Mobile Object Security Developments

8 April 1998

Takanori Ugai

Secure Mobile Object Model

- Local Resource Security for hosts
  - protecting the local resources (memory, CPU, files ...) from hostile mobile objects
- **Secure Communication**
  - **protecting communications and migration from hostile third parties**.
- **Secure Mobile Object (Migration and Contents)**
  - **protecting mobile objects from hostile hosts**.

Secure Communication

- host to host
  - Using an existing SSL implementation for the communication layer
  - Providing security policy API
- object to object
  - require object identity
  - object must reveal proof of identity to its host
    - We assume some public key infrastructure and use the X509 Certificates for objects
  - only reasonable at trusted hosts

Host to Host Policy

- Trust relationships are based on Certificates
  - Need a public key infrastructure to use SSL
    - key management, certificate management, CA....
  - We do not assume a particular public key infrastructure
  - We will provide a sample implementation with demonstration programs.
  - We will develop a service provider interface for the application programmer to use the public key infrastructure.

Object to Object Policy

- Reflexive access to supplied credentials

```
boolean checkAccess(Object o,
                    Method m,
                    Object args,
                    Certificate client)
```

Policy Example

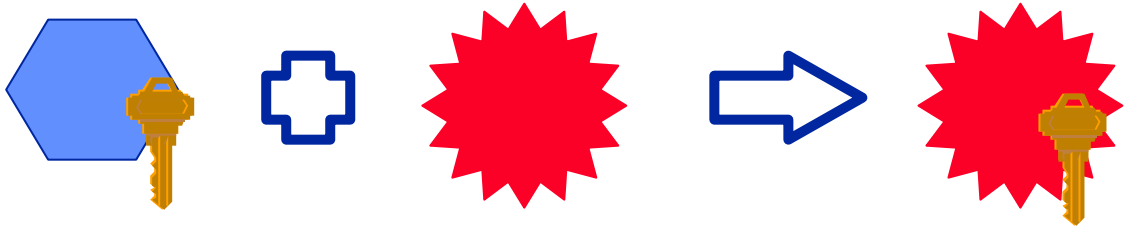
```
public class MyPolicy extends ObjectPolicy {
  public checkAccess(...) {
     if (certificate is certified by ANSA) {
       if (method is read) {
         return ;
       }
     } else if (certificate is certified by APM) {
       return ;
     } else {
       throw new PolicyException(Not Permitted)
     }
  }}
```

Mobile Object Problem

- A mobile object wishes to act on behalf of a user
  - carry with it passwords, credit card details etc.
- A mobile object may be dissected by any host it passes through
  - need to encrypt secret data to prevent access
  - we must not reveal secret information to host that can misuse it.

Agent Integrity Problem

- Hosts must not be able to break object apart and build new ones
  - want to check agents are not modified



- Object's data will be updated during use
  - Object must be modifiable

Code Integrity

- Classes may be maliciously modified
- We prevent this by identifying classes via a secure hash code
- This also provides version management

Data Integrity

- An object may make an integrity statement to its host, that must be validated
  - if we don’t do this, a malicious host could remove secret information from one object and splice it into another
  - the integrity statement has two purposes
    - to allow the host to decide whether to allow the object to run
    - to allow the host to trust secret information revealed to it.

Secure Object

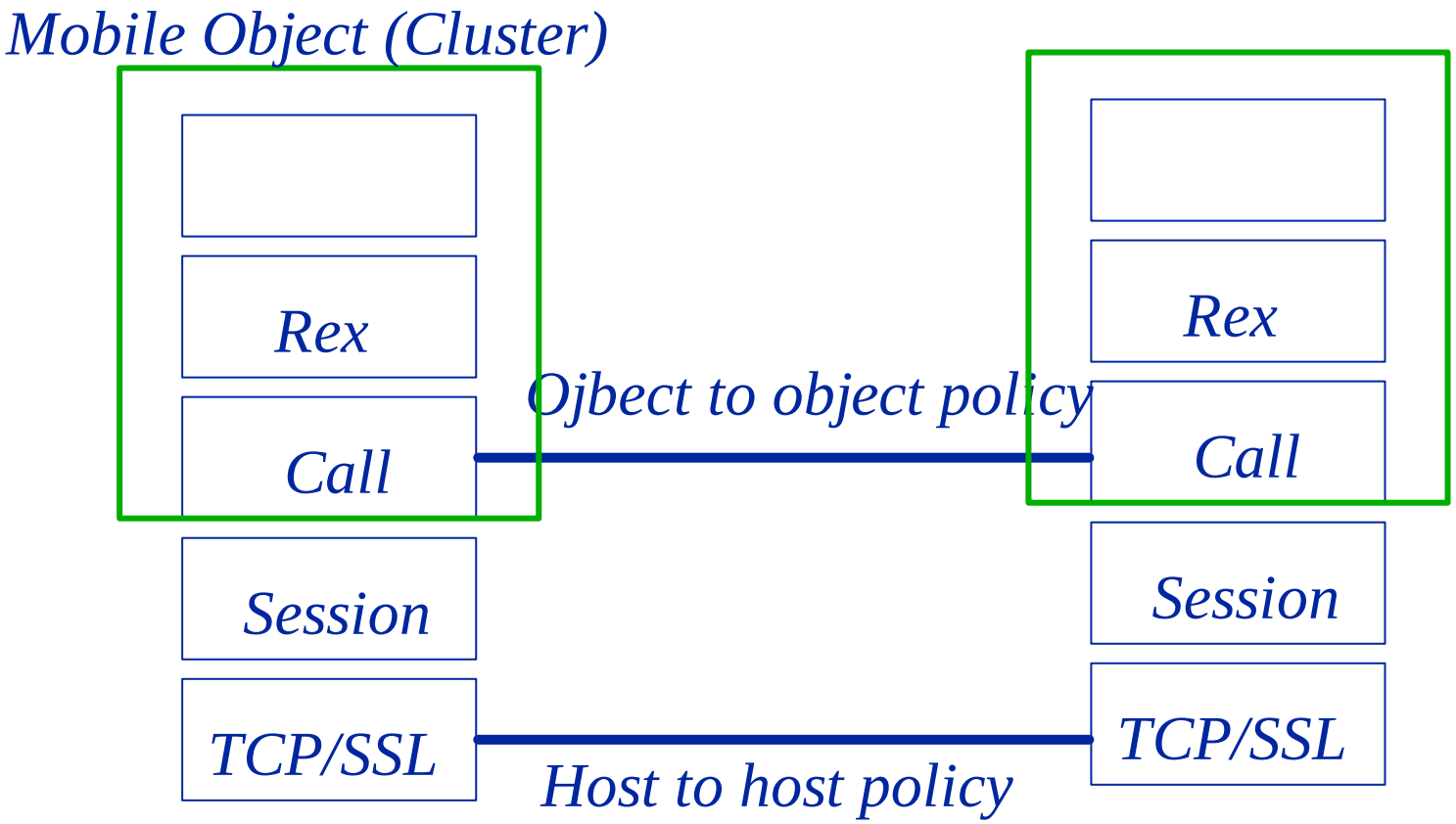
- Revealing/modifying secure data will be transparent
  - get() and put() methods
  - will fail if access is against policy

Implementation Choice

- Caller and ClientCall Layer
  - When the move is called, arguments are committed, and sealed. ClientCall extract the arguments and check verify the arguments.
- Serialiser and Deserialiser Layer

- Class loader is responsible for code integrity. Moving objects can keep some evidence like fingerprint of class data.

Implementation



Current status and working plan

- **SSL FlexiNet with simple security policy (done)**
- **TCP MOW (done)**
- **SSL FlexiNet with Interface/Method base security policy (done)**
- **SSL MOW with simple security policy (2 weeks)**
  - host to host security (done)
  - object to object security
- **SSL MOW with Interface/Method base security policy (2 - 4 weeks)**
- **Secure Object Infrastructure (4-6 weeks)**
- **Demonstoration Programs**
- **Declarative Mobile Security Pre-Processor**

Performance (RPC)

- UDP (default)
  - 7.5 msec/nullcall 400kbps through put
- TCP
  - 8.1 msec/nullcall 500kbps through put
- SSL
  - 130 msec/nullcall 66kbps through put
    - RSA_RC4_SHA
    - handshake 335 msec

166MHz SuperSparc, JDK1.1.5 MOW 1.0

Demonstration Plan

- Voting system (Anonymity)
- Flight Booking system (Information gathering)
- Payment system / Purchasing (User Preference)

Future works

- Negotiable security policy with FlexiNet Framework
- Dynamic security policy with security policy object + policy expression language
- Domain security policy with Java domain security model + enterprise security model

- Security Policy Object Interface
- Policy Expression Language

SSL implementation comparison

- SSLeay
  - Free
  - Faster than IAIK package
  - written in C
  - Java interface is not stable enough
- IAIK
  - Not free ($400 per license)
  - written in Java
  - lots of algorithm and PKCS are implemented.
- JCP
  - Commercial Product ($1500 research license)
  - written in Java
  - less of algorithm and PKCS are implemented.

My requirements to SSL implementation

- Java level X509 Certification manipulation
- Java level SSL parameter manipulation
- Crypto package for implementing Secure Mobile Object
- SSL negotiation algorithm