

# **Formal Methods and Verification of Protocols and Systems**



Dr. David von Oheimb  
Siemens Corporate Technology, Security  
<http://www.ct.siemens.com/>

Guest lecture on invitation by Prof. Sauerbrey at the  
EI department of TU Munich, Germany, 27 Jan 2012

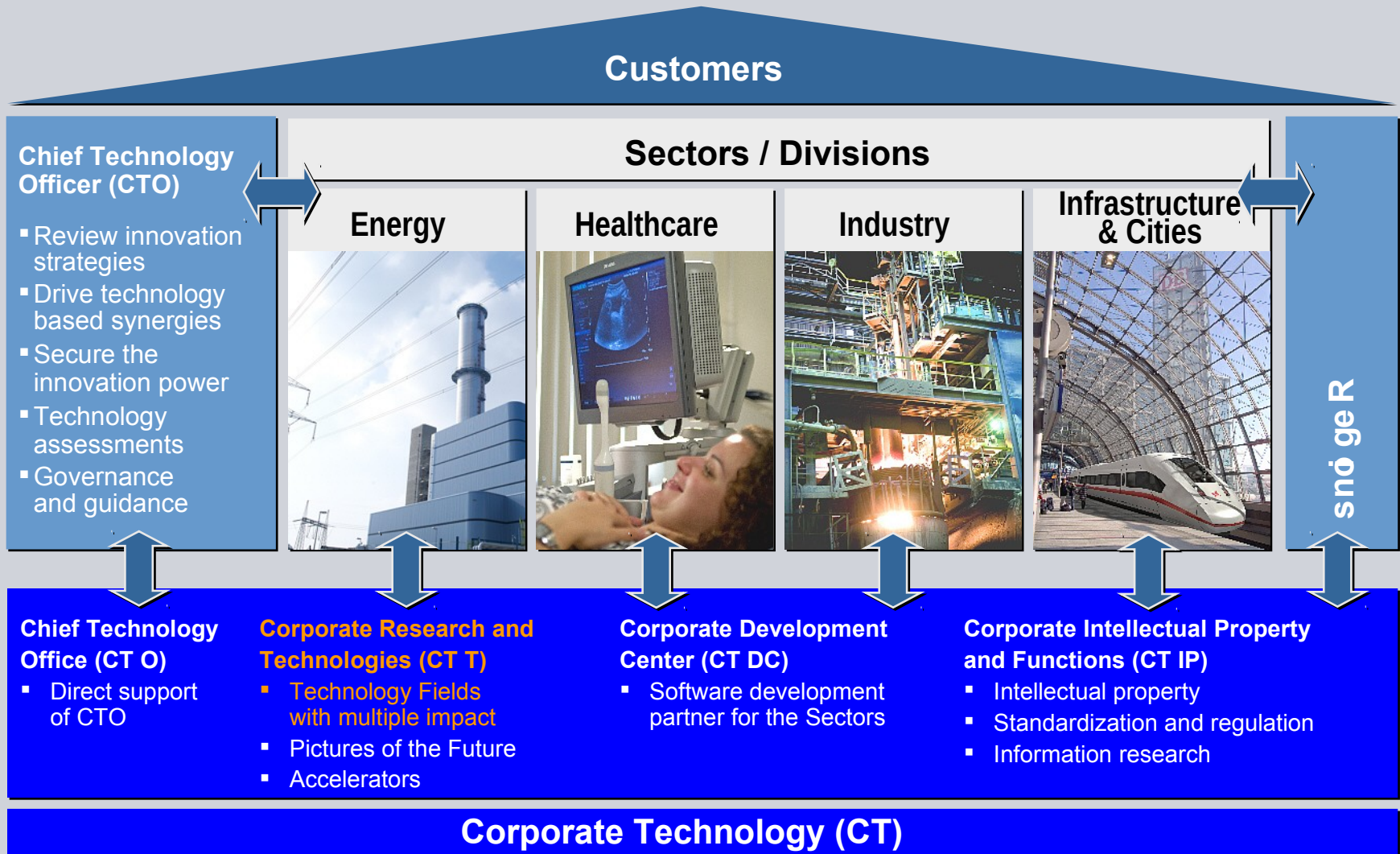
<http://www.sec.ei.tum.de/en/courses/selected-topics-in-system-security/>

## Overview

- **IT Security at Siemens Corporate Technology**
- Software distribution systems
- Common Criteria certification
- Formal security analysis
- Research project AVANTSSAR
- Needham-Schroeder protocol

# Siemens Corporate Technology (CT)

Networking the integrated technology company

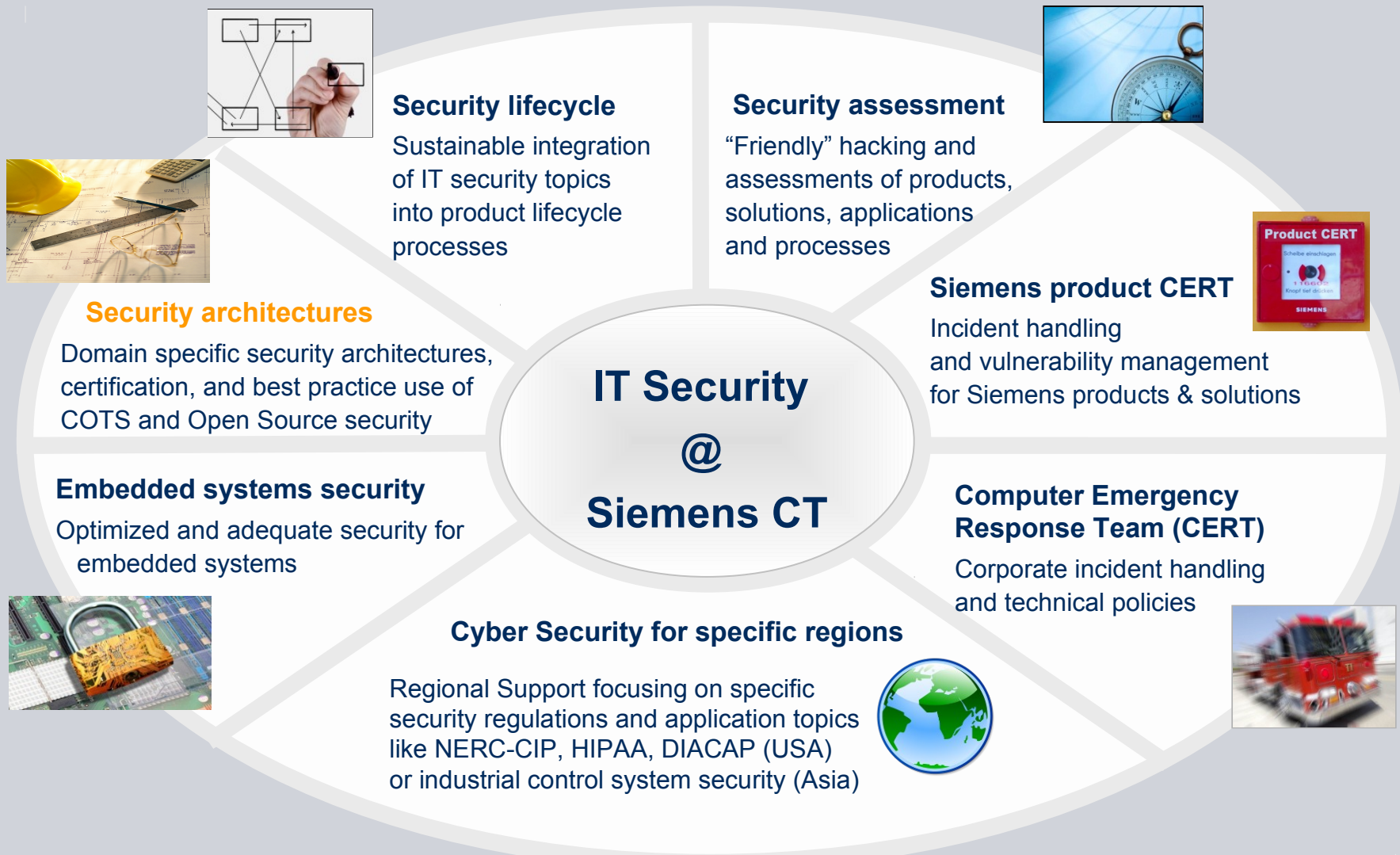


# Siemens corporate R&T: around 1,800 researchers Present in all leading markets and technology hot spots

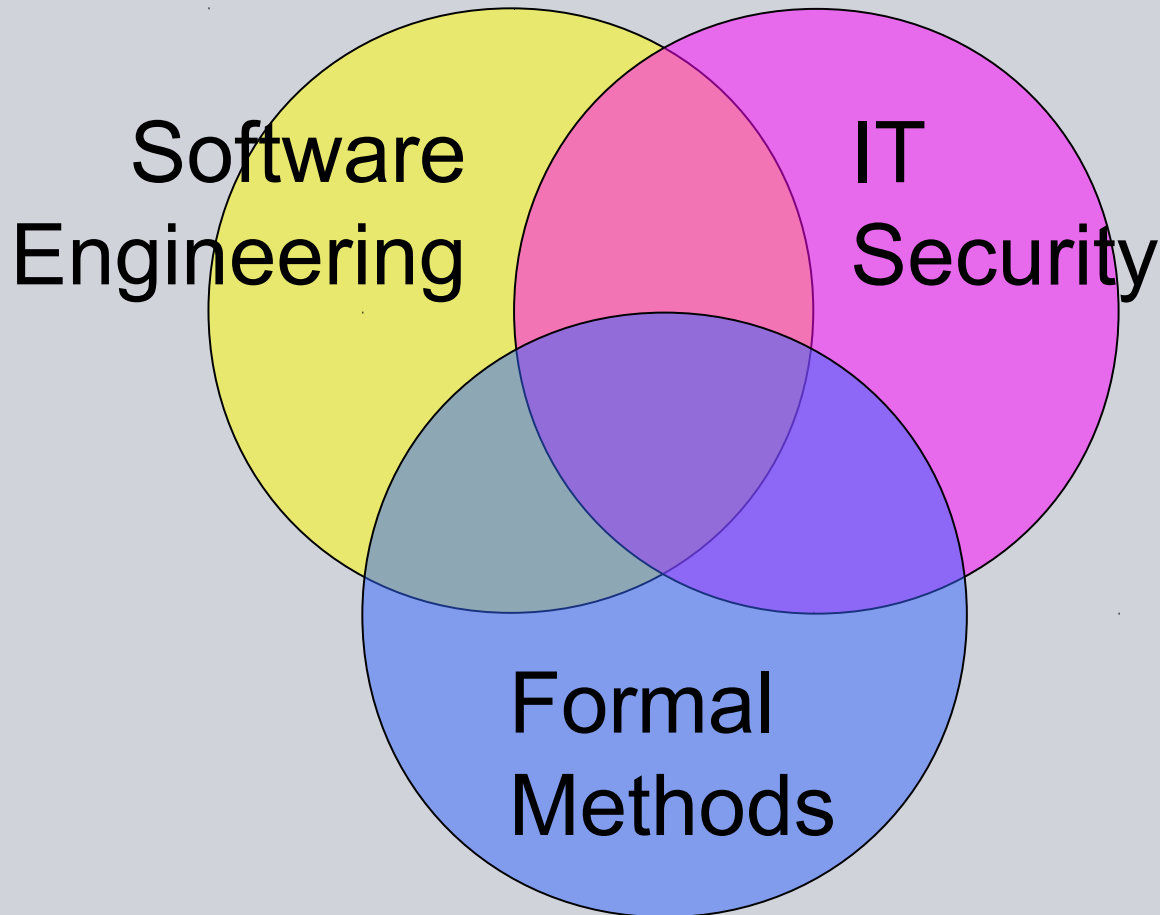
**SIEMENS**



## IT Security topics at Siemens Corporate Technology



## Fields

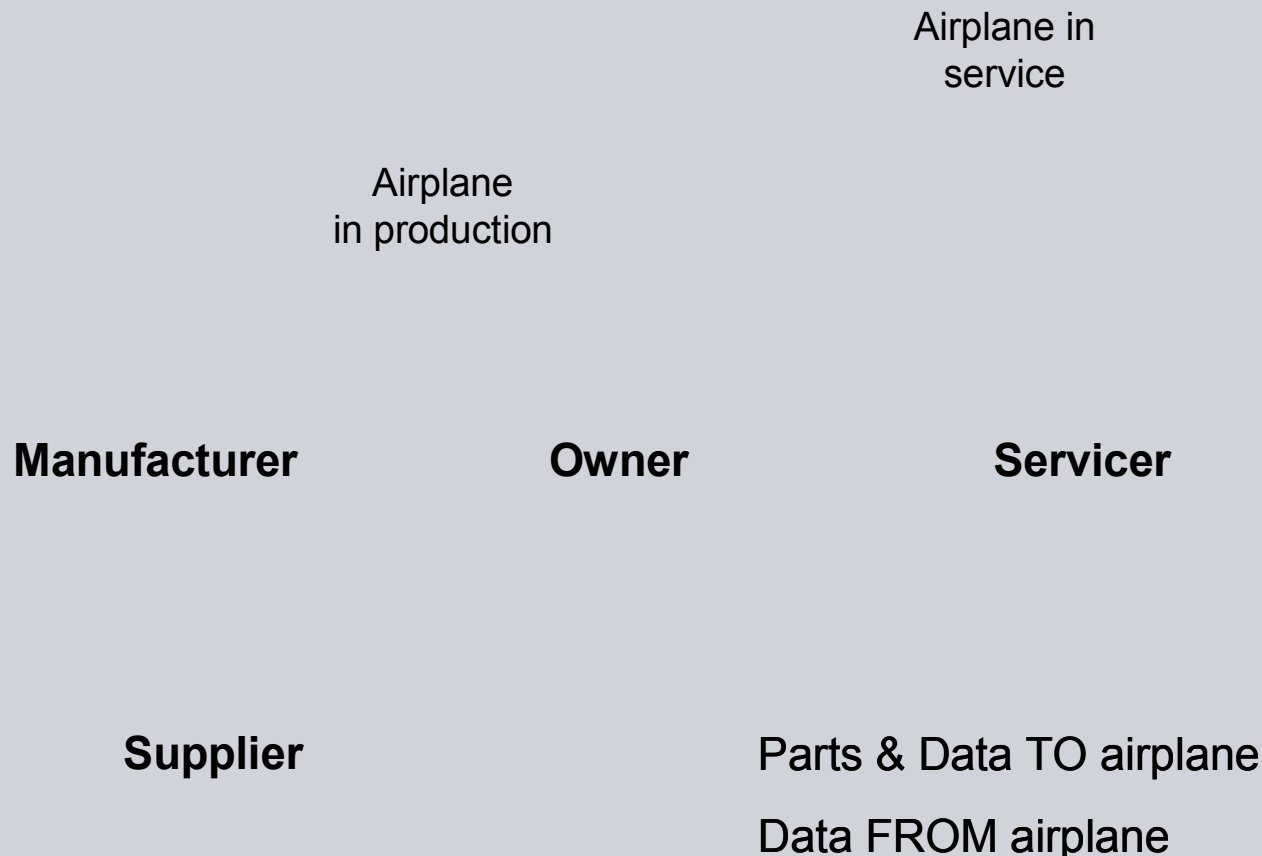


## Overview

- IT Security at Siemens Corporate Technology
- **Software distribution systems**
- Common Criteria certification
- Formal security analysis
- Research project AVANTSSAR
- Needham-Schroeder protocol

## Airplane Assets Distribution System (AADS)

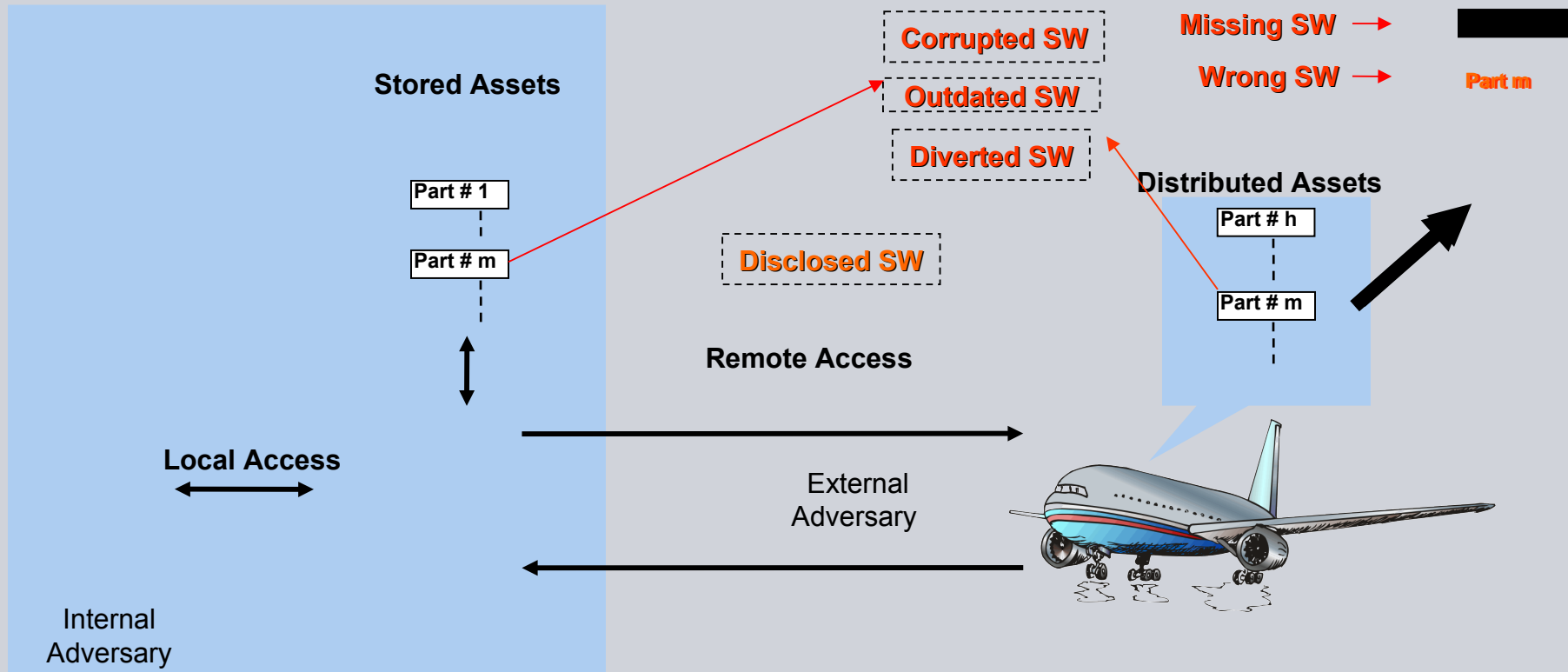
AADS is a system for storage and distribution of airplane assets, including *Loadable Software Airplane Parts (LSAP)* and airplane health data





# Safety-related security threats at the AADS example

**Attacker's objective:** lower airplane safety margins by tampering software that will be executed onboard an aircraft



Corruption/Injection

Wrong Version

Diversion

Disclosure

## IT Security as a System Engineering Problem

- **IT security** aims at preventing, or at least detecting, unauthorized actions by agents in an IT system.

In the AADS context, security is a prerequisite of safety.

- **Safety** aims at the absence of accidents (→ airworthiness)

**Situation:** security loopholes in IT systems **actively exploited**

**Objective:** **thwart attacks** by eliminating vulnerabilities

**Difficulty:** IT systems are very complex. Security is interwoven with the whole system, so **very hard to assess**.

**Remedy:** evaluate system following the **Common Criteria** approach

- address security **systematically in all development phases**
- perform document & code reviews and tests
- for maximal assurance, use **formal modeling and analysis**

## Overview

- IT Security at Siemens Corporate Technology
- Software distribution systems
- **Common Criteria certification**
- Formal security analysis
- Research project AVANTSSAR
- Needham-Schroeder protocol

## Common Criteria (CC) for IT security evaluation



product-oriented methodology  
for IT security assessment

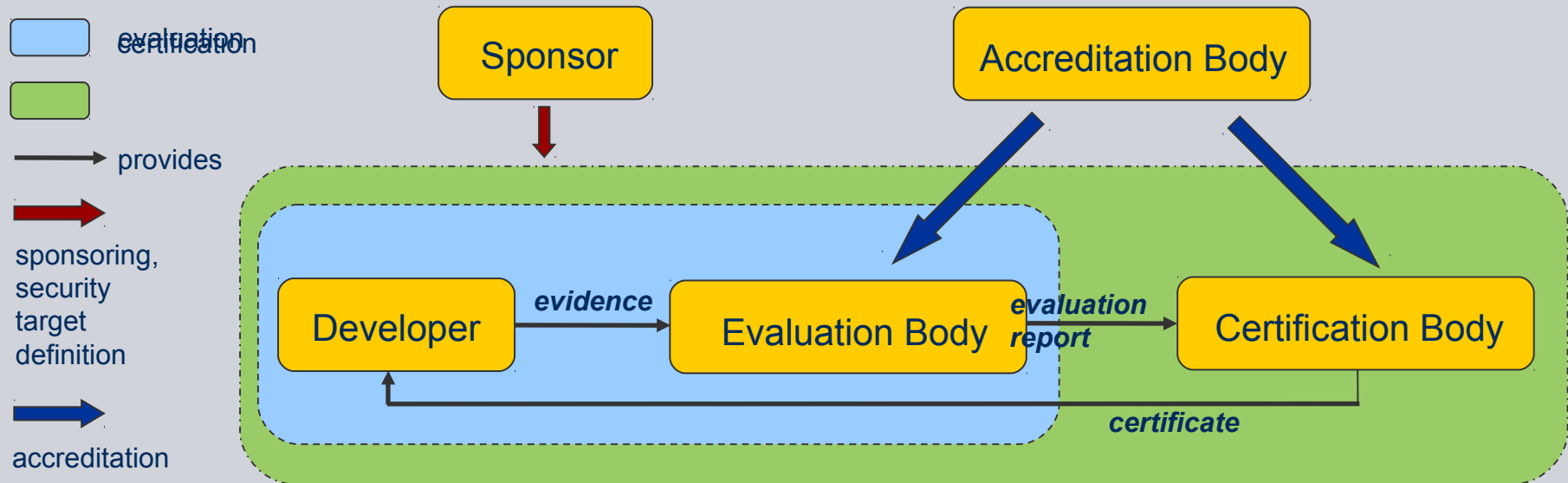
**ISO/IEC standard 15408**

Current version: 3.1R3 of Jul 2009

**Aim:** gain **confidence** in the security of a system

- What are the **objectives** the system should achieve?
- Are the **measures** employed **appropriate** to achieve them?
- Are the measures **implemented and deployed correctly**?

## Common Criteria process overview



Certification according to the Common Criteria is a rather **complex**, **time consuming** and **expensive** process, providing **systematic assurance**.

A successful, approved evaluation is awarded a **certificate**.

Lifetime of certificates is **theoretically not bounded**, but their applicability is **limited by technical progress** (→ re-certification).

## CC: Security Targets

**Security Target (ST):** defines extent and depth of the evaluation  
for a specific product called *Target of Evaluation (TOE)*

**Protection Profile (PP):** defines extent and depth of the evaluation  
for a whole class of products, i.e. firewalls

STs and PPs may inherit (*claim*) other PPs.

ST and PP specifications use **generic** “construction kit”:

- Building blocks for defining *Security Functional Requirements (SFRs)*
- Scalable in depth and rigor: *Security Assurance Requirements (SARs)*  
layered as *Evaluation Assurance Levels (EALs)*

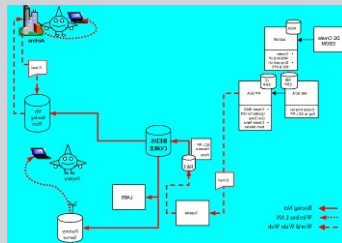
## Overview

- IT Security at Siemens Corporate Technology
- Software distribution systems
- Common Criteria certification
- **Formal security analysis**
- Research project AVANTSSAR
- Needham-Schroeder protocol

## Formal Security Analysis: Approach and Benefits

Mission: security analysis with **maximal precision**

Approach: **formal modeling and verification**



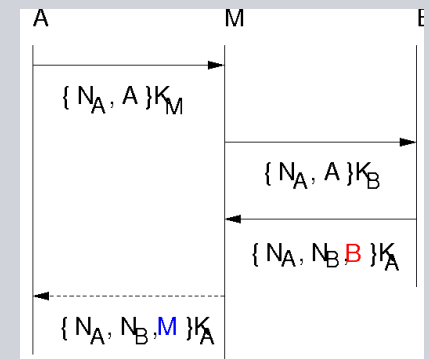
Spec.

Impl.



Abstraction

Interpretation



Improving the **quality of the system specification**

Checking for the existence of **security loopholes**

High-level protocol/system specification lang.  
Model checkers (e.g., **AVANTSSAR tools**)

HOL, Interacting State Machines, etc.  
Interactive theorem provers (e.g., **Isabelle**)

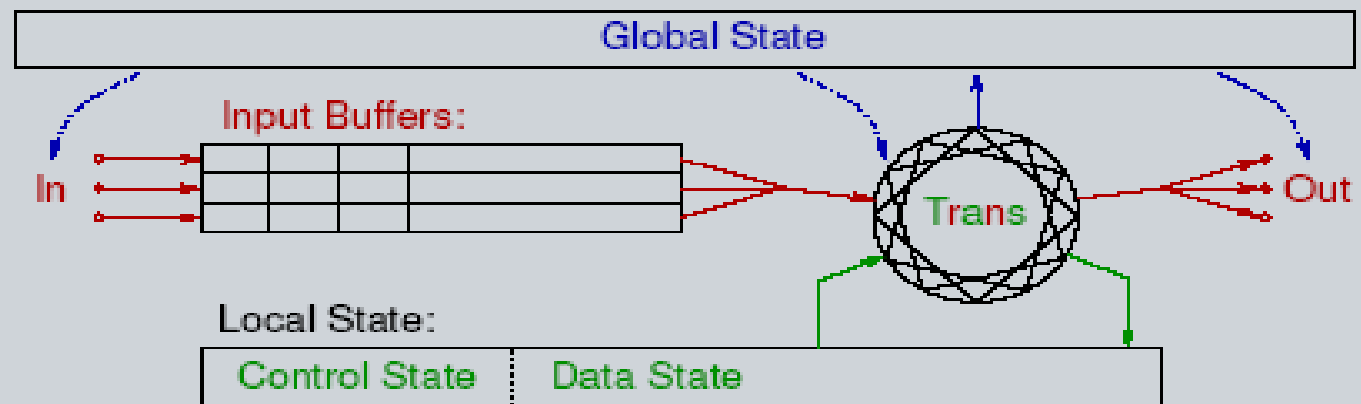


## Formal Security Models

- ▶ A **security policy** defines **what is allowed** (actions, data flow, ...) typically by a relationship between **subjects** and **objects**.
- ▶ A **security model** is a **(+/- formal) description** of a policy and enforcing mechanisms, usually in terms of system **states** or state sequences (**traces**).
- ▶ **Security verification** proves that **mechanisms enforce policy**.
- ▶ Models focus on **specific characteristics** of the reality (policies).
- ▶ Types of formal security models
  - ▶ **Automata** models
  - ▶ **Access Control** models
  - ▶ **Information Flow** models
  - ▶ **Cryptoprotocol** models

## Interacting State Machines (ISMs)

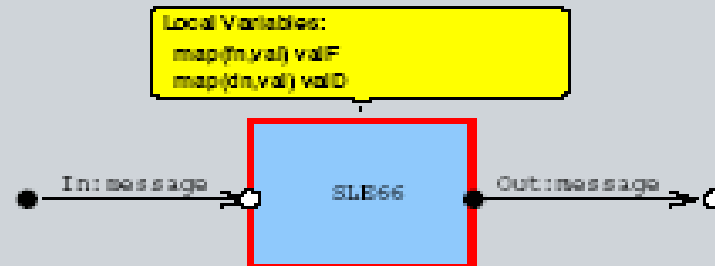
Automata with (nondeterministic) **state transitions** + **buffered I/O, simultaneously** on multiple connections.



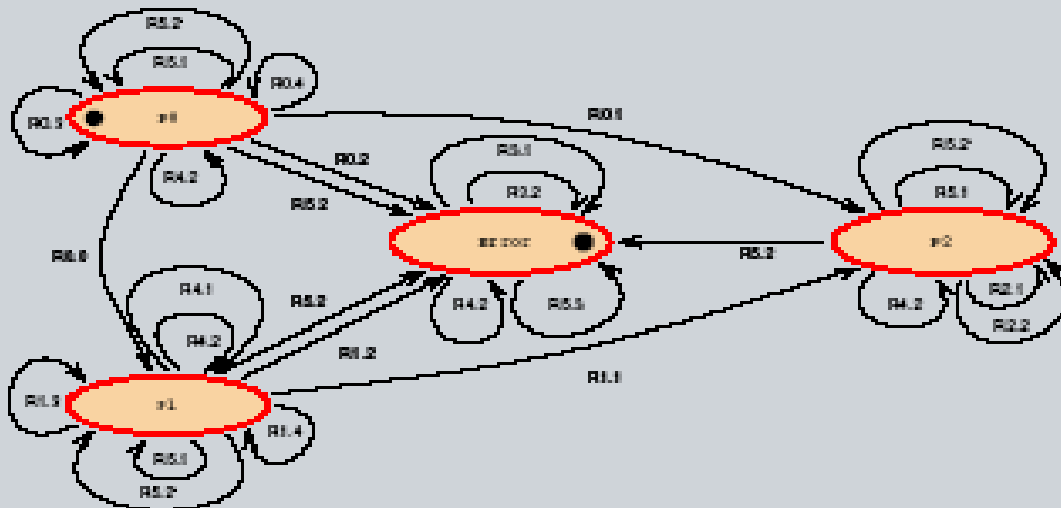
Transitions definable in executable and/or axiomatic style.  
 An ISM system may have changing **global state**.  
 Applicable to a large **variety of reactive systems**.  
*By now, not much verification support (theory, tools).*

## Formal model of Infineon SLE 66 Smart Card Processor

System Structure Diagram:



State Transition Diagram (abstracted):



First higher-level (EAL5) certification for a smart card processor!

# Formal RBAC model of Complex Information System

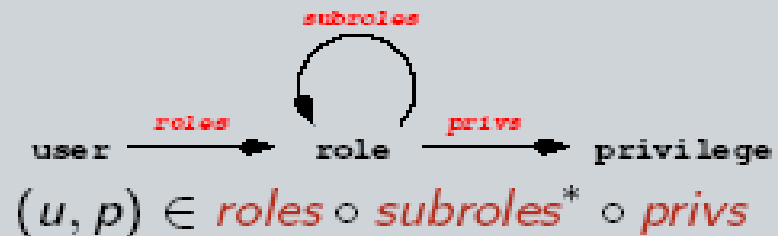
Is the security design (with emergency access etc.) sound?

Privileges:

$$roles \subseteq user \times role$$

$$subroles \subseteq role \times role$$

$$privs \subseteq role \times privilege$$



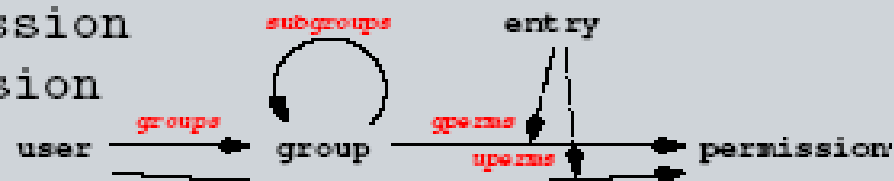
Permissions:

$$groups \subseteq user \times group$$

$$subgroups \subseteq group \times group$$

$$gperms \subseteq group \times permission$$

$$uperms \subseteq user \times permission$$



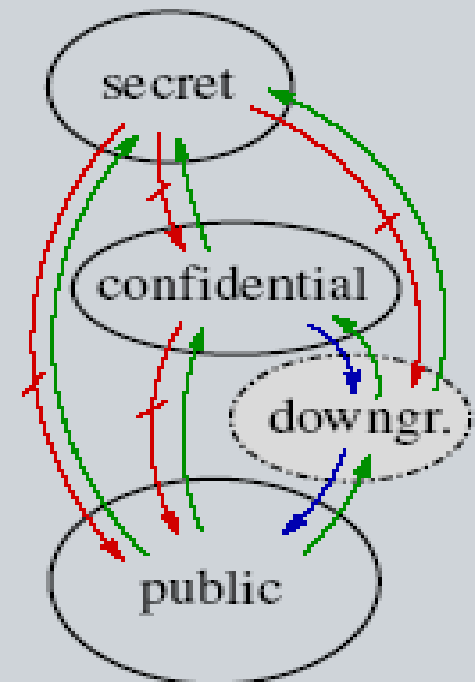
$$(u, p) \in (groups \circ subgroups^* \circ gperms(e)) \cup uperms(e)$$

“nagging questions”  $\rightsquigarrow$  clarifications improving specification quality.

Open issue: relation between model and implementation ( $\rightsquigarrow$  testing).

## Information Flow Models

- ▶ Identify knowledge/information domains
  - ▶ Specify **allowed flow** between domains
  - ▶ Check the **observations** that can be made about state and/or actions
  - ▶ Consider also **indirect and partial flow**
- 
- ▶ Classical model:  
Noninterference (Goguen & Meseguer)
  - ▶ Many variants:  
Non-deducability, Restrictiveness, Non-leakage, ...



**Very strong**, but rarely used in practice

*Available:* connection with ISMs

## Language-based Information Flow Security

**Policy:** no assignments of **high**-values  
to low-variables, enforced by type system

**Semantically:** take  $(x, y)$  as elements of the **state space**  
with high-level data (**on left**) and low-level data (on right).

**Step function**  $S(x, y) = (S_H(x, y), S_L(x, y))$

does not leak information from high to low

if  $S_L(x_1, y) = S_L(x_2, y)$  (functional **independence**).

**Observational equivalence**  $(x, y) \stackrel{L}{\sim} (x', y') \iff y = y'$

allows re-formulation:

$$s \stackrel{L}{\sim} t \longrightarrow S(s) \stackrel{L}{\sim} S(t) \quad (\text{preservation of } \stackrel{L}{\sim})$$

**Generalization** to action sequences  $\alpha$  and arbitrary policies  $\rightsquigarrow$

# Cryptoprotocol models

- ▶ Describe **message exchange** between processes or principals



- ▶ Take **cryptographic operations** as **perfect** primitives
- ▶ Describe system with specialized modeling languages
- ▶ State **secrecy, authentication, ...** goals
- ▶ Verify (mostly) **automatically** using model-checkers

EU project **AVISPA** , **AVANTSSAR**

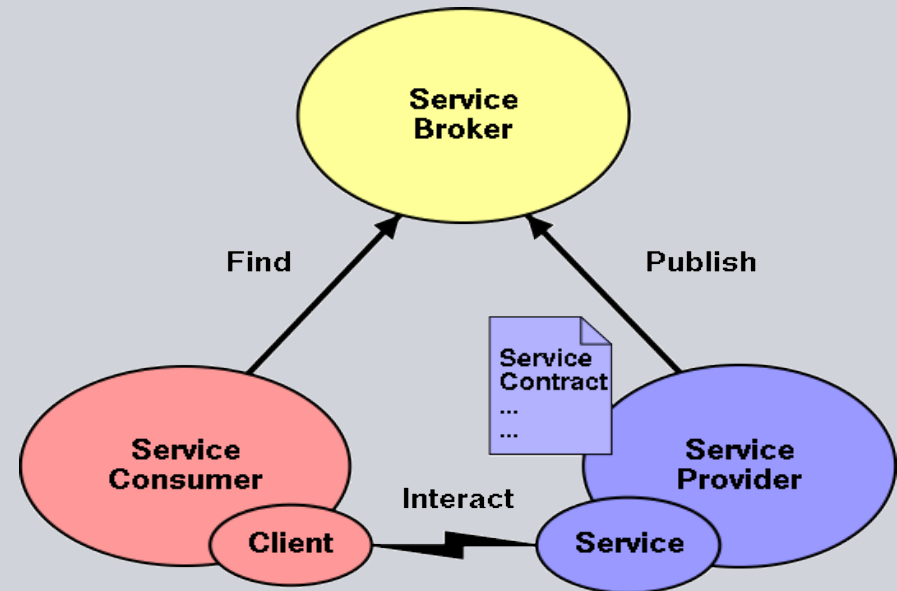
## Overview

- IT Security at Siemens Corporate Technology
- Software distribution systems
- Common Criteria certification
- Formal security analysis
- **Research project AVANTSSAR**
- Needham-Schroeder protocol



avantssar.eu

## Model-checking SOA security — research project AVANTSSAR<sup>1</sup>



<sup>1</sup>Automated Validation of Trust and Security of Service-oriented Architectures

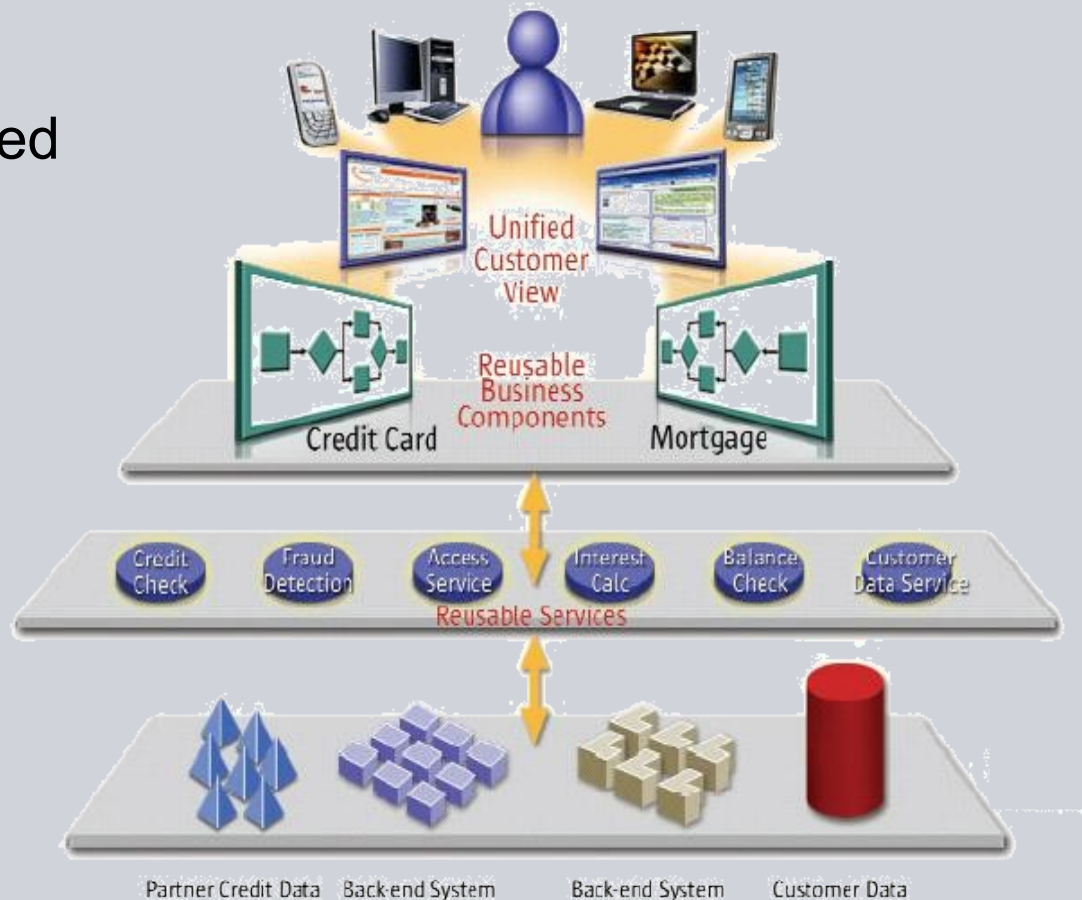
**FP7-2007-ICT-1, ICT-1.1.4, STREP project no. 216471  
Jan 2008 - Dec 2010, 590 PMs, 6M€ budget, 3.8M€ EC contribution**

# AVANTSSAR project motivation

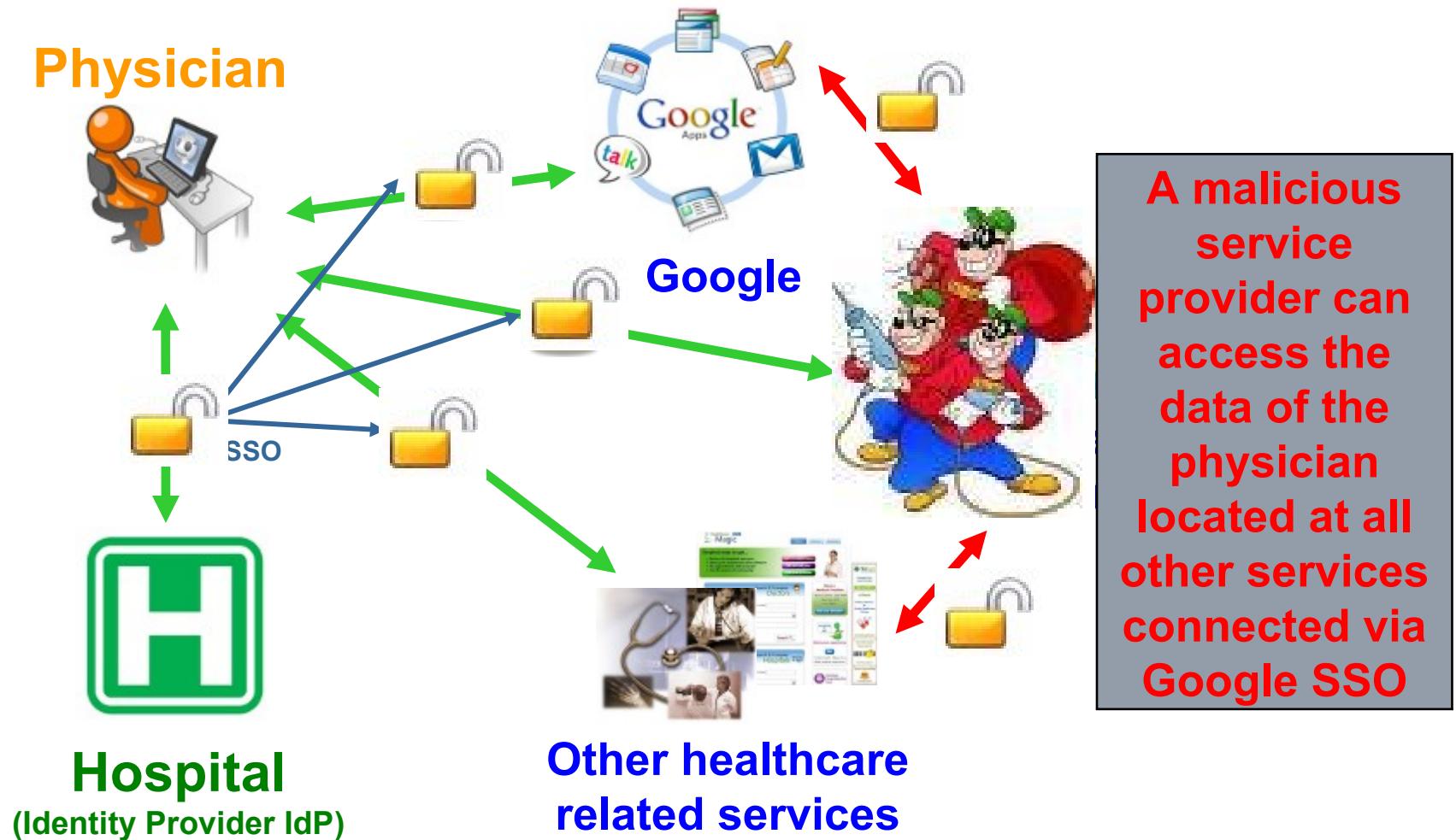
ICT paradigm shift: from components to **services**, composed and reconfigured dynamically in a demand-driven way.

Trustworthy service may **interact** with others causing novel trust and security problems.

For the composition of individual services into service-oriented architectures, **validation** is dramatically needed.



## Example 1: Google SAML-based Single Sign-On (SSO)



# Example 1: Google SAML SSO protocol flaw

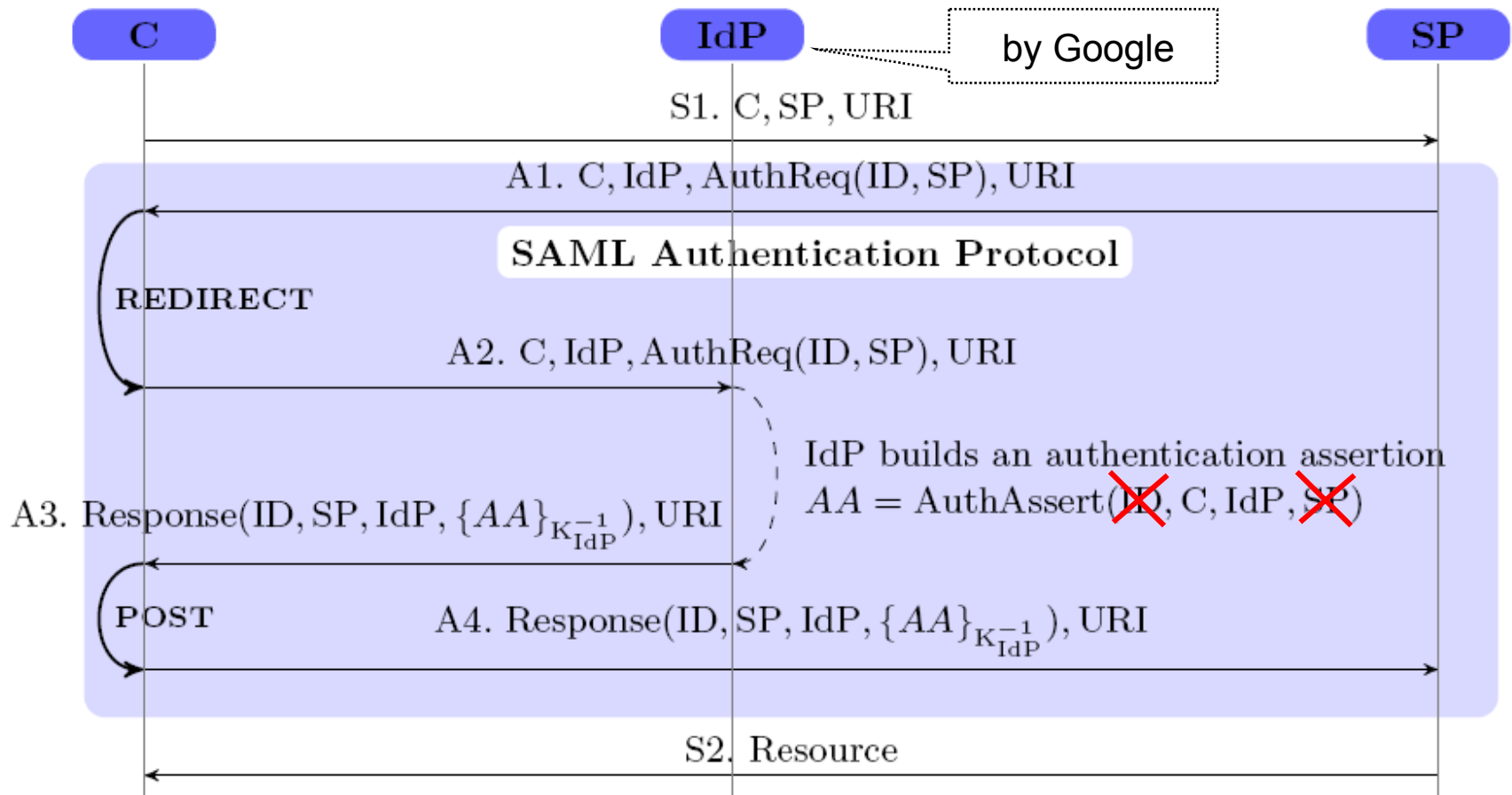


Fig. 1. SP-Initiated SSO with Redirect/POST Bindings

# AVANTSSAR consortium

## Industry

*SAP Research France, Sophia Antipolis*

*Siemens Corporate Technology, München*

IBM Zürich Research Labs (initial two years)

OpenTrust, Paris

## Academia

Università di Verona

*Università di Genova*

*ETH Zürich*

*INRIA Lorraine*

UPS-IRIT, Toulouse

IEAT, Timișoara

## Expertise

Service-oriented enterprise architectures

Security solutions

Standardization and industry migration

Security engineering

Formal methods

Automated security validation

# AVANTSSAR main objectives and principles

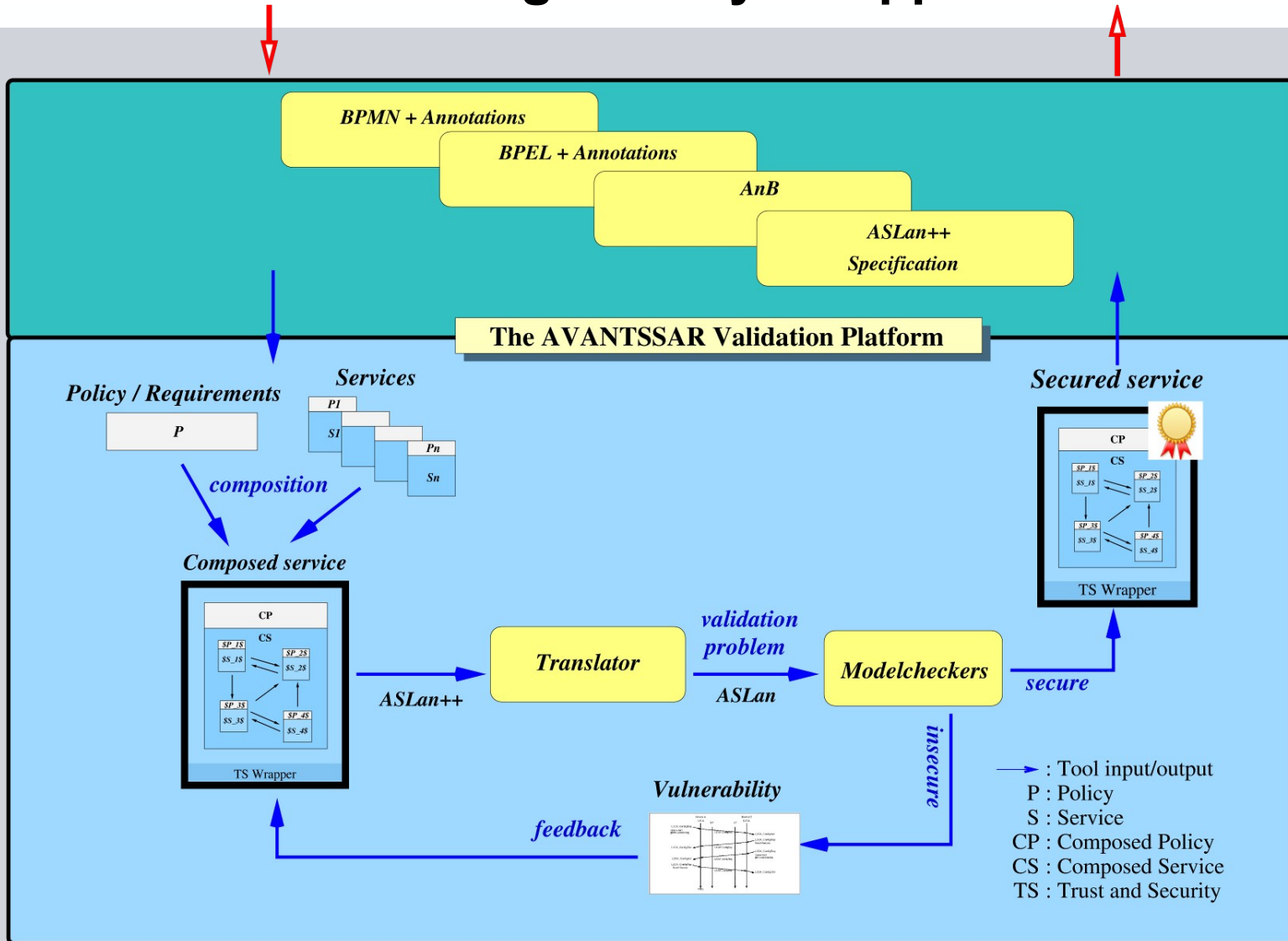
## AVANTSSAR product: Platform for formal specification and automated validation of trust and security of SOAs

- **Formal language** for specifying trust and security properties of services, their policies, and their composition into service-oriented architectures
- **Automated toolset** supporting the above
- **Library** of validated industry-relevant case studies

## Migration of platform to industry and standardization organizations

- **Speed up development** of new service infrastructures
- **Enhance** their **security** and robustness
- **Increase public acceptance** of SOA-based systems

## AVANTSSAR modeling & analysis approach with ASLan++



## Example 2: Process Task Delegation (PTD)

### Authorization and trust management via token passing

There are three roles in the protocol (**C**, **A**, **TS**) and potentially several instances for each role

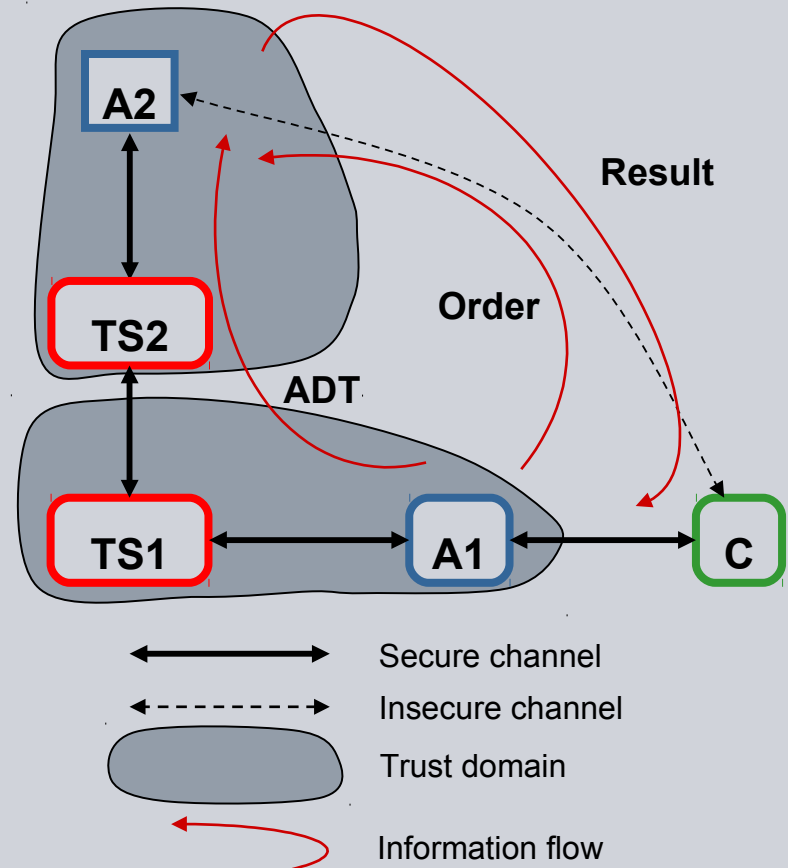
The *client C* (or *user*) uses the system for authorization and trust management, e.g. SSO

Each *application A* is in one domain, each domain has exactly one active *trust server TS*

**A1** uses the system to pass to **A2** some **Order** and an **ADT (Authorization Decision Token)**

- **Order** contains:
  - workflow task information
  - application data
  - information about the client **C** and his current activity to be delivered securely (integrity and confidentiality)
- **ADT** is mainly authorization *attributes* and *decisions*
  - sent via **TS1** and **TS2**, who may weaken it
  - must remain unaltered, apart from weakening by **TS**
  - must remain confidential among intended parties

**C**, **A1**, and **A2** must be authenticated among each other

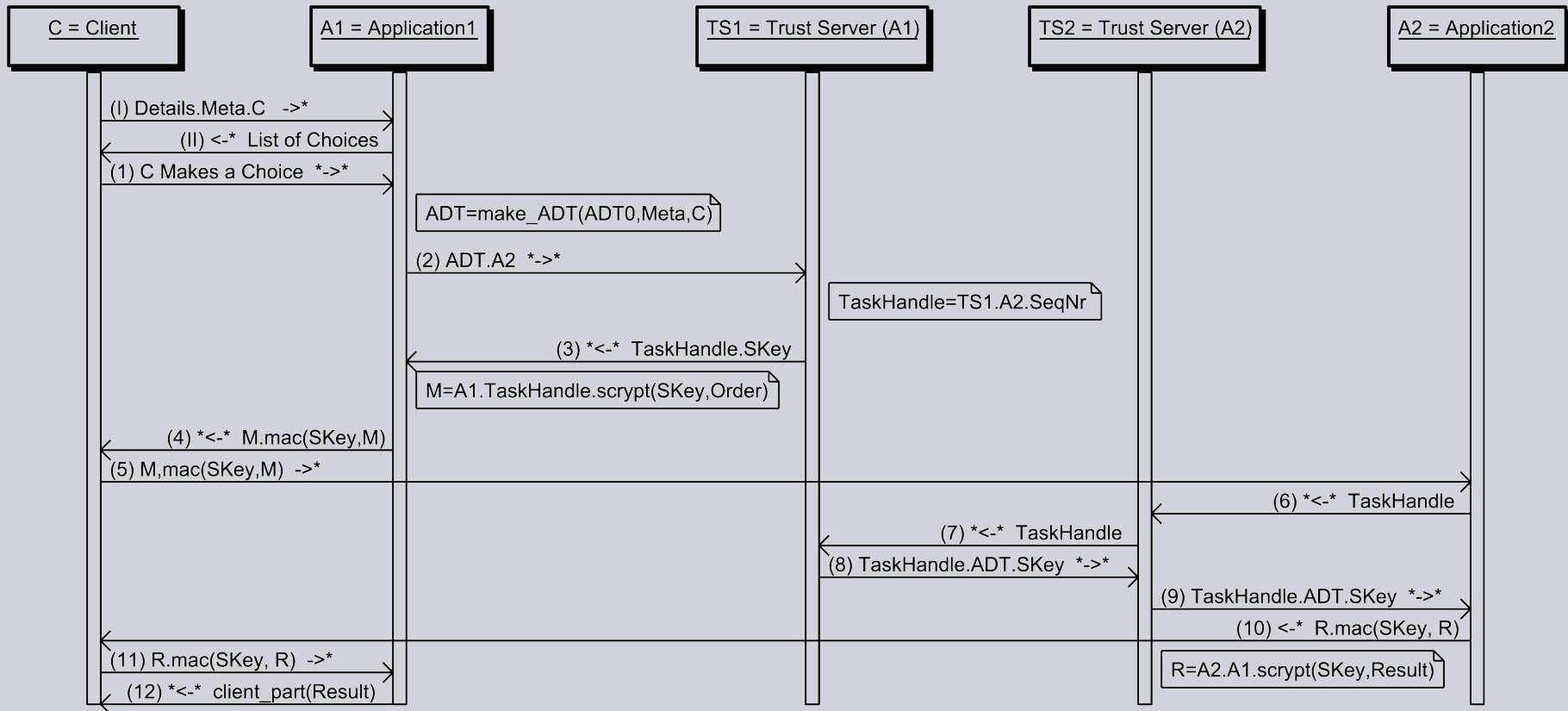


### Security prerequisites:

PKI used for **A** and **TS**, username & passwd for **C**  
 The **TS** enforce a strict time-out



# Example 2: Message Sequence Chart of PTD



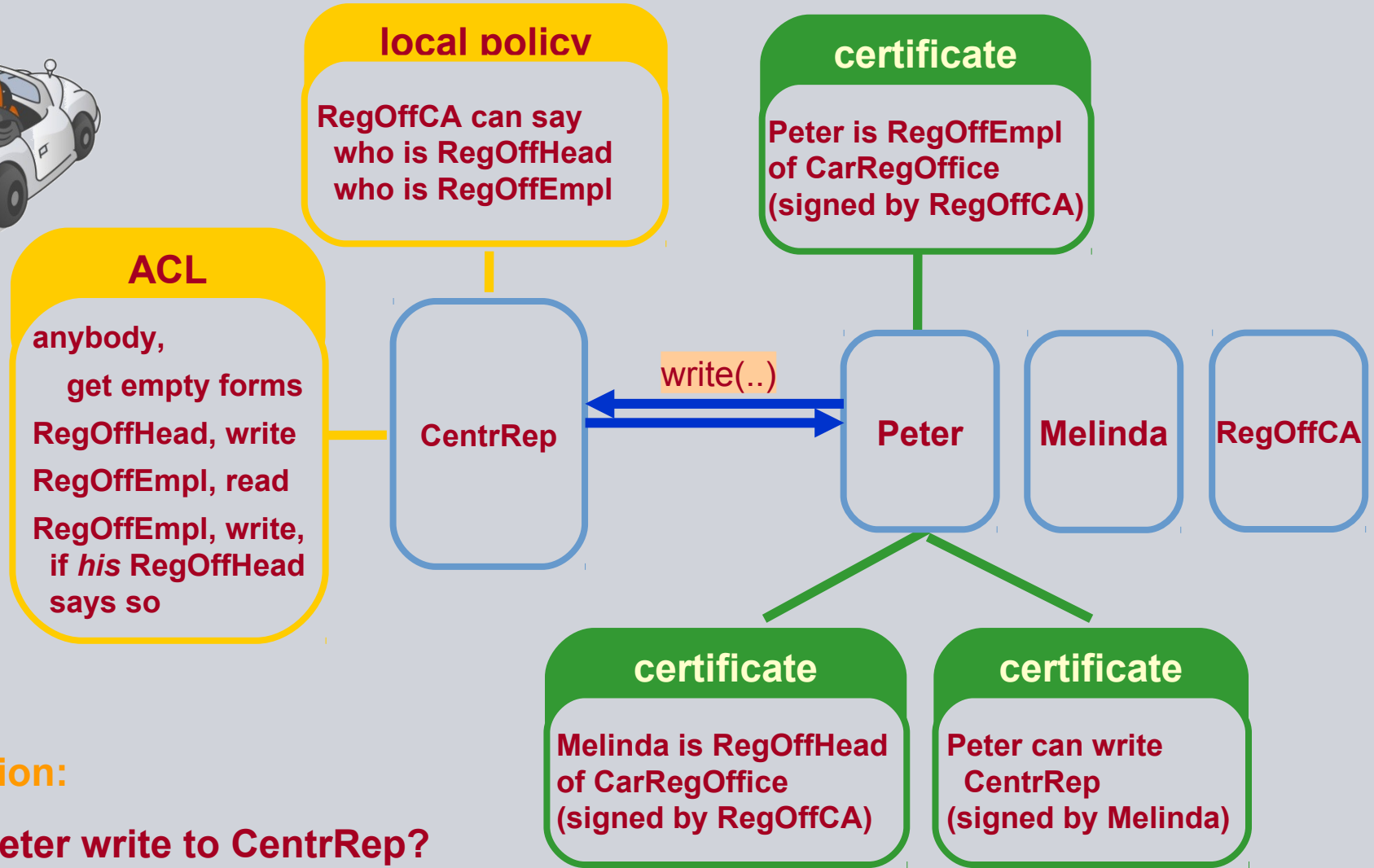
## Example 2: ASLan++ model of PTD Application A2

```

entity A2 (Actor: agent, TS2: agent) { % Application 2, connected with Trust Server 2
symbols
  C0,C,A1: agent;
  CryptedOrder, Order, Details, Results, TaskHandle, ADT, MAC: message;
  SKey: symmetric_key;
body { while (true) {
  select {
    % A2 receives (via some C0) a package from some A1. This package includes encrypted and
    % hashed information. A2 needs the corresponding key and the Authorization Decision Token.
    on (?C0 -> Actor: (?A1.Actor.?TaskHandle.?CryptedOrder).?MAC): {
      % A2 contacts its own ticket server (TS2) and requests the secret key SKey and the ADT.
      Actor *->* TS2: TaskHandle;
    }
    % A2 receives from A1 the SKey and checks if the decrypted data corresponds to the hashed data
    on (TS2 *->* Actor: (?ADT.?SKey).TaskHandle & CryptedOrder = script(SKey,?,?Details.?C)
      & MAC = hash(SKey, A1.Actor.TaskHandle.CryptedOrder)): {
      % A2 does the task requested by A1, then sends to A1 via C the results encrypted with the secret key.
      Results := fresh(); % in general, the result depends on Details etc.
      Actor -> C: Actor.C.A1. script(SKey,Results);
    }
  }
}
goals
  authentic_C_A2_Details: C *-> Actor: Details;
  secret_Order: secret (Order, {Actor, A1});
}

```

# Example 3: Electronic Car Registration policies



Question:

May Peter write to CentrRep?

## Example 3: On-the-fly inferences via Horn clauses

**DKAL-style trust inference**, e.g. trust application:

```
trustapp(P,Q,Anything) :
  P->knows(Anything) :-
    P->trusts(Q,Anything) &
    P->knows(Q->said(Anything));
```

**Basic facts**, e.g. the central repository fully trusts the CA

```
centrRepTrustCA(Anything) :
  centrRep->trusts(theCA,Anything);
```

**State-dependent (evolving) facts**, e.g. department head manages a set of trusted employees:

```
trustedEmplsCanStoreDoc(Head): forall Empl.
  Head->knows(Empl->canStoreDoc) :-
    contains(TrustedEmpls, Empl);
```

**Use of certificates**, e.g. the central repository trusts the department head on employee's rights:

```
centrRepTrustHead(Head, Empl) :
  centrRep->trusts(Head, Empl->canStoreDoc) :-
    centrRep->knows(theCA->said(Head->hasRole(head))) &
    centrRep->knows(theCA->said(Empl->hasRole(employee)));
```

# Overview

- IT Security at Siemens Corporate Technology
- Software distribution systems
- Common Criteria certification
- Formal security analysis
- Research project AVANTSSAR
- **Needham-Schroeder protocol**

## Example: Needham-Schroeder Public Key Protocol

[Needham-Schroeder 1978]

[http://en.wikipedia.org/wiki/Needham-Schroeder\\_protocol](http://en.wikipedia.org/wiki/Needham-Schroeder_protocol)

Simplified version without key server, assuming that A and B already know the public key of their peers:

$$A \rightarrow B: \{Na . A\}_{pk(B)}$$
$$B \rightarrow A: \{Na . Nb\}_{pk(A)}$$
$$A \rightarrow B: \{Nb\}_{pk(B)}$$

Goal: strong mutual authentication

## Example: ASLan++ model NSPK\_Cert (1): Alice & Bob

```

specification NSPK_Cert
...
entity Alice (Actor, B: agent) {
  symbols
    Na, Nb: message;
  body {
    if(trusted_pk(B)) {
      Na := fresh();
      Actor -> B: {secret_Na:(Na).Actor}_pk(B);
      B -> Actor: {Alice_strong_auth_Bob_on_Na:(Na).secret_Nb:(?Nb)}_pk(Actor);
      Actor -> B: {Bob_strong_auth_Alice_on_Nb:(Nb)}_pk(B); } }
  }
entity Bob (Actor: agent) {
  symbols
    A: agent;
    Na, Nb: message;
  body {
    ?A -> Actor: {secret_Na:(?Na).?A}_pk(Actor); % Bob learns A here!
    if (trusted_pk(A)) {
      Nb := fresh();
      Actor -> A: {Alice_strong_auth_Bob_on_Na:(Na).secret_Nb:(Nb)}_pk(A);
      A -> Actor: {Bob_strong_auth_Alice_on_Nb:(Nb)}_pk(Actor); } }
  } ...
}

```

## Example: ASLan++ model NSPK\_Cert (2): certificates

```
specification NSPK_Cert channel_model CCM
entity Environment {
```

### symbols

```
trusted_pk(agent): fact;
trusted_agent(agent): fact;
root_ca, ca: agent;
issued(message): fact;
```

### macros

```
A->signed(M) = {M}_inv(pk(A)).M;
C->cert(A,PK) = C->signed(C.A.PK); % no validity period etc.
```

### clauses

```
trusted_pk_direct(C):
  trusted_pk(C) :-
  trusted_agent(C);

trusted_pk_cert_chain(A,B):
  trusted_pk(A) :-
  trusted_pk(B) & issued(B->cert(A,pk(A)));
```



## Example: ASLan++ model NSPK\_Cert (3): sessions

```

entity Session (A, B: agent) {
  entity Alice (Actor, B: agent) {...}
  entity Bob (Actor: agent) {...}
  body {
    issued(ca->cert(A,pk(A)));
    issued(ca->cert(B,pk(B)));
    new Alice(A,B);
    new Bob(B);
  }
  goals
    secret_Na: {A,B};
    secret_Nb: {A,B};
    Alice_strong_auth_Bob_on_Na: B *->> A;
    Bob_strong_auth_Alice_on_Nb: A *->> B;
}
body { % need two sessions for Lowe's attack
  trusted_agent(root_ca);
  issued(root_ca->cert(ca,pk(ca))); % root-signed CA certificate
  issued(      ca->cert(i ,pk(i ))); % CA-signed intruder cert
  any A B. Session(A,B) where A!=B;
  any A B. Session(A,B) where A!=B; } }

```

## Example: Lowe's attack on NSPK

### [Lowe 1995] Man-in-the-middle attack

1.1 A -  $\{Na.A\}_{pk(i)} \rightarrow i$

2.1  $i(A)$  -  $\{Na.A\}_{pk(B)} \rightarrow B$

2.2  $i(A) \leftarrow \{Na.Nb\}_{pk(A)} - B$

1.2 A  $\leftarrow \{Na.Nb\}_{pk(A)} - i$

1.3 A -  $\{Nb\}_{pk(i)} \rightarrow i$

2.3  $i(A)$  -  $\{Nb\}_{pk(B)} \rightarrow B$

In the first session, Alice talks with some party, e.g. Chuck, who in fact is an intruder.

In the second session, Bob thinks that he was contacted by Alice but **actually talks to the intruder**.

Therefore, also his nonce **Nb gets leaked** to the intruder.