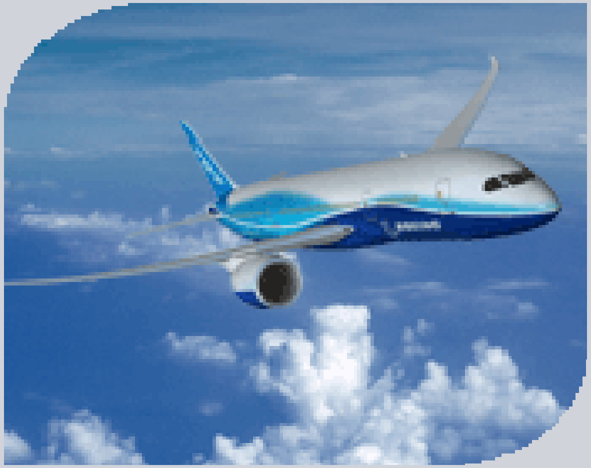
SIEMENS

Corporate Technology

Formal security analysis and product certification in industry, at practical examples



Dr. David von Oheimb
Siemens Corporate Technology, IT Security

Guest lecture in the Software Security series on invitation by Prof. Posegga, Univ. Passau, Germany, 11 Jun 2012

http://web.sec.uni-passau.de/teaching/ »Software-Sicherheit«

Overview

- **IT Security at Siemens Corporate Technology**
- Software distribution systems
- Common Criteria certification
- Formal security analysis
- Research project AVANTSSAR
- Conclusion on Formal Security Analysis

Siemens Corporate Technology (CT)

Networking the integrated technology company

SIEMENS

Customers

Chief Technology Officer (CTO)

- Review innovation strategies
- Drive technology based synergies
- Secure the innovation power
- Technology assessments
- Governance and guidance

Sectors / Divisions

Energy

Healthcare

Industry

Infrastructure & Cities

Regions

Chief Technology Office (CT O)
- Direct support of CTO

Corporate Research and Technologies (CT T)
- Technology Fields with multiple impact
- Pictures of the Future
- Accelerators

Corporate Development Center (CT DC)
- Software development partner for the Sectors

Corporate Intellectual Property and Functions (CT IP)
- Intellectual property
- Standardization and regulation
- Information research

Corporate Technology (CT)

Siemens corporate R&T: around 1,800 researchers

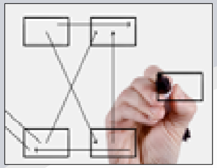
Present in all leading markets and technology hot spots

SIEMENS

SIEMENS

IT Security topics at Siemens Corporate Technology

IT Security @ Siemens CT



Security lifecycle

Sustainable integration of IT security topics into product lifecycle processes

Security assessment

“Friendly” hacking and assessments of products, solutions, applications and processes

Siemens product CERT

Incident handling and vulnerability management for Siemens products & solutions

Security architectures

Domain specific security architectures, certification, and best practice use of COTS and Open Source security

Embedded systems security

Optimized and adequate security for embedded systems

Computer Emergency Response Team (CERT)

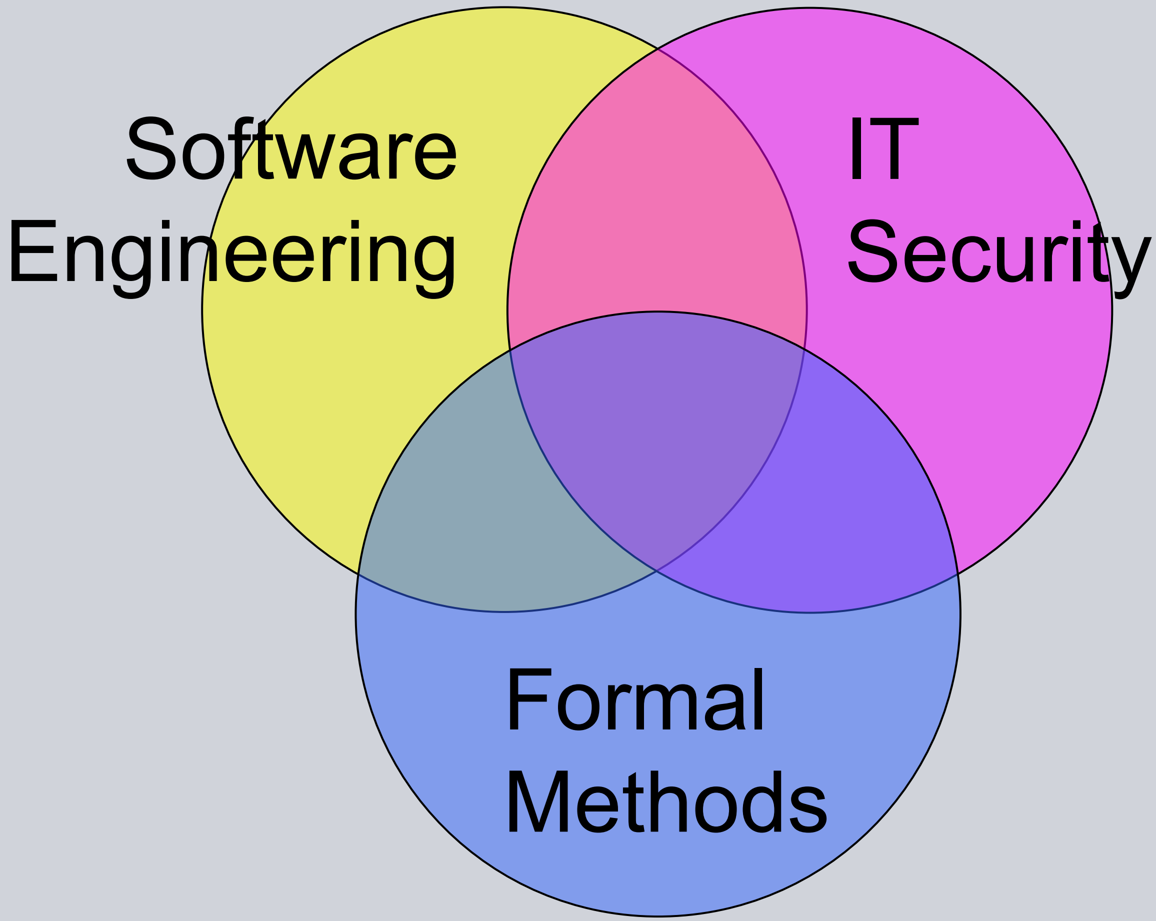
Corporate incident handling and technical policies

Cyber Security for specific regions

Regional Support focusing on specific security regulations and application topics like NERC-CIP, HIPAA, DIACAP (USA) or industrial control system security (Asia)

SIEMENS

Fields

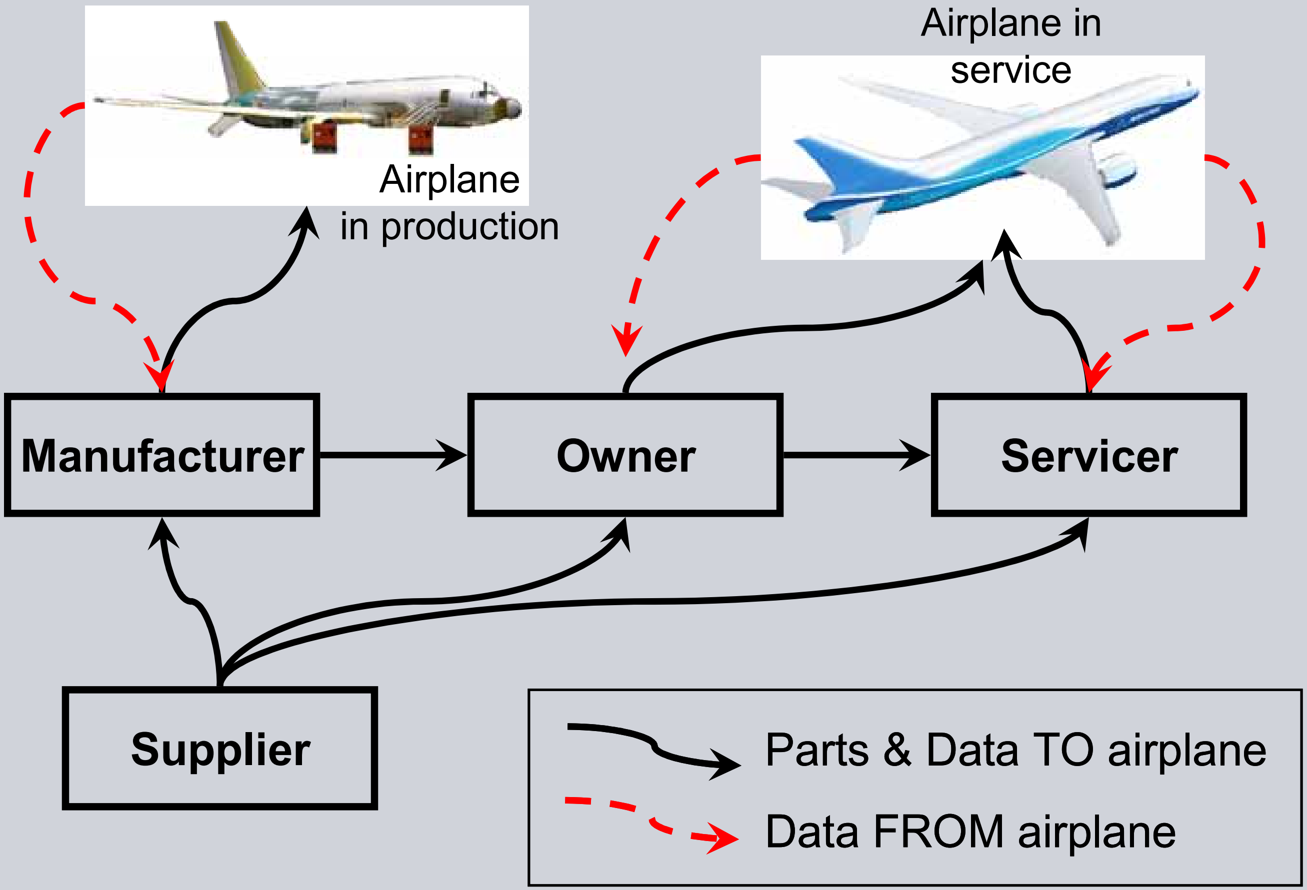


Overview

- IT Security at Siemens Corporate Technology
- **Software distribution systems**
- Common Criteria certification
- Formal security analysis
- Research project AVANTSSAR
- Conclusion on Formal Security Analysis

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



Airplane Assets Distribution System architecture

A complex distributed store-and-forward middleware with OSS components

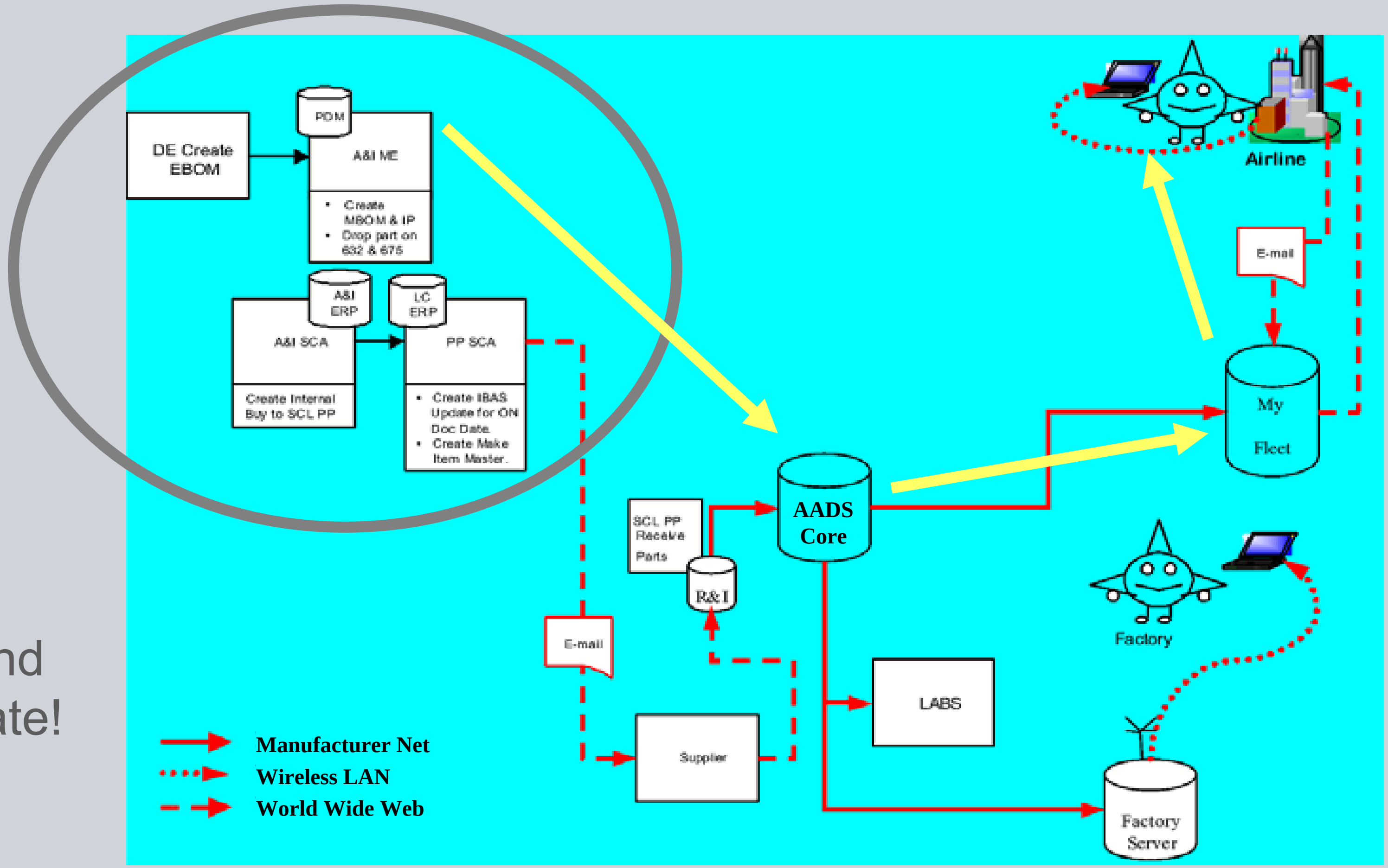
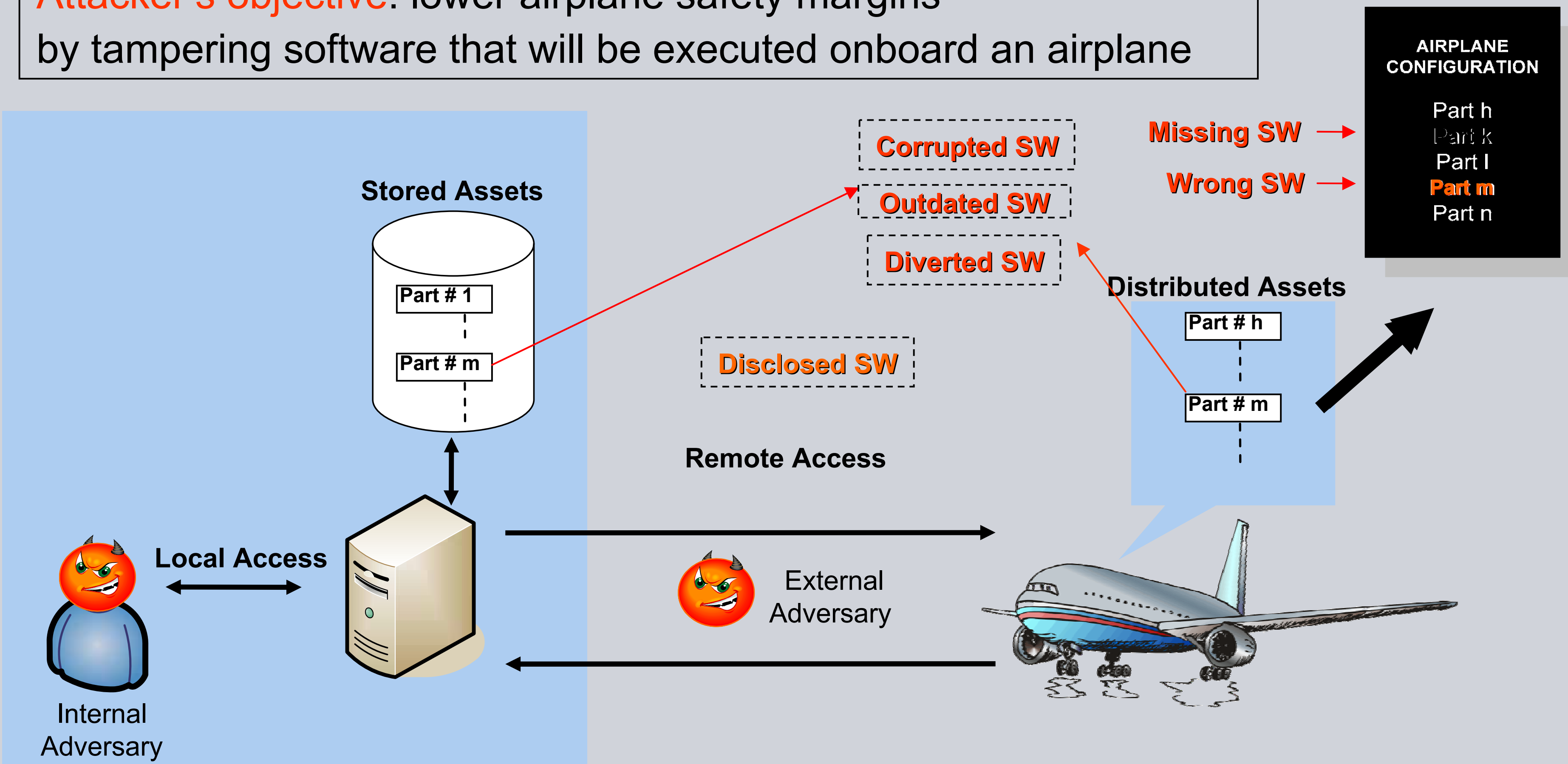


Figure is simplified and not up-to-date!

SIEMENS

Security threats at the AADS example

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

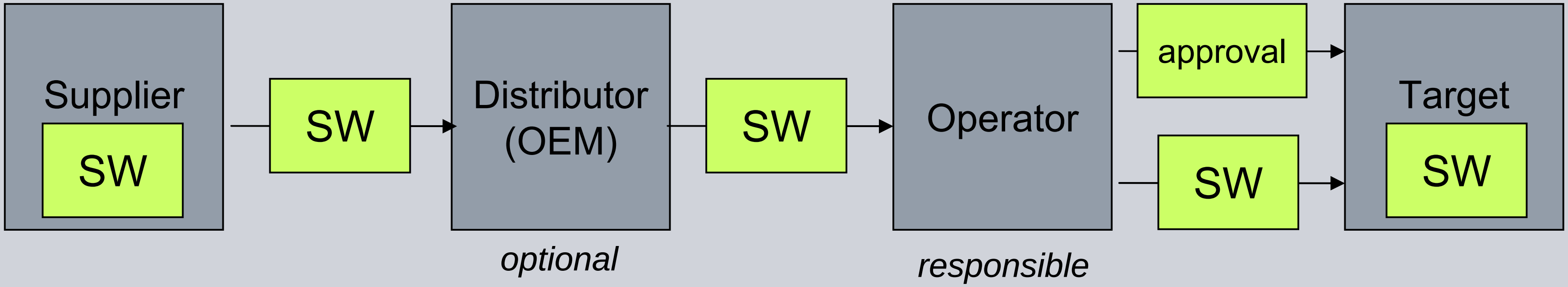


Corruption/Injection **Wrong Version** **Diversion** **Disclosure**

Software Distribution System (SDS)

ICT systems with networked devices in the field performing safety-critical and/or security-critical tasks.
Field devices require secure software update.

→ Software Distribution System (SDS):
System providing secure distribution of software (SW) from software supplier to target devices in the field



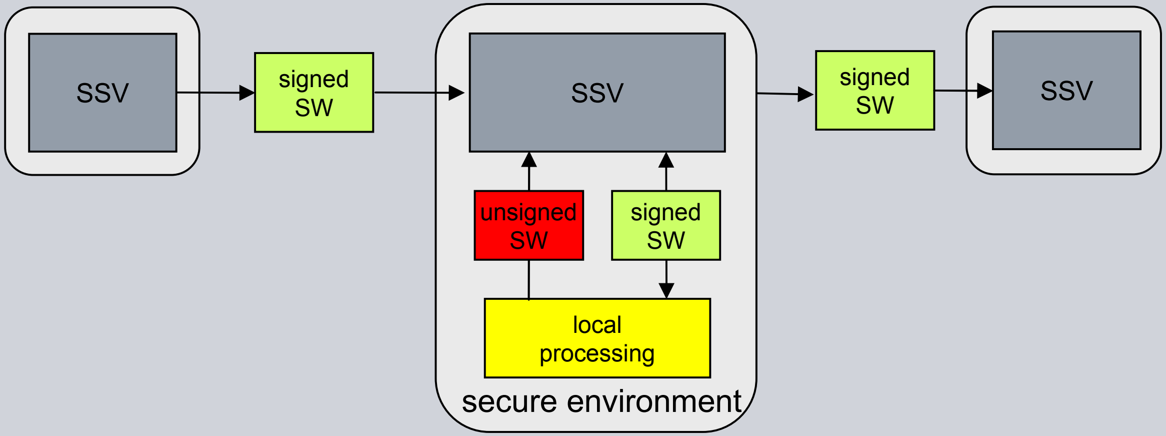
Transition from media-based (CD-ROMs etc.) to networked SW transport increases security risks due to transport over open, untrusted networks

SIEMENS

Software Signer Verifier (SSV)

Each node in SDS runs an SSV instance, used for:

- Introducing unsigned software into the SDS, by digitally signing and optionally encrypting it
- Verifying the signature on software received from other SSVs, checking integrity, authenticity and authorization of the sender
- Approving software by adding an authorized signature
- Delivering software out of the SDS after successfully verifying it



Overview

- IT Security at Siemens Corporate Technology
- Software distribution systems
- **Common Criteria certification**
- Formal security analysis
- Research project AVANTSSAR
- Conclusion on Formal Security Analysis

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

Common Criteria (CC) for IT security evaluation

Common Criteria

product-oriented methodology

for IT security assessment

ISO/IEC **standard** 15408

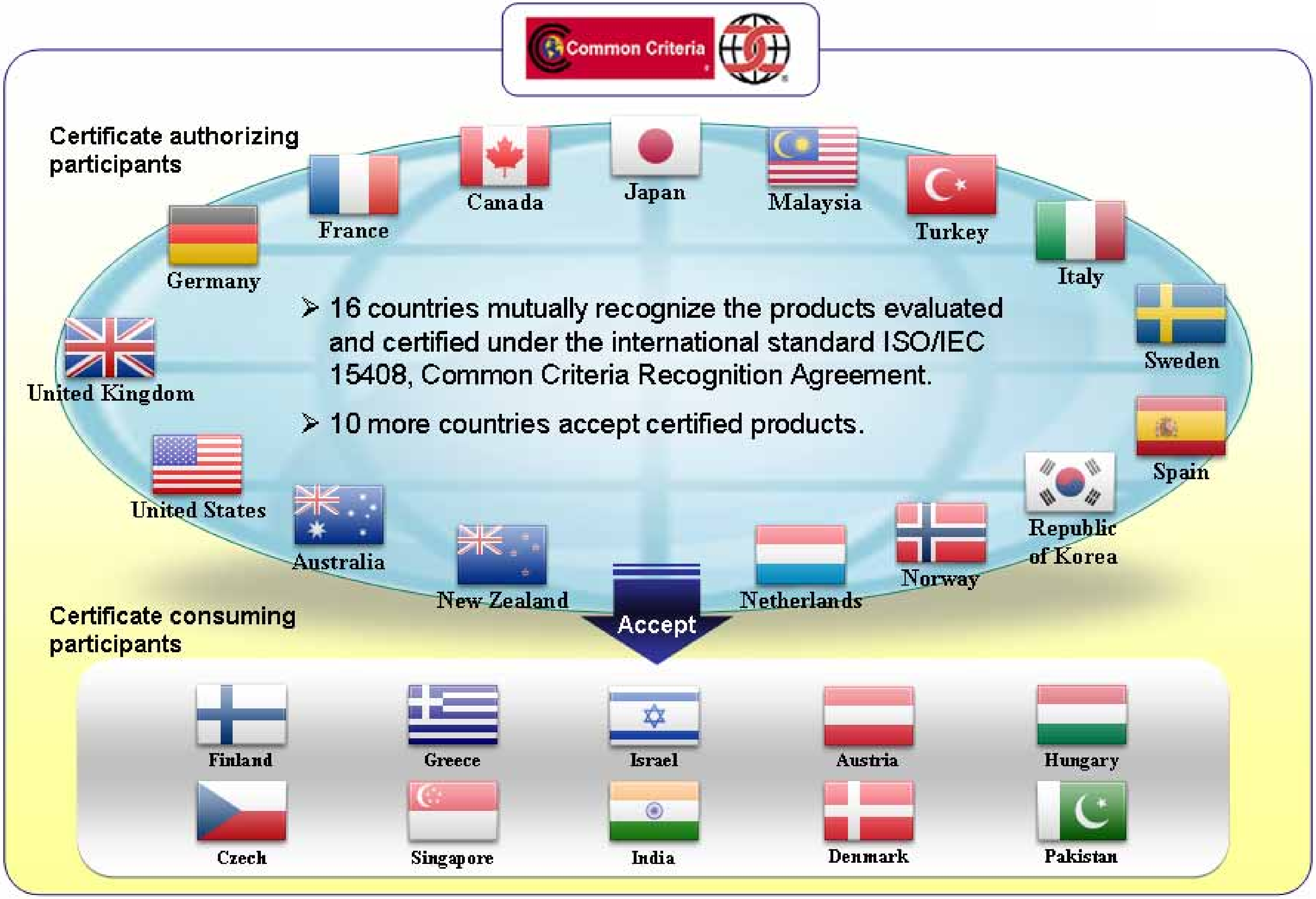
Current version: 3.1R3 of Jul 2009

Aim: gain confidence in the security of a system

Approach: assessment of system and documents by neutral experts

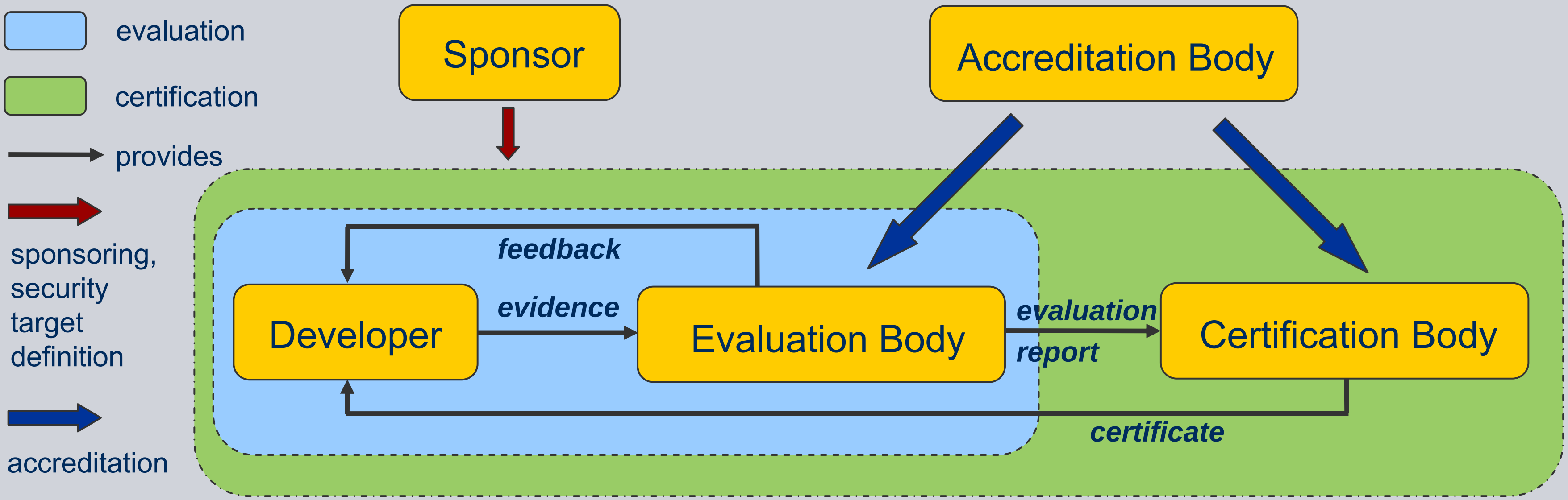
- What are the objectives the system should achieve?
- Are the measures employed appropriate to achieve them?
- Are the measures implemented and deployed correctly?

CC: authorization and international acceptance of certificates



SIEMENS

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).

SIEMENS

CC: Security requirements documents

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.

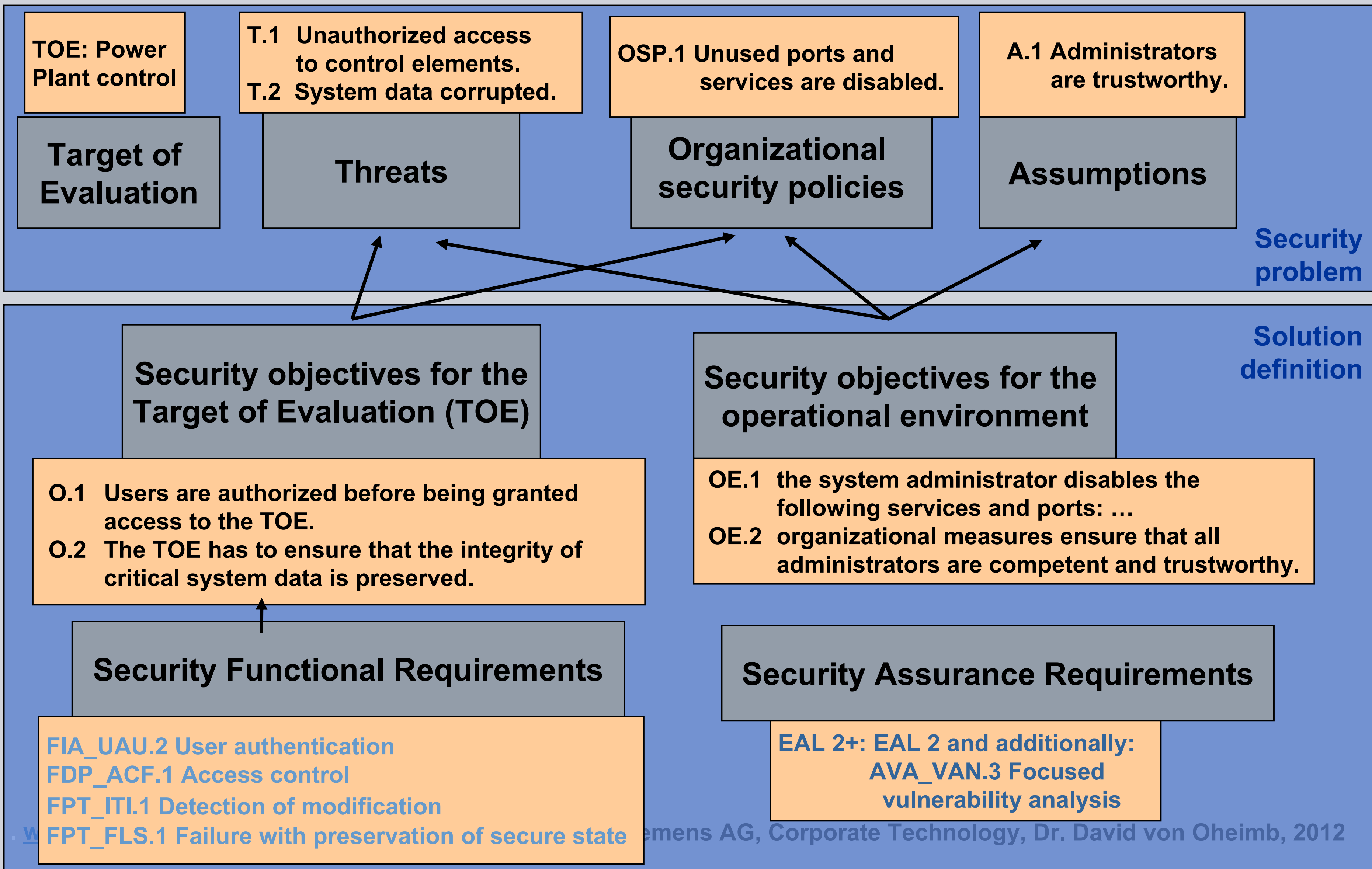
ST and PP specifications use **generic** “construction kit”:

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

layered as *Evaluation Assurance Levels (EALs)*

SIEMENS

CC: Security Target or Protection Profile example overview



SIEMENS

Threats Addressed by the AADS Security Objectives

Objectives	Threats	Safety-relevant				Business-relevant			
		Corruption	Misconfiguration	Diversion	Staleness	Unavailability	Late Detection	False Alarm	Repudiation
Safety-relevant	Integrity	√							
	Correct Destination			√					
	Latest Version				√				
	Authentication	√	√						√
	Authorization	√	√						
	Timeliness				√				
Business-Relevant	Availability					√			
	Early Detection						√		
	Correct Status							√	
	Traceability	√	√						√
	Nonrepudiation								√
Environment	Part_Coherence	√	√	√					
	Loading_Interlocks	√	√	√					
	Protective_Channels	√							
	Network_Protection				√	√			
	Host_Protection	√							√
Assumptions	Adequate_Signing	√							
	Configuration		√						
	Development	√	√	√	√	√	√	√	√
	Management	√	√						√

SIEMENS

CC: Security Functional Requirements (SFRs) overview

FAU: Security audit

- Security audit automatic response (FAU_ARP)
- Security audit data generation (FAU_GEN)
- Security audit analysis (FAU_SAA)
- Security audit review (FAU_SAR)
- Security audit event selection (FAU_SEL)
- Security audit event storage (FAU_STG)

FCO: Communication

FCS: Cryptographic support

FDP: User data protection

FIA : Identification and authentication

FMT: Security management

FPR: Privacy

FPT: Protection of the TSF

FRU: Resource utilization

FTA: TOE access

FTP: Trusted path/channels

SIEMENS

CC: Evaluation Assurance Levels

Assurance requirements are grouped as Evaluation Assurance Levels:

	EAL designation
EAL1	functionally tested
EAL2	structurally tested
EAL3	methodically tested and checked
EAL4	methodically designed, tested and reviewed
EAL5	semiformally designed and tested
EAL6	semiformally verified design and tested
EAL7	formally verified design and tested

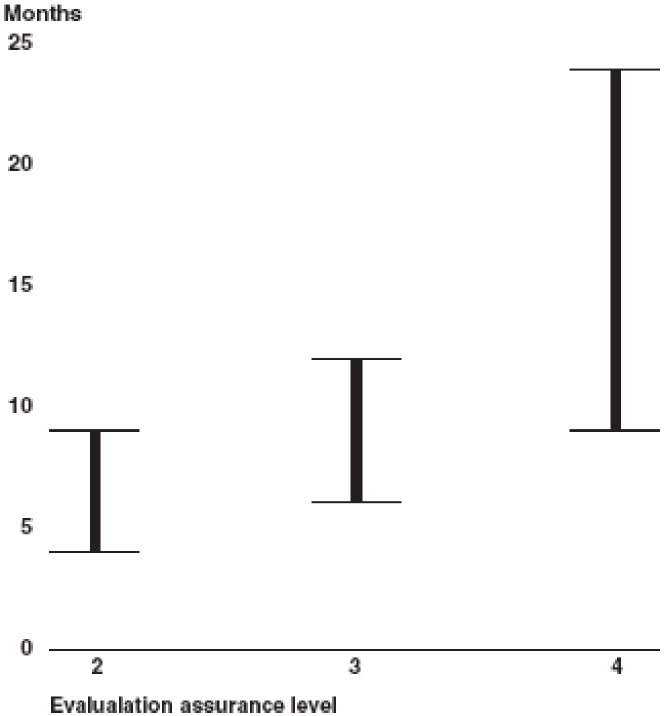
Increasing requirements on scope, depth and rigor of evaluation.

EAL does not say how secure a product is, but how well its requirements are checked.

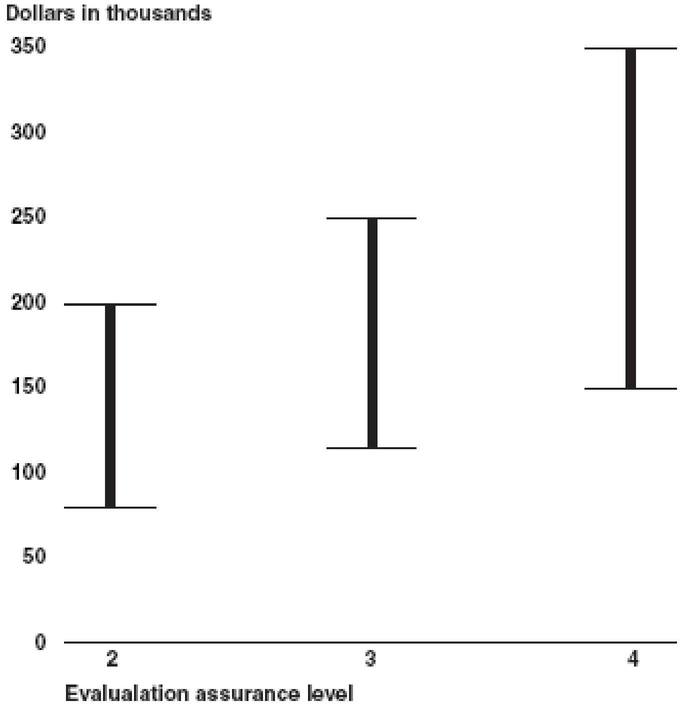
Assurance is grounds for confidence that an IT product meets its security objectives.

CC: Factors determining the evaluation effort

- Boundary of TOE vs. TOE environment
- Definition of Threats and Security Objectives for the TOE
- Definition of Security Functional Requirements (SFRs)
- Selection of Evaluation Assurance Level (EAL)



Source: GAO analysis of data provided by laboratories.



Source: GAO analysis of data provided by laboratories.

SIEMENS

Selection of Evaluation Assurance Level (EAL) for AADS

	Flight safety	**Airline business**
Threat Level assume sophisticated adversary with moderate resources who is willing to take XXX risk	**T5**: XXX = significant e.g. intl. terrorists	**T4**: XXX = little e.g. organized crime, sophisticated hackers, intl. corporations
Information Value violation of the protection policy would cause YYY damage to the security, safety, financial posture, or infrastructure of the organization	**V5**: YYY= exceptionally grave Risk: loss of lives	**V4**: YYY = serious Risk: airplanes out of service, or damage airline reputation
Evaluation Assurance Level for the given Treat Level and Information Value	**EAL 6**: semiformally verified design and tested	**EAL 4**: methodically designed, tested, and reviewed

Evaluating the whole AADS at EAL 6 would be extremely costly.
Currently available Public Key Infrastructure (PKI) certified only at EAL 4.

Two-level approach: evaluate only LSAP integrity & authenticity at EAL6.

SIEMENS

Conclusion (1) on AADS

- Challenges for AADS development
  - pioneering system design and architecture
  - complex, heterogeneous, distributed system
  - security is critical for both safety and business
- Common Criteria (CC) offer widely accepted, adequate methodology for assessment, at least for small products / systems components
- Systematic approach, in particular formal analysis, enhances
  - understanding of the security issues
  - quality of specifications and documentation
  - confidence (of Boeing, customers, FAA, etc.) in the security solutions

SIEMENS

Conclusion (2) on AADS

- Experience with AADS evaluation
  - CC offer good guidance for systematic security problem definition:
    threats, assumptions, organizational policies, objectives
  - Shape system architecture to alleviate security evaluation
  - Use formal analysis where cost/benefit ratio is best
  - Problem of compositional security evaluation not solved

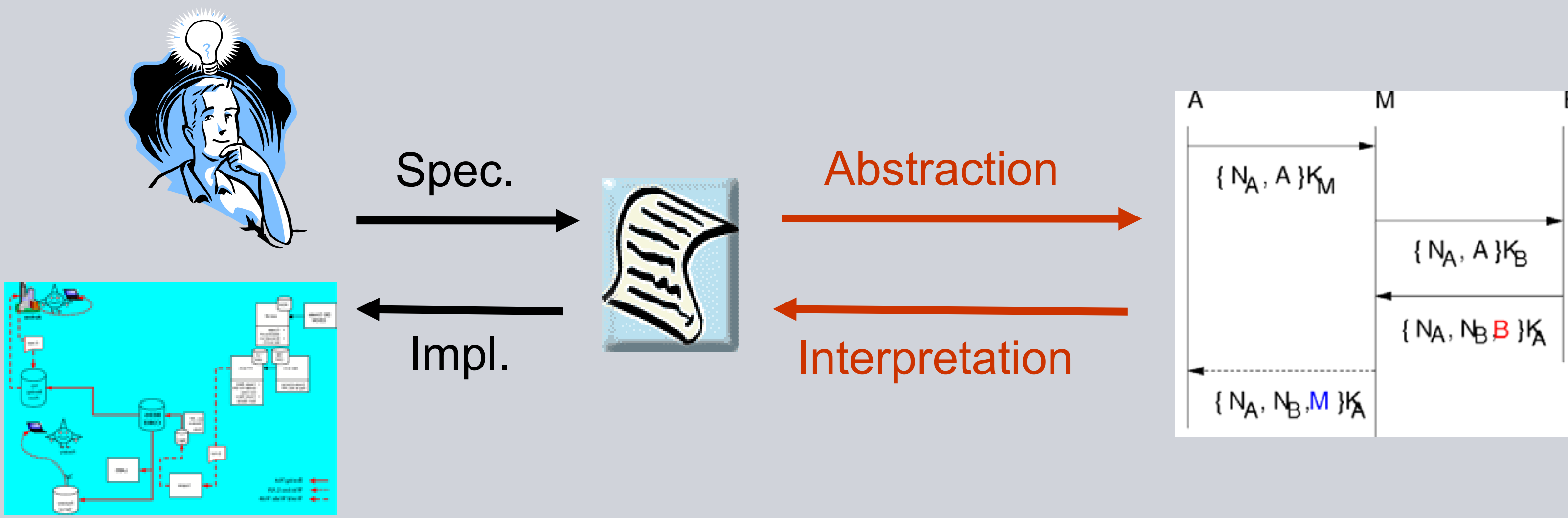
- Aspects omitted so far:
  - Key management
    Public Key Infrastructure (PKI) components etc.
  - Configuration management
    with installation instructions and status/completion reports

Overview

- IT Security at Siemens Corporate Technology
- Software distribution systems
- Common Criteria certification
- **Formal security analysis**
- Research project AVANTSSAR
- Conclusion on Formal Security Analysis

Formal Security Analysis: Approach and Benefits

Mission: security analysis with maximal precision
Approach: formal modeling and verification



Improving the quality of the system specification

Checking for the existence of security loopholes

AVANTSSAR Specification Language
- Model checkers (AVANTSSAR Tool)

Interacting State Machines
- Interactive theorem prover (Isabelle)

Tools of the avantssar.eu project

Needham-Schroeder Public Key Protocol

[Needham-Schroeder 1978]
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:

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

Goal: strong mutual authentication

SIEMENS

Example: ASLan++ model NSPK_Cert (1): Alice and 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); } }
  } …
}
```

SIEMENS

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

$$
\begin{array}{llll}
1.1 & A - \{Na.A\}_{pk(i)} \rightarrow i & & \\
2.1 & & i(A) - \{Na.A\}_{pk(B)} \rightarrow B & \\
2.2 & & i(A) \leftarrow \{Na.Nb\}_{pk(A)} - B & \\
1.2 & A \leftarrow \{Na.Nb\}_{pk(A)} - i & & \\
1.3 & A - \{Nb\}_{pk(i)} \dashrightarrow i & & \\
2.3 & & i(A) - \{Nb\}_{pk(B)} \dashrightarrow B &
\end{array}
$$

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.

Formal Security Models

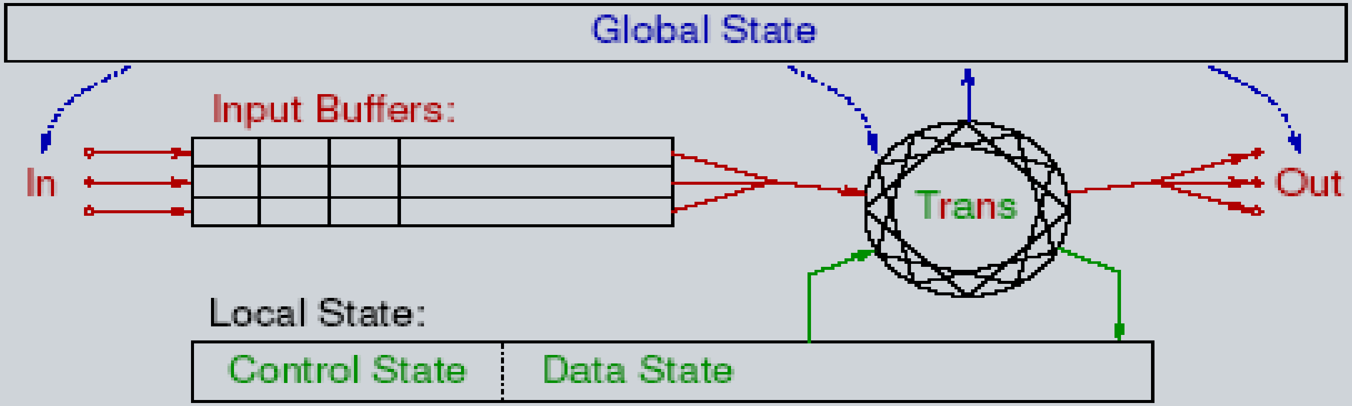
SIEMENS

- 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

SIEMENS

Interacting State Machines (ISMs)

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



Transitions definable in executable and/or axiomatic style.

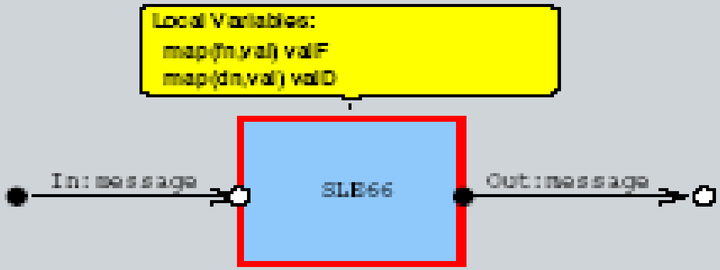
An ISM system may have changing global state.

Applicable to a large variety of reactive systems.

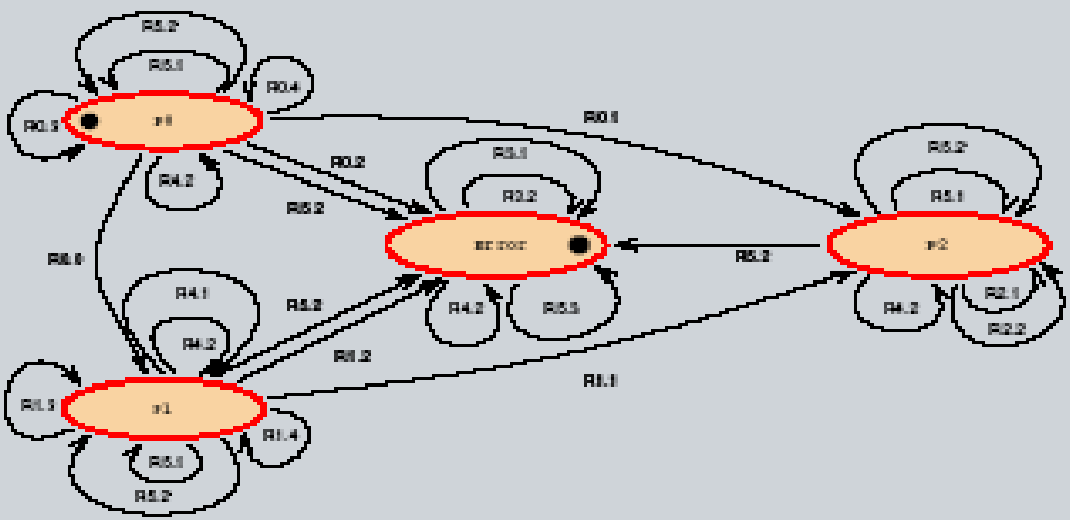
By now, not much verification support (theory, tools).

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

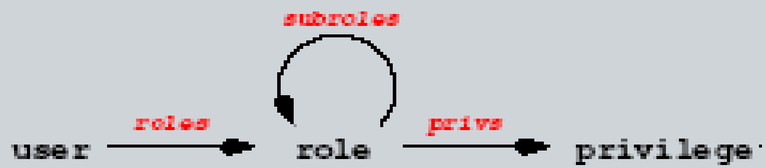
Is the security design (with emergency access etc.) sound?

Privileges:

$\mathit{roles} \subseteq \mathtt{user} \times \mathtt{role}$

$\mathit{subroles} \subseteq \mathtt{role} \times \mathtt{role}$

$\mathit{privs} \subseteq \mathtt{role} \times \mathtt{privilege}$



$$(u, p) \in \mathit{roles} \circ \mathit{subroles}^* \circ \mathit{privs}$$

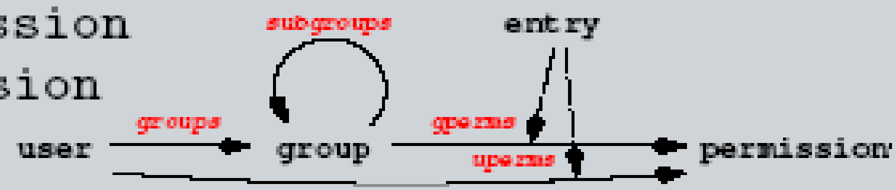
Permissions:

$\mathit{groups} \subseteq \mathtt{user} \times \mathtt{group}$

$\mathit{subgroups} \subseteq \mathtt{group} \times \mathtt{group}$

$\mathit{gperms} \subseteq \mathtt{group} \times \mathtt{permission}$

$\mathit{uperms} \subseteq \mathtt{user} \times \mathtt{permission}$



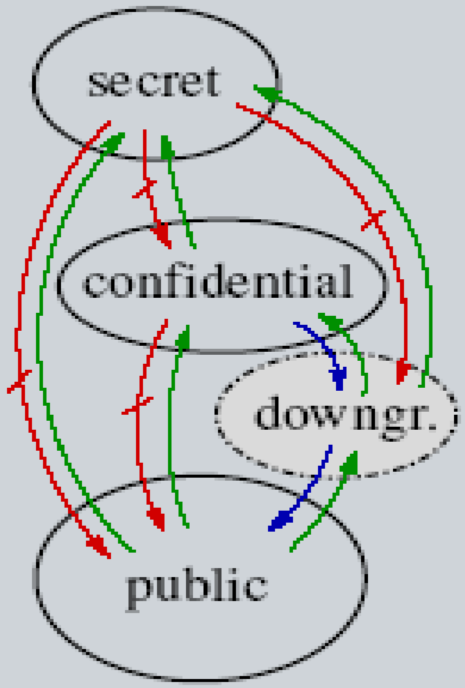
$$(u, p) \in (\mathit{groups} \circ \mathit{subgroups}^* \circ \mathit{gperms}(e)) \cup \mathit{uperms}(e)$$

"nagging questions" $\leadsto$ clarifications improving specification quality.

Open issue: relation between model and implementation ($\leadsto$ 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 $(x, y) \stackrel{L}{\sim} (x', y') \;:\longleftrightarrow\; y = y'$ allows re-formulation:

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

Generalization to action sequences α and arbitrary policies $\rightsquigarrow$

SIEMENS

Cryptoprotocol models

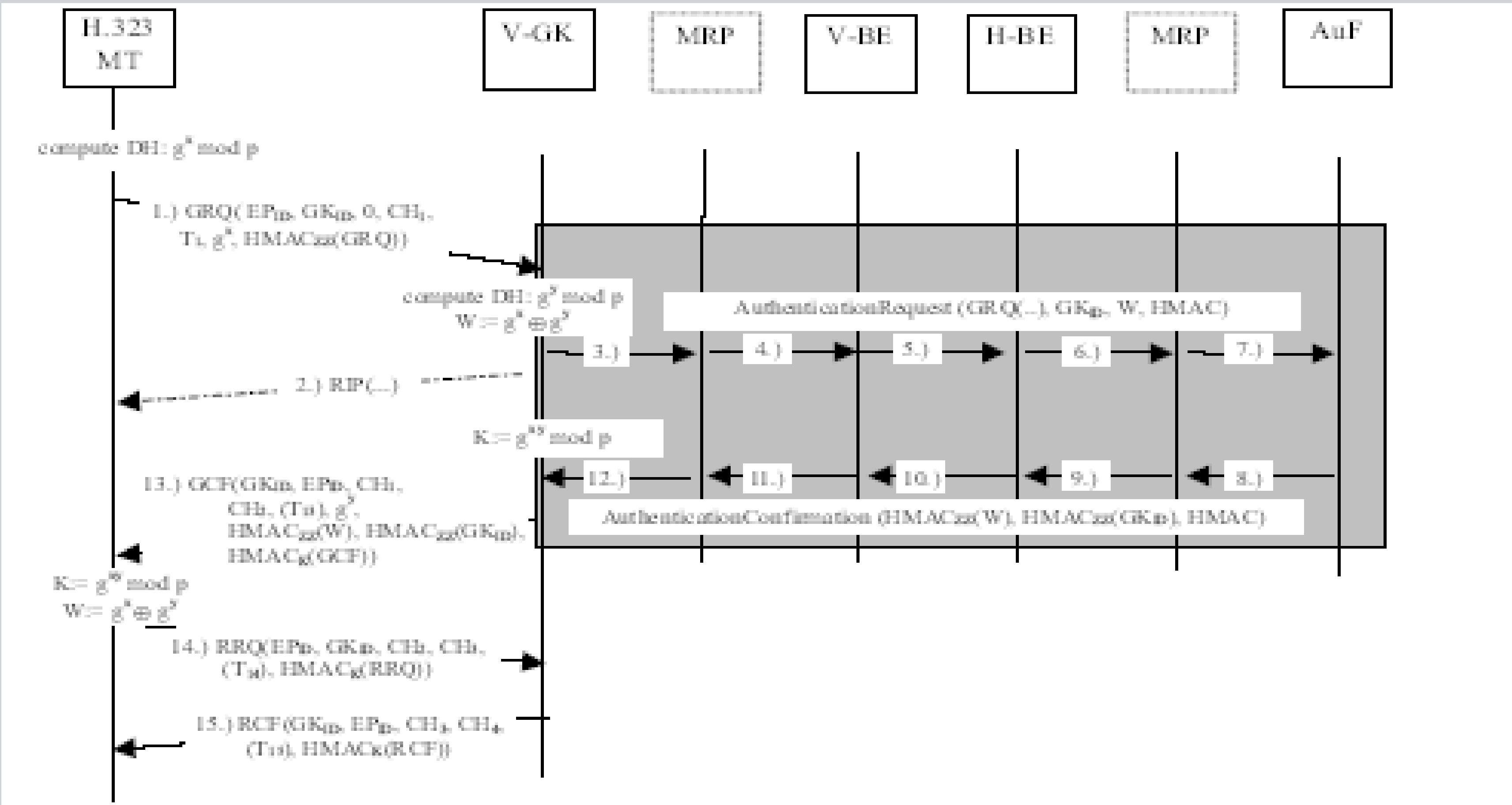
- Describe message exchange between processes or principals

- 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 , ...

SIEMENS

Example: H.530 Mobile Roaming Authentication



Two vulnerabilities found and corrected. Solution standardized.

Overview

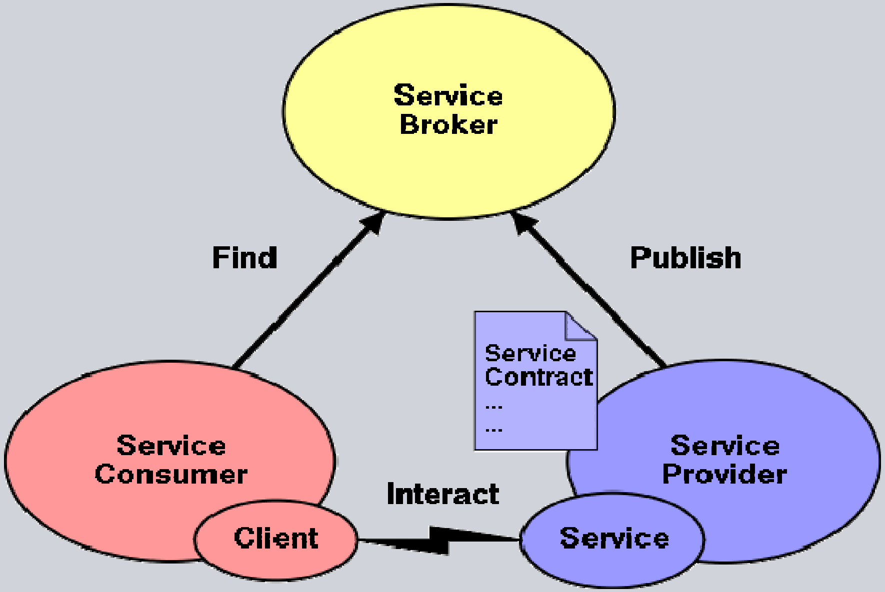
- IT Security at Siemens Corporate Technology
- Software distribution systems
- Common Criteria certification
- Formal security analysis
- **Research project AVANTSSAR**
- Conclusion on Formal Security Analysis

AVANTSSAR

SIEMENS

avantssar.eu

Model-checking SOA security — research project AVANTSSAR[1]



[1] **A**utomated **V**alidatio**N** of **T**rust and **S**ecurity of **S**ervice-oriented **Ar**chitectures

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

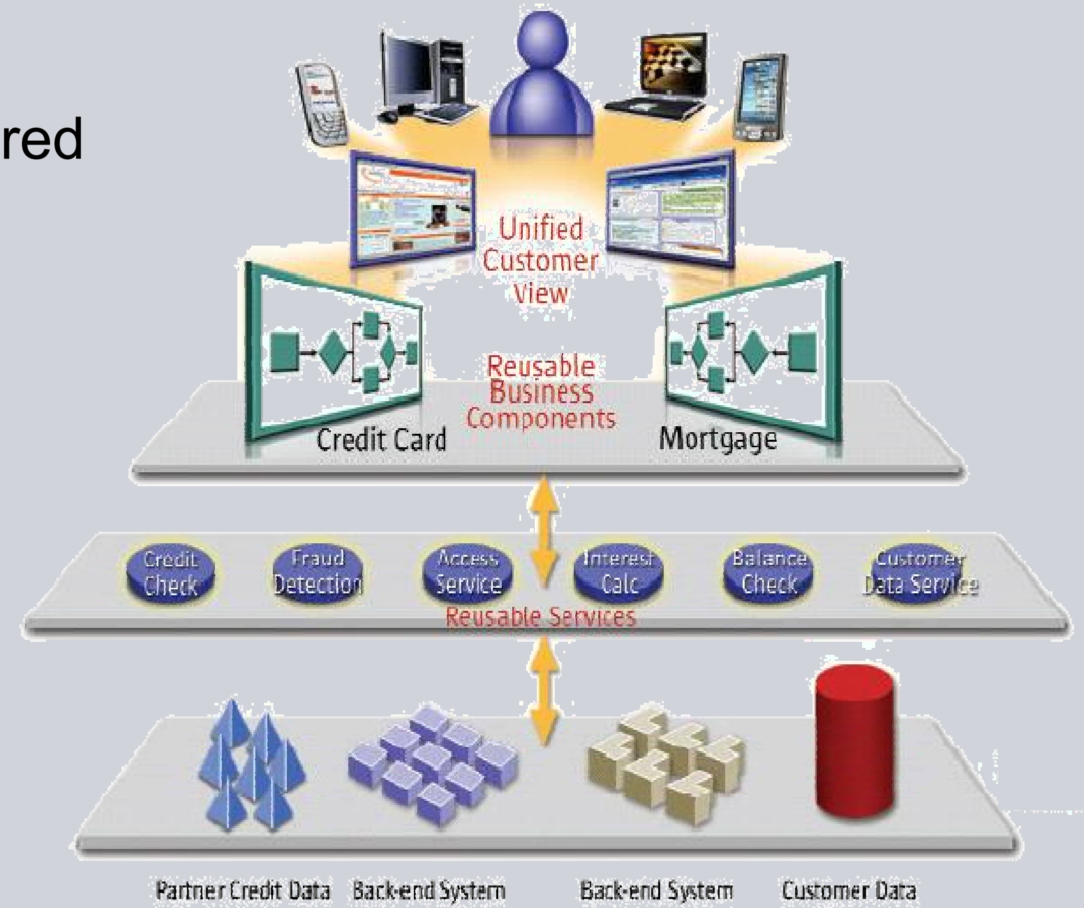
SIEMENS

AVANTSSAR project motivation

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

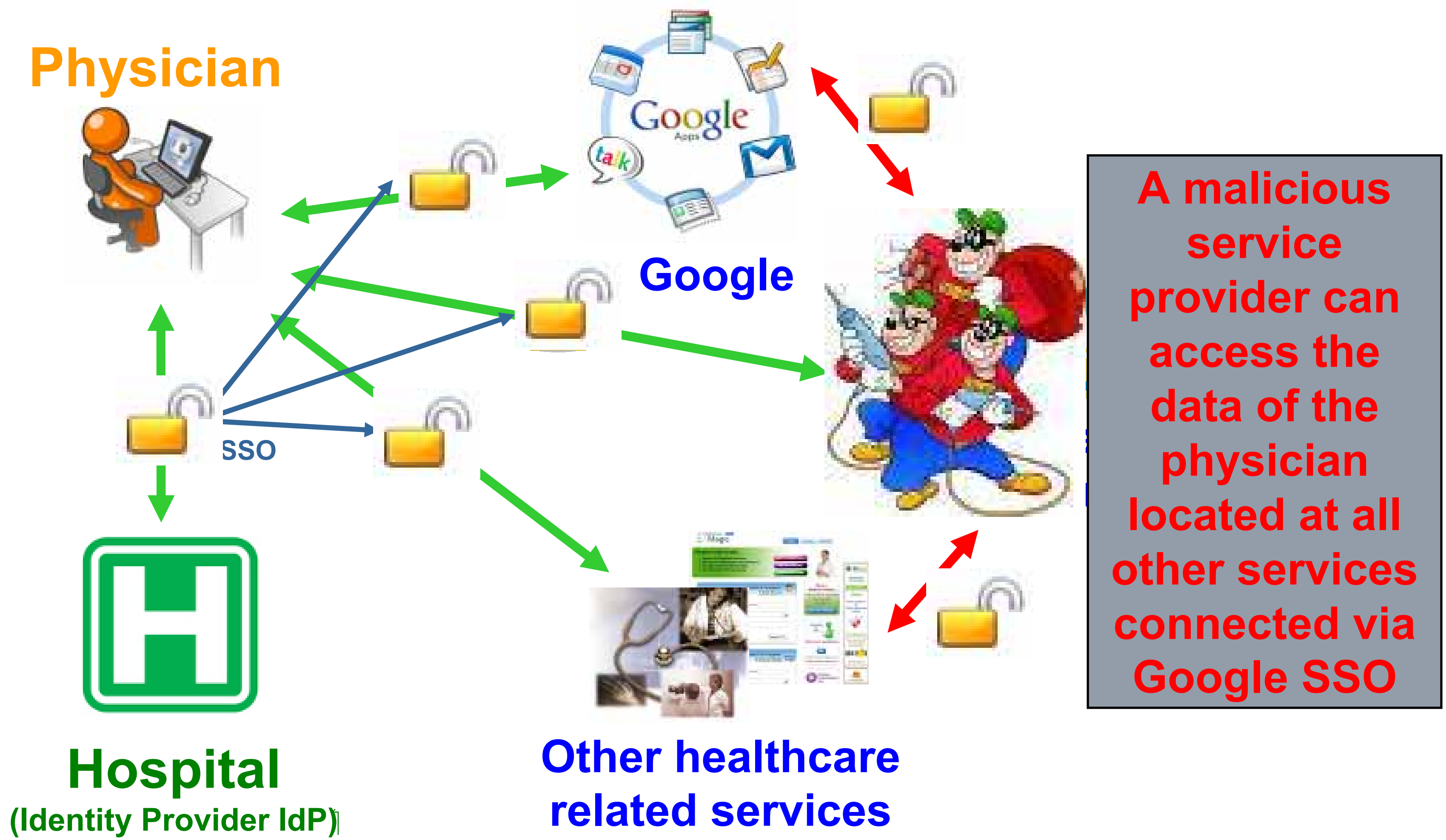
Trustworthy service may interact with others causing novel trust and security problems.

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



SIEMENS

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



Example 1: Google SAML SSO protocol flaw

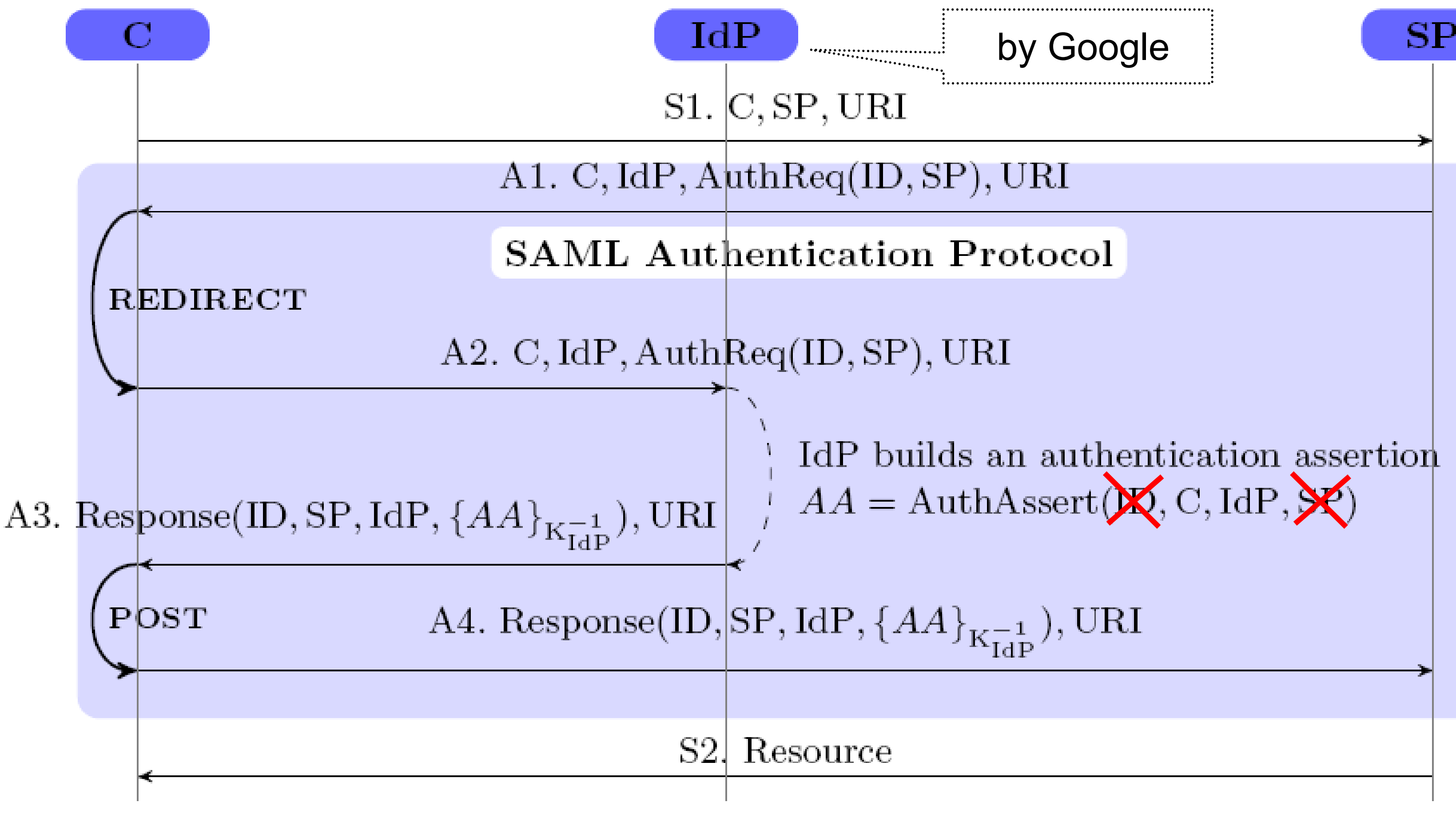


Fig. 1. SP-Initiated SSO with Redirect/POST Bindings

AVANTSSAR consortium

SIEMENS

Industry

SAP Research France, Sophia Antipolis

Siemens Corporate Technology, München

IBM Zürich Research Labs (part time)

OpenTrust, Paris

Academia

Università di Verona

Università di Genova

ETH Zürich

INRIA Lorraine

UPS-IRIT Toulouse

IEAT Timişoara

Expertise

Service-oriented enterprise architectures

Security solutions

Standardization and industry migration

Security engineering

Formal methods

Automated security validation

SIEMENS

AVANTSSAR main objectives and principles

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

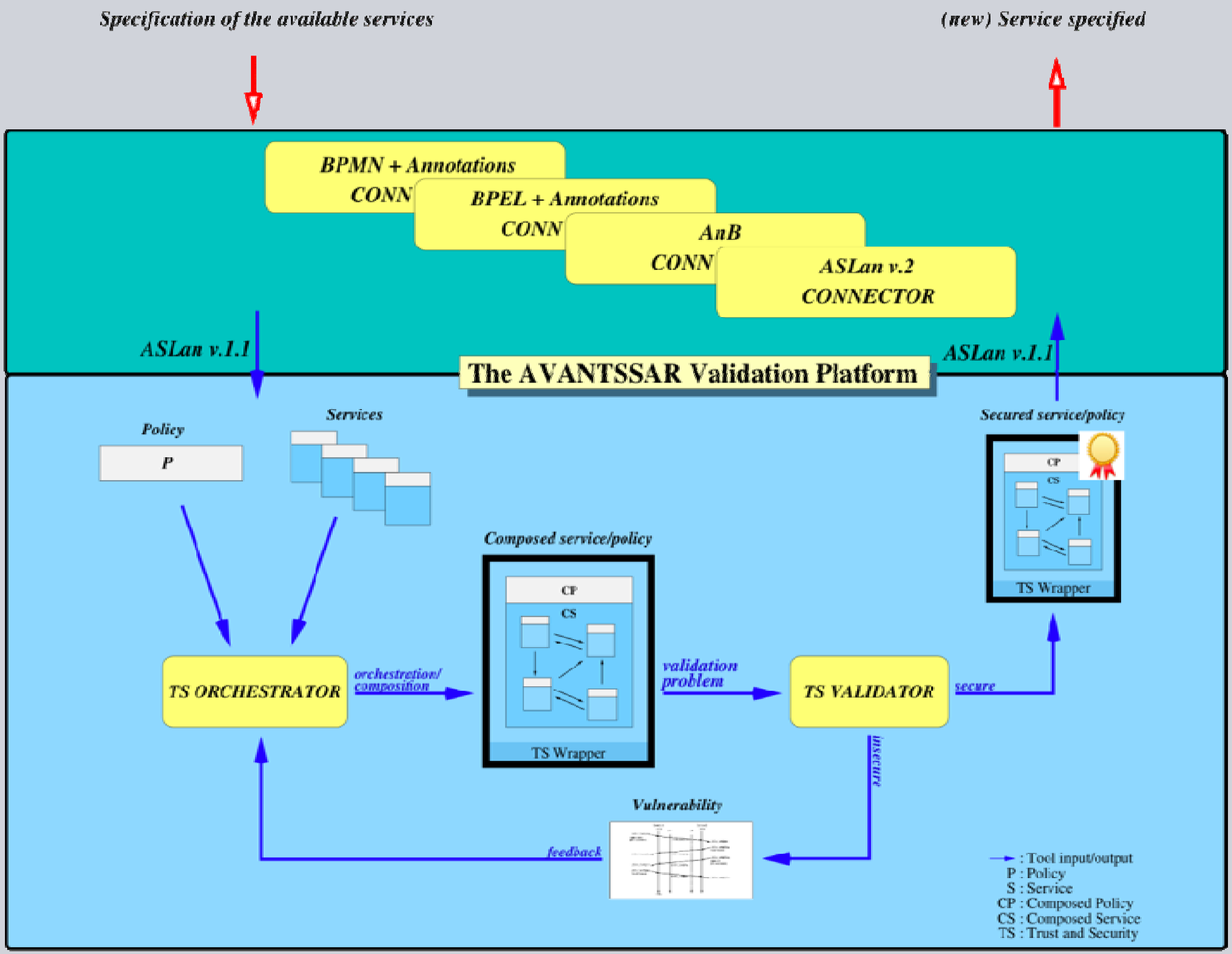
- Formal language for specifying trust and security properties of services, their policies, and their composition into service-oriented architectures
- Automated toolset supporting the above
- Library of validated industry-relevant case studies

Migration of platform to industry and standardization organizations

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

SIEMENS

AVANTSSAR project results and innovation

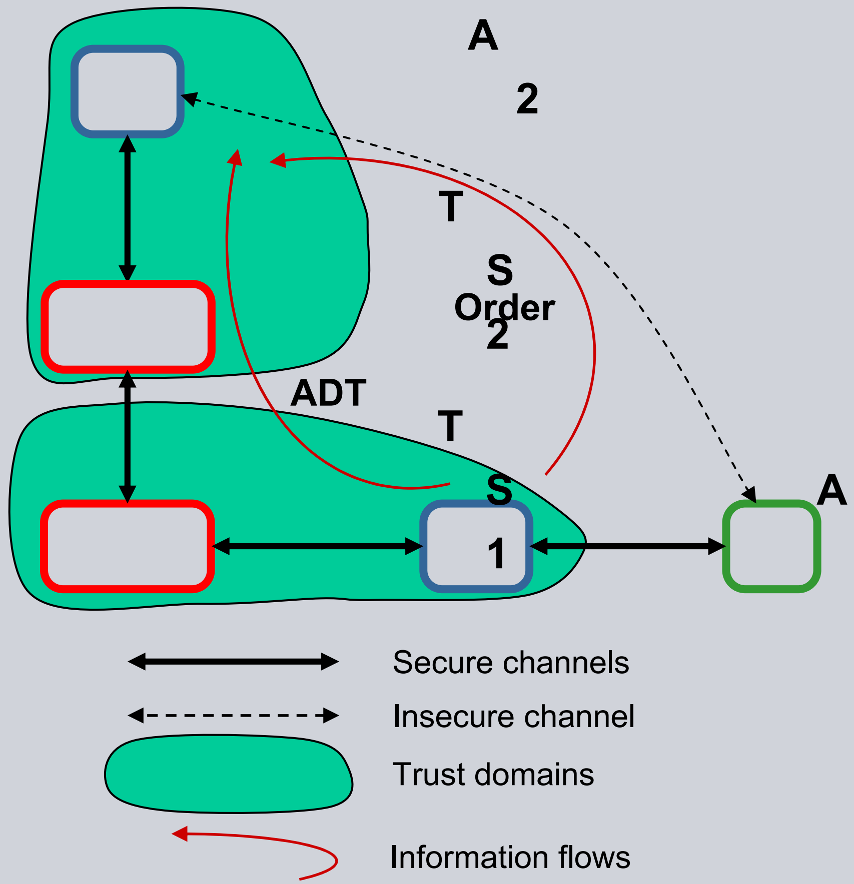


SIEMENS

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 SSO, authorization and trust management
- Each *application* **A** is in one domain, each domain has exactly one active *token 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

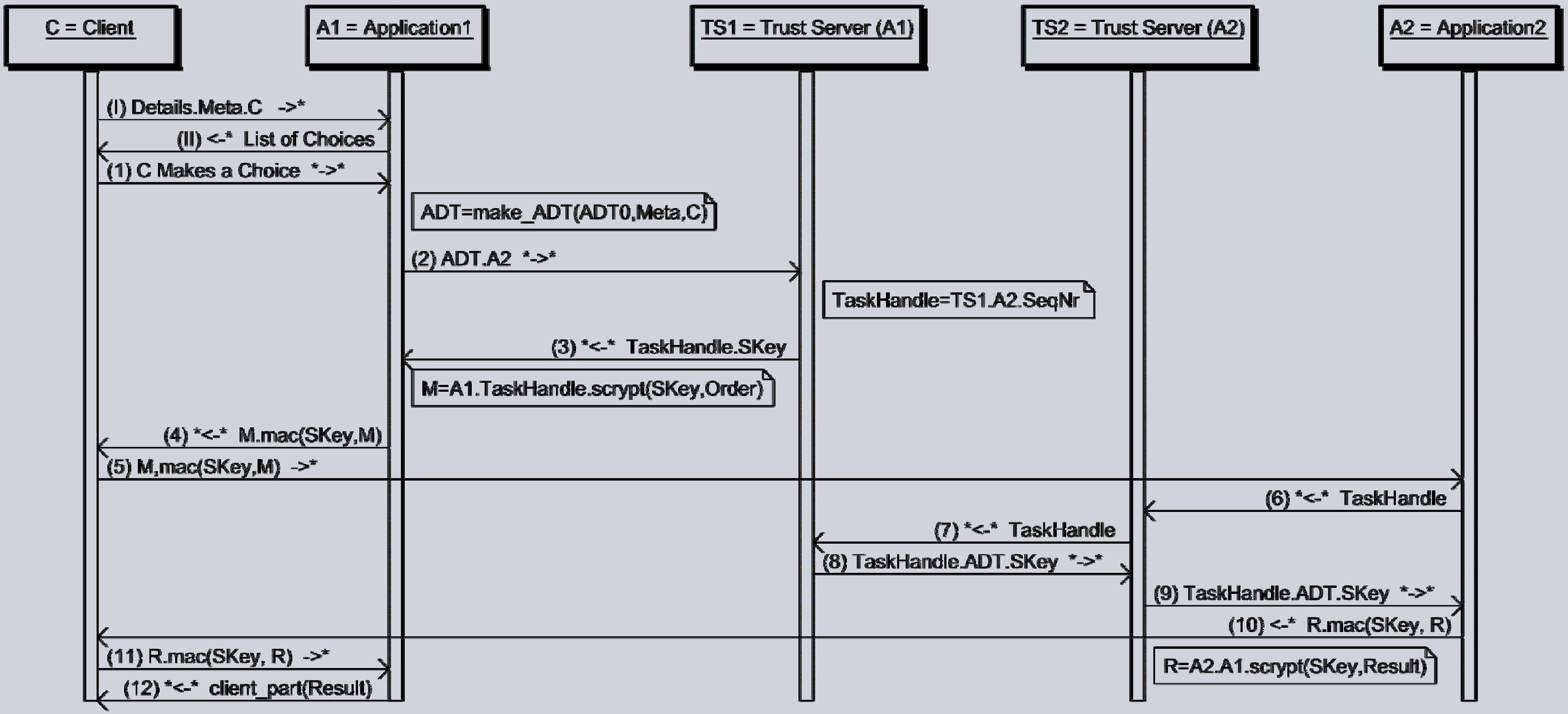
    to be delivered securely (integrity and confidentiality)
  - **ADT** is mainly authorization *attributes* and *decisions*
    - sent via **TS1** and **TS2**, <u>who may weaken it</u>
    - must remain unaltered, apart from weakening by **TS**
    - must remain confidential among intended parties
- **C**, **A1**, and **A2** must be authenticated among each other



- **Security prerequisites**:
- PKI is used for **A** and **TS**, username & pwd for **C**
- **TS** enforces a strict time-out

Example 2: Message Sequence Chart of PTD

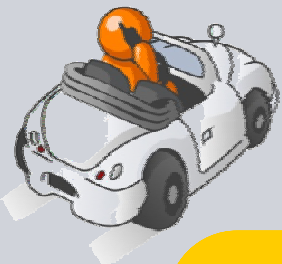
SIEMENS



Example 2: ASLan++ model of A2

```
entity A2 (Actor: agent, TS2: agent) {    % Applicaton2, connected with TokenServer2
  symbols
    C0,C,A1: agent;
    CryptedOrder, Order, Order0, Details, Results, TaskHandle, ADT, HMAC: 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).?HMAC): {
        % 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 = scrypt(SKey,?Order0,?Details.?C)
          & HMAC = hmac(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. scrypt(SKey,Results);
   } } }
  goals
    authentic_C_A2_Details: C  *-> Actor: Details;
    secret_Order: secret (Order0,Details.C, {Actor, A1});
}
```

Example 3: Electronic Car Registration policies



local policy

RegOffCA can say
who is RegOffHead
who is RegOffEmpl

certificate

Peter is RegOffEmpl
of CarRegOffice
(signed by RegOffCA)

ACL

anybody,
get empty forms
RegOffHead, write
RegOffEmpl, read
RegOffEmpl, write,
if *his* RegOffHead
says so

CentrRep

write(..)

Peter

Melinda

RegOffCA

certificate

Melinda is RegOffHead
of CarRegOffice
(signed by RegOffCA)

certificate

Peter can write
CentrRep
(signed by Melinda)

Question:

May Peter write to CentrRep?

SIEMENS

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)));
```

AVANTSSAR final status

SIEMENS

WP2: ASLan++ supports the formal specification of trust and security related aspects of SOAs, and of static service and policy composition

WP3: Techniques for: satisfiability check of policies, model checking of SOAs w.r.t. policies, different attacker models, compositional reasoning, abstraction

WP4: Deploy first prototype of **AVANTSSAR Platform**

WP5: Formalization of **industry-relevant problem cases** as ASLan++ specifications and their validation

WP6: Ongoing dissemination and migration into scientific community and industry

AVANTSSAR Tool demo (part 2)

Try the AVANTSSAR platform

pre-release at ddvo.net/AVANTSSAR

- TLS Client and Server model

Overview

- IT Security at Siemens Corporate Technology
- Software distribution systems
- Common Criteria certification
- Formal security analysis
- Research project AVANTSSAR
- **Conclusion on Formal Security Analysis**

SIEMENS

Formal Security Analysis: Information Required

- Overview: system architecture (components and interfaces),
  e.g. databases, authentication services, connections,…
- Security-related concepts: actors, assets, states, messages, …
- Threats: which attacks have to be expected.
- Assumptions: what does the environment fulfill.
- Security objectives: what the system should achieve.
  Described in detail such that concrete verification goals can be set up
  e.g. integrity: which contents shall be modifiable by whom, at which times,
  by which operations (and no changes otherwise!)
- Security mechanisms: relation to objectives and how they are achieved.
  e.g. who signs where which contents, and where is the signature checked
  Described precisely but at high level (no implementation details required),
  e.g. abstract message contents/format but not concrete syntax

Shaping a Formal Model

SIEMENS

Formality Level: should be adequate:

- the more formal, the more precise,
- but requires deeper mastering of formal methods

Choice of Formalism: dependent on ...

- application domain, modeler's experience, tool availability, ...
- formalism should be simple, expressive, flexible, mature

Abstraction Level: should be ...

- high enough to achieve clarity and limit the effort
- low enough not to loose important detail

refinement allows for both high-level and detailed description

SIEMENS

Development Phases and the Benefits of Formal Analysis

Requirements analysis:

understanding the security issues

- abstraction: concentration on essentials, to keep overview
- genericity: standardized patterns simplify the analysis

Design, documentation:

quality of specifications

- enforces preciseness and completeness

Implementation:

effectiveness of security functionality

- formal model as precise reference for testing and verification