



# Secure Mobile Agents in JADE

Francesco Librizzi

---

The 2006 miniWorkshop on Security Frameworks

- Security in Mobility -



# What 's Jade?



- ▶ JADE: Java Agent DEvelopment Framework;
- ▶ It represents the environment where the agents live;
- ▶ It provides a library of classes for developing MAS;





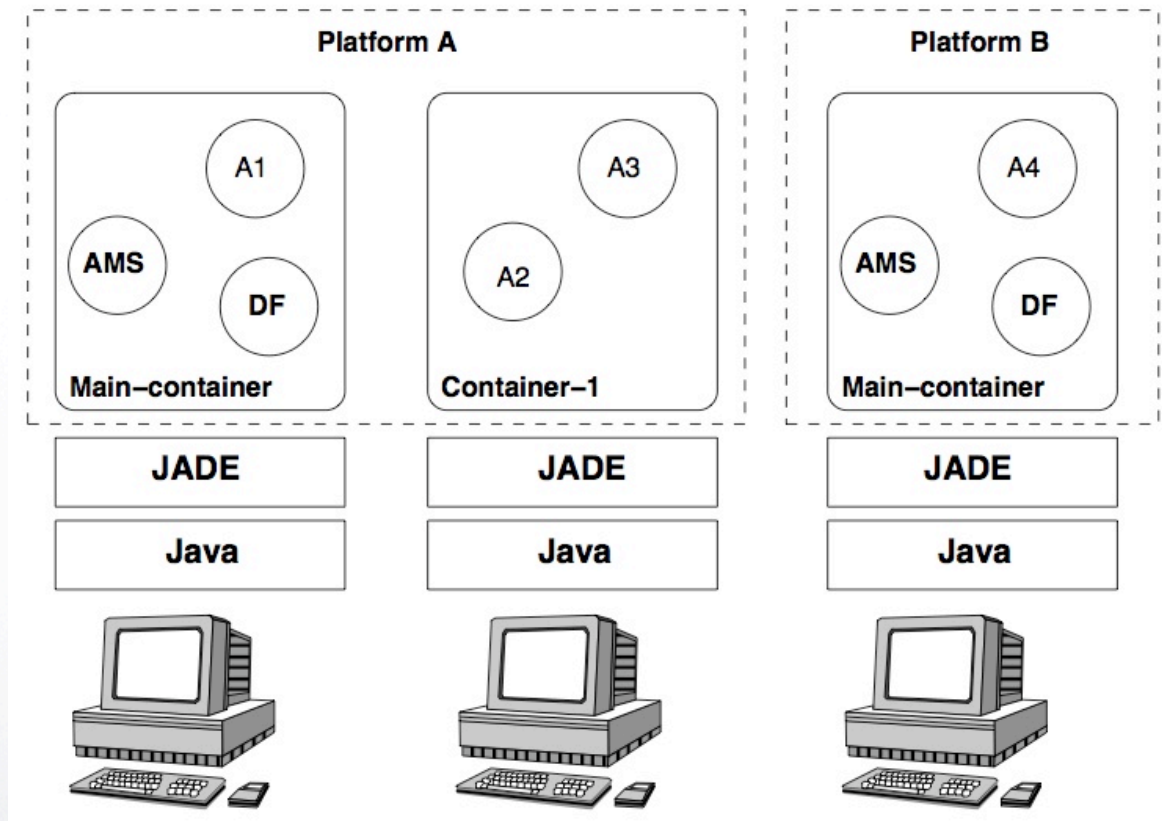
# Services of Jade



- ▶ Communication;
- ▶ Naming and addressing (“White Pages”);
- ▶ Task (Behaviour) scheduling execution;
- ▶ Yellow Pages / Ontology services;
- ▶ Mobility.



# Jade's Architecture




- ✓ AMS: Agent Management System (white pages);
- ✓ DF: Directory Facilitator (yellow pages);





# Mobility of Agent



- ▶ An agent can move to others machines to retrieve information or interact with others agents;
- ▶ Move agents  Move code agents

## ✓ Pro:

- Bandwith Saving;
- Network Fault-tolerance;
- Disconnected Operations;

## ✓ Contra:

- Code VS Message;
- Killer-Application;
- Security.



# Attacks on Jade



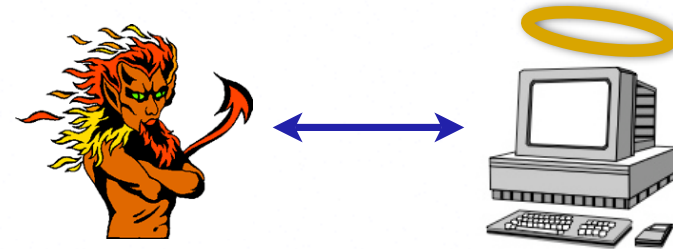
There are three types of attacks:

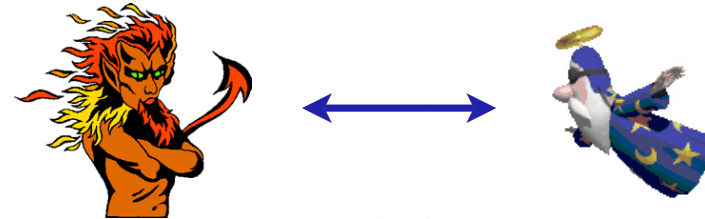
- ▶ Agent VS Platform;
- ▶ Agent VS Agent;
- ▶ Platform VS Agent.



Three possible attacks:

- ▶ DoS (Denial of Service);
- ▶ Trojan Horse;
- ▶ Flying Dutchman;

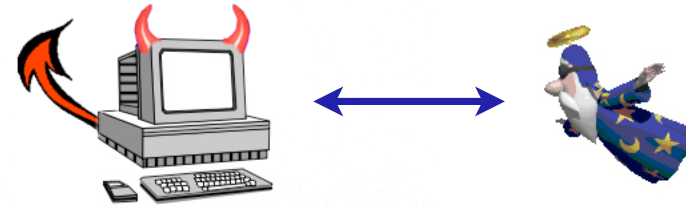




- ▶ DoS:
  - direct attack;
  - indirect attack;
  
- ▶ Spoofing/Man-in-the-middle;



- ▶ **Code/Data Alteration** (byte-code recompiled into source code);
- ▶ **Replay Attacks;**
- ▶ **Malicious Routing;**
- ▶ **Misinformation/DoS.**





# SAgent



- ▶ Security Framework for Jade;
- ▶ It focuses on the security of mobile agents;
- ▶ It offers two different views:
  - point of view for applications developer;
  - point of view for who develops protection techniques;
- ▶ It is distributed freely under the GNU LGPL license;





## SAgent (2)



- ▶ The actors of the scenario:
  - Originator;
  - Mobile Agents;
  - Host;
  
- ▶ SAgent supports the dynamic routing;



## SAgent (3)



- ▶ Goals of SAgent are privacy and integrity for:
  - Agent code;
  - Agent State;
  - Host Data Input;





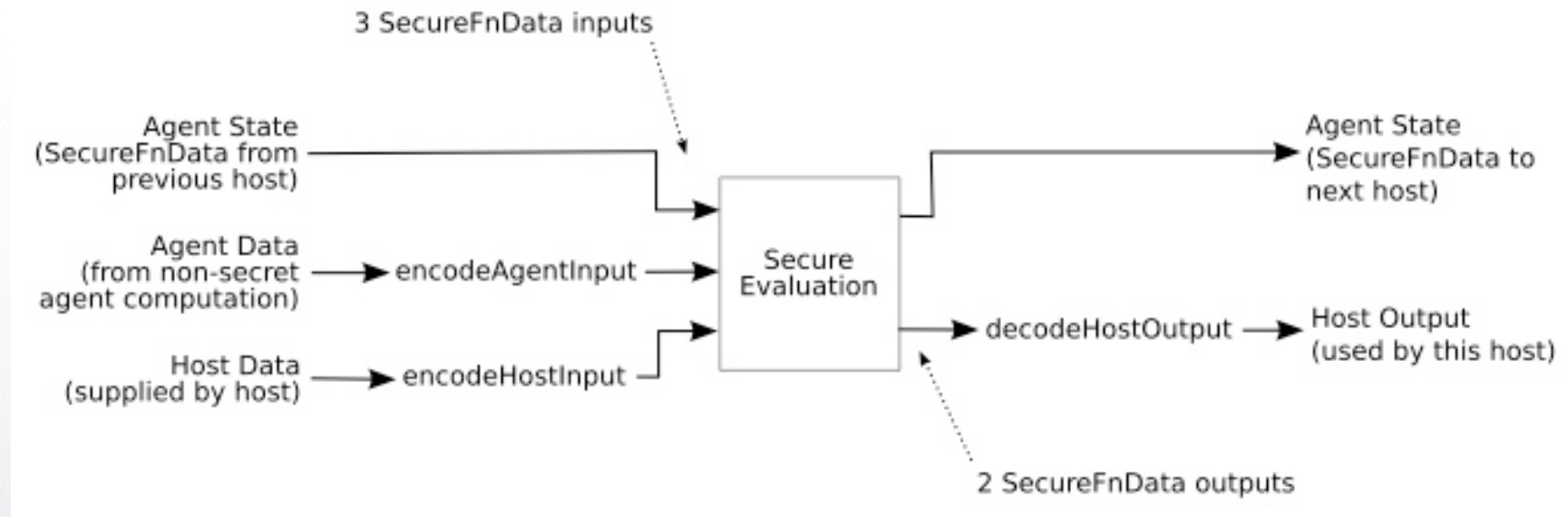
# Security Model



- ▶ SAgent separates the computations in:
  - protected computations;
  - unprotected computations;
  
- ▶ The protected computations are defined in the next image;

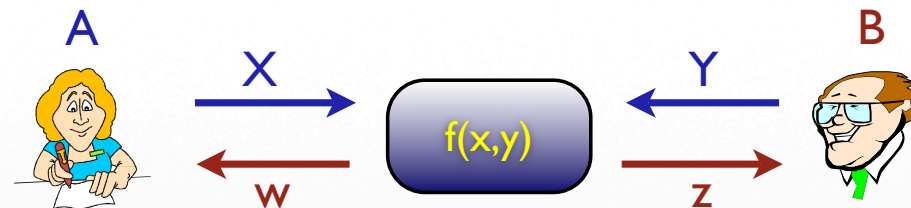


# Security Model (2)



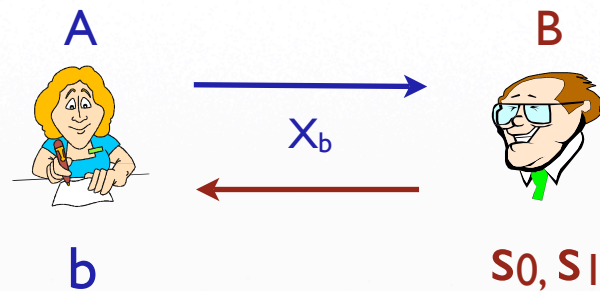


## two-party Secure Function Evaluation (Yao's protocol):



- ▶ The SFE guarantees that:
  - A will know only  $w$ ;
  - B will know only  $z$ ;

- ▶ It guarantees that the input host remains unknown to the Originator.



After the execution of the protocol:

- ▶ A receives  $s_b$  without learning anything about  $s_{1-b}$ ;
- ▶ B doesn't learn  $b$ ;





# How does SAgent do it?



- ▶ SAgent offers three different ways to achieve security:
  - using ACCK and TX Protocols (encrypted circuit);
  - making a personal protocol using the homomorphic encryption;
  - using trusted hardware;



# The SAgent protocols



- ▶ **ACCK: Uses a trusted third party (TTP)**  
due to Algesheimer, Cachin, Camenisch, and Karjoth. May 2001;
- ▶ **TX: Uses threshold cryptography and multiple agents to obviate need for TTP**  
due to Tate and Xu. Year 2003.





# The phases of the protocols



## 1. Initialization:

- The originator creates the encrypted circuit;
- It encodes the agent state as signals for the encrypted circuit;
- It encodes all possible input signals;

## 2. Evaluation:

- Host gets the signal for its input;
- Protocols ensure that only a single input can be retrieved;
- Encrypted circuit is evaluated;
- Hosts can decode their outputs;

## 3. Finalization:

- The originator decodes the agent state by using the semantics of the signals.



# Notations



- ▶  $C$  = encrypted circuit;
- ▶  $L$  = signal for the agent state;
- ▶  $K$  = set of the inputs signal for the host input;
  
- ▶  $PK$  = Public Key
- ▶  $PK^{-1}$  = Private Key





# ACCK protocol

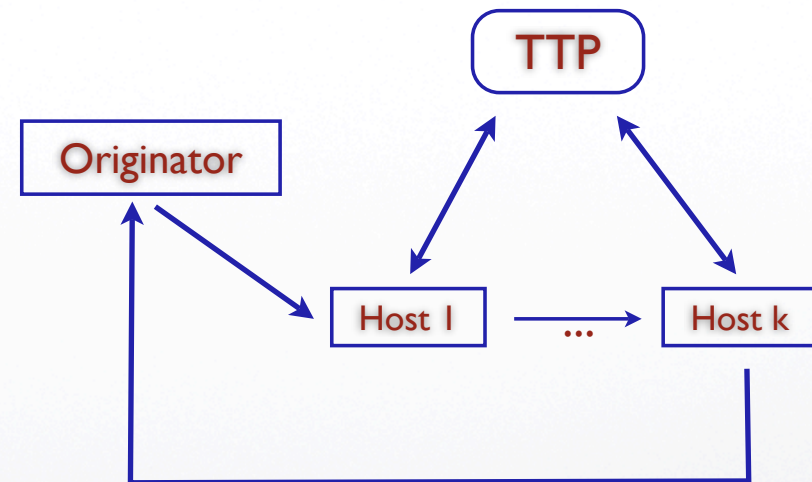


1.  $O \rightarrow H: \{C, L, K\} PK^{-1}_O$

$$K = \{\{K_i\} PK_{TTP}\}_{i=1 \dots 2^{nx}}$$

2.  $H \rightarrow TTP: \{K_i\} PK_{TTP}$

3.  $TTP \rightarrow H/MA: y_i$  (input signal)





# TX protocol



- ▶ To eliminate the TTP, TX protocol makes use of Threshold cryptosystem;
- ▶ TX protocol requires multiple agents, that work together;
- ▶ To achieve oblivious transfer without TTP, it uses OTD (Oblivious Threshold Decryption)





# OTD protocol



It makes use of a trust dealer D.

1. OTD-Setup: D creates PK, VK and n shares  $SK_i$  ( $i=1, \dots, n$ );
2. OTD-Distribute: the agent  $A_i$  sends  $s_0$  and  $s_1$  to  $t-1$  agents;
3. OTD-Share-Creation: the agents verify the identity of  $s_0$  and  $s_1$ . If OK then they decrypt else nothing;
4. OTD-Share-Combination:  $A_i$  “runs” oblivious transfer, verifies the validity of the  $t$  decryption shares and obtains  $s_b$ .



# Phases of TX protocol



▶ The trusted dealer is the Originator (O isn't TTP).

▶ The originator creates:

- PK;
- VK;
- SK<sub>i</sub>;
- C;
- disjoint subsets.

1. O → H: {C,L,K}PK<sup>-1</sup><sub>O</sub>  
K = {{K<sub>i</sub>}PK} i= 1...2<sup>nx</sup>
2. H → tMA: OTD-Distribute(K)
3. tMA:
  - Verify(K)
  - OTD-Creation(K)
  - OTD-Share-Combination(K)
4. H/MA: SFE()
5. H → H+I: {C,L,K}PK<sup>-1</sup><sub>O</sub>  
OR  
H → O: {C,L,K}PK<sup>-1</sup><sub>O</sub>





## TX protocol (2)



### ✓ Advantages:

- No collusion TTP;
- Fault-tolerance;
- Parallelism;

### ✓ Disadvantages:

- It is slower than ACCK protocol;



# Programming with SAgent



- ▶ The framework provides two main interfaces:

```
SecureFnPublic  
  
+initializeSecurity( thisAgent : Agent ) : void  
+encodeHostInput( input : Object, thisAgent : Agent ) : SecureFnData  
+encodeAgentInput( input : Object ) : SecureFnData  
+decodeHostOutput( data : SecureFnData ) : Object  
+evaluate( agentInput : SecureFnData, hostInput : SecureFnData ) : SecureFnData  
+secureMove( thisAgent : Agent, whereTo : Location ) : void
```

```
SecureFnPrivate  
  
+getPublicPart( agentNum : int ) : SecureFnPublic  
+setAgentState( agentNum : int, state : Object ) : void  
+decodeAgentState( agentNum : int, encState : SecureFnData ) : Object
```

- ▶ SecureFnData: is an opaque data to represent protected data, it doesn't provide any method.





# Conclusion



- ▶ With SAgent we can create small applications based on mobile agents;
- ▶ The MAS application can be developed from two teams: a team for the application development and another for secure layer development.
- ▶ We can implement our personal protocols.



# References



▶ Jade:

- TILab (<http://jade.tilab.com/>);
- Lecture notes Corrado Santoro (<http://www.diit.unict.it/users/csanto/>);

▶ SAgent:

- Gunupudi, Tate - SAgent: A security Framework for JADE;
- Gunupudi, Tate and Xu - Experimental Evaluation of Security Protocols in SAgent;
- Tate, Xu - Mobile Agent Security Through Multi-Agent Cryptographic Protocols
- Reference site: <http://cops.csci.unt.edu/sagent/index.html>





THE  
END