

Model-based Run-time Checking of Security Permissions using Guarded Objects

Jan Jürjens

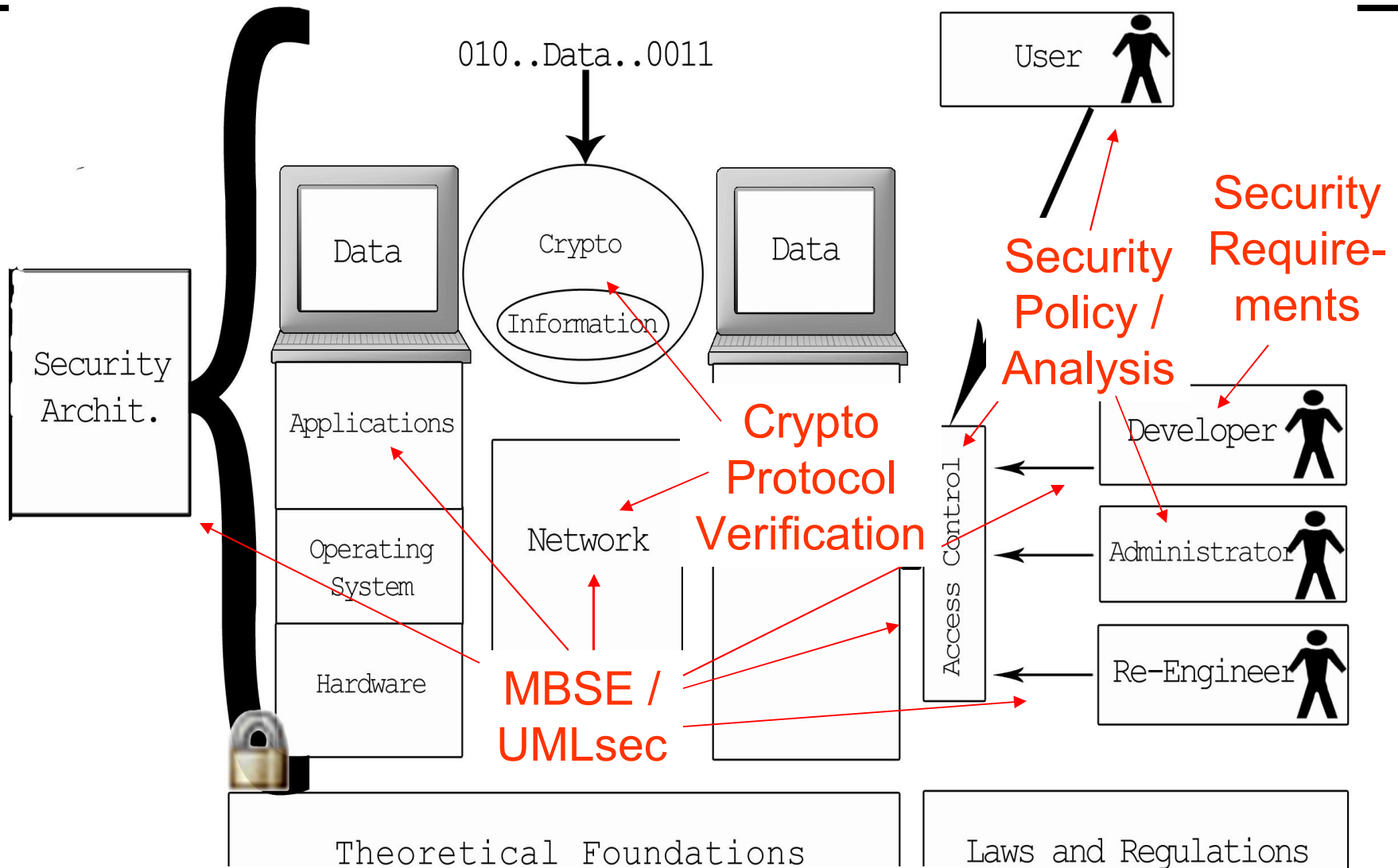
Computing Department
The Open University, GB



J.Jurjens@open.ac.uk

<http://www.jurjens.de/jan>

Security: Systems View



Formal Methods, vs Comput. Complex. Privacy Regulations

Why Run-time Checking of Security Permissions?

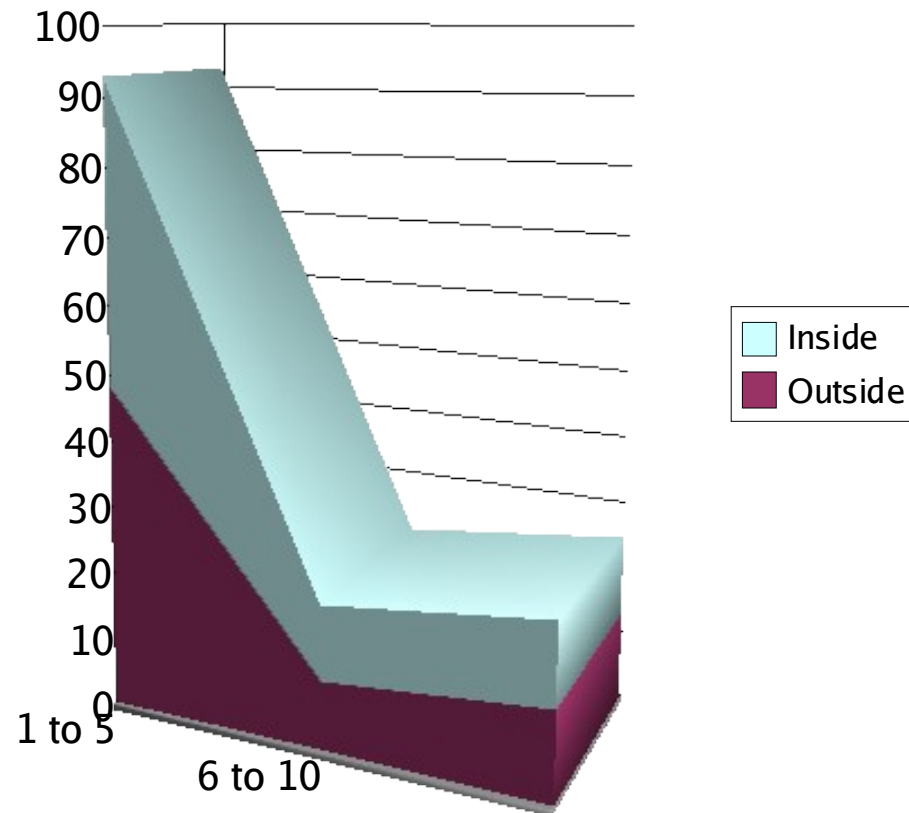
Computer Crime:

- 99% detected breaches
- 223 firms loose \$ 455,848,000
- 50% inside attacks

Automated Analysis

- Manual review of permissions impossible
- Huge amount of data
- Attacks from reviewer ?

Where Attackers come from



Check configuration data

Example: check SAP permissions for security rules. Cannot do this manually:

- large data set (60.000 entries)
- complex interrelations between permissions on different levels
- dynamic changes, delegation
- manual analysis trustworthy ?

Automated check increases security at central point.

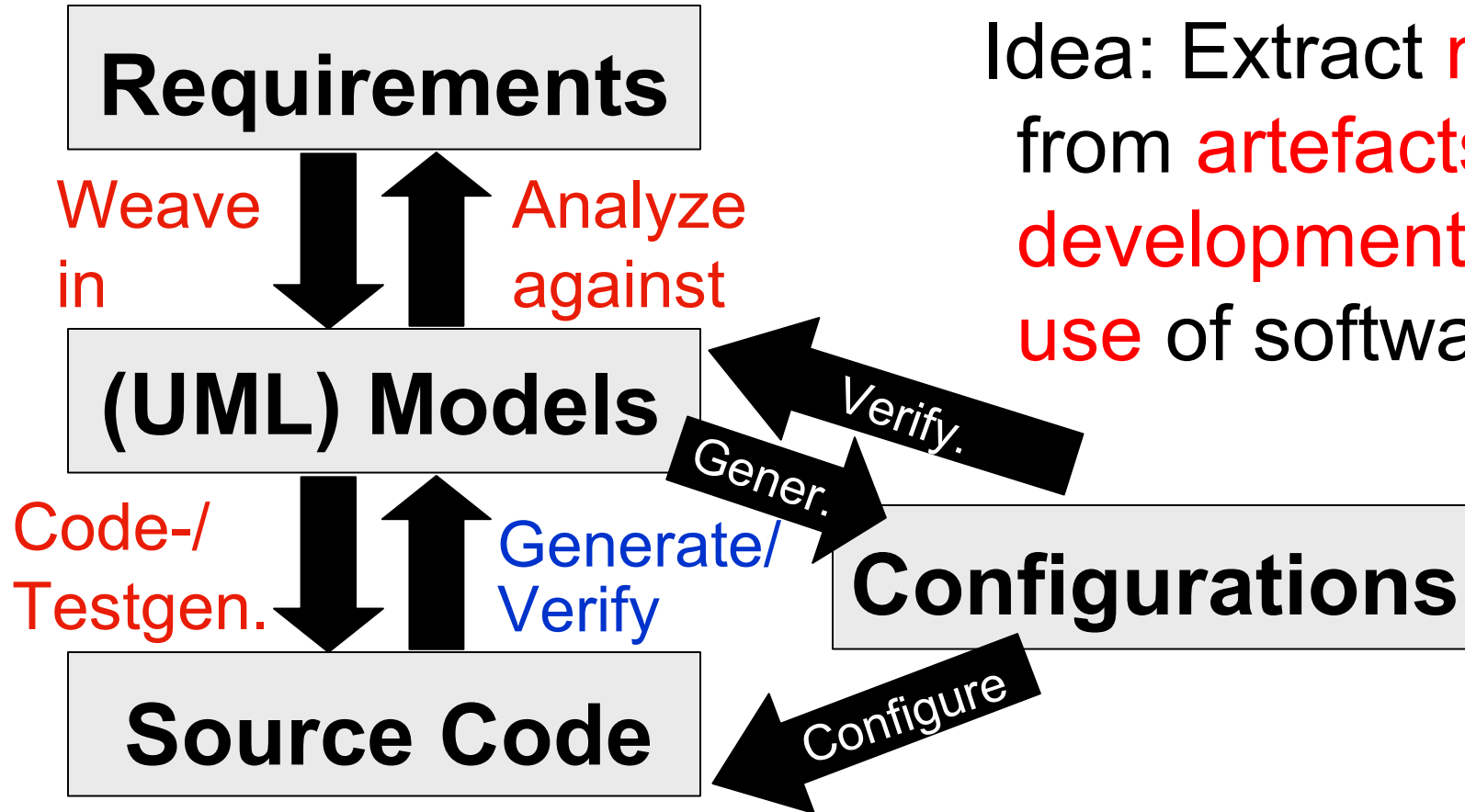


Goals

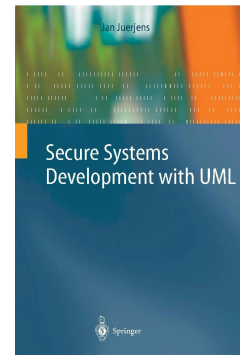
- Read configuration from business application
- Generate report of detected weaknesses
- Flexible configuration of report's data
- Easily configurable for different business applications / use-cases
- Check large-scale databases
- Checks based on freely configurable rules



Background: Model-based Security Engineering

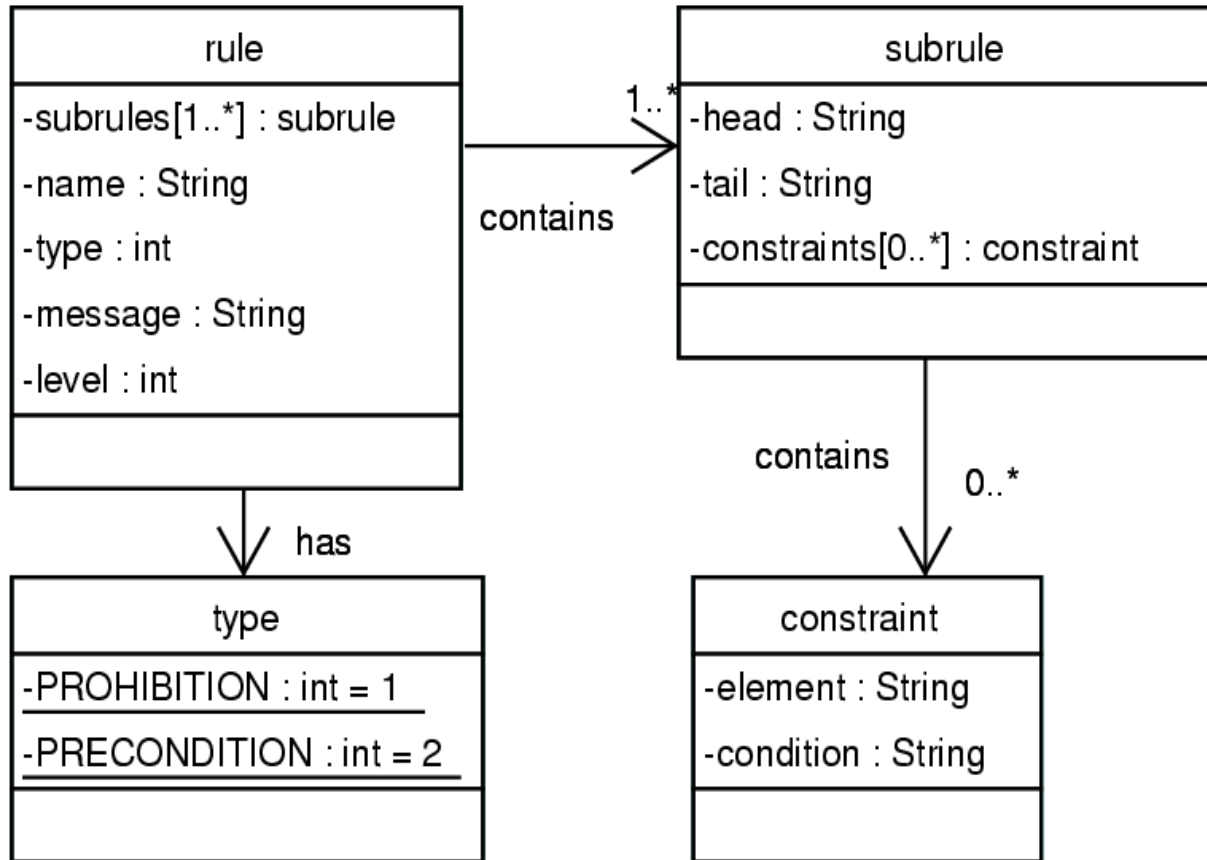


Idea: Extract **models** from **artefacts** in **development** and **use** of software.



→ Long-term goal: Tool-supported, theoretically sound, efficient automated security design & analysis.

Security Permission Rules



Rule elements

- Name
- Type
- Message
- Priority
- Sub-rules
 - Head
 - Target
 - Constraints

Example: a simple rule

Check whether user with permission `read_data_xyz` does not have role `writer`.

Rule:

- Name: `check_user_role_perm`
- Type: prohibited
- Message: User ... has role ...
- Priority: warning
- Sub-rule 1: Head: permission, Target: user
- Sub-rule 2: Head: user, Target: role,
Constraint: `role != role_writer`



Evaluating Rules

Rules evaluated using Prolog

- Every graph node atomic term
- Every rule gets associating term
- Prolog finds unification for set of parameters (A, B, C)

Example:

Graph node:

- `user(john, 501)`.

Associating Term:

- `user_role(A, B, C) :- user(A, B), role(B, C)`.



Example: Separation of duty

Karin: create credit (K1)

Susanne: grant credit (K2)

Stefan: assign vacation substitutes

Rule: no employee has rights K1 and K2
for same data object



Example: SAP Transactions

Transactions in SAP

implicitly grant access
to sub-transactions

„Transitive“ permissions
error-prone

Goal: Find users with
access to „dangerous“
transaction_xyz

Transactions part of
UML model

Rule:

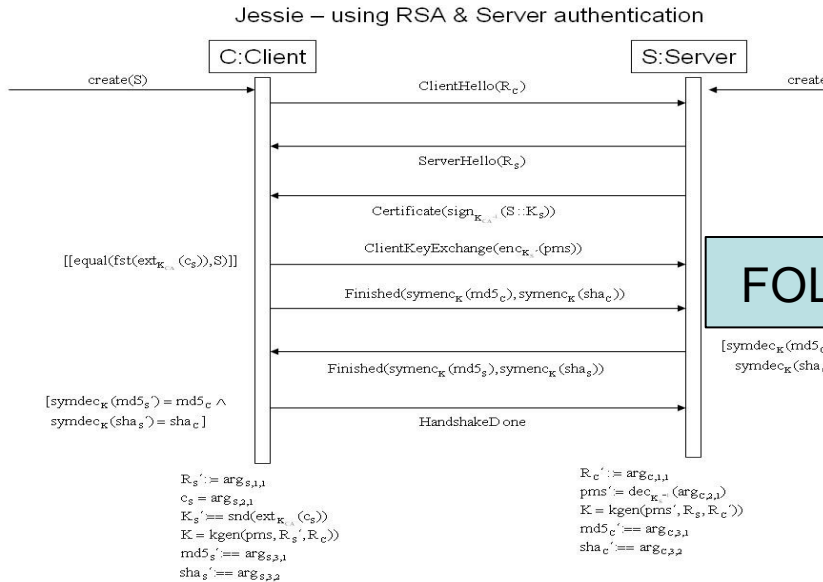
- Type: prohibited
- Message: user ... can
access ...

Subrule:

- Head:
transaction_xyz
- Target: user



Static Checking



FOL

```

...
((
  knows (ArgC_3)
  & (equal (fst (ArgC_3), type_serverkeyexchange))
  & (equal (snd (ext (snd (snd (ArgC_3))), k_ca), skey))
  & (equal (snd (ext (snd (ArgC_2), k_ca)), fst (snd (ArgC_3))))
))
=> (
  ((knows (ArgC_4_1)
    & equal (ArgC_4_1, type_serverhellodone))
    => (
      ( ( true & equal (ClientKeyExchange, enc (premasterkey, skey))
        )
      )
    )
  )
...
%----- Conjecture -----
input_formula (attack, conjecture, (
  knows (mastersecret) ) ).
  
```

ATP

```

analyzing results ...
model found/total failure
time limit information: 19 total / 18 strategy
(leaving wrapper).
task myUML_PID1491 on atbroyl has status SUCCESS
(model found by strategy 300) consuming 1 seconds
deleting temporary files.
e-SETHEO done. exiting
  
```



Reasoning

Theorem.

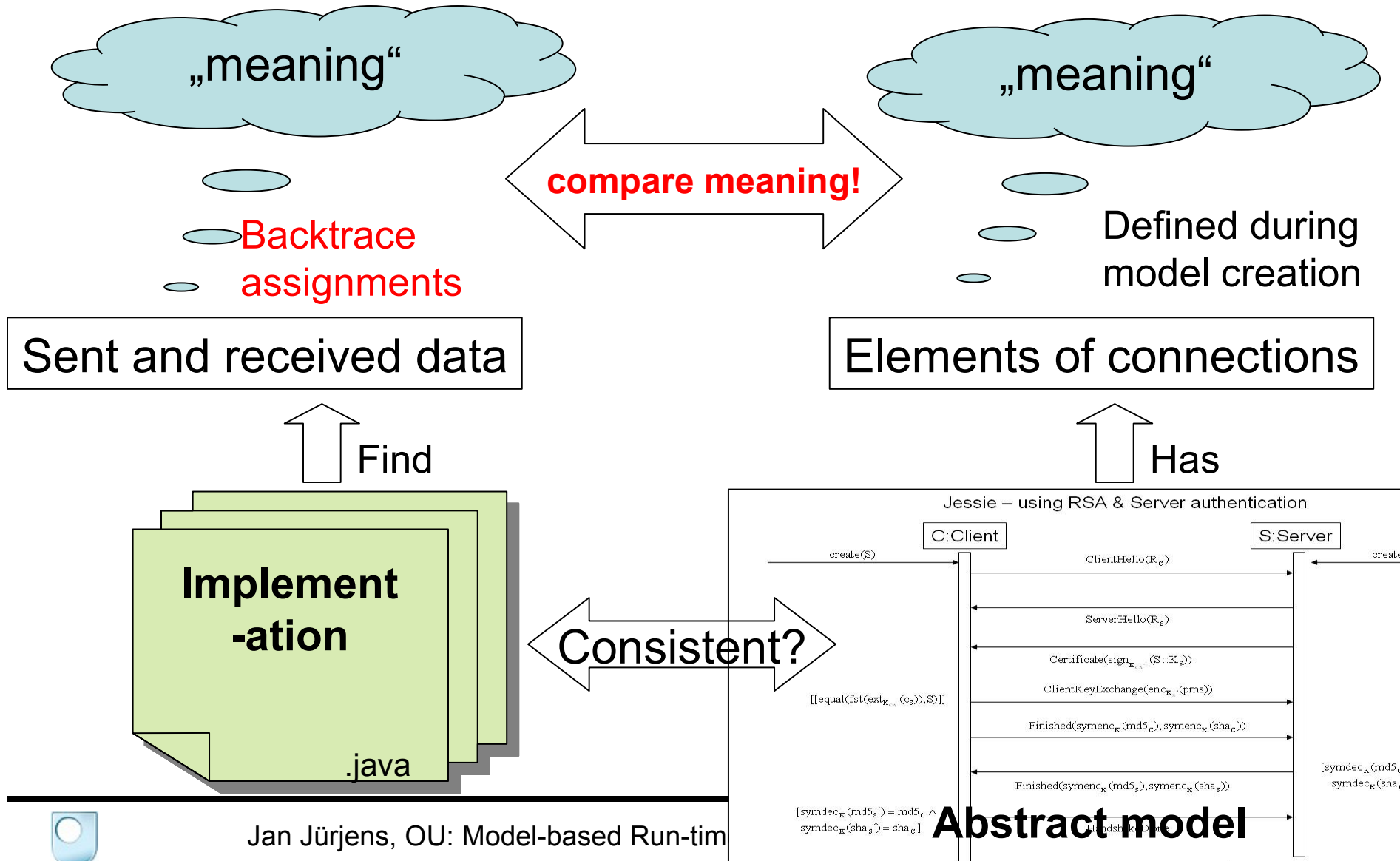
Suppose access to resource according to **Guard** object specifications granted only to objects signed with K .

Suppose all components keep secrecy of K .

Then **only** objects **signed** with K are granted **access**.



Static vs. Run-time Checking



Java Security Architecture

Originally (JDK 1.0): sandbox.

Too **simplistic** and **restrictive**.

JDK 1.2/1.3: more fine-grained security control, signing, sealing, guarding objects, . . .)

BUT: complex, thus use is **error-prone**.



Java Security policies

Permission entries consist of:

- protection domains (i. e. URL's and keys)
- target **resource** (e.g. files on local machine)
- corresponding **permissions** (e.g. read, write, execute)

Signed and Sealed Objects

Need to protect **integrity** of objects used as authentication tokens or transported across JVMs.

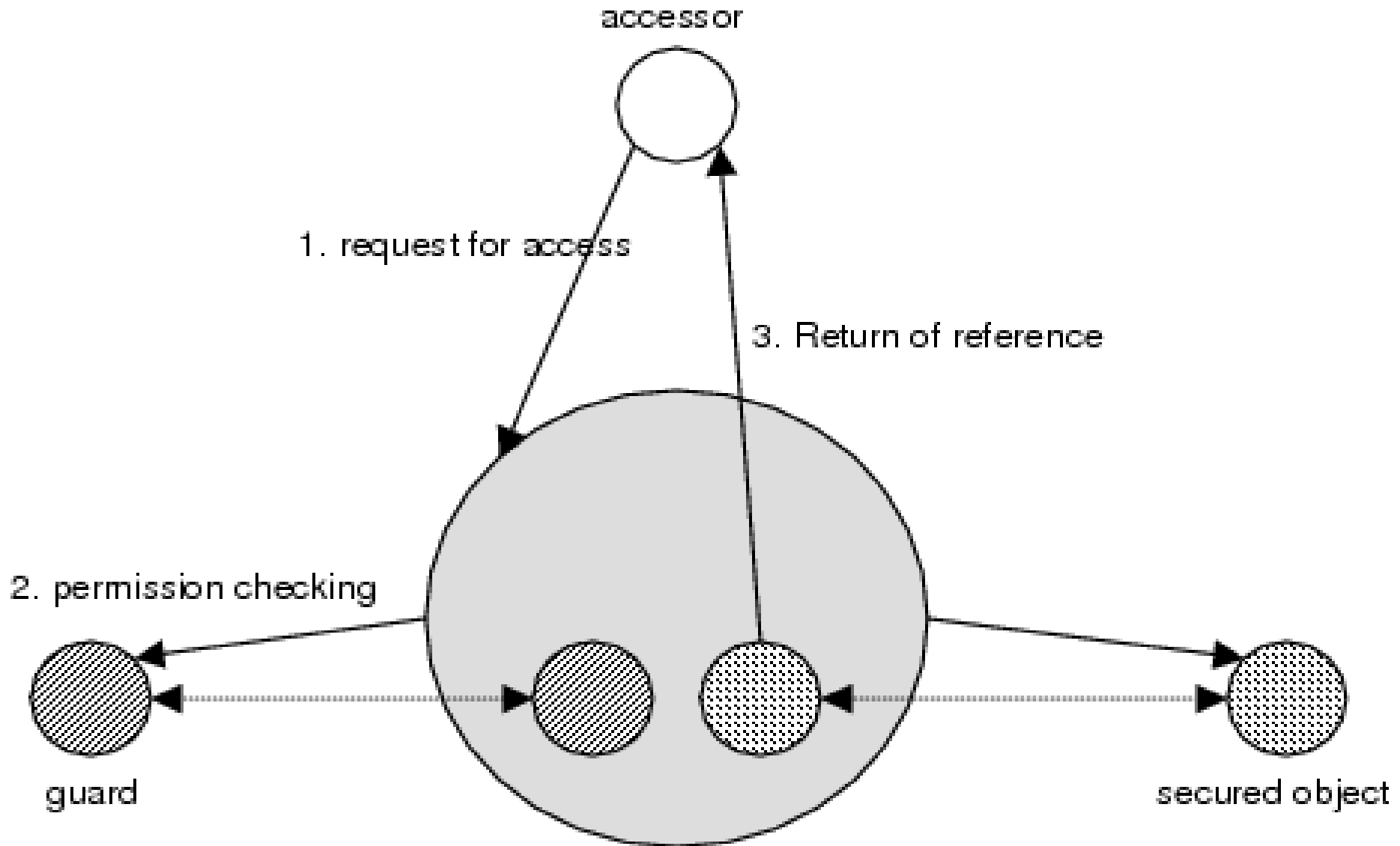
A **SignedObject** contains an object and its signature.

Similarly, need **confidentiality**.

A **SealedObject** is an encrypted object.



Guarded Objects



Problem: Complexity

- Granting of permission depends on **execution context**.
- Access control decisions may rely on **multiple threads**.
- A thread may involve several **protection domains**.
- Have method **doPrivileged()** **overriding** execution context.
- Guarded objects defer access control to **run-time**.
- **Authentication** in presence of adversaries can be subtle.
- **Indirect** granting of access with capabilities (keys).
 - **Difficult** to see which objects are granted permission.
 - ⇒ use **UMLsec**

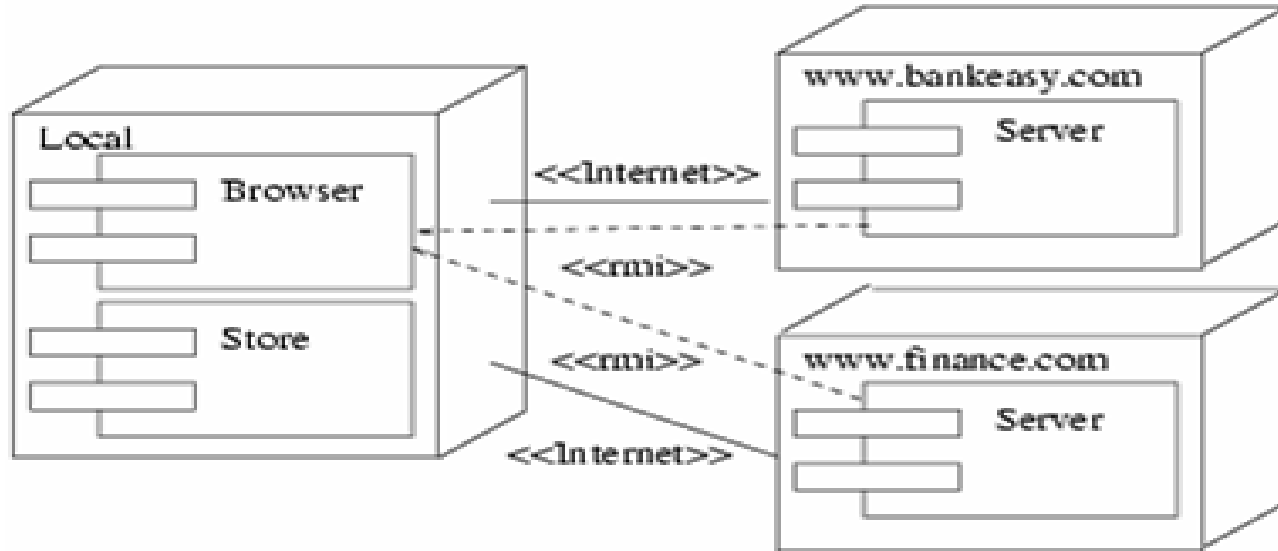


Checking Security Permissions

- (1) Formulate access control **requirements** for sensitive objects.
- (2) Give **guard objects** with appropriate access control checks.
- (3) Check that guard objects **protect** objects **sufficiently**.
- (4) Check that access control is consistent with **functionality**.
- (5) Check **mobile objects** are sufficiently protected.



Example: Financial Application



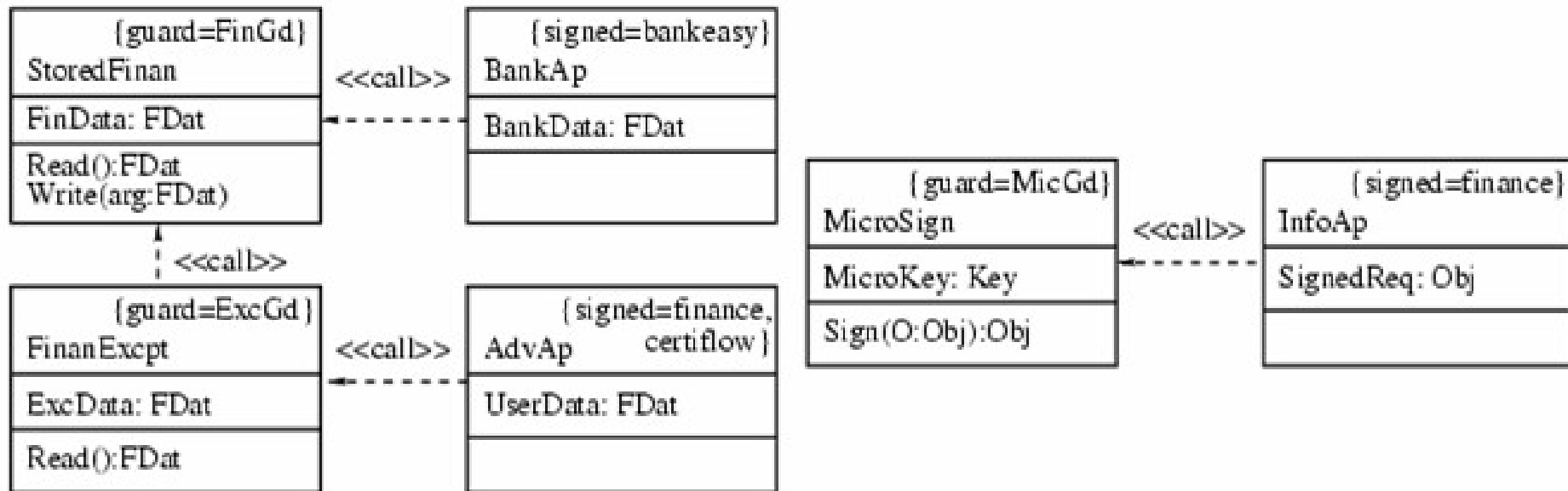
Internet bank, Bankeasy, and financial advisor, Finance, offer services to local user. Applets need certain Privileges (step1).

- Applets from and signed by bank **read** and **write** financial data between 1 pm and 2 pm.
- Applets from and signed by Finance **use** micropayment key five times a week.

Financial Application: Class Diagram

Sign and **seal** objects sent over Internet for Integrity and confidentiality.

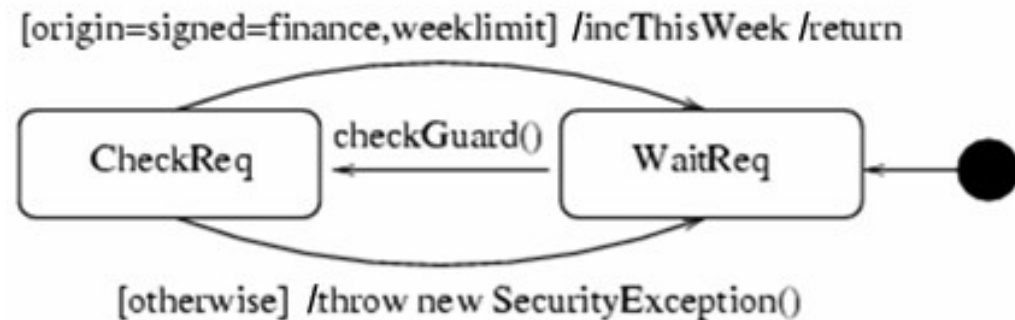
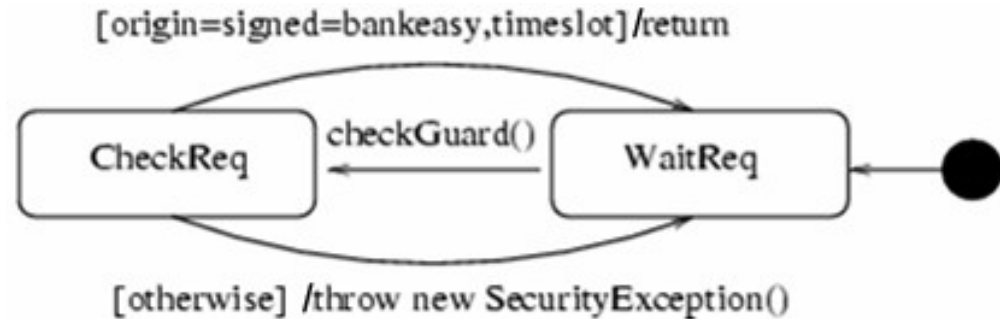
GuardedObjects control access.



Financial Application: Guard Objects (step 2)

timeslot true between 1pm and 2pm.

weeklimit true until access granted five times; **inc ThisWeek** increments counter.



Financial Application: Validation

Guard objects give **sufficient protection** (step 3).

Proposition. UML specification for guard objects only grants permissions implied by access permission requirements.

Access control consistent with **functionality** (step 4).
Includes:

Proposition. Suppose applet in current execution context originates from and signed by Finance. Use of micropayment key requested (and less than five times before). Then permission granted.

Mobile objects sufficiently protected (step 5), since objects sent over Internet are signed and sealed.

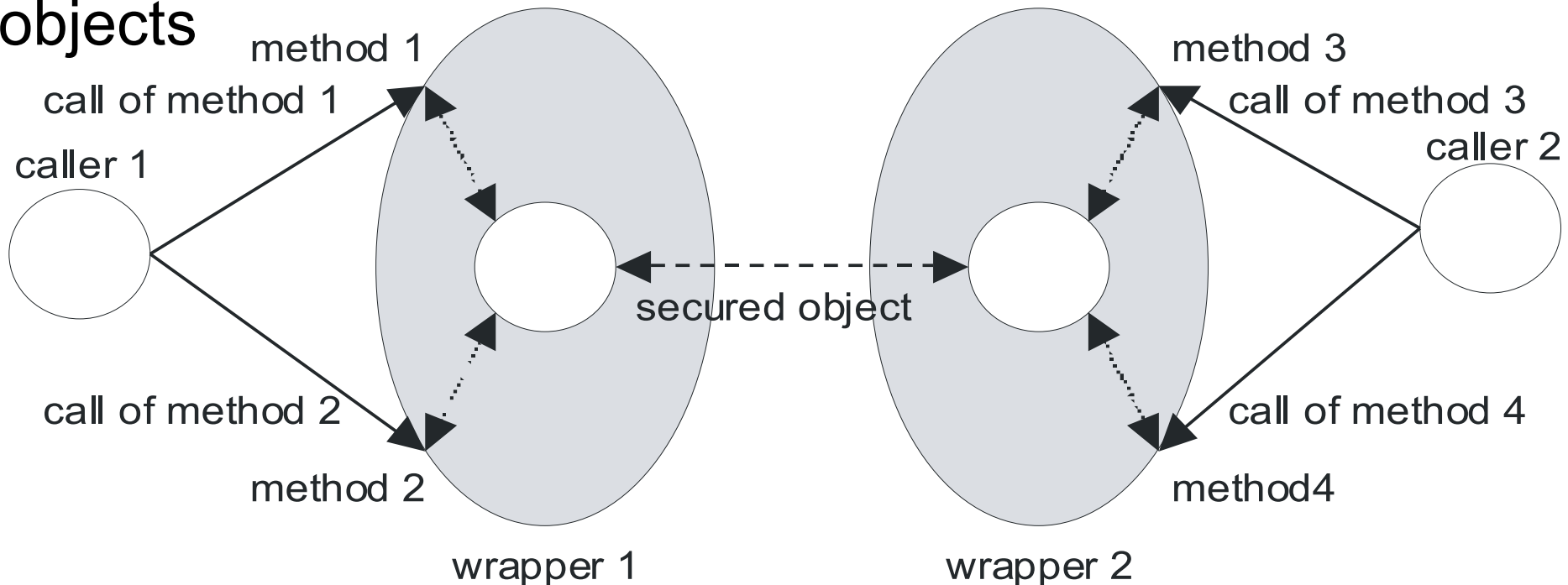


Delegation using Certificates

GuardedObject manages certificates

- Passed as parameter to getObject() method
- Secured by asymmetric key

Per-method permission check by returning wrapper objects



Some Applications

T-Systems

Analyzed designs / implementations / configurations e.g. for

- Biometry- or smart-card-based identification
- authentication (crypto protocols)
- authorization (user permissions, e.g. SAP systems)

Analyzed security policies, e.g. for privacy regulations.

Allianz 

Deutsche Bank 

HypoVereinsbank 

CEPS™

BMW Group



Related Projects

- PhD project on Verifying Implementations of Cryptoprotocols in C (MSR Cambridge / A. Gordon)
- RoySoc JIP with TU Munich on Formal Model-based Analysis of Cryptoprotocol Implementations
- RoySoc JIP with NII (Tokyo) on Security Requirements vs Design
- PhD project on IT security risk assessment with Munich Re
- PhD project on Adaptive Security for Ambient Technology
- PhD project on fuzzy reasoning for IT security risks
- PhD project on model-based development for avionics



Conclusion

Model-based approach for linking static and run-time checking of security permissions.

Here only initial steps; more work needed (collaboration welcome !).

Related: LTL based run-time verification for crypto protocols (with Andi Bauer, SESS08@ICSE).

Don't miss: presentation at FASE on Tuesday !



Questions ?

More information
(papers, slides, tool
etc.):

J.Jurjens@open.ac.uk