

Tools for Secure Systems Development with UML: Security Analysis with ATPs

Jan Jürjens and Pasha Shabalin

Software & Systems Engineering
TU Munich, Germany



juerjens@in.tum.de

<http://www.umlsec.org>



Security Analysis with UMLsec

Analysis wrt. security requirements in class diagram: Verify sequence / statechart diagrams against attacker model from threat scenarios in deployment diagrams who (Dolev, Yao 1982):

- may **participate** in some protocol runs,
- **knows** some data in advance,
- may **intercept** messages on some links,
- **injects** messages that it can produce in some links
- may access certain nodes.

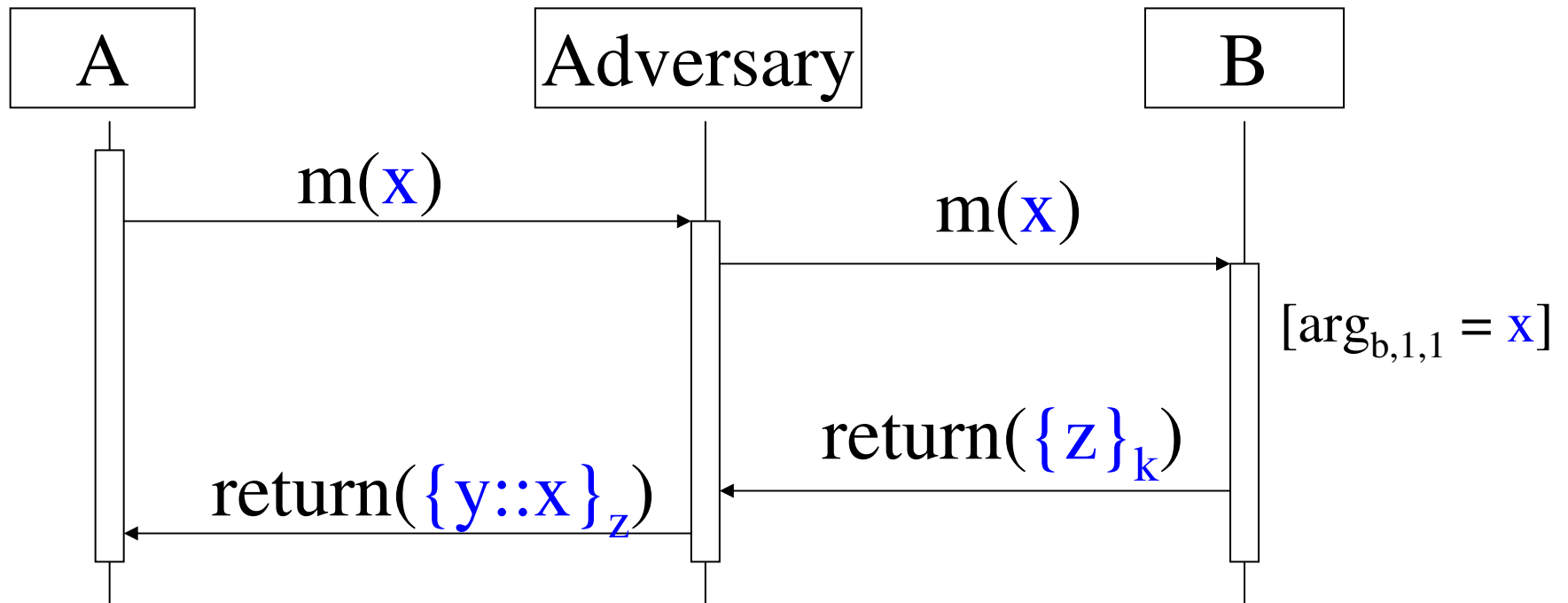
Symbolic Crypto

Exp: term algebra generated by $\text{Var} \cup \text{Keys} \cup \text{Data}$ and

- $_ \ :: _$ (concatenation) and empty expression \mathcal{E} ,
- $\{ _ \} _$ (encryption)
- $Dec ()$ (decryption)
- $Sign ()$ (signing)
- $Ext_ ()$ (extracting from signature)
- $Hash(_)$ (hashing)

by factoring out the equations $Dec_{K^{-1}}(\{E\}_k) = E$ and $Ext_K(Sign_{K^{-1}}(E)) = E$ (for $K \in \text{Keys}$).

Adversary: Simulation



Adversary
knowledge:

k^{-1}, y, x
 $\{z\}_k, z$

- $\forall e, k. Dec_{k^{-1}}(\{e\}_k) = e$

Security Analysis in First-order Logic

Idea: **approximate** set of possible **data values** flowing through system **from above**.

Predicate *knows*(E) meaning that the adversary may get to know E during the execution of the protocol.

For any secret s , check whether can derive *knows*(s) using automated theorem prover.

First-order Logic: Basic Rules

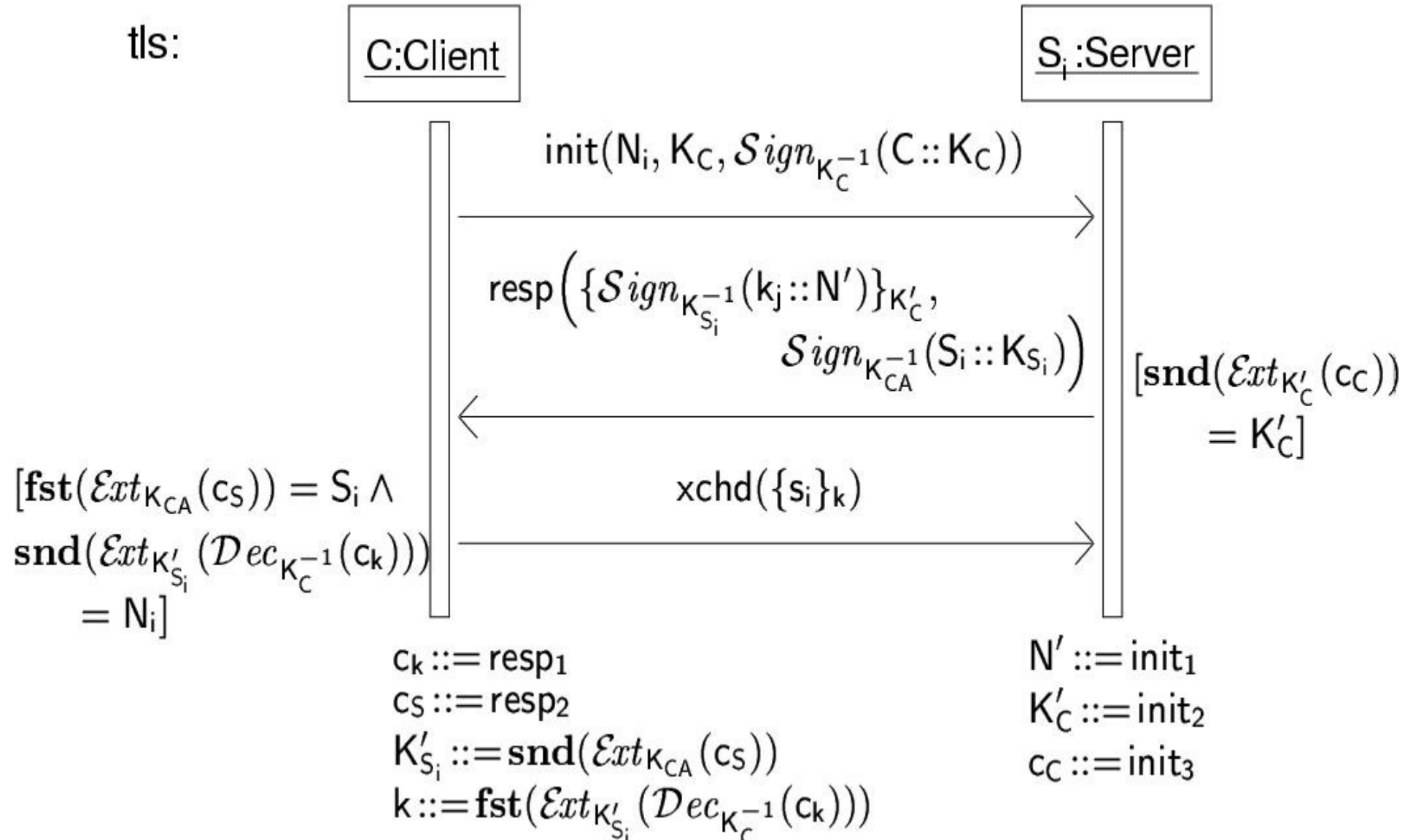
For initial adversary knowledge (K^0): Define $knows(E)$ for any E initially known to the adversary (protocol-specific, e.g. K_A, K_A^{-1}). Define above equations.

For evolving knowledge (K^n) define

$$\forall E_1, E_2. (knows(E_1) \wedge knows(E_2) \Rightarrow \\ knows(E_1 :: E_2) \wedge knows(\{E_1\}_{E_2}) \wedge \\ knows(Dec_{E_2}(E_1)) \wedge knows(Sign_{E_2}(E_1)) \wedge \\ knows(Ext_{E_2}(E_1)))$$

$$\forall E. (knows(E) \Rightarrow \\ knows(head(E)) \wedge knows(tail(E)))$$

Given Sequence Diagram ...



... Translate to 1st Order Logic

Connection (or statechart transition)

$TR1 = (in(msg_in), cond(msg_in), out(msg_out))$

followed by $TR2$ gives predicate $PRED(TR1) =$

$$\begin{aligned} & \forall msg_in. [knows(msg_in) \wedge cond(msg_in) \\ & \quad \Rightarrow knows(msg_out) \\ & \quad \wedge PRED(TR2)] \end{aligned}$$

(Assume: order enforced (!).)

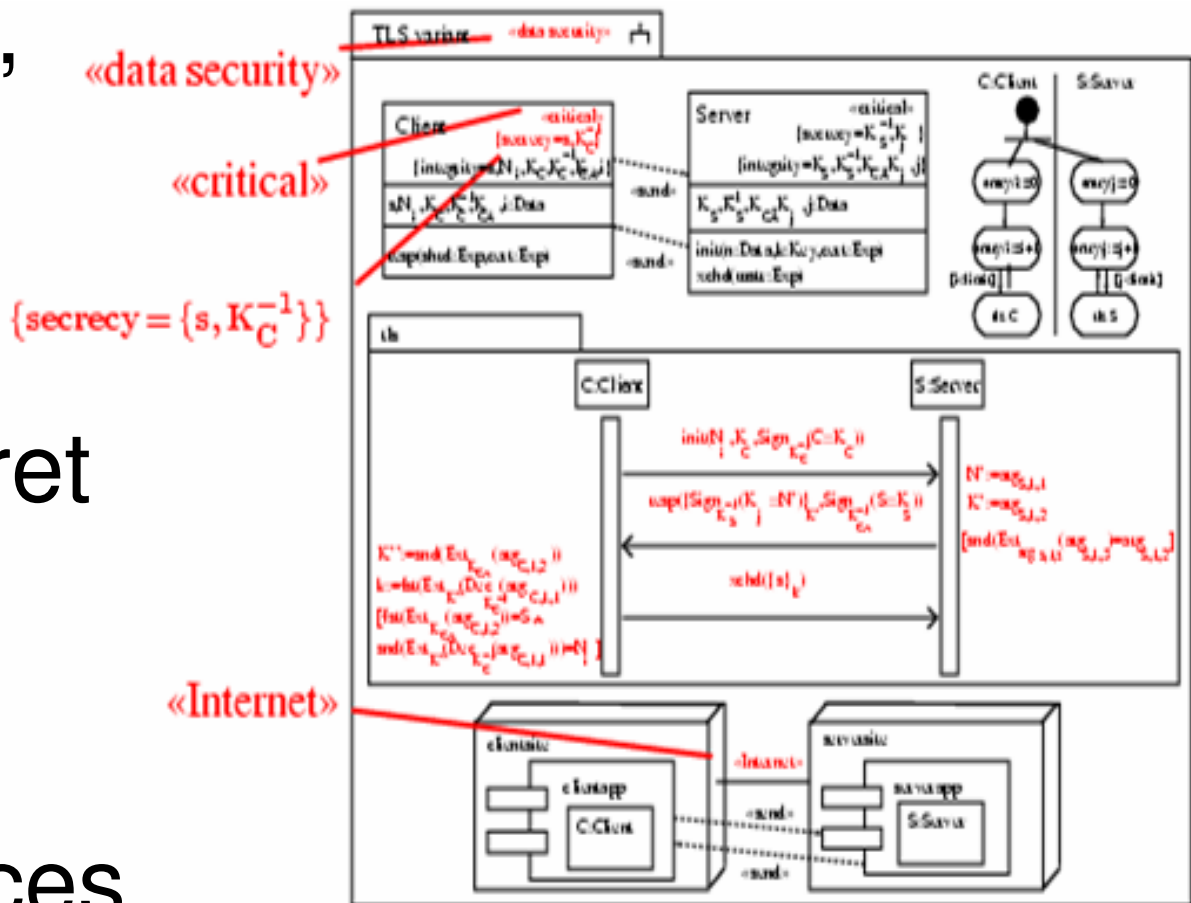
Can include senders, receivers in messages.

Abstraction: find all attacks, may have false positives.

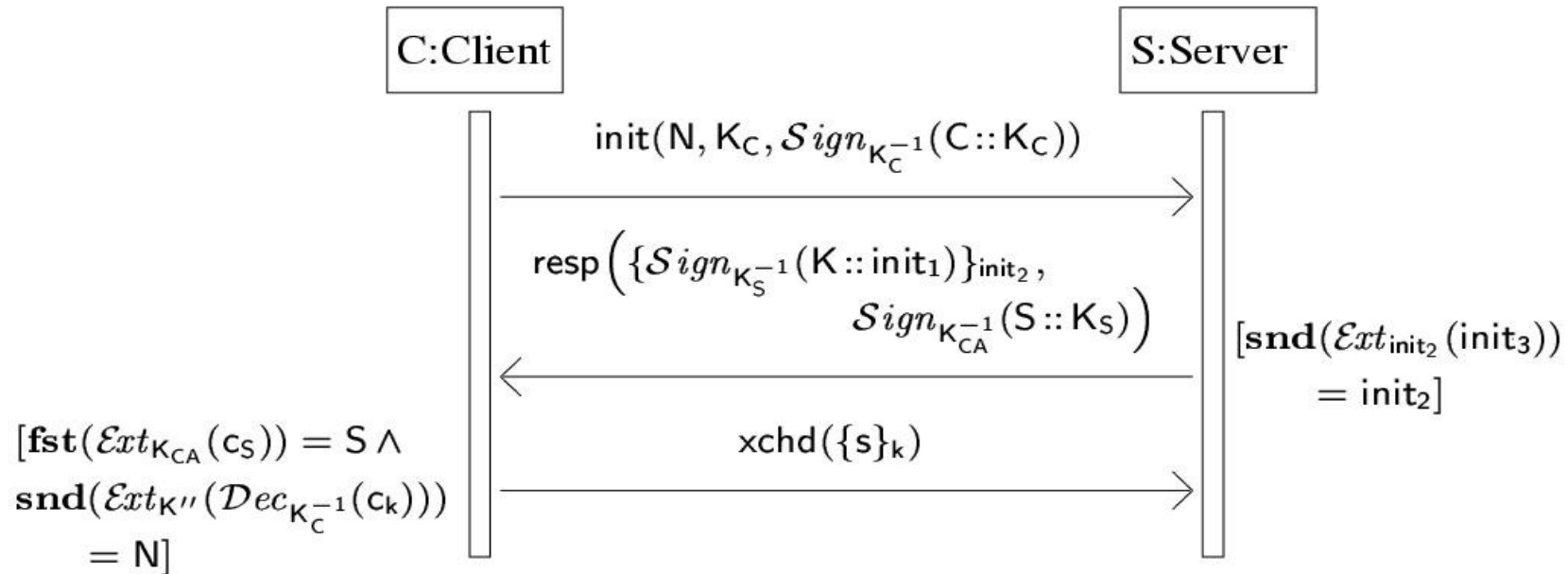
Example: Proposed Variant of TLS (SSL)

Apostolopoulos,
Peris, Saha;
IEEE Infocom
1999.

Goal: send secret
protected by
session key
using fewer
server resources.

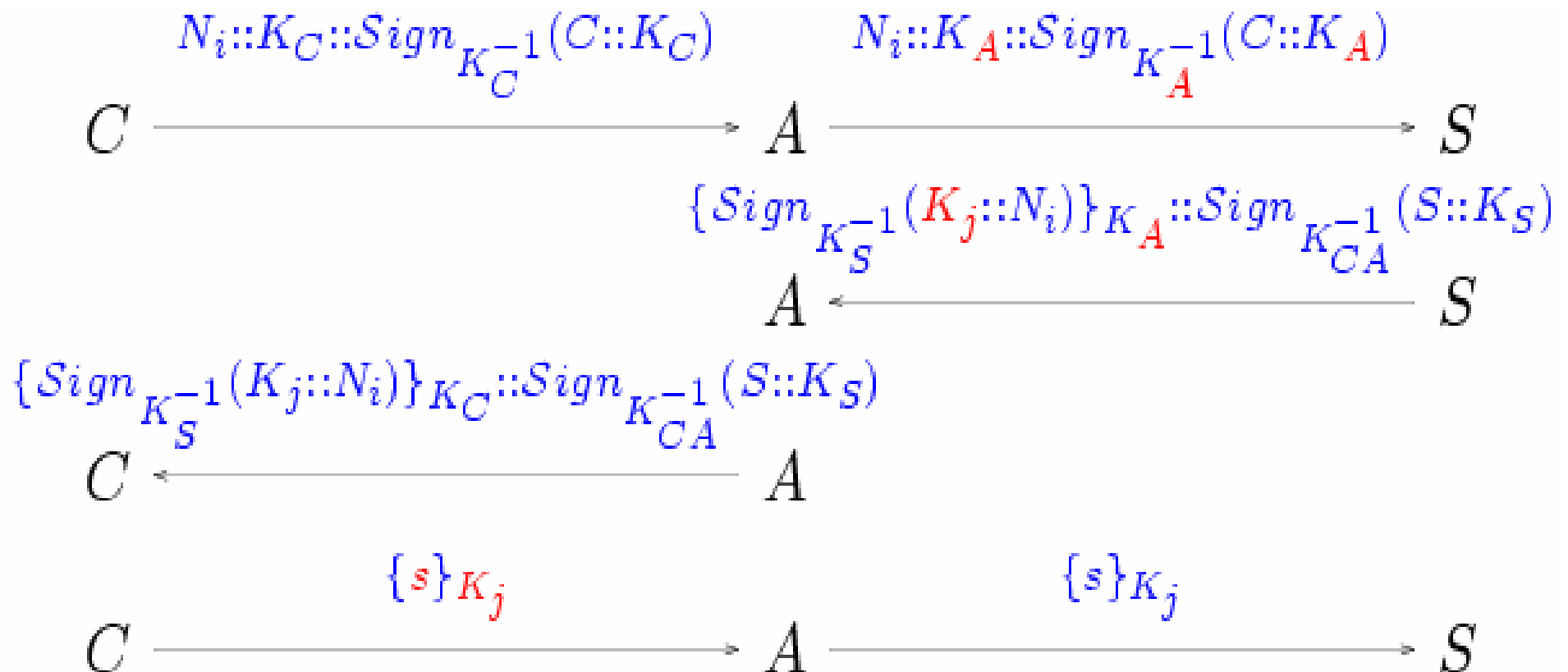


Example: Translation to Logic



$\text{knows}(N) \wedge \text{knows}(K_C) \wedge \text{knows}(\text{Sign}_{K_C^{-1}}(C::K_C))$
 $\wedge \forall \text{init}_1, \text{init}_2, \text{init}_3. [\text{knows}(\text{init}_1) \wedge \text{knows}(\text{init}_2) \wedge$
 $\text{knows}(\text{init}_3) \wedge \text{snd}(\text{Ext}_{\text{init}_2}(\text{init}_3)) = \text{init}_2$
 $\Rightarrow \text{knows}(\{\text{Sign}_{K_S^{-1}}(\dots)\}_{\dots}) \wedge [\dots] \wedge [\dots \Rightarrow \dots] \dots]$

Man-in-the-Middle Attack



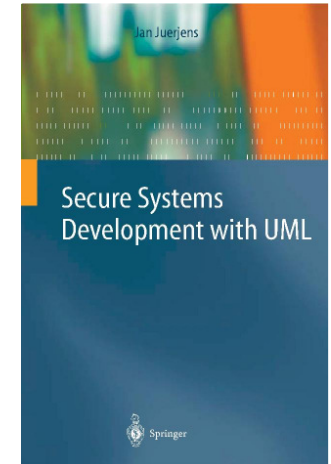
Formal semantics for UML fragment

Diagrams in **context** (using subsystems).
Model **actions** and internal **activities** explicitly.

Message exchange between objects or components (incl. event dispatching).

For UMLsec: include **adversary model** arising from threat scenario in deployment diagram.

Currently updated to UML 2.0 with Selic (IBM Rational), Broy, Cengarle (TUM), Rumpe (TU-BS)



Tool-support: Pragmatics

Commercial modelling tools: so far mainly **syntactic** checks and **code-generation**.

Goal: sophisticated analysis. Solution:

- Draw UML models with editor.
- Save UML models as **XMI** (XML dialect).
- Connect to **verification** tools (automated theorem prover, model-checker ...), e.g. using XMI Data Binding.

Tool-support: Tool Binding

Several possibilities:

- General purpose language with integrated XML parser (Perl, ...)
- Special purpose XML parsing language (XSLT, ...)
- Data Binding (Castor; XMI: e.g. MDR)

MDR (netbeans): XMI file into Java Objects, following UML meta-model

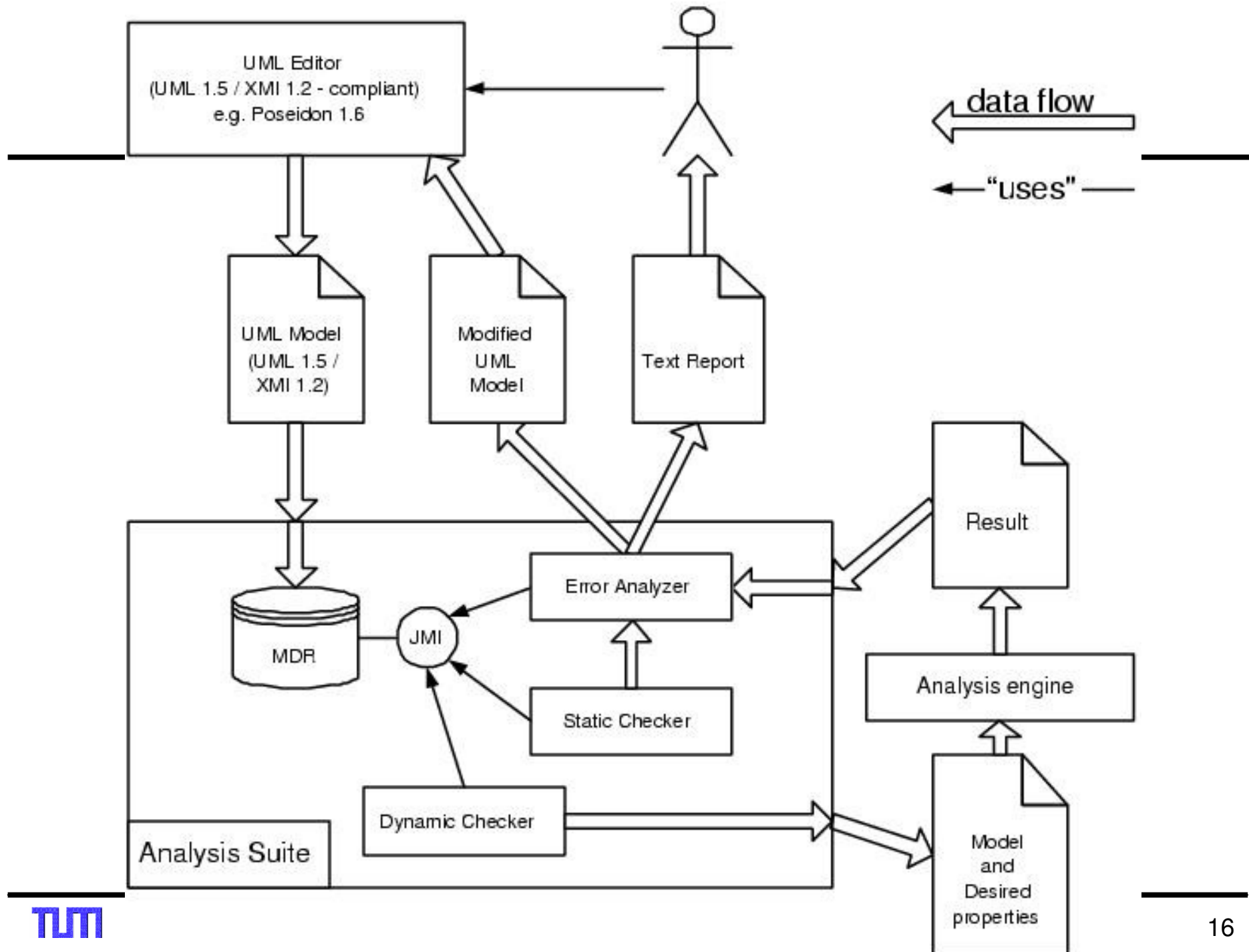
CSDUML Framework

Framework for analysis plug-ins to access UML models on conceptual level over various UI's. Exposes a set of commands. Has internal state (preserved between command calls).

Framework and analysis tools accessible and available at <http://www.umlsec.org> .

Upload UML model (as .xmi file) on website.

Analyse model for included critical requirements. Download report and UML model with highlighted weaknesses.



Applications

- Biometric Access Control System (T-Systems; BMBF-Project Verisoft): three attacks found + corrected so far (ICSE 05)
- Common Electronic Purse Specifications (BMW Project Fairpay): attacks in purchase and load protocols (HASE 04)
- Attack in TLS variant proposed at IEEE Infocom `99
- Multi-layer security protocol for HypoVereinsbank web application, telematic automobile emergency application with BMW, Electronic signature architecture with Allianz, Electronic purse for Oktoberfest

Conclusions

Model-based Security Engineering using UMLsec:

- formally based approach
- industrially used notation
- automated tool support
- successful industrial applications
- integrated approach (source-code, configuration data)

Resources

Jan Jürjens, Secure Systems Development with UML, Springer 2004

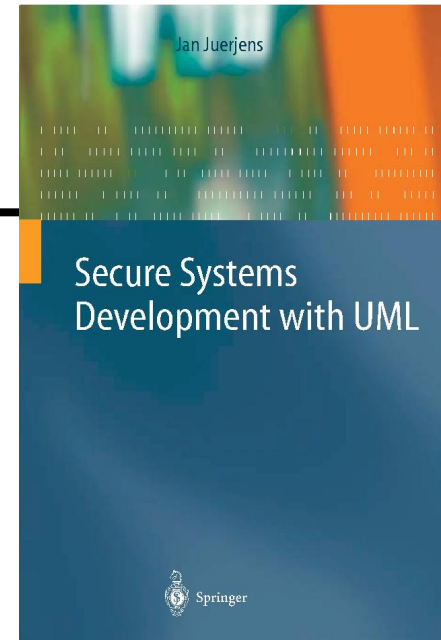
Spring School: May 2005, Carlos III Univ. Madrid

Workshops: FoMSESS Jahrestreffen (Kiel, Juni); CSDUML@SAFECOMP05 (Norwegen, Sept.), WITS06@ETAPS06 (April, Vienna)

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

<http://www.umlsec.org>

(user Participant, password Iwasthere)



Backup

Adversaries

Model classes of **adversaries**.

May **attack** different parts of the system according to threat scenarios.

Example: **insider** attacker may intercept communication links in LAN.

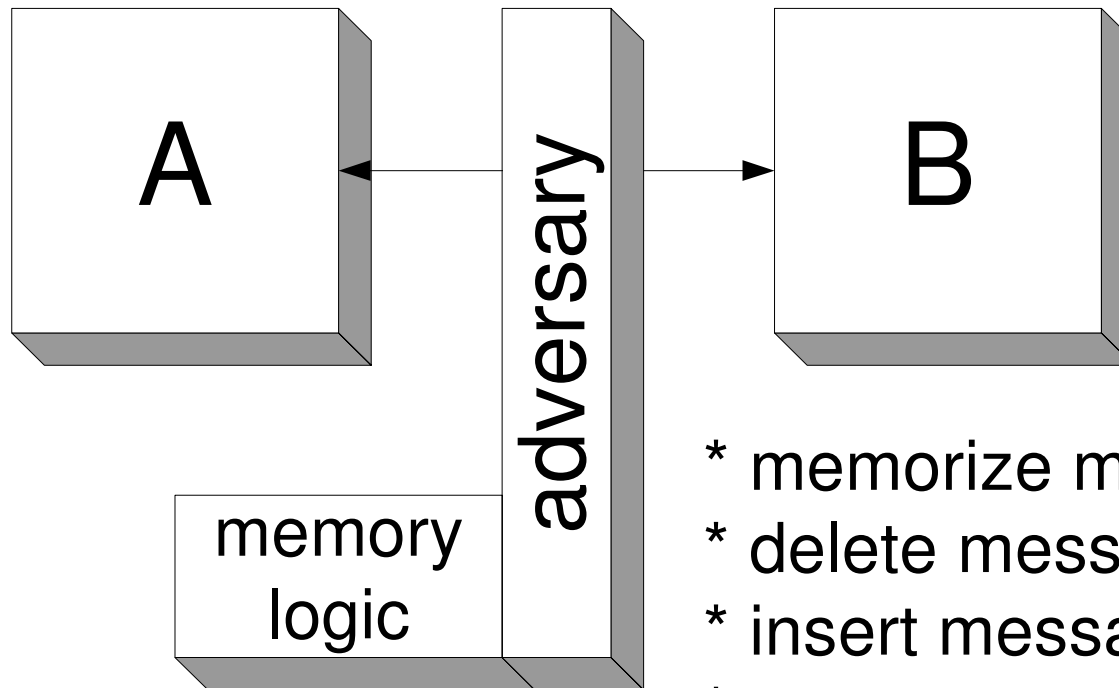
To evaluate security of specification, simulate jointly with adversary model.

Cryptography

Keys are **symbols**, crypto-algorithms are **abstract** operations.

- Can only decrypt with **right** keys.
- Can only compose with **available** messages.
- Cannot perform **statistical** attacks.

Adversary Model



- * memorize message
- * delete message
- * insert message
- * compose own message
- * use cryptographic primitives

Adversary Knowledge

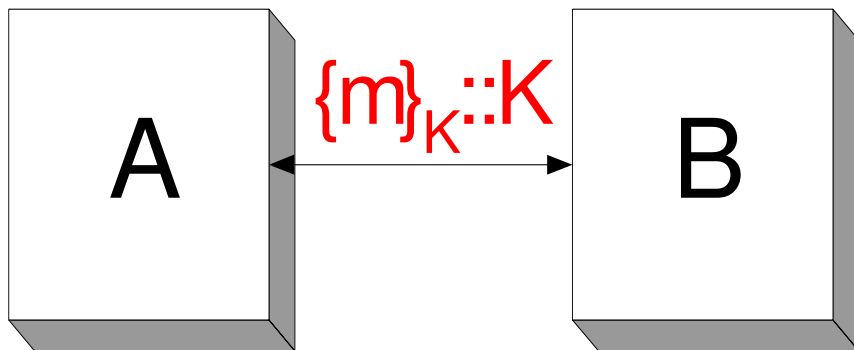
Specify set K_A^0 of **initial knowledge** of an adversary of type A . Let K_A^{n+1} be the **Exp**-subalgebra generated by K_A^n and the expressions received after $n+1$ st iteration of the protocol.

Definition (Dolev, Yao 1982).

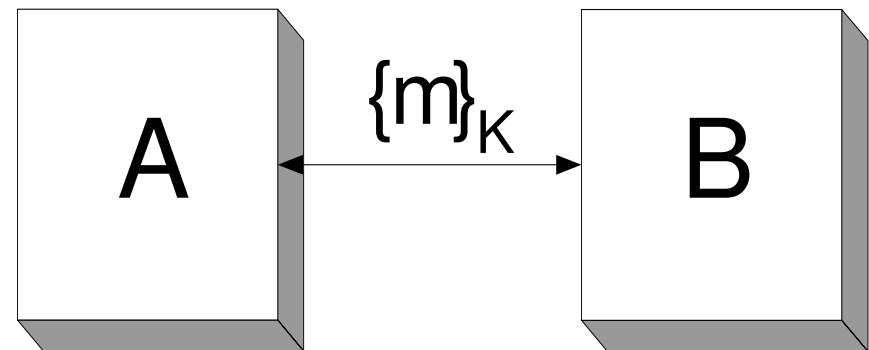
S keeps secrecy of M against attackers of type A if there is no n with $M \in K_A^n$.

Example: Secrecy

Against attacker who can read messages:

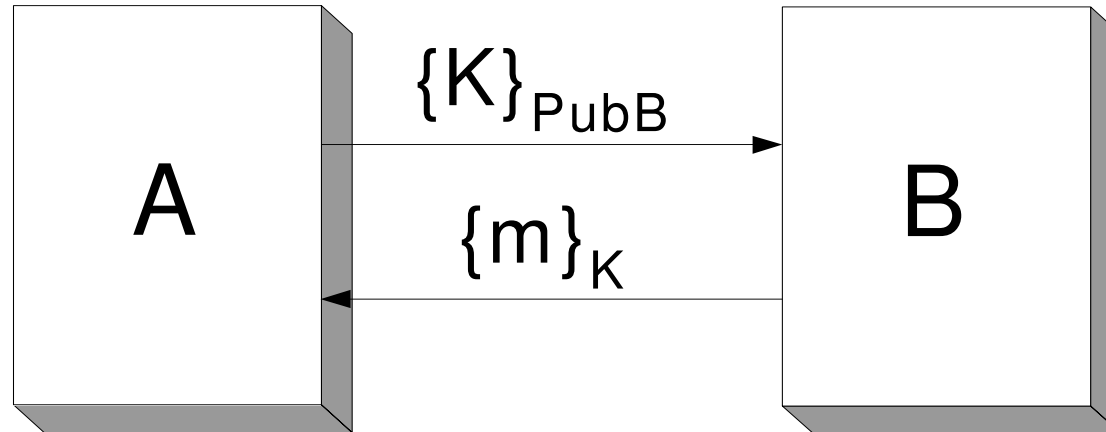


Secrecy of m , K not preserved.



Secrecy of m , K preserved.

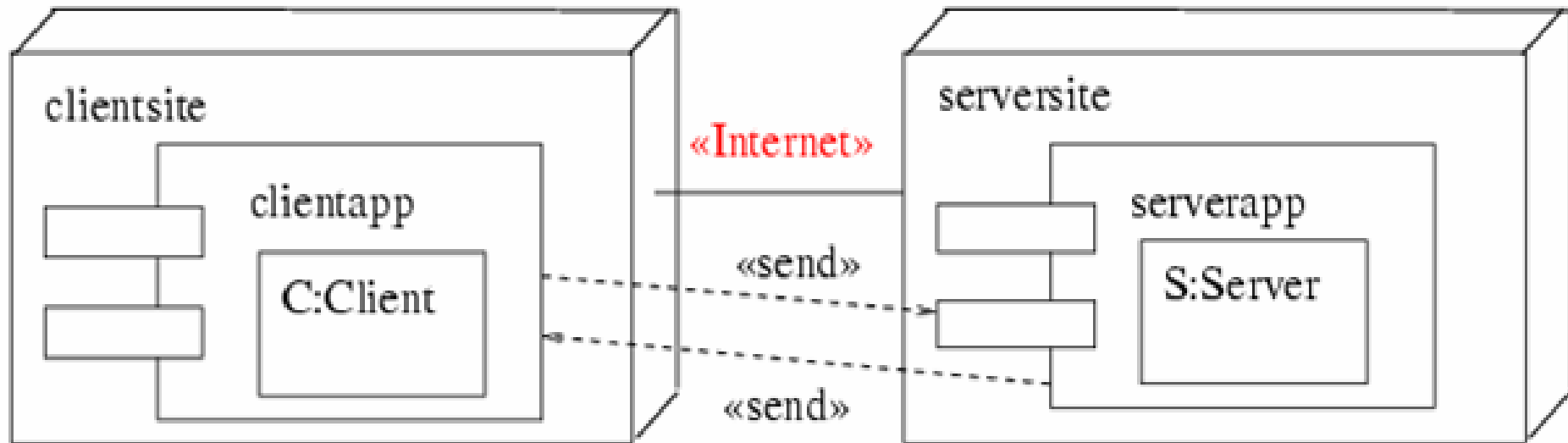
Example: Hybrid Scheme



Secrecy of m **not preserved** against attacker who can **delete** and **insert** messages.

Security of m **preserved** against attacker who can **listen**, but not alter messages.

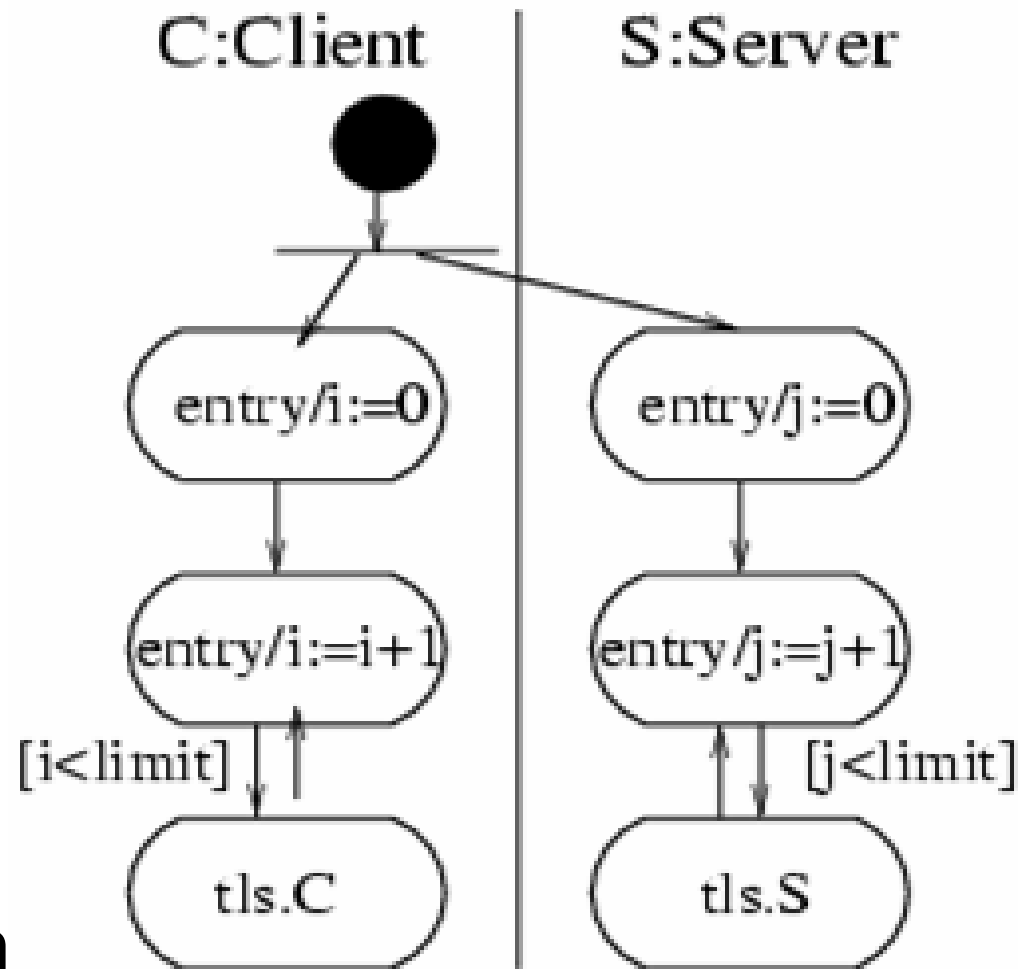
... and Physical Layer Model ...



Deployment diagram.

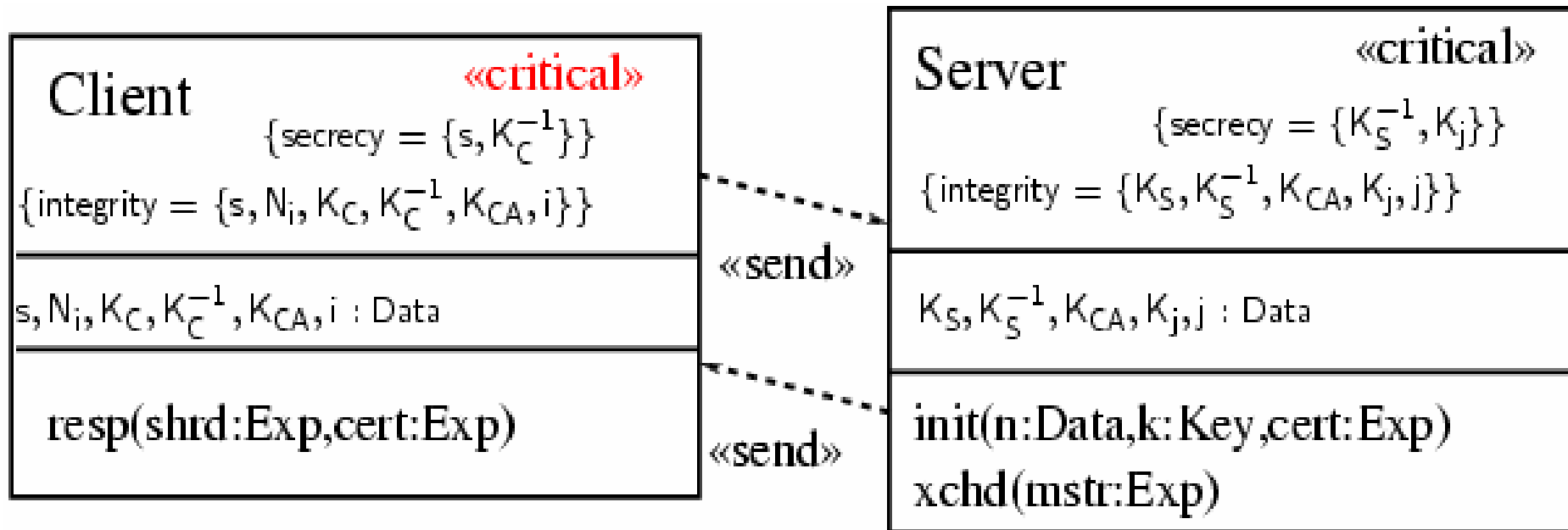
Derived adversary model: **read**, **delete**, **insert** data.

Execute in System Context



Activity diagram

Formulate Data Security Requirements



Class diagram.

Gives conjecture: *knows(s)* derivable ?

TLS Variant in TPTP Notation I

```
input_formula(tls_abstract_protocol, axiom, (
  ![ArgS_11, ArgS_12, ArgS_13, ArgC_11, ArgC_12] : (
    ![DataC_KK, DataC_k, DataC_n] : (
      % Client -> Attacker (1. message)
      (
        knows(n)
        & knows(k_c)
        & knows(sign(conc(c, k_c), inv(k_c) ) ) )
    & % Server -> Attacker (2. message)
    (
      (
        knows(ArgS_11)
        & knows(ArgS_12)
        & knows(ArgS_13)
        & ( ? [X] : equal( sign(conc(X, ArgS_12), inv(ArgS_12) ),
                          ArgS_13 ) ) )
    => (
      knows(enc(sign(conc(kgen(ArgS_12), ArgS_11), inv(k_s) ),
                ArgS_12 ) )
      & knows(sign(conc(s, k_s), inv(k_ca) ) ) ) ) )
```

TLS Variant in TPTP Notation II

```
& % Client -> Attacker (3. message)
  ( ( knows(ArgC_11)
    & knows(ArgC_12)
    & equal(sign(conc(s, DataC_KK), inv(k_ca)), ArgC_12 )
    & equal(enc(sign(conc(DataC_k, DataC_n), inv(DataC_KK) ),
                k_c), ArgC_11 )
    & ( ? [DataC_ks] : equal(sign(conc(s, DataC_ks), inv(k_ca) ),
                            ArgC_12 ) )
    & equal(enc(sign(conc(DataC_k, n), inv(DataC_KK) ), k_c),
            ArgC_11 )
    & equal(enc(sign(conc(DataC_k, DataC_n), inv(DataC_KK) ), k_c),
            ArgC_11 )
  )
=> ( knows(symenc(secret, DataC_k)) ) )
) ) ).
```

Surprise ...

E-SETHEO csp03 single processor running on host ...
(c) 2003 Max-Planck-Institut fuer Informatik and
Technische Universitaet Muenchen

```
tlsvariant-freshkey-check.tptp
...
time limit information: 300 total (entering statistics module).
problem analysis ...
testing if first-order ...
first-order problem
...
statistics: 19 0 7 46 3 6 2 0 1 2 14 8 0 2 28 6
...
schedule selection: problem is horn with equality (class he).
schedule:605 3 300 597
...
entering next strategy 605 with resource 3 seconds.
...
analyzing results ...
proof found
time limit information: 298 total / 297 strategy (leaving wrapper).
...
e-SETHEO done. exiting
```

Attack

... Which Means:

Can derive *knows(s)* (!).

That is: Protocol does **not** preserve secrecy of *s* against adversaries.

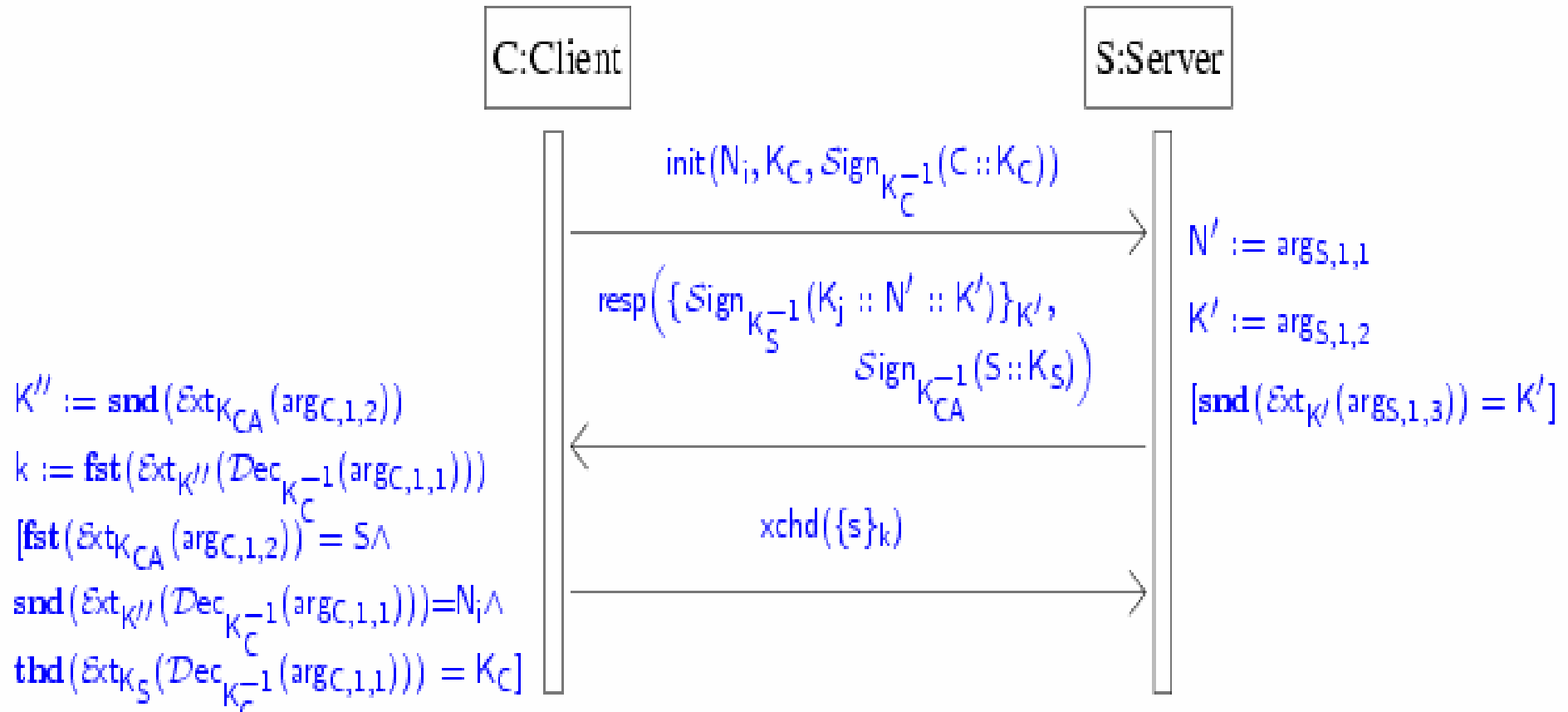
→ Completely insecure wrt stated goals.

But why ?

Could look at proof tree.

Or: use prolog-based attack generator.

The Fix



e-Setheo: *knows(s)* not derivable. Thus secure.

Execution Semantics

Behavioral interpretation of a UML subsystem:

- (1) Takes **input** events.
- (2) Events distributed from **input** and **link** queues between subcomponents to intended **recipients** where they are processed.
- (3) Output distributed to **link** or **output** queues.
- (4) Apply **adversary** / **failure model**.

Data-binding with MDR

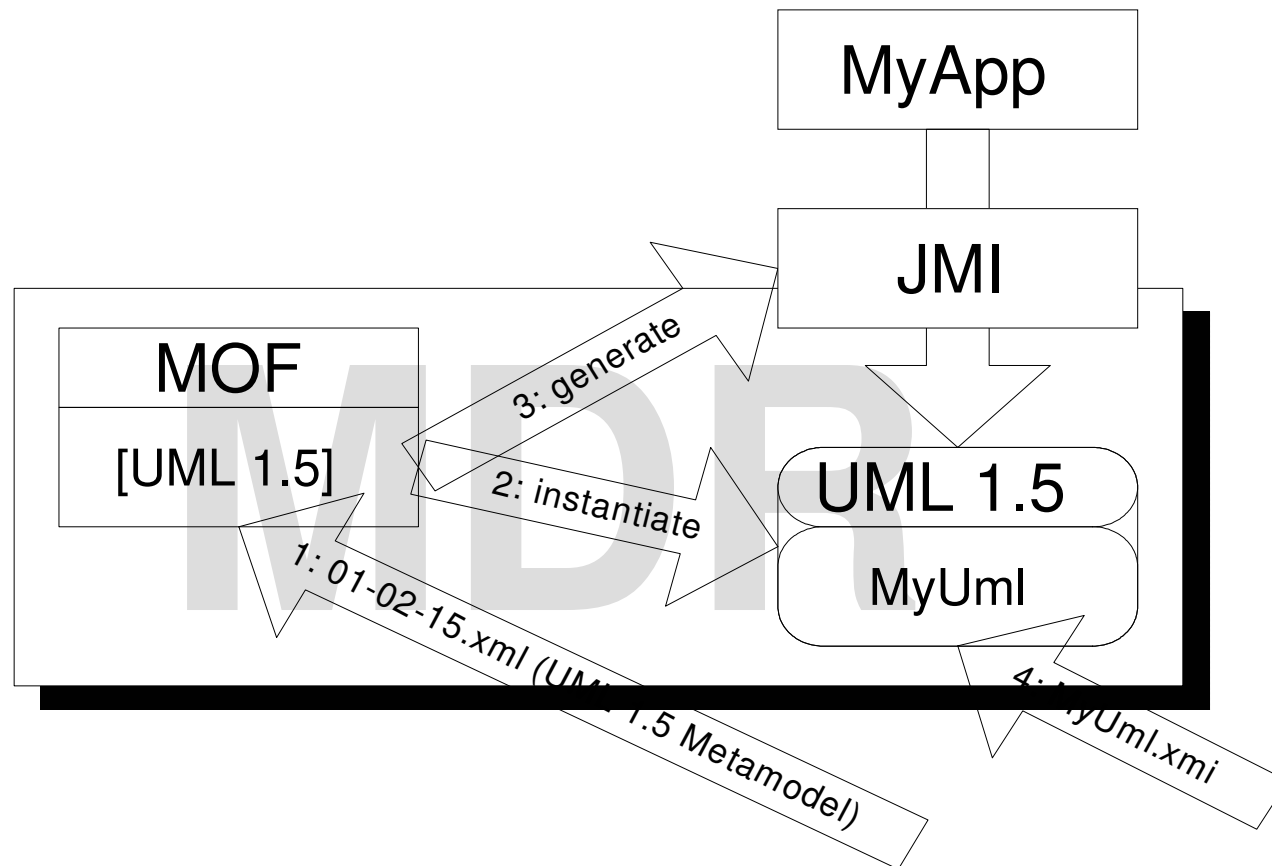
MDR: MetaData Repository,
Netbeans library (www.netbeans.org)

Extracts data from XMI file into Java
Objects, following UML meta-model.

Access data via methods.

Advantage: No need to worry about XML.

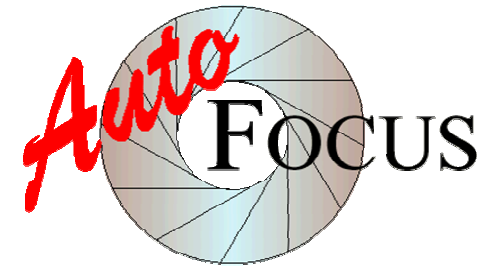
UML Processing



Connection with Analysis Tool

Industrial CASE tool with UML-like notation:

AUTOFOCUS (<http://autofocus.informatik.tu-muenchen.de>)



- Simulation
- Validation (Consistency, Testing, Model Checking)
- Code Generation (e.g. Java, C, Ada)
- Connection to Matlab

Connect UML tool to underlying analysis engine.

GUI

