Declarative Security Policy

Takanori Ugai

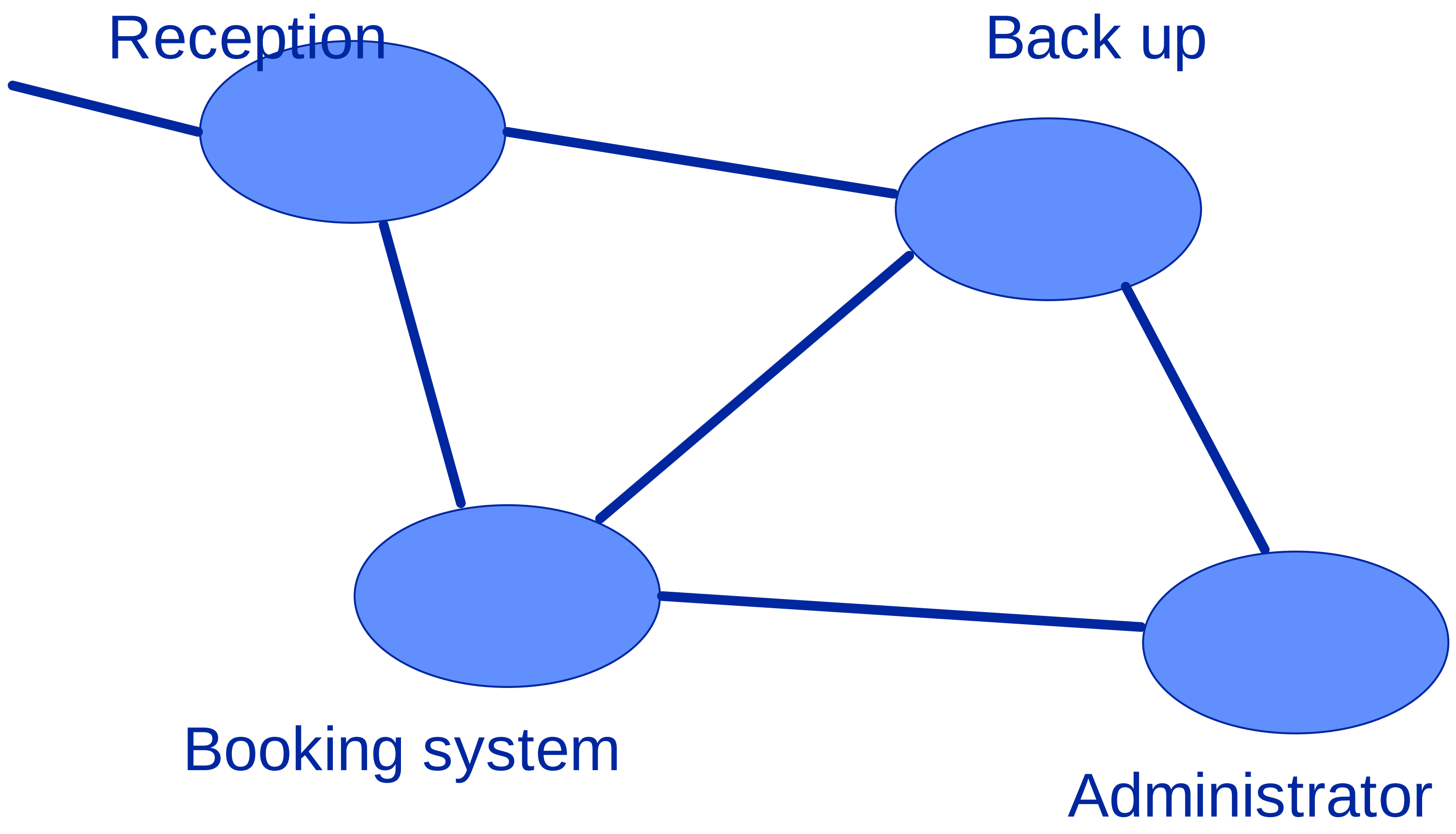
14 October 1997

Motivation

- Self Defense Security Model
  - Recent technology allows enforcement of policies that involve complex interactions between different roles with different agendas and permits conflict resolution between different administrative domains.
  - Traditional policy modeling using access control lists is insufficient to cope with such generality.
- Mobile code (Mobile Agent) security
  - Because a host uploads the class files and state for the agents, they or anyone sniffing the network during transit could potentially read private information or even alter the agent's code and state.

Self Defense Security Modeling

Flight Booking System



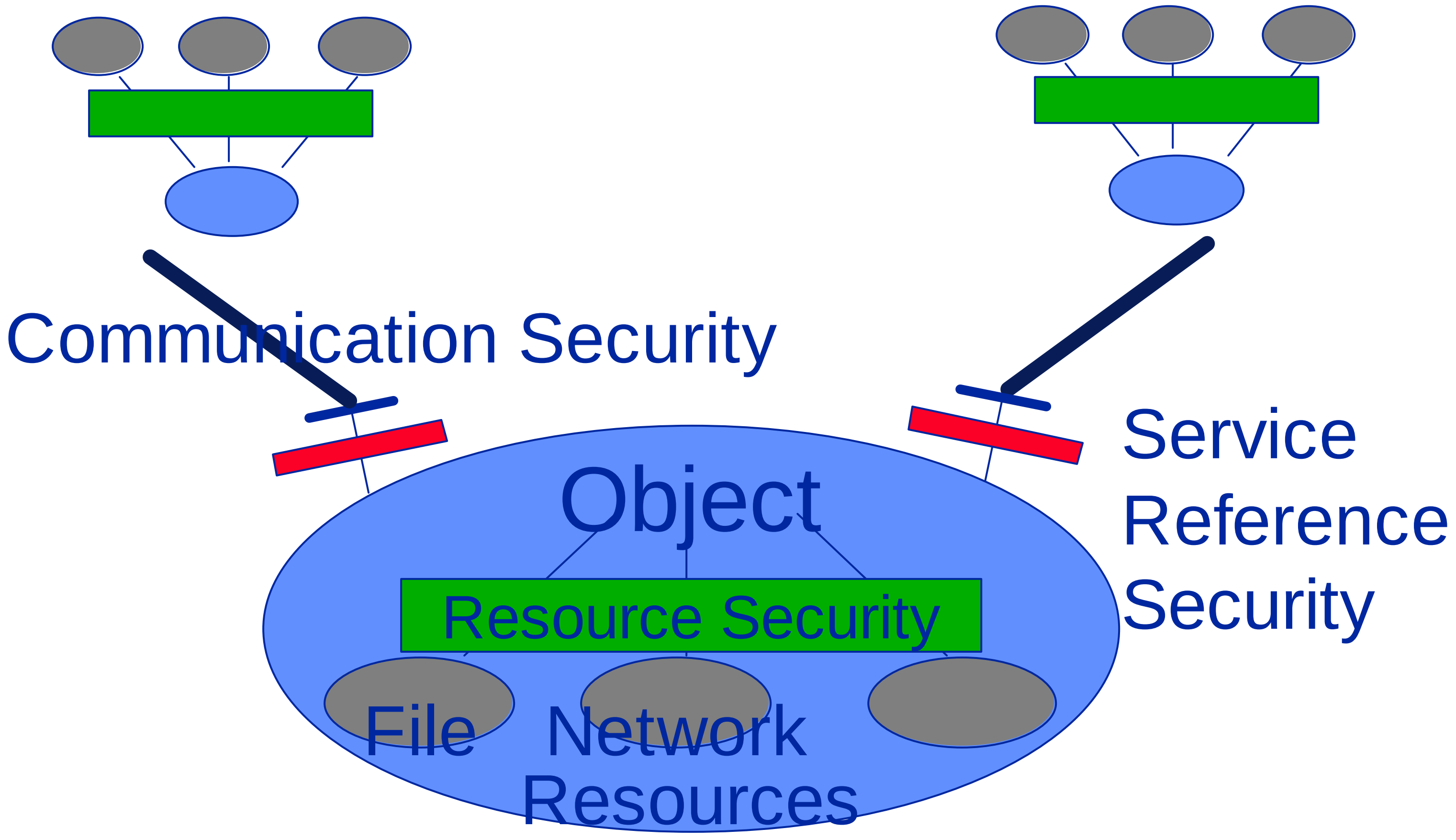
Booking System's security policy

- Receptionist can perform query and update..
- Administrator and Back up can esquire about status.
- Back up can access the data.
- Physical separated room from the reception
- Administrator cannot access the readable data.
- Card + Password Authentication
- 64 bit encryption communication.

Problem

- How organize these peace of policies?
- How achieve a global security policy?

Self Defense Security Modeling



Resource Security

- Authorization control for accessing internal data, and infrastructure resources.

Service Reference Security

- Service reference (Interface reference) access control
  - on using the service
- Service reference configuration control
  - on setting, removing and modifying the service

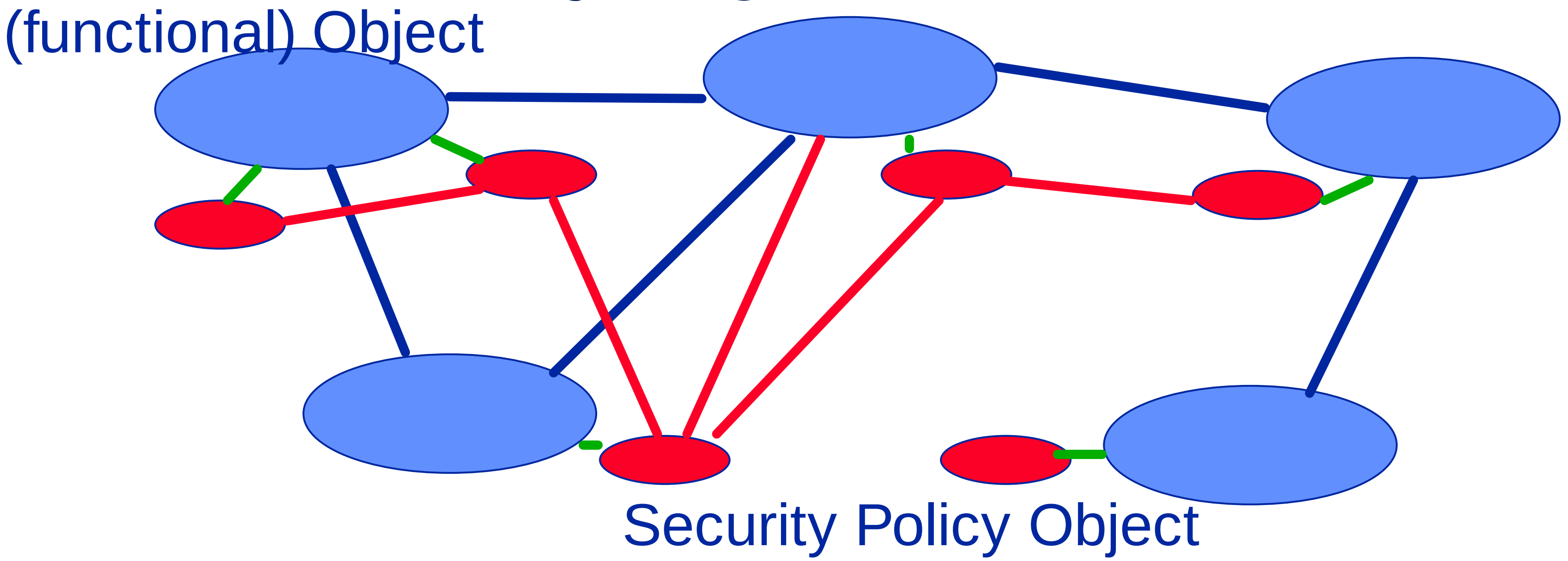
Communication Security

- Authentication protocol
- Confidentiality policy (Encryption algorithm)
- Key management

Worker’s policy

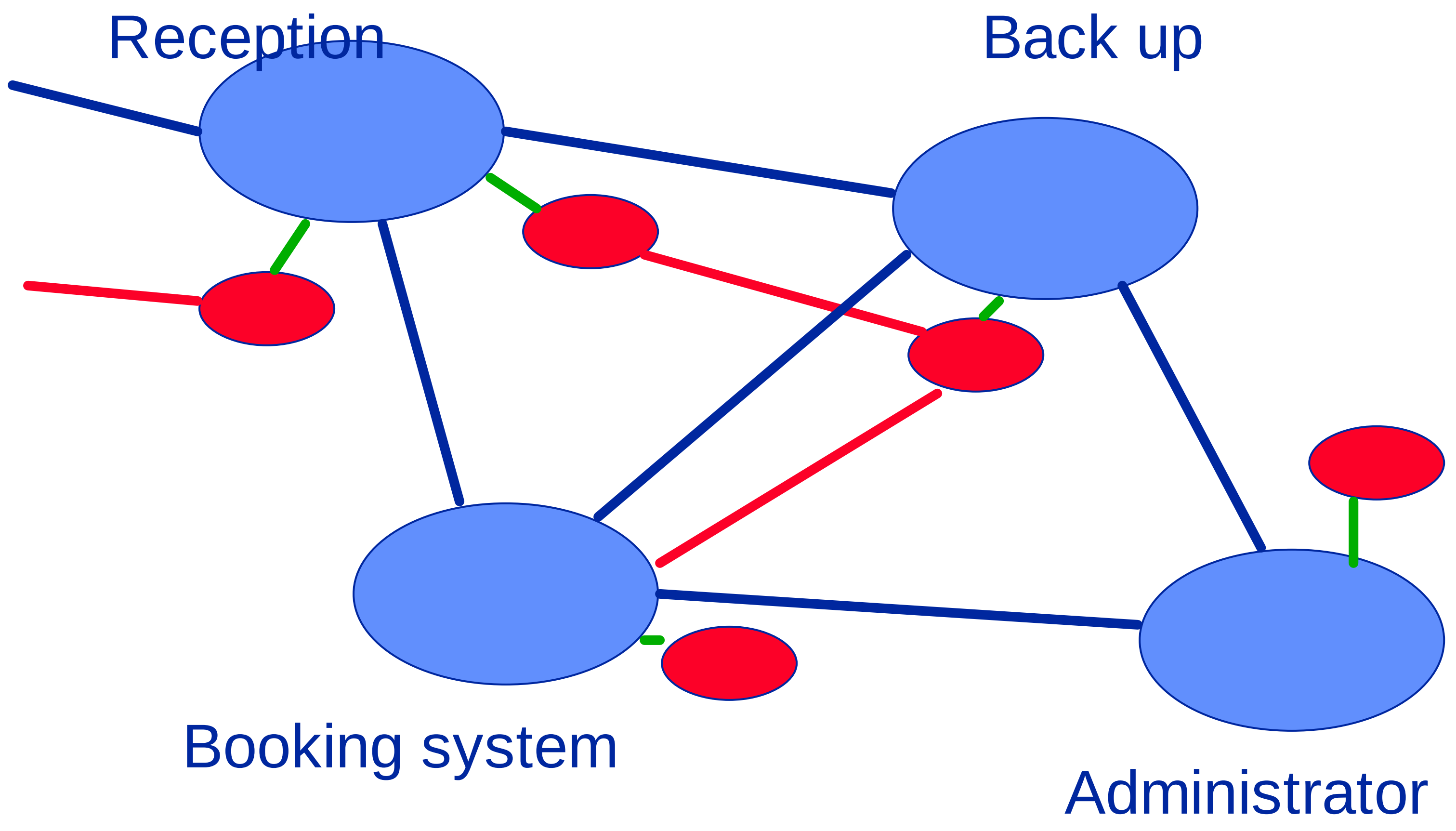
- Service Reference Security
  - Receptionist can perform query and update..
  - Administrator and Back up can esquire about status.
  - Back up can access the data.
- Resource Security
  - Physical separated room from the reception
  - Administrator cannot access the readable data.
  - Card + Password Authentication
- Communication Security
  - 64 bit encryption communication.

Policy Object Network



▬	Communication between (functional) objects
🟥	Communication between security policy objects and between (functional) object and security policy object
🟩	Control message for security policy object to manage the object

Flight Booking System



Self Defense Security Modeling (Summery)

- System security policy can be captured and expressed as three aspects :
  - Resource security (Computation model)
  - Service reference security (Enterprise Model)
  - Communication security (Information model)
- A policy describes the action which may or must be taken by a principal when acting in a particular role. Policy associated with a role is expressed in one or more related policy objects.
- An overall policy could be designed and implemented as a network of policy object.

Kafka

- 100% pure Java Agent Library
- Dynamic service interface support based on Runtime Reflection
- Mobile code support based on Remote Evaluation
- Distributed Naming Service
- Java RMI and HORB (Hirano's ORB produced by Dr. Hirano at ETL, Japan) support
- Free with all sources for internal and research use.

Implementation for Kafka

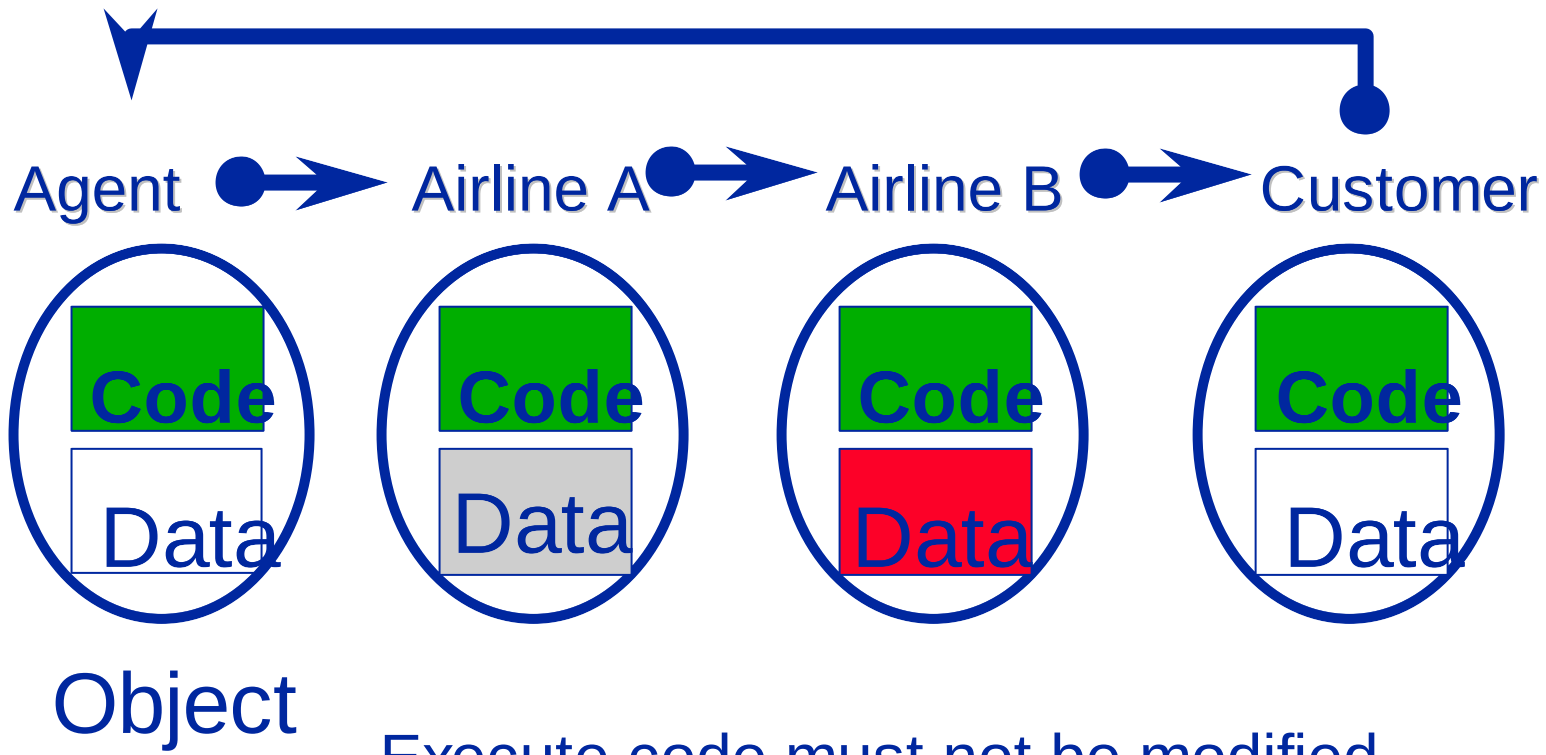
- Resource Security
  - build by Java (safe language)
  - Agent security manager (an extension of the Java security manager)
- Service Reference Security
  - Access Control Class, which support authorization by each method call.
- Communication Security
  - 3 kind of authentication modules
  - 2 kind of encryption communication modules

Current Issues

- Developing an security policy expression language. (currently OMT notation is used)
  - Translator to the configuration of administration tools
- Developing tools supporting the model
- Mathematical model supporting the self defense model.
  - Security requirements validation
  - Solving security requirements conflicts

Self Defense Mobile Agent

Flight Booking Agent



Execute code must not be modified.
Data may be modified.

If B can read/write data freely, B might modify offer by A

Specification

- There are four hosts : a customer host, a travel agency host, and two servers owned by competing airlines which we assume for the sake of this example do not share a common reservation system.
- The mobile agent is programmed by a travel agency by a customer’s request.
- The customer dispatches the agent to the Airline A server where the agent queries the flight database.
- With the results stored in its environment, the agent then migrates to the Airline B server where again it queries the flight database.
- The agent compares flight and fare information, decides on a flight plan, migrates to the appropriate airline host, and reserves the desired flights.
- Finally, the agent returns to the customer with the results.

Assumptions & Problem

- The customer can expect that the individual airlines will provide true information on flight schedules and fares in an attempt to win her business, just as we assume nowadays that the reservation information the airlines provide over the telephone is accurate, although it is not always complete.
- The airline servers are in a competitive relation with each other. The airline servers illustrates a crucial principle: For many of the most natural and important applications of mobile agents, we cannot expect the participants to trust one another.

Generalized Problem

- Because a host uploads the class files and state for agents, they or anyone sniffing the network during transit could potentially read private information or even alter the agent's code and state.

Signed Mobile Code

- Code : signed by Originator (Travel Agency)
- Data :
  - Each data field has the modifier’s signature
  - Each data field has an access control list

Data Fields and Access Control List

- User Preference
  - A: READ, B: READ
  - User: READ,WRITE ;Agent: READ,
- Offer A
  - A: READ,WRITE
  - User: READ ;Agent: READ
- Offer B
  - B: READ,WRITE
  - User: READ ; Agent: READ
- User Confirmation
  - User : READ,WRITE
  - Agent: READ

Offer A

- Data is made by airline A
  - put the A’s signature
  - encrypted using a new secret key
    - add the key encrypted by User’s key
    - add the key encrypted by Agent’s key
    - add the key encrypted by A’s key

$$\Downarrow$$

B cannot read/modify the data

Experimental Implementation

- Kafka Agent support
- DSA (Digital Signature Algorithm) support
- experimental implementation of JCE 1.2 Signed object class.
- RSA support for public key system
- IDEA support for the secret key

Current Issues

- Implementation of Kafka on the FlexiNet.
- Keeping the trace of agent’s movement.

References

- Takashi Nishigaya, “Kafka : Yet Another Multi-Agent Library for Java”, http://www.fujitsu.co.jp/hypertext/free/kafka/index.html
- Iida, I., Nishigaya, T., and Murakami, K., 1995. DUET: Agent-based Personal Communications Network, Proceedings of ISS'95, Vol. 1, 119-123.
- David J Otway, “Design for Secure Remote Method Invocation”, DRA/CIS3/PROJ/CORBA/SC/1/CSM436-07/0/1 APM Ltd 97/03
- Sun Microsystems, Inc. Java Remote Method Invocation Specification.
- Schneier,B “Applied Cryptography”, Wiley ISBN 0-471-12845-7
- Java Cryptography Extension: API Specification, http://java.sun.com/security/JCE1.1/earlyaccess/index.html