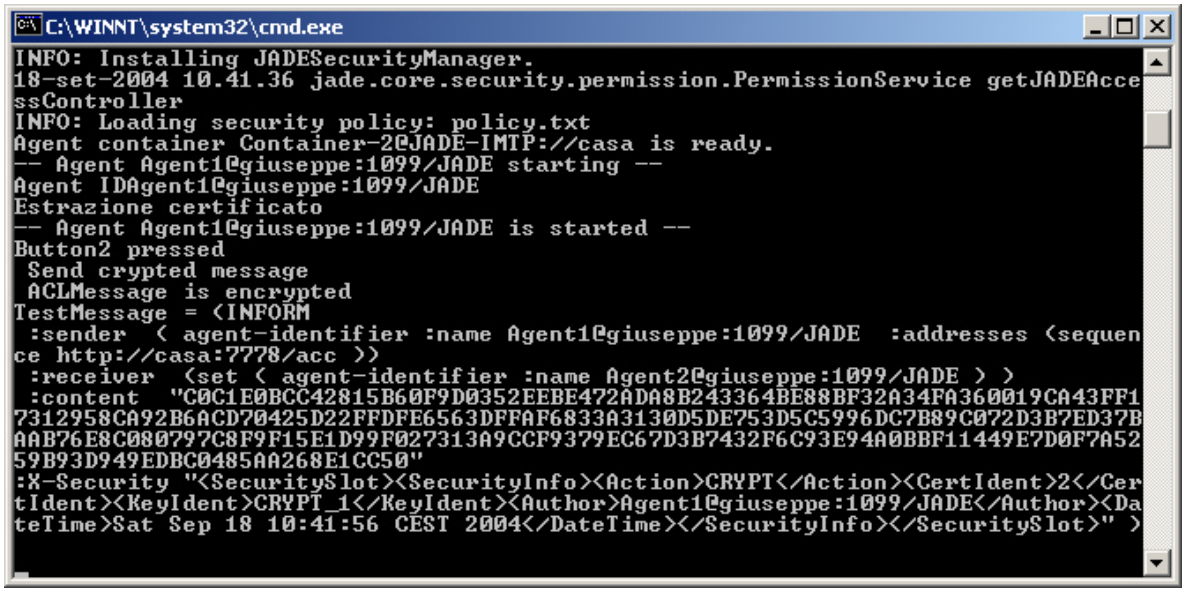


Figure 3.10: Sessione dos Agent1

Analizzando i messaggi contenuti nella sessione dos dell'Agent2 possiamo vedere cosa viene ricevuto e quali sono le azioni che vengono compiute per risalire al testo originale.
Si vede subito che l'Agent2, ricevuto un messaggio, individua immediatamente chi è il mittente. Si accorge che la struttura del messaggio contiene lo slot "X-Security" e si comporta di conseguenza. Nella sessione dos compare il messaggio "Incoming ACL Message was secured".
A questo punto utilizza la propria chiave privata per l'estrazione del contenuto del messaggio e lo mostra a video.

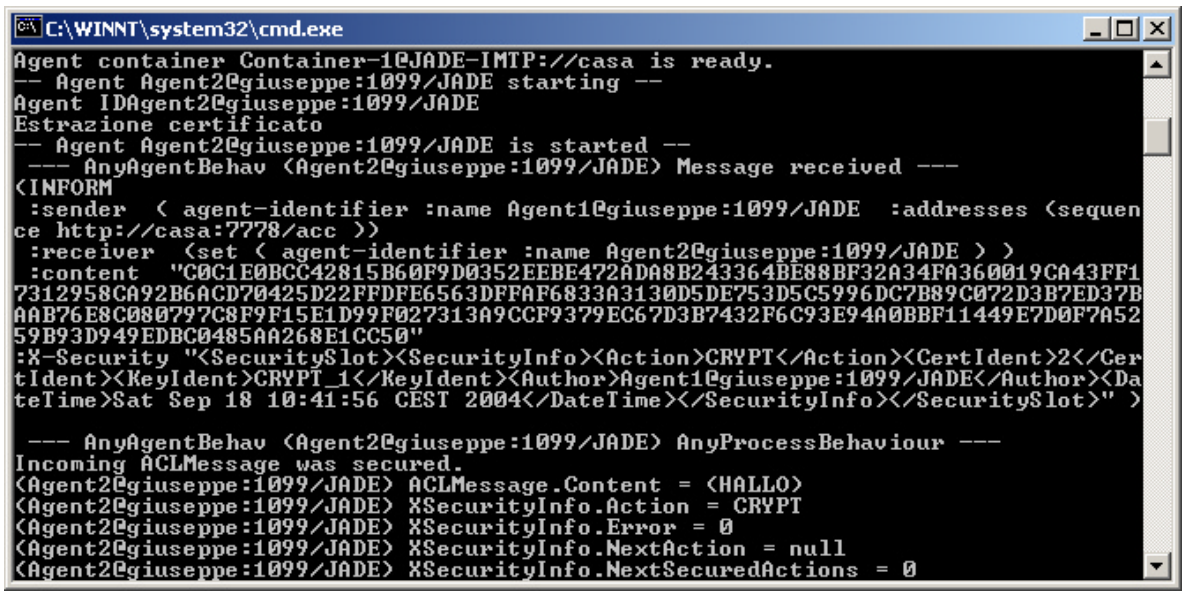


Figure 3.11: Sessione dos dell'agente receiver

Stesso discorso vale per i messaggi che vanno nella direzione inversa.

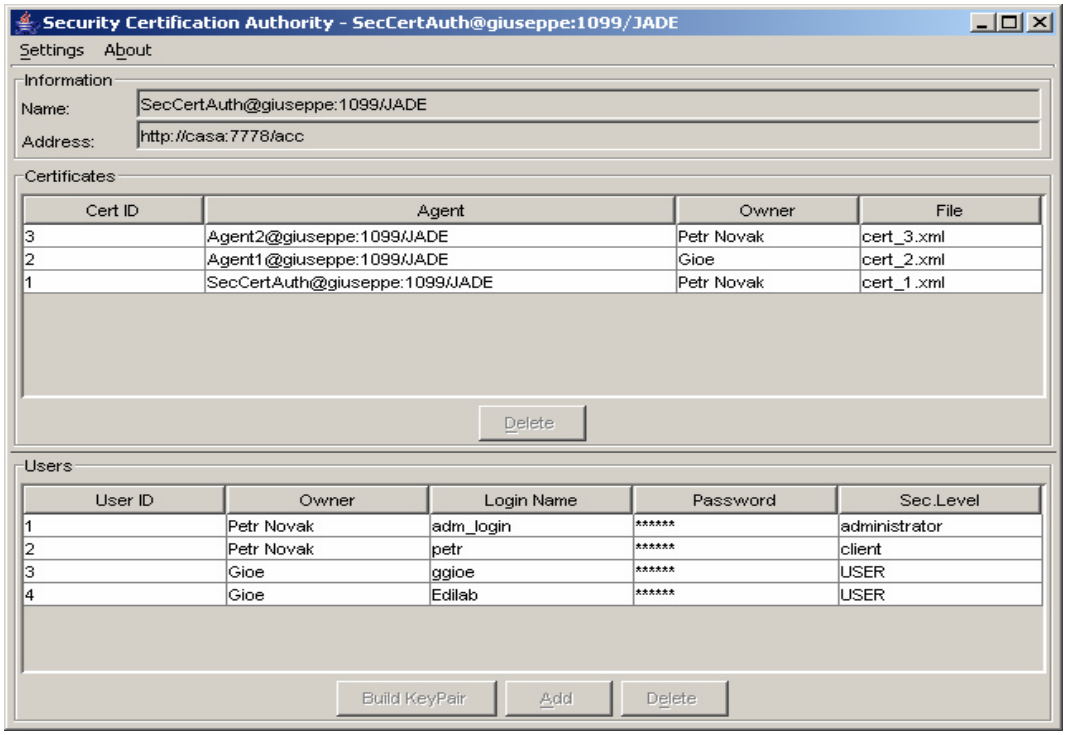


Figure 3.12: Agenti registrati nella SCA

3.8 Considerazioni sull'uso delle impronte digitali

Come ripetutamente detto si è usata l'impronta digitale per far si che ad eseguire il processo di autenticazione potesse essere chi realmente possiede i requisiti per accedere alla piattaforma e non da un impostore che venga in possesso di username e password. Si è voluto legare l'autorizzazione e la conseguente assegnazione di ownership a un qualcosa di fisico dell'individuo, l'impronta digitale è stata quella che si è pensato potesse essere migliore in fase di sperimentazione perché è quella il cui prelevamento richiede meno problemi dal punto di vista degli utenti, cioè è la caratteristica fisica che un individuo è meno restio a prestare tra le possibili.
Al posto dell'impronta digitale si sarebbero potute infatti usare altre caratteristiche che individuano univocamente l'uomo come ad esempio la scansione della retina.
Durante il corso della sperimentazione si sono evidenziate diverse problematiche dovute alla diffidenza degli individui che sono stati invitati a prestare la loro impronta digitale.
Diversi si sono mostrati restii a farsi prelevare l'impronta digitale, pur sapendo che era per soli fini sperimentali, probabilmente il fatto di rendere pubblico un qualcosa che è intimo e legato alla propria persona blocca l'individuo a prestarsi per tali fini.
Una conclusione potrebbe essere che l'uomo, per farsi prelevare l'impronta digitale o un'altra sua caratteristica biometria, deve avere un riscontro che ne valga lo sforzo che sta eseguendo. Un sistema che prevede il riconoscimento tramite impronta digitale, quindi, deve offrire un servizio per cui l'utente decida che vale la pena consegnare la propria impronta digitale.

Conclusioni

In questo lavoro sono state analizzate le minacce alla sicurezza a cui possono essere soggetti i sistemi distribuiti ed in particolar modo quelli ad agenti mobili. Un sistema ad agenti mobili è costituito da una moltitudine di entità che, per lo studio della sicurezza, possono essere catalogate in due gruppi: gli agenti e le piattaforme che li ospitano. Ogni entità appartenente a uno di questi due gruppi può essere sia un potenziale attaccante che oggetto di minaccia; per questo motivo lo stesso tipo di attacco (per es. il masquerading) va analizzato per ogni possibile combinazione attaccante-attaccato per poter meglio individuare le possibili soluzioni.
Le quattro possibili categorie di attacco che si possono avere all'interno di un sistema ad agenti mobili sono:

- agente-piattaforma,
- agente- agente,
- piattaforma – agente
- qualunque altro ente esterno - piattaforma

Oltre agli attacchi tra i componenti del sistema (i primi tre) in un sistema aperto in rete si deve tener conto anche degli attacchi che possono venire dall'esterno (il quarto) e che possono minacciare sia le singole entità che l'intero sistema ad agenti.
Le soluzioni illustrate in questo rapporto comprendono sia strumenti già utilizzabili e collaudati, sia idee non ancora implementate. Il particolare che le accomuna tutte, e che va sempre tenuto presente quando si parla di agenti mobili, è:
"più sicurezza si introduce più si limita la libertà di movimento degli agenti, si diminuisce la velocità delle trasmissioni in rete e si rallenta l'esecuzione dei processi".
La piattaforma realizzata consente di neutralizzare alcune minacce alla sicurezza dei sistemi ad agenti mobili riportate nel capitolo 2. Ricordiamo che una delle cause principali che frena l'utilizzo in applicazioni reali degli agenti mobili è appunto la mancanza di fiducia nei meccanismi di sicurezza. In particolare, la piattaforma realizzata consente di individuare i seguenti tipi di attacchi.

- Piattaforma – Agente;
- Agente – Piattaforma;
- Agente – Agente.

Attraverso l'uso di JAAS e delle policy di sicurezza è possibile fare in modo che un agente, legato ad un utente autenticato, non possa attaccare la piattaforma. Per default tutti i permessi sono negati, l'amministratore della piattaforma deve abilitare i permessi ai singoli agenti con riferimento all'identità del soggetto che lo utilizza. Ricordiamo che JAAS consente di associare permessi diversi al codice a seconda dell'utente che lo utilizza. Stesso discorso vale per gli attacchi piattaforma contro agente e gli attacchi agente contro agente.
Altro aspetto molto importante da considerare è il fatto che è ciascun agente è associato un utente responsabile delle sue azioni. Nella piattaforma realizzata si utilizza un paradigma di autenticazione degli utenti legato a quello che sono, quindi siamo certi che

non è possibile lanciare agenti nella piattaforma senza essere la persona che dice di essere.
Anche nei casi in cui l'utilizzo dell'informazione biometrica non è sufficiente ad avere la certezza dell'identità della persona che si sta autenticando (la verifica dell'impronta ritorna un risultato di affinità prossimo al 100% ma non uguale) il possesso del certificato consente di potere assicurare l'accesso alla piattaforma. Analogamente, anche se la verifica dell'impronta viene superata la mancata disponibilità del certificato impedisce comunque l'accesso alla piattaforma.
Mentre non sono stati trattati gli attacchi del tipo "Altri – Sistema ad agenti". L'host che ospita la piattaforma deve essere reso sicuro per mezzo di altri sistemi di sicurezza. Ad esempio è possibile utilizzare firewall, proxy, antivirus, reti client /server con protocolli di comunicazione sicuri. Gli stessi certificati X509 e l'autenticazione biometrica può essere utilizzata per riconoscere l'utente che vuole aprire una sessione in quella macchina
.

Bibliografia

1. V. Conti, G. Pilato, S. Vitabile, F. Sorbello, “A Robust System for Fingerprints Identification”, Knowledge-Based Intelligent Information Engineering System & Allied Technologies, Crema 16-18 September 2002, pp. 1162-1166.

2. S. Vitabile, V. Conti, G. Pilato, C. Ferrara, F. Sorbello (2003). “Agents Ownership Setting by User Fingerprints”, Proc. of the Workshop Dagli Oggetti agli Agenti - Sistemi Intelligenti e Computazione Pervasiva - (WOA'03), Pitagora Editrice Bologna, ISBN 88-371-1413-3, Villasimius (CA), Italy, 10-11 Settembre 2003.

3. V. Conti, S. Vitabile, G. Pilato, and F. Sorbello, "An Enhanced Authentication System for the JADE-S Platform", WSEAS Transaction on Information Science and Applications, Issue 1, Vol.1, pp. 178 – 183, July 2004, ISSN 1790-0832

4. S. Vitabile, G. Pilato, V. Conti, and F. Sorbello, “Fingerprint in User Authentication Process and Agent Ownership” accepted to Accademia di Scienze, Lettere ed Arti, 2004

5. S. Vitabile, G. Pilato, V. Conti, G. Gioè, F. Sorbello, “Biometric Features for Mobile Agents Ownership”, IPSI BgD Transactions on Internet Research, Issues in Computer Science and Engineering, a publication of IPSI Bgd Internet Research Society New York, Frankfurt, Tokyo, Belgrade, 2004

6. S. Vitabile, V. Conti, G. Pilato, F. Sorbello (2003). “A Fingerprint Based Authentication System for the JADE-S Platform”, Agentcities ID3, Feb. 6-8, 2003, Barcelona, Spain.

7. V. Conti, G. Pilato, S. Vitabile, F. Sorbello, “Verification of Ink-on-paper Fingerprints by Using Image Processing Techniques and a New Matching Operator”, AI*IA Siena 10-13 September 2002.

8. Sucurity Working Group: Security Requirements for the Agentcities Network - http://www.agentcities.org/out/00023/actf-out-00023a.pdf

9. W. Jansen, T.Karygiannis,” NIST Special Publication 800-19 – Mobile Agent Security”, National Institute of Standards and Technology,Computer Security Division, Gaithersburg MD 20899

10. F. Bellifemmine , A. Poggi , G. Rimassa. “JADE - A FIPA compliant agent framework”, 4th International Conference and Exhibition on the Pratical Application of Intelligent Agents and Multi-Agents, UK, 1999.

11. Poggi A., G. Rimassa, M. Tomaiuolo “Multi-User and Security Support for Multi-Agent Systems”, AI*IA – TABOO, Modena 4-5 September 2001.

12. D. Chess. “Security Issues in Mobile Code Systems”, High Integrity Computing Lab, IBM T.J. Watson Research Center, Hawtone, Ny, USA 1998

13. G. Vitaglione, “Jade Tutorial Security Administrator Guide”.

14. G. Vigna. Mobile Code Technologies, Paradigms and Application. Ph. D. Thesis Politecnico di Milano, 1998

15. http://sharon.cselt.it/projects/jade

16. http://java.sun.com/security/

17. http://sharon.cselt.it/projects/jade/doc/tutorials/SecurityAdminGuide.pdf.

18. W. Jansen. Countermeasures for Mobile Agent Security. In Computer Communications, Special Issue on Advances in Research and Application of Network Security, November 2000.

19. R. Wahbe, S. Lucco, T. Anderson. Efficient Software-based Fault Isolation Proceedings of the 14th ACM Symposium on Operatine System Principles, dicembre 1993.

20. S. Poslad & M. Calisti, Towards improved trust and security in FIPA agent platforms, Autonomous Agents 2000 Workshop on Deception, Fraud and Trust in Agent Societies, Barcelona, June 2000.

21. Corradi, A., R., Montanari and C., Stefanelli. Security issues in mobile agent technology. Distributed Computing Systems, 1999. Proc. 7th IEEE Workshop on Future Trends of distributed computing systems, 3 -8, (1999).

22. L., Korba. Towards Secure Agent Distribution and Communication, Proc. 32nd Hawaii International Conference on System Sciences, 10pp, (1999).

23. Farmer, W.M., J. D. Guttman, and V Swarup. Security for mobile agents: Authentication and state appraisal. Proc. 4th European Symposium on Research in Computer Security, Springer-Verlag Lecture Notes in Computer Science No. 1146, pages 118-130, (1996).

24. P. Novak, M. Rollo, J. Hodik, T. Vlcek, M. Pechoucek, “X-Security Architecture in AgentCities”.

25. Http://java.sun.com/j2se/1.4/docs/guide/security/spec/security-spec.doc1.html

26. T. D. Peddireddy, J. M. Vidal: “Multiagent Network Security System using FIPA-OS”, Proceedings IEEE SoutheasCon 2002

27. Yip . J. Cunningham, “Some Issues on Agent Ownership”, LEA 2002 Workshop On The Law Of Electronic Agents

28. B.S. Yee. A Sanctuary for Mobile Agents, aprile 1998. J. Vitek, C.D. Jensen, Secure Internet Programming. Lecture Notes in Computer Science N. 1603

29. S. A. DeLoach, Multiagent System Engineering: A Methodology And Language for Designing Agent Systems, In. Proc. Of Agent-Oriented Information System (AOIS)'99

30. Foundation for Intelligent Physical Agents, FIPA Agent Management Specification, Geneva Switzerland, October 2000

31. Foundation for Intelligent Physical Agents, FIPA Agent Management Support for Mobility Specification, Geneva, Switzerland, October 2000

32. W. Farmer, S. Guttman, V. Swarup. Security Mobile Agents: Authentication and State Appraisal, settembre 1996.

33. J.J. Ordille. When Agents Roam, Who can you trust?, Proceedings of the 1st Conference of Emerging Techonologies and Applications in Communications, maggio 1996.

34. V. Roth. Secure Recording of Itineraries Throuh Cooperating Agents. Proc. Of the ECOOP Workshop on distributed object security and 4th workshop on mobile object systems: Secure Internet computation, pp. 147-154, INRIA Francia, 1998.

35. g. Vigna. Protecting Mobile Agent through Tracing, Proceedings of the 3rd ECOOP Workshop on Mobile Object Systems, giugno 1997.

36. T. Sander, C. Tschudin. Protecting Mobile Agents Against Mailicius Hosts. G. Vigna Ed., Mobile Agents and Secuirty. Lecture Notes in Computer Science No. 1419.