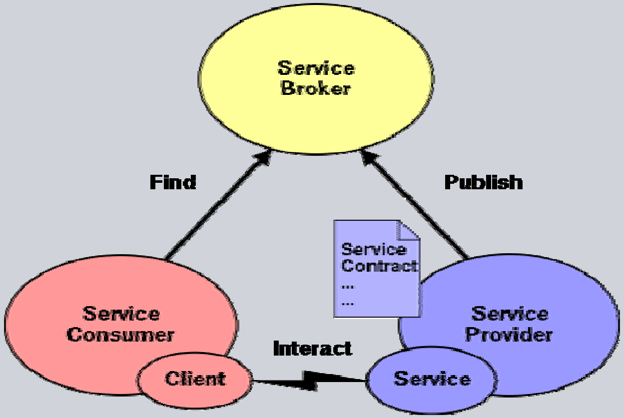
SIEMENS

avantssar.eu

Model checking SOA Security: a report on work in progress in AVANTSSAR[1]



Presented at DFKI Formal Methods Group, Saarbrücken, Germany, 2010-07-19

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

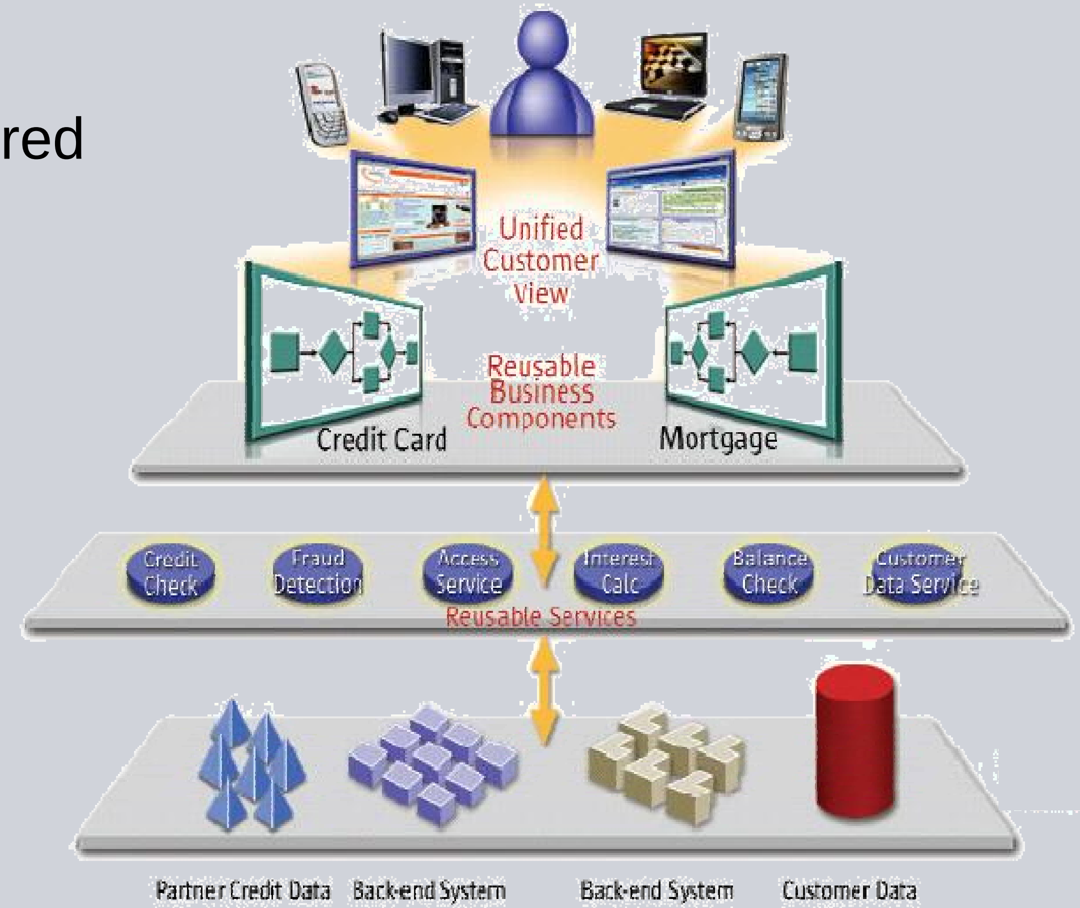
SIEMENS

AVANTSSAR project motivation

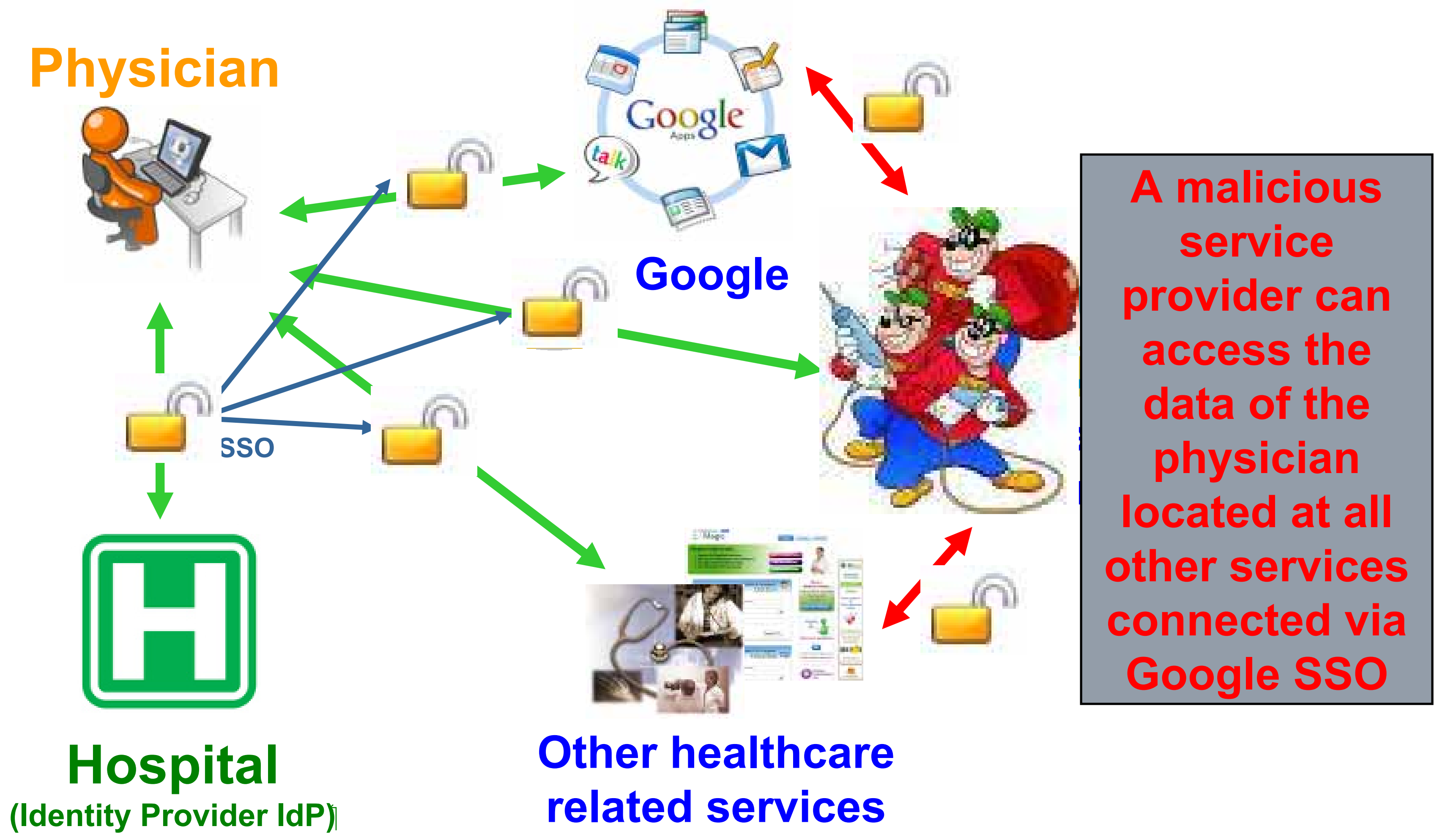
ICT paradigm shift: from components to services, composed and reconfigured dynamically in a demand-driven way.

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.



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



Example 1: Google SAML SSO protocol flaw

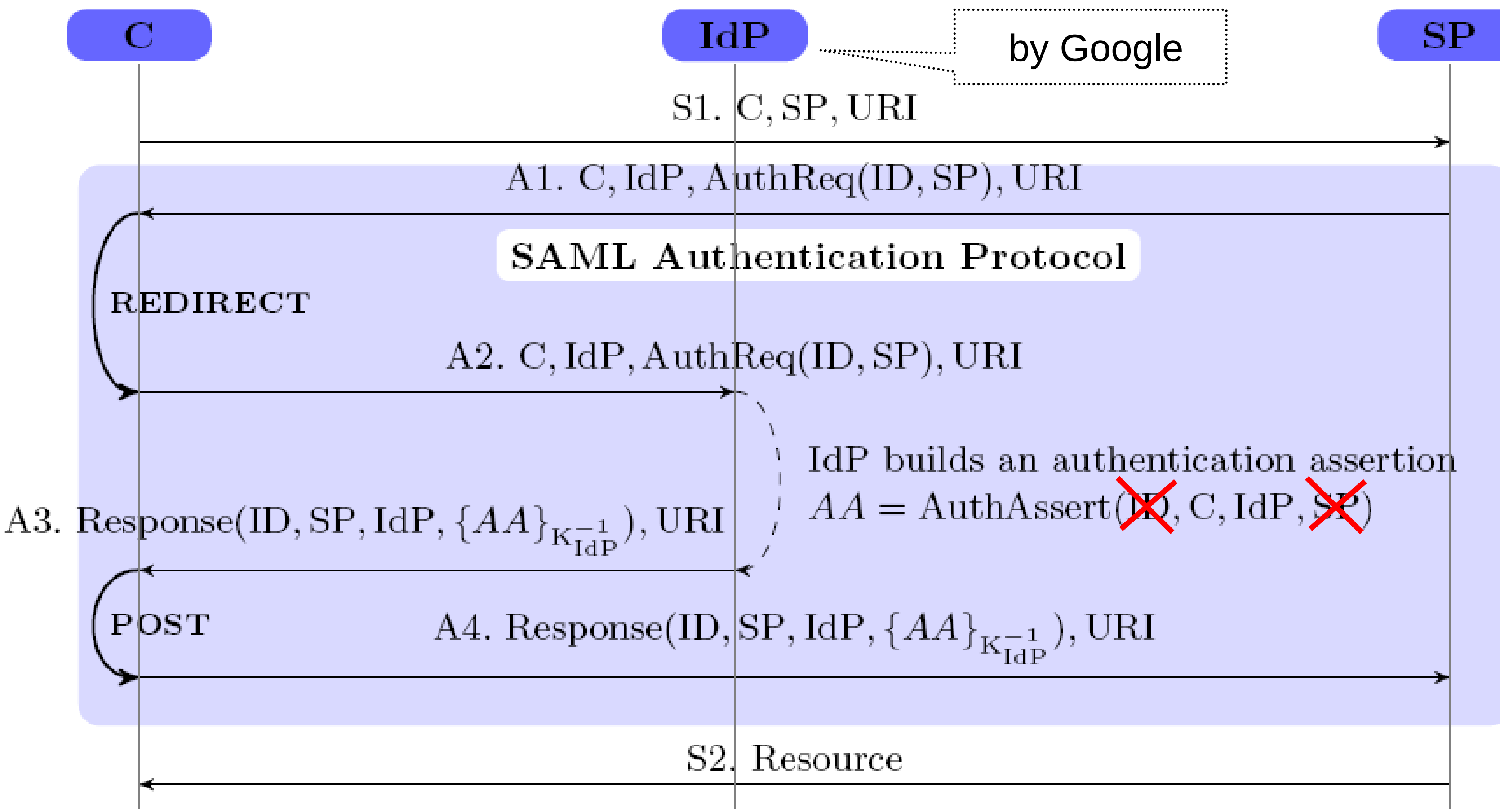


Fig. 1. SP-Initiated SSO with Redirect/POST Bindings

AVANTSSAR consortium

Industry

SAP Research France, Sophia Antipolis

Siemens Corporate Technology, München

IBM Zürich Research Labs (part time)

OpenTrust, Paris

Academia

Università di Verona

Università di Genova

ETH Zürich

INRIA Lorraine

UPS-IRIT Toulouse

IEAT Timisoara

Expertise

Service-oriented enterprise architectures

Security solutions

Standardization and industry migration

Security engineering

Formal methods

Automated security validation

SIEMENS

AVANTSSAR main objectives and principles

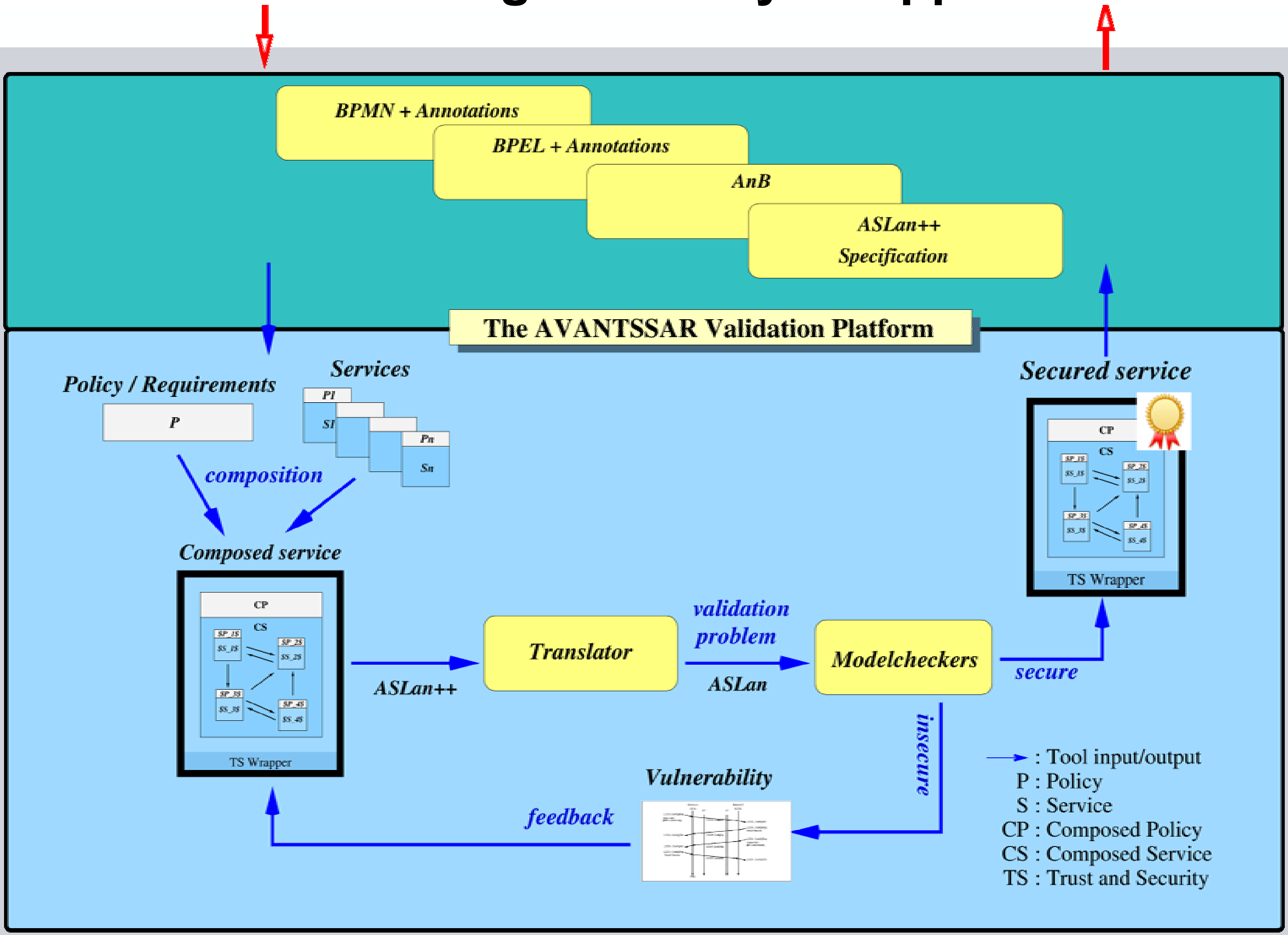
AVANTSSAR product: Platform for formal specification and automated validation of trust and security of SOAs

- 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

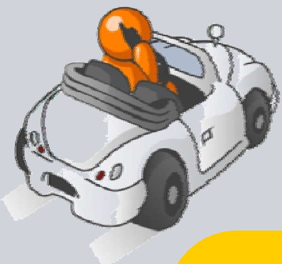
- Speed up development of new service infrastructures
- Enhance their security and robustness
- Increase public acceptance of SOA-based systems

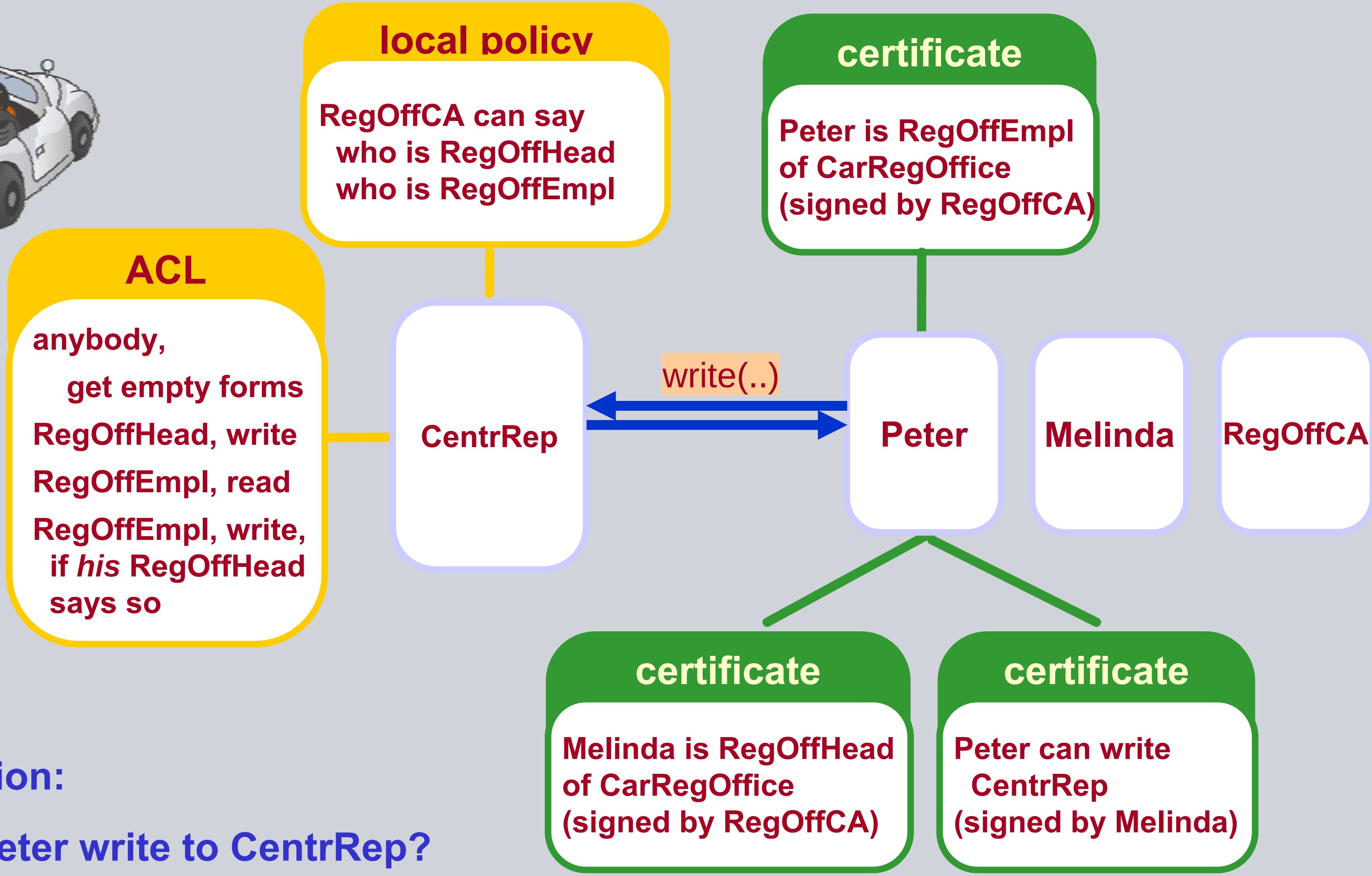
AVANTSSAR modeling and analysis approach



Example 2: Electronic Car Registration policies

SIEMENS





Question:

May Peter write to CentrRep?

On-the-fly inferences: 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. the 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)));
```

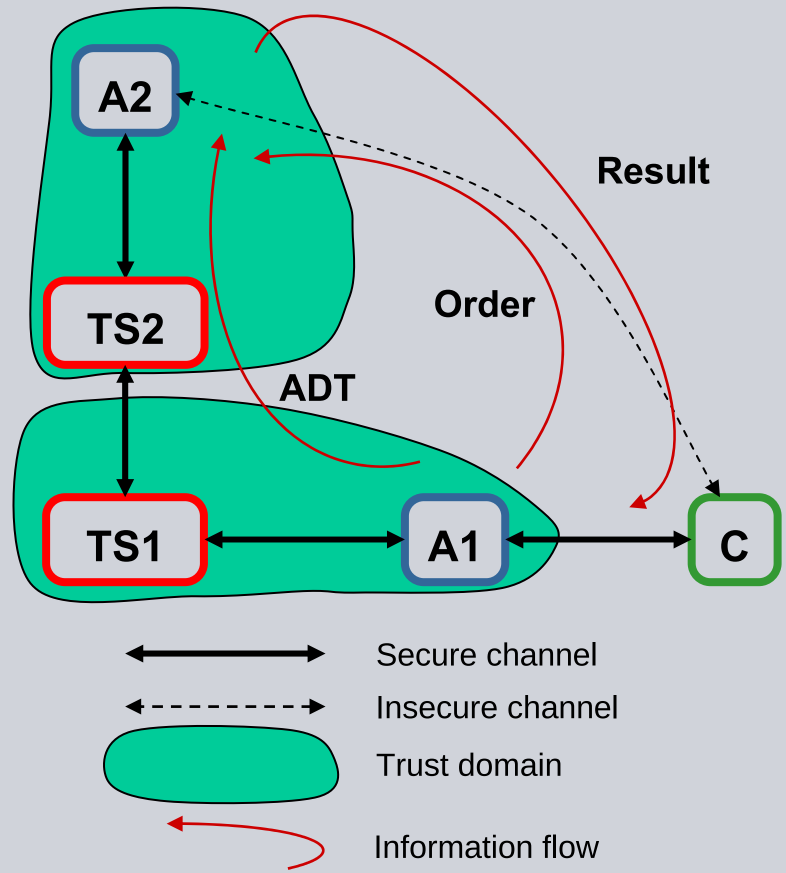

SIEMENS

Example 3: 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**, <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



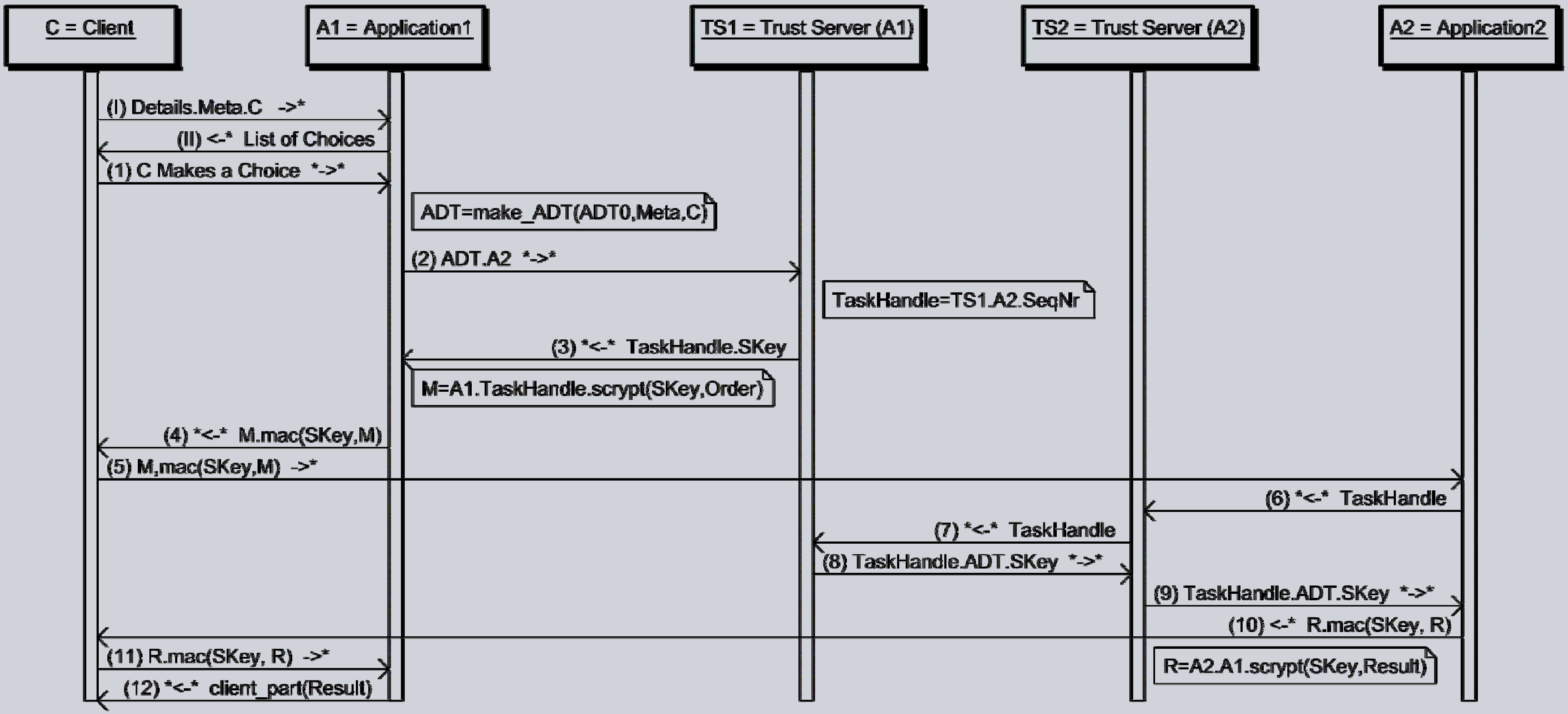
Security prerequisites:

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

Example 3: Message Sequence Chart of PTD

SIEMENS

..\..\deliv\5.2\figures\PTD_generic.pdf



SIEMENS

Example 3: 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 = scrypt(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. scrypt(SKey,Results);
 }}}
 goals
  authentic_C_A2_Details: C  *-> Actor: Details;
  secret_Order: secret (Order, {Actor, A1});
}
```


Optimization: Merging transitions on translation

A series of transmission and internal computation ASLan++ commands like

```
receive(A, ?M);
N := fresh();
send(A, N);
```

could bet translated into individual ASLan transitions like:

```
state_entity(Actor, IID, 1, dummy, dummy) . iknows(M) =>
state_entity(Actor, IID, 2, M    , dummy)

state_entity(Actor, IID, 2, M    , dummy) =[exists N]=>
state_entity(Actor, IID, 3, M    , N    )

state_entity(Actor, IID, 3, M    , N    ) =>
state_entity(Actor, IID, 4, M    , N    ) . network(N)
```

but can be \`compressed´ into a single atomic ASLan transition:

```
state_entity(Actor, IID, 1, dummy, dummy) . iknows(M) =[exists N]=>
state_entity(Actor, IID, 4, M    , N    ) . network(N)
```

Even internal computations containing loops etc. can be \`glued together´ to avoid interleaving. This dramatically reduces the search space because a lot of useless branching is avoided.

SIEMENS

Semantics of channel goals as LTL formulas

A channel goal requiring authentication, directedness, freshness, and confidentiality:

```
secure_Alice_Payload_Bob: A *->>* B: Payload;
```

On the sender side: `Actor -> B: ...Payload...;`

```
witness(Actor,B,auth_Alice_Payload_Bob,Payload);
secret(Payload,secr_Alice_Payload_Bob,{Actor,B});
```

On the receiver side: `A -> Actor: ...?Payload...;`

```
request(Actor,A,auth_Alice_Payload_Bob,Payload,IID);
secret(Payload,secr_Alice_Payload_Bob,{A,Actor});
```

Semantics of the **authentication** and **directedness** part:

```
forall A,B,P,M,IID. [] (request(B,A,P,M,IID) =>
 (<-> (witness(A,B,P,M)) || (dishonest(A) & iknows(M))))
```

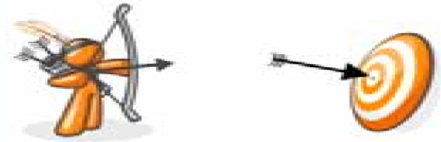
Semantics of the **freshness** (replay protection) part:

```
forall A,B,P,M,IID IID'. [] (request(B,A,P,M,IID) =>
 (!(<-> (request(B,A,P,M,IID') & !(IID=IID'))  ||  dishonest(A)))
```

Semantics of the **confidentiality** part:

```
forall M,P,As. [] ((secret(M,P,As) & iknows(M)) => contains(i, As))
```


AVANTSSAR: current status



SIEMENS

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

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

WP4: First prototype of the **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

SIEMENS

AVANTSSAR conclusion and industry migration

Contemporary SOA has complex structure and security requirements including dynamic trust relations and application-specific policies.

On integration of the AVANTSSAR Platform in industrial development, a rigorous demonstration that the security requirements are fulfilled will:

- assist developers with security architecture, analysis and certification
- increase customers' confidence in modern service-oriented architectures

The AVANTSSAR Platform will advance the security of industrial vendors' service offerings: validated, provable, traceable.

AVANTSSAR will thus strengthen the competitive advantage of the products of the industrial partners.