LEHRSTUHL 14
SOFTWARE ENGINEERING

Willkommen zur Vorlesung

Methodische Grundlagen des Software-Engineering

im Sommersemester 2011

Prof. Dr. Jan Jürjens

TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

22. Kryptographische Protokolle: Sicherheitsanalyse

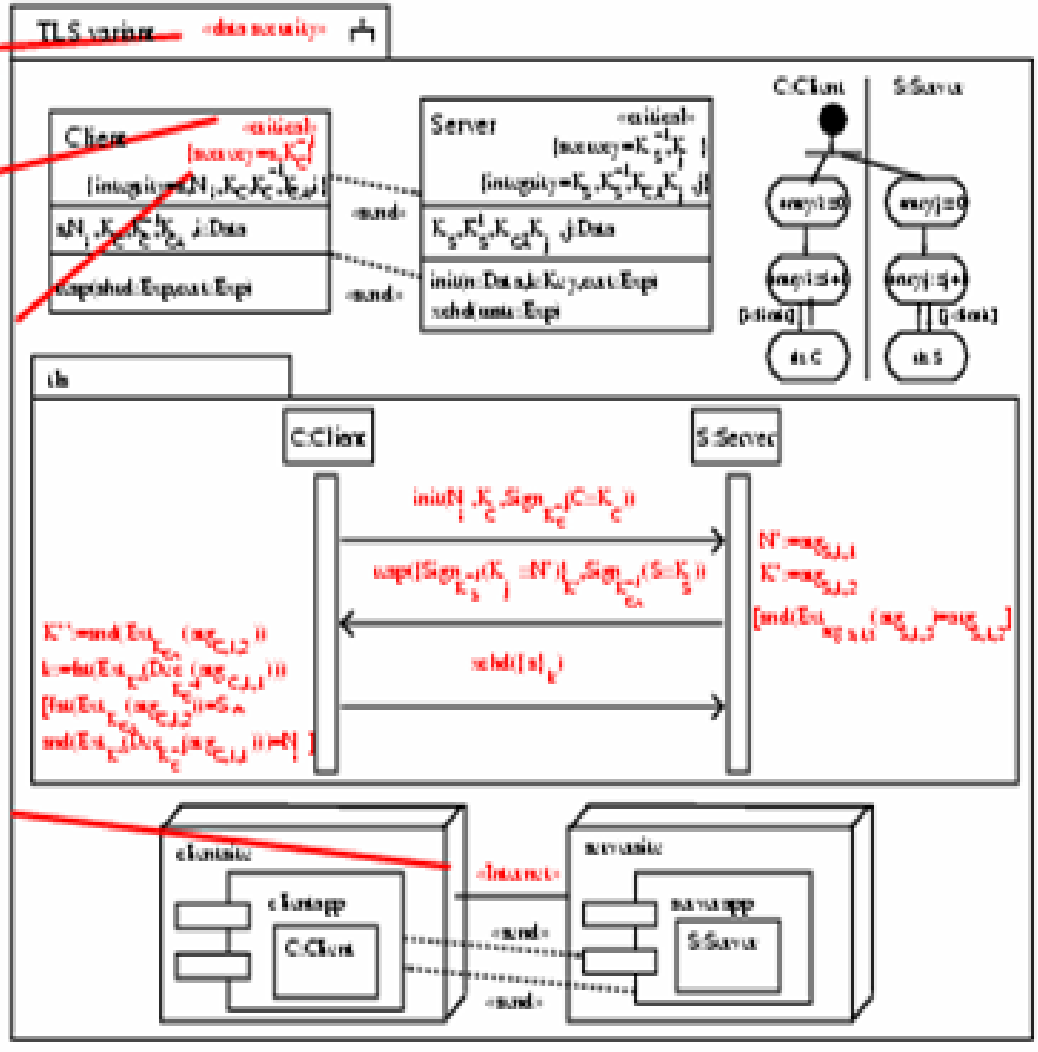
Einordnung
Protokolle mit UMLsec

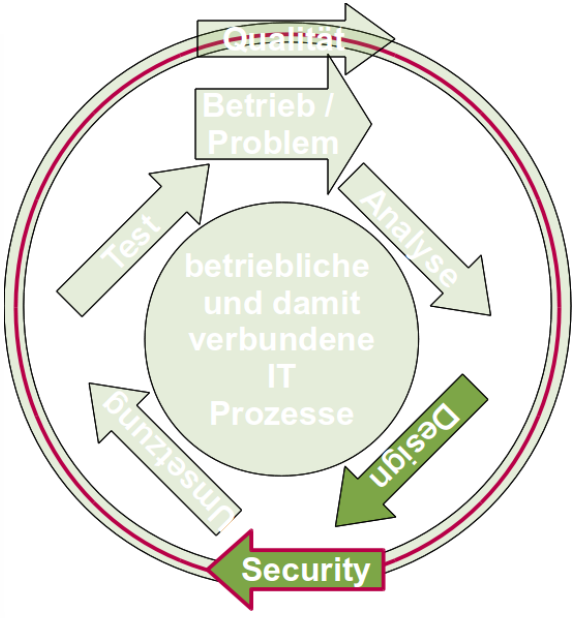
«data security»

«critical»

$\{secrecy = \{s, K_C^{-1}\}\}$

«Internet»





Einordnung
Einführung UML / UMLsec

- Business Prozesse
- Qualitätsmanagement
- Testen
- Sicherheit
- Sicheres Software Design
  - Einführung UML / UMLsec
  - Kryptographische Protokolle
  - Biometrische Authentifizierung
  - Elektronische Geldbörse
  - Weitere Anwendungsbeispiele

Angreifer-Wissen

Gegeben sei die Menge K_A^0 als initiales Wissen eines Angreifers des Typs *A*. Sei K_A^{n+1} die Menge der Krypto-Terme, die aus der Menge K_A^n von Krypto-Termen und aus den Ausdrücken, die nach der *n+1*sten Iteration des Protokolls empfangen wurden, durch Anwendung der Krypto-Operationen gebildet werden kann.

Definition (Dolev, Yao 1982).

Ein Datenwert *M* bleibt vertraulich gegen einen Angreifer des Typs *A*, wenn es kein *n* mit $M \in K_A^n$ gibt.

Sicherheitsanalyse in Logik erster Stufe

Idee: Menge der Datenwerte, die der Angreifer möglicherweise kennenlernen kann, von oben approximieren.

Das logische Prädikat *knows(E)* bedeutet, dass der Angreifer den Ausdruck *E* während der Ausführung des Protokolls möglicherweise kennenlernen kann.

Für jedes Geheimnis *s* kann man dann überprüfen, ob *knows(s)* aus den logischen Formeln, die das Systemverhalten repräsentieren, abgeleitet werden kann.

Sicherheitsanalyse: Grundlegende Regeln

Für initiales Angreiferwissen (K^0): Definiere Axiom $knows(E)$ für jeden Datenwert E, der dem Angreifer initial bekannt ist (protokoll-spezifisch, z.B. K_A , K_A^{-1}).

Modellieren, dass der Angreifer selber Krypto-Algorithmen einsetzen kann: Definiere Axiome:

$$\forall E_1,E_2.(knows(E_1) \wedge knows(E_2) \Rightarrow knows(E_1::E_2) \wedge knows(\{E_1\}_{E2}) \wedge$$

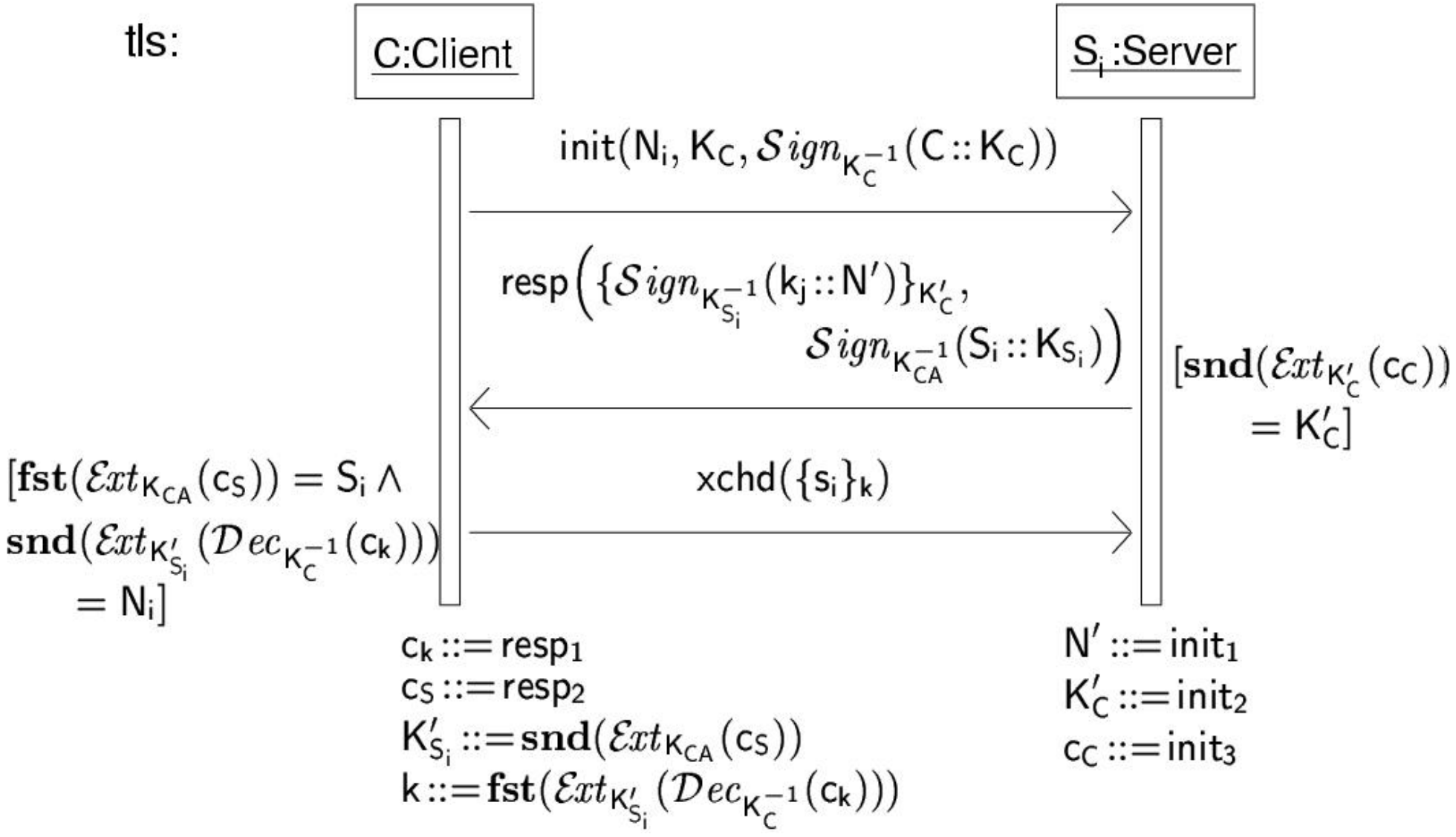
$$knows(Dec_{E2}(E_1)) \wedge knows(Sign_{E2}(E_1)) \wedge$$

$$knows(Ext_{E2}(E_1)))$$

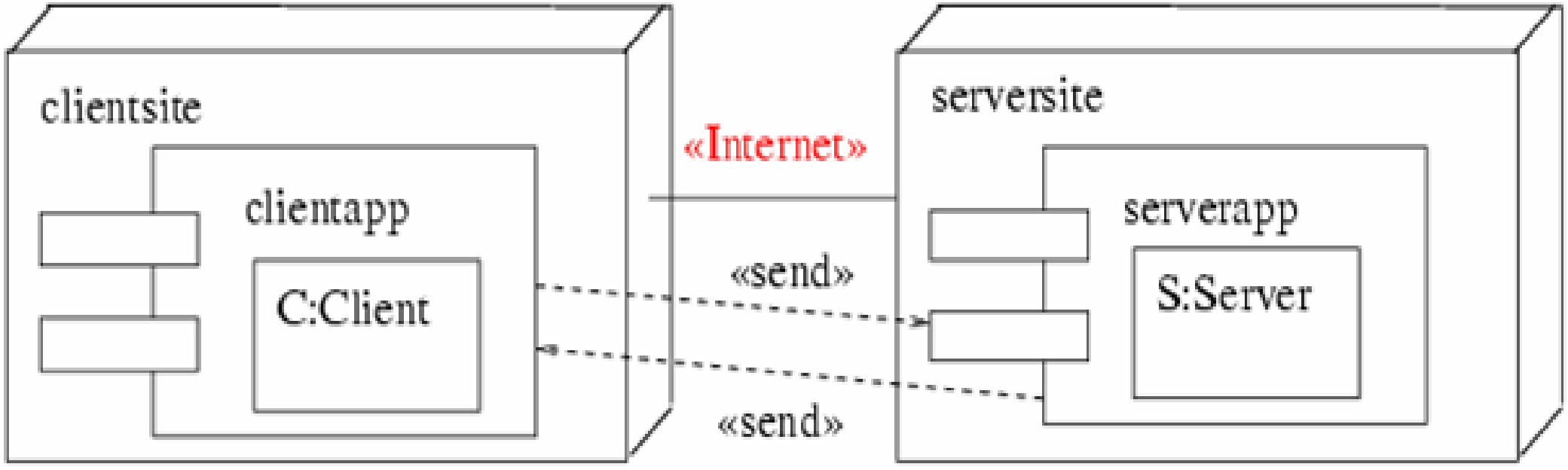
$$\forall E.(knows(E) \Rightarrow knows(head(E)) \wedge knows(tail(E)))$$

Gegebenes Sequenzdiagramm für o.g. Variante von TLS …

tls:



… und Verteilungsdiagramm …



Abgeleitete Angreifer-Aktionen auf Kommunikationsdaten: read, delete, insert.

… Übersetzung in Logik erster Stufe

Gegeben eine Nachrichtenspezifikation

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

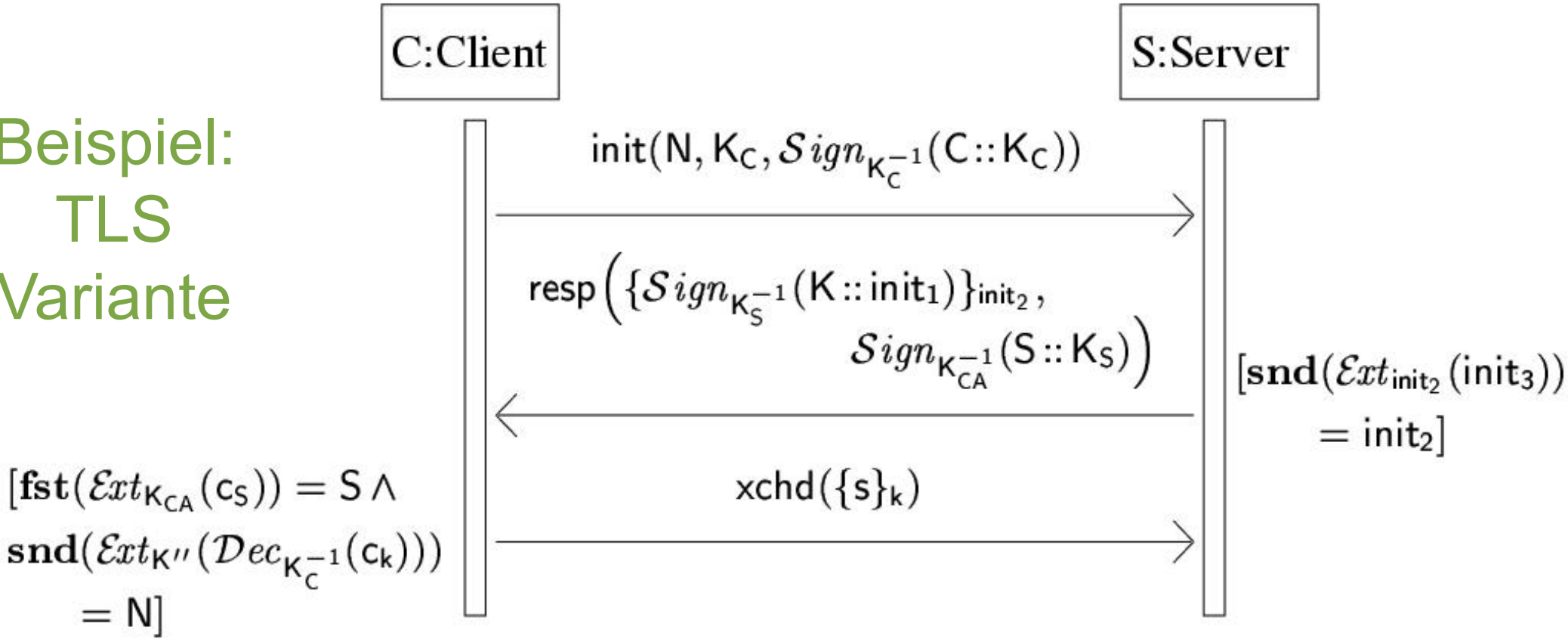
im Sequenzdiagramm, gefolgt von der Nachrichtenspezifikation *TR2*, ergibt Prädikat

$$PRED(TR1) = \forall\ msg_in.\ [knows(msg_in) \wedge cond(msg_in) \Rightarrow knows(msg_out) \wedge PRED(TR2)]$$

(Annahme: Nachrichtenreihenfolge überprüft (!).)

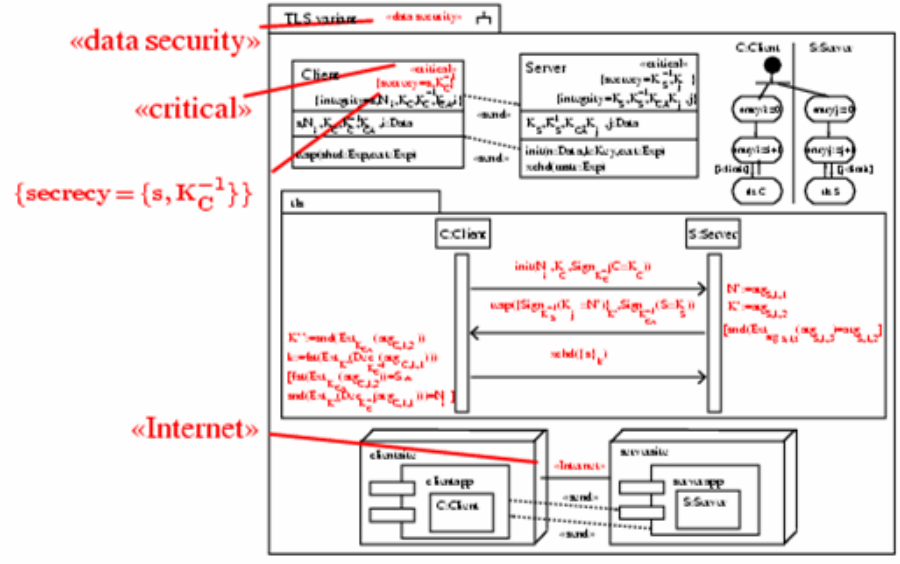
Hohe Abstraktion (zum Beispiel vom Sender und Empfänger der Nachrichten) mit dem Ziel einer effizienten automatischen Analyse. Nachteil: Es können “falsch-positive“ Ergebnisse auftreten (d.h. angebliche Angriffsmöglichkeiten, die sich bei näherer Untersuchung als unpraktikabel herausstellen).

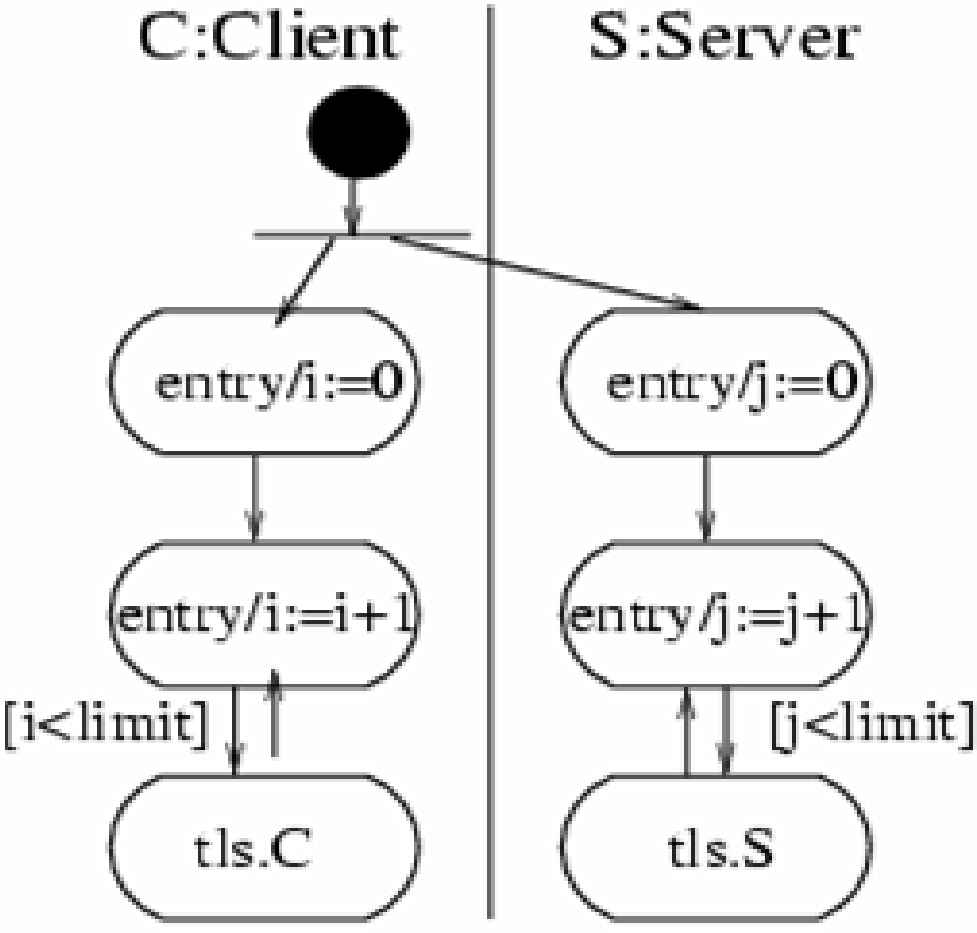
Beispiel: TLS Variante



$$knows(N) \wedge knows(K_C) \wedge knows(Sign_{K_C^{-1}}(C::K_C))$$
$$\wedge \forall init_1, init_2, init_3.[knows(init_1) \wedge knows(init_2) \wedge$$
$$knows(init_3) \wedge snd(Ext_{init\ 2}(init_3)) = init_2$$
$$\Rightarrow knows(\{Sign_{K_S^{-1}}(\ldots)\}_{\ldots}) \wedge [knows(Sign\ldots)] \wedge$$
$$.\forall resp_1, resp_2.[\ldots \Rightarrow \ldots]]$$

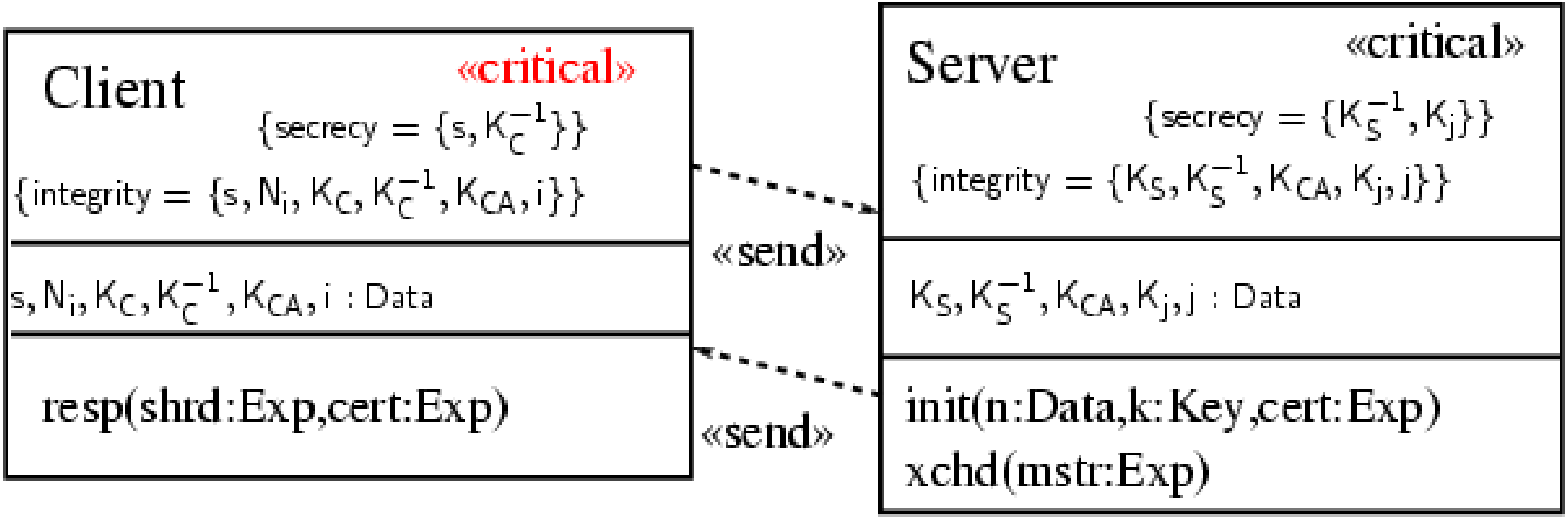
Ausführung im Systemkontext





Aktivitätsdiagramm

Sicherheitsanforderungen an Daten



Klassendiagramm.

Frage: Vertraulichkeit von Sitzungsschlüssel *s* bewahrt, d.h. *knows(s)* ableitbar ?

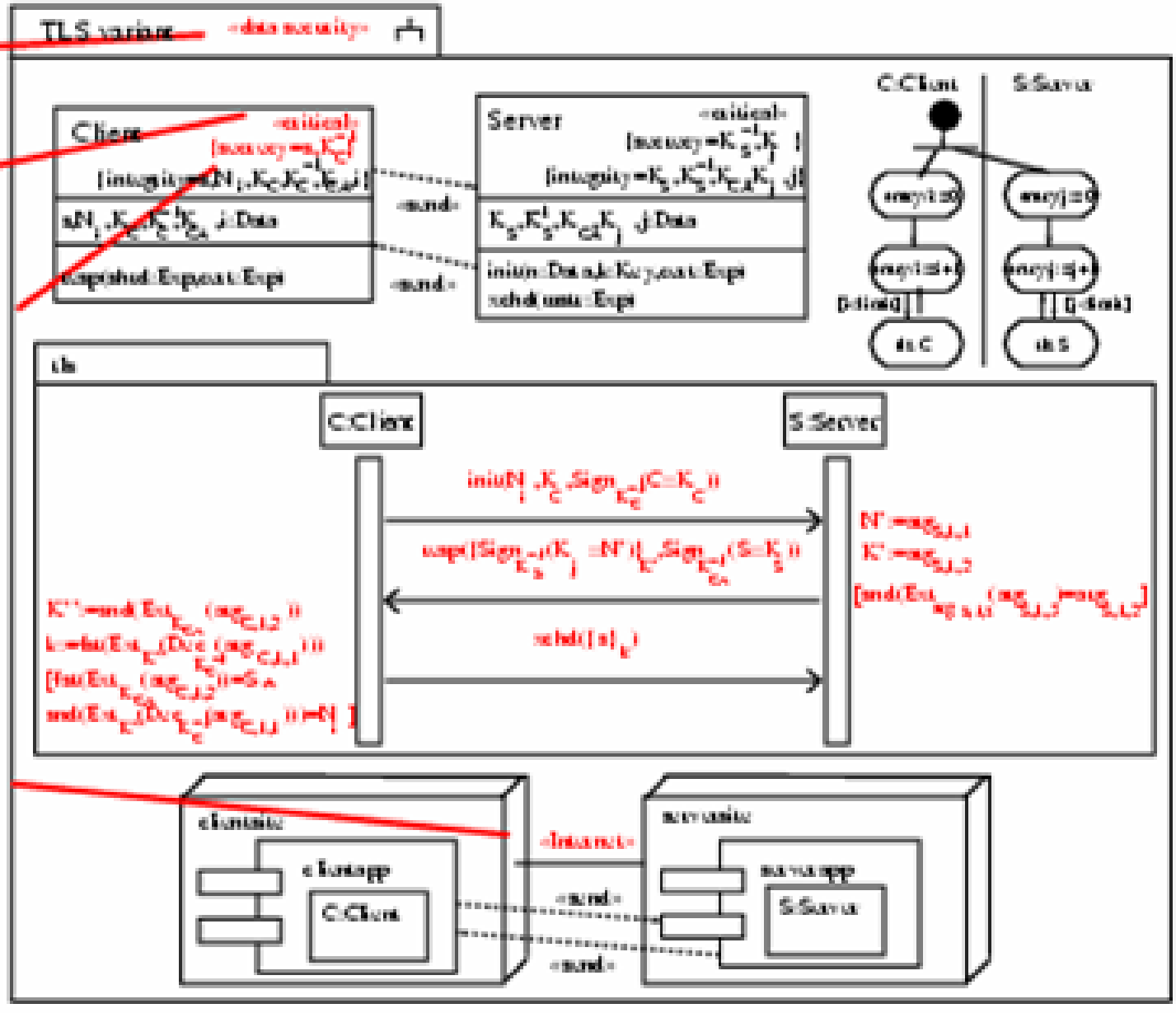
Überblick Variante von TLS: Gesamtes Modell

«data security»

«critical»

$\{secrecy = \{s, K_C^{-1}\}\}$

«Internet»



Übersetzung in Logik in Theorembeweiser-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) ) ) )  )
```

Übersetzung in Logik in Theorembeweiser-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)) )  )
) )  )).
```

Überraschung …


```
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

… Was bedeutet das ?

Kann *knows(s)* ableiten (!).

Daher: Protokoll bewahrt nicht die Vertraulichkeit des Sitzungsschlüssels *s* gegen Angreifer.

➔ Absolut unsicher in Bezug auf festgesetzte Ziele.

Aber wieso ?

Kann Angriffszenario mittels prologbasierten Angriffsgenerator erzeugen.

Man-in-the-Middle Angriff

$$C \xrightarrow{N_i::K_C::Sign_{K_C^{-1}}(C::K_C)} A \xrightarrow{N_i::K_A::Sign_{K_A^{-1}}(C::K_A)} S$$

$$A \xleftarrow{\{Sign_{K_S^{-1}}(K_j::N_i)\}_{K_A}::Sign_{K_{CA}^{-1}}(S::K_S)} S$$

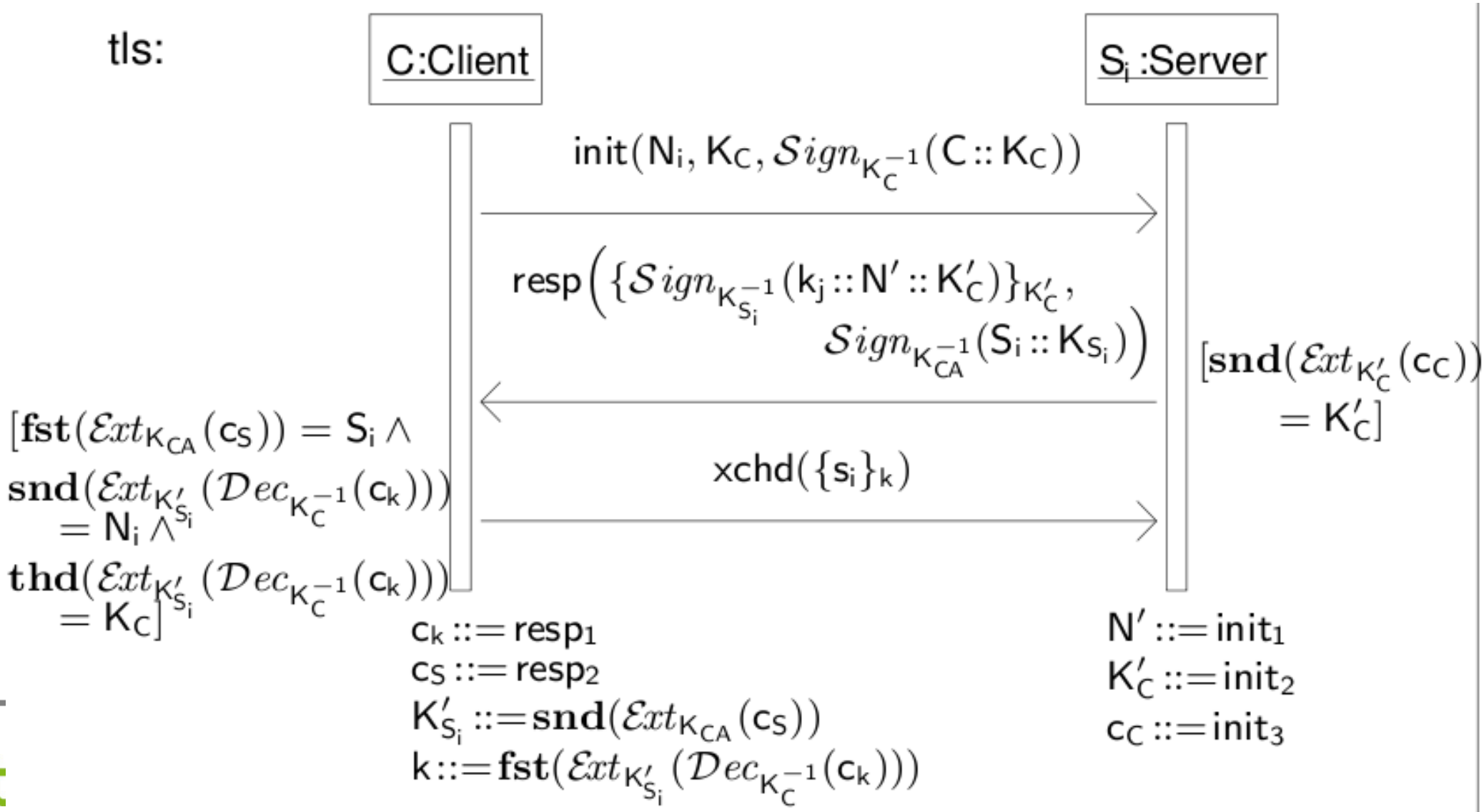
$$C \xleftarrow{\{Sign_{K_S^{-1}}(K_j::N_i)\}_{K_C}::Sign_{K_{CA}^{-1}}(S::K_S)} A$$

$$C \xrightarrow{\{s\}_{K_j}} A \xrightarrow{\{s\}_{K_j}} S$$

Korrigiertes Protokoll

Analyse: *knows(s)* nicht ableitbar, d.h. Vertraulichkeit von s bewahrt.

tls:



Zusammenfassung

- TLS-Variante
- Protokollmodellierung
- Sicherheitsanalyse
- Korrigierte Version