VALIkrypt

Fraunhofer Institut Sichere Telekooperation

Valikrypt - Validierung kryptographischer Protokolle

Sigrid Gürgens, Carsten Rudolph und Peter Ochsenschläger

FhG – Institut für Sichere Telekooperation SIT

Darmstadt

Inhalt

- Einleitung
- Protokollspezifikation mit APA
- Übergang Spezifikation – Analyse
- Ein Angriff
- Zusammenfassung

Einleitung

Was liefert Validierung:

Einleitung

Was liefert Validierung:

Aufzeigen möglicher Angriffe im betrachteten Szenario

Einleitung

Was liefert Validierung:

Aufzeigen möglicher Angriffe im betrachteten Szenario

Annahme: unsichere Übertragungswege

VALIkrypt

Einleitung

Fraunhofer Institut Sichere Telekooperation

Was liefert Validierung:

Aufzeigen möglicher Angriffe im betrachteten Szenario
Annahme: unsichere Übertragungswege

Was kann Validierung nicht liefern:

Einleitung

Was liefert Validierung:

Aufzeigen möglicher Angriffe im betrachteten Szenario
Annahme: unsichere Übertragungswege

Was kann Validierung nicht liefern:

- **Kryptoanalyse der im Protokoll verwendeten Algorithmen (Annahme “perfect encryption”)**

Einleitung

Was liefert Validierung:

Aufzeigen möglicher Angriffe im betrachteten Szenario
Annahme: unsichere Übertragungswege

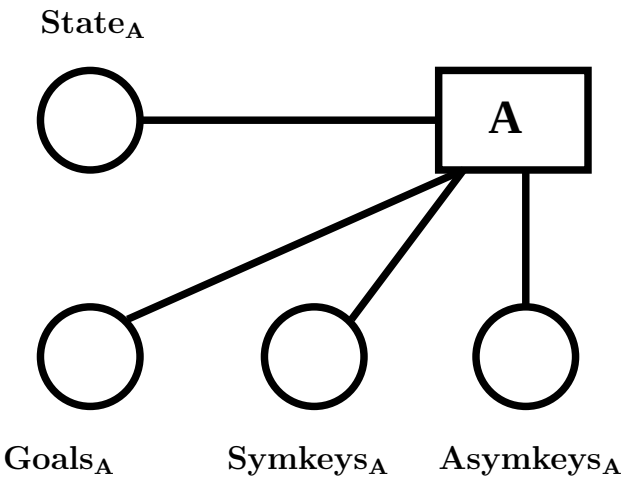
Was kann Validierung nicht liefern:

- **Kryptoanalyse der im Protokoll verwendeten Algorithmen (Annahme “perfect encryption”)**
- **Beweise für Sicherheitseigenschaften**

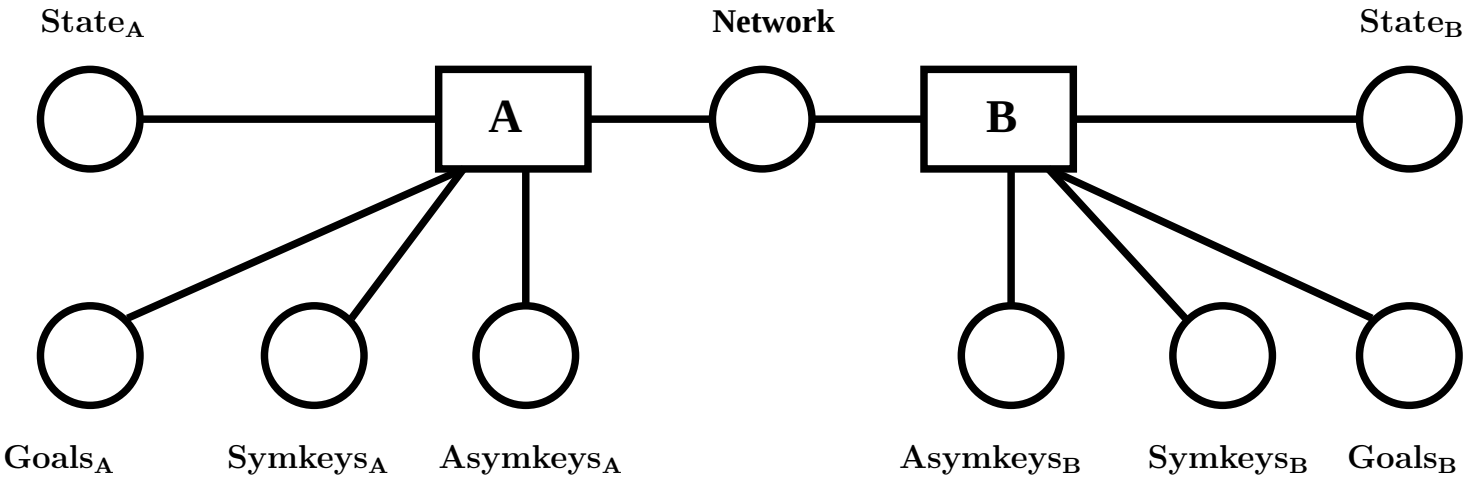
  kein Angriff gefunden $\Rightarrow$??

APA Struktur für Protokolle

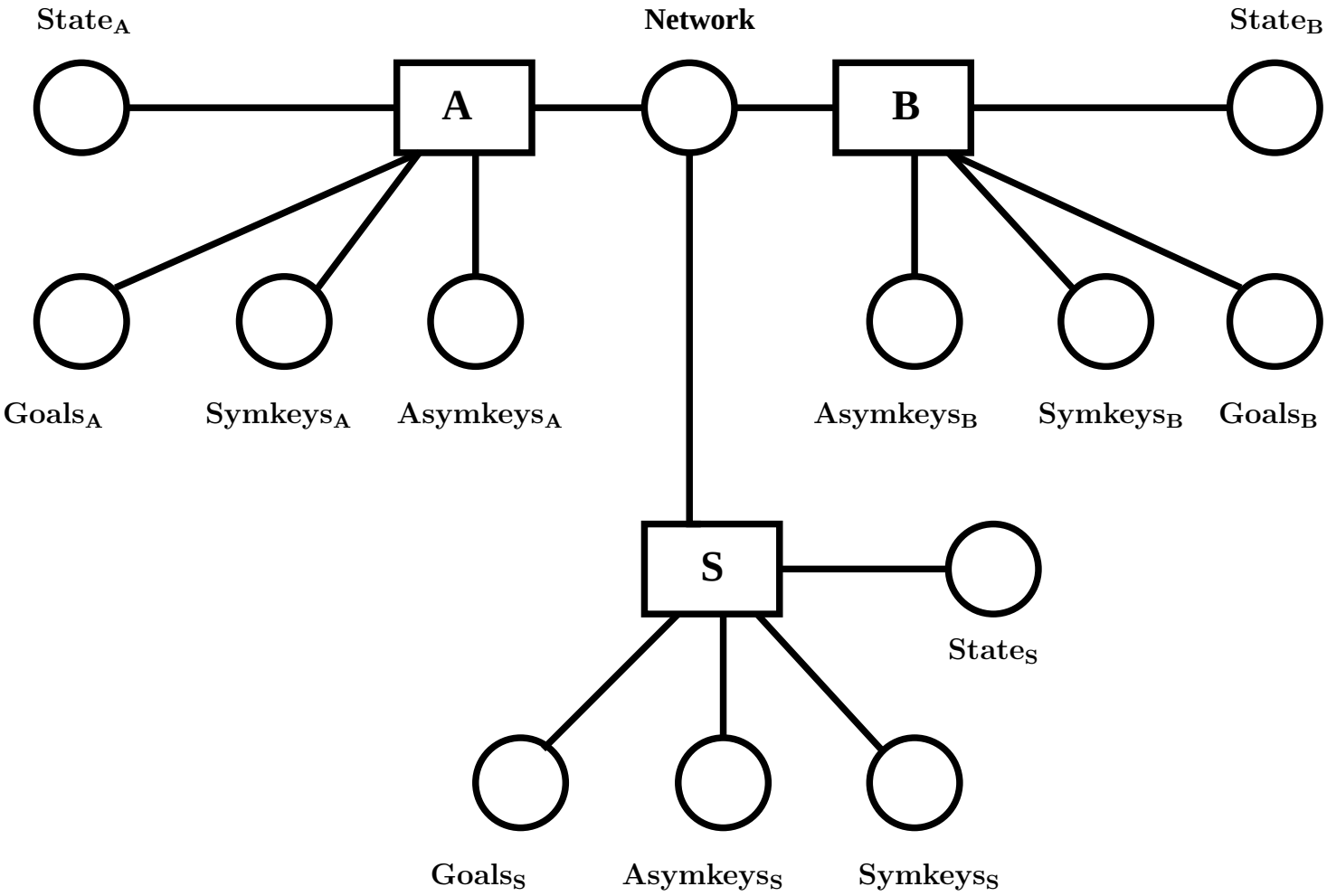
APA Struktur für Protokolle



APA Struktur für Protokolle



APA Struktur für Protokolle



APA Spezifikation

Um einen APA komplett zu spezifieren, wird benötigt:

APA Spezifikation

Um einen APA komplett zu spezifieren, wird benötigt:

- **Elementarautomaten, Zustandskomponenten, Nachbarschaftsrelation**

APA Spezifikation

Um einen APA komplett zu spezifieren, wird benötigt:

- **Elementarautomaten, Zustandskomponenten, Nachbarschaftsrelation**
- **Definitionsbereiche für Zustandskomponenten**

APA Spezifikation

Um einen APA komplett zu spezifieren, wird benötigt:

- **Elementarautomaten, Zustandskomponenten, Nachbarschaftsrelation**
- **Definitionsbereiche für Zustandskomponenten**
- **Zustandsübergangsrelation**

APA Spezifikation

Um einen APA komplett zu spezifieren, wird benötigt:

- **Elementarautomaten, Zustandskomponenten, Nachbarschaftsrelation**
- **Definitionsbereiche für Zustandskomponenten**
- **Zustandsübergangsrelation**
- **Initialzustand**

Definitionsbereiche, $\mathcal{M}essages$

- **Network: (A,B,message)**

Definitionsbereiche, $\mathcal{M}essages$

- **Network: (A,B,message)**

Grundmengen für $\mathcal{M}essages$:

Definitionsbereiche, $\mathcal{M}essages$

- **Network: (A,B,message)**

Grundmengen für $\mathcal{M}essages$:

N	**Menge der natürlichen Zahlen**
$\mathcal{A}gents$	**Menge von Agentennamen**
$\mathcal{I}dentity$	$\mathcal{A}gents \cup$ **N**
$\mathcal{N}once$	**Menge von nonces**
$\mathcal{C}onstants$	**Menge von Konstanten zur Definition der Zustandsübergangsrelationen**

Definitionsbereiche, $\mathcal{M}essages$

$\mathcal{K}eynames$ **Menge von Schlüsselnamen,**

$$\mathcal{A}gents \subseteq \mathcal{K}eynames$$

$$\mathcal{A}gents \times \mathcal{A}gents \subseteq \mathcal{K}eynames$$

Definitionsbereiche, $\mathcal{M}essages$

$\mathcal{K}eynames$	**Menge von Schlüsselnamen,**
	$\mathcal{A}gents \subseteq \mathcal{K}eynames$
	$\mathcal{A}gents \times \mathcal{A}gents \subseteq \mathcal{K}eynames$
$\mathcal{S}ymflags$	$\{sym, symmac, \ldots\}$
$\mathcal{A}symflags$	$\{pub, priv, pubverify, privsign, \ldots\}$

Definitionsbereiche, $\mathcal{M}essages$

$\mathcal{K}eynames$	**Menge von Schlüsselnamen,**
	$\mathcal{A}gents \subseteq \mathcal{K}eynames$
	$\mathcal{A}gents \times \mathcal{A}gents \subseteq \mathcal{K}eynames$
$\mathcal{S}ymflags$	$\{sym, symmac, \ldots\}$
$\mathcal{A}symflags$	$\{pub, priv, pubverify, privsign, \ldots\}$
$\mathcal{K}eys$	$\{(w, f, n) \mid w \in \mathcal{K}eynames, f \in \mathcal{S}ymflags \cup \mathcal{A}symflags, n \in \mathbf{N}\}$

Definitionsbereiche, $\mathcal{M}essages$

$\mathcal{K}eynames$	**Menge von Schlüsselnamen,**
	$\mathcal{A}gents \subseteq \mathcal{K}eynames$
	$\mathcal{A}gents \times \mathcal{A}gents \subseteq \mathcal{K}eynames$
$\mathcal{S}ymflags$	$\{sym, symmac, \ldots\}$
$\mathcal{A}symflags$	$\{pub, priv, pubverify, privsign, \ldots\}$
$\mathcal{K}eys$	$\{(w, f, n) \mid w \in \mathcal{K}eynames, f \in \mathcal{S}ymflags \cup \mathcal{A}symflags, n \in \mathbf{N}\}$
$\mathcal{F}uncs$	**Symbolische Funktionen für Kryptographie**
	$(encrypt, decrypt, crypt, sign, hash, \ldots)$

Messages

1. $\mathbf{N} \cup \mathcal{A}gents \cup \mathcal{I}dentity \cup \mathcal{N}once \cup \mathcal{C}onstants \cup \mathcal{K}eys \subseteq \mathcal{M}essages$

$\mathcal{M}essages$

1. **N** $\cup$ $\mathcal{A}gents \cup \mathcal{I}dentity \cup \mathcal{N}once \cup \mathcal{C}onstants \cup \mathcal{K}eys \subseteq \mathcal{M}essages$
2. **Erzeugung weiterer Nachrichten durch**
   - **eine Liste von Elementen in** $\mathcal{M}essages$

$\mathcal{M}essages$

1. **N** $\cup$ $\mathcal{A}gents \cup \mathcal{I}dentity \cup \mathcal{N}once \cup \mathcal{C}onstants \cup \mathcal{K}eys \subseteq \mathcal{M}essages$
2. **Erzeugung weiterer Nachrichten durch**
   - **eine Liste von Elementen in** $\mathcal{M}essages$
   - **Anwendung symbolischer Funktionen**

Die symbolischen Funktionen $encrypt, decrypt, crypt, sign$ haben die folgenden Eigenschaften:

Für alle $P \in \mathcal{A}gents$ und $k, m, n \in \mathcal{M}$ gilt

1. $decrypt(k, encrypt(k, m)) = m$
2. $encrypt(k, decrypt(k, m)) = m$
3. $crypt((P, priv, n), crypt((P, pub, n), m)) = m$
4. $recover((P, pub, n), crypt((P, priv, n), m)) = m$.

Definitionsbereiche, Beispiele

- **Network: (A,B,message)**

Definitionsbereiche, Beispiele

- **Network: (A,B,message)**
- **Symkeys: (B,sym,((A,B),sym,1))**

Definitionsbereiche, Beispiele

- Network: (A,B,message)
- Symkeys: (B,sym,((A,B),sym,1))
- Asymkeys: (S,pub,(S,pub,1))

Definitionsbereiche, Beispiele

- Network: (A,B,message)
- Symkeys: (B,sym,((A,B),sym,1))
- Asymkeys: (S,pub,(S,pub,1))
- Goals: zur Spezifikation von Sicherheitszielen.

Definitionsbereiche, Beispiele

- Network: (A,B,message)
- Symkeys: (B,sym,((A,B),sym,1))
- Asymkeys: (S,pub,(S,pub,1))
- Goals: zur Spezifikation von Sicherheitszielen.
  - Authentizität des Senders einer Nachricht
  - Frische von Schlüsseln
  - Vertraulichkeit von Daten

Definitionsbereiche, Beispiele

- Network: (A,B,message)
- Symkeys: (B,sym,((A,B),sym,1))
- Asymkeys: (S,pub,(S,pub,1))
- Goals: zur Spezifikation von Sicherheitszielen.
  - Authentizität des Senders einer Nachricht
  - Frische von Schlüsseln
  - Vertraulichkeit von Daten

    Beispiel:

    $\forall P \in \mathcal{A}gents \setminus \{A, B\} : \textbf{\textit{not\_knows}}(P, K)$

Zustandsübergangsrelationen

Zustandsübergangsrelationen

Beispiel: Needham-Schroeder Protokoll

1. $A \longrightarrow S: \quad A, B, R_A$

Zustandsübergangsrelationen

Beispiel: Needham-Schroeder Protokoll

1. $A \longrightarrow S: \quad A, B, R_A$
2. $S \longrightarrow A: \quad encrypt(K_{AS}, (R_A, B, K_{AB}, encrypt(K_{BS}, (K_{AB}, A))))$

Zustandsübergangsrelationen

Beispiel: Needham-Schroeder Protokoll

1. $A \longrightarrow S : \quad A, B, R_A$
2. $S \longrightarrow A : \quad encrypt(K_{AS}, (R_A, B, K_{AB}, encrypt(K_{BS}, (K_{AB}, A))))$
3. $A \longrightarrow B : \quad encrypt(K_{BS}, (K_{AB}, A))$

Zustandsübergangsrelationen

Beispiel: Needham-Schroeder Protokoll

1. $A \longrightarrow S : \quad A, B, R_A$
2. $S \longrightarrow A : \quad encrypt(K_{AS}, (R_A, B, K_{AB}, encrypt(K_{BS}, (K_{AB}, A))))$
3. $A \longrightarrow B : \quad encrypt(K_{BS}, (K_{AB}, A))$
4. $B \longrightarrow A : \quad encrypt(K_{AB}, (R_B))$

Zustandsübergangsrelationen

Beispiel: Needham-Schroeder Protokoll

1. $A \longrightarrow S : \quad A, B, R_A$
2. $S \longrightarrow A : \quad encrypt(K_{AS}, (R_A, B, K_{AB}, encrypt(K_{BS}, (K_{AB}, A))))$
3. $A \longrightarrow B : \quad encrypt(K_{BS}, (K_{AB}, A))$
4. $B \longrightarrow A : \quad encrypt(K_{AB}, (R_B))$
5. $A \longrightarrow B : \quad encrypt(K_{AB}, (R_B - 1))$

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

  $encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$

$\xrightarrow{\mathrm{B}}$

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

  $encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$

  Variables: $X, A, S, M, K_{BS}, K_{AB}, R_B$

$\xrightarrow{\mathrm{B}}$

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

  $encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$

  Variables: $X, A, S, M, K_{BS}, K_{AB}, R_B$

  $(X, B, M) \in$ **Network**

$\xrightarrow{\mathrm{B}}$

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

  $encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$

  Variables: $X, A, S, M, K_{BS}, K_{AB}, R_B$

  $(X, B, M) \in$ **Network**

  $(A,$ ***agent***$) \in$ **State$_B$**

$\xrightarrow{B}$

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

  $encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$

  Variables: $X, A, S, M, K_{BS}, K_{AB}, R_B$

  $(X, B, M) \in$ **Network**

  $(A,$ ***agent***$) \in$ **State$_B$**

  $(S,$ ***server***$) \in$ **State$_B$**

$\xrightarrow{B}$

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

  $encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$

  Variables: $X, A, S, M, K_{BS}, K_{AB}, R_B$

  $(X, B, M) \in$ **Network**

  $(A,$ ***agent***$) \in$ **State$_B$**

  $(S,$ ***server***$) \in$ **State$_B$**

  $(S, sym, K_{BS}) \in$ **Symkeys$_B$**

  $\xrightarrow{B}$

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

  $encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$

  Variables: $X, A, S, M, K_{BS}, K_{AB}, R_B$

  $(X, B, M) \in$ **Network**

  $(A,$ ***agent***$) \in$ **State$_B$**

  $(S,$ ***server***$) \in$ **State$_B$**

  $(S, sym, K_{BS}) \in$ **Symkeys$_B$**

  $elem(2, decrypt(K_{BS}, M)) = A$

  $\xrightarrow{B}$

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

  $$encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$$

  $$\xrightarrow{\mathrm{B}}$$

  $$K_{AB} := elem(1, decrypt(K_{BS}, M))$$

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

$$encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$$

$$\xrightarrow{\mathrm{B}}$$

$$K_{AB} := elem(1, decrypt(K_{BS}, M))$$

$$R_B := random\_nonce$$

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

  $encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$

  $\xrightarrow{\mathrm{B}}$

  $K_{AB} := elem(1, decrypt(K_{BS}, M))$

  $R_B := random\_nonce$

  (X, B, M) $\hookleftarrow$ **Network**

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

  $encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$

  $\xrightarrow{\mathrm{B}}$

  $K_{AB} := elem(1, decrypt(K_{BS}, M))$

  $R_B := random\_nonce$

  (X, B, M) $\hookleftarrow$ **Network**

  $(\textbf{\textit{new session key}}, A, K_{AB}, R_B)$ $\hookrightarrow$ **State$_B$**

Zustandsübergangsrelationen

- **Transitionsmuster:**

  step 4

  $encrypt(K_{BS}, (K_{AB}, A)),\ encrypt(K_{AB}, (R_B))$

  $\xrightarrow{\text{B}}$

  $K_{AB} := elem(1, decrypt(K_{BS}, M))$

  $R_B := random\_nonce$

  (X, B, M) $\hookleftarrow$ **Network**

  $(\textbf{\textit{new session key}}, A, K_{AB}, R_B)$ $\hookrightarrow$ **State$_B$**

  $(B, A, (encrypt(K_{AB}, R_B)))$ $\hookrightarrow$ **Network**

VALIkrypt

Übergang zur Analyse

Fraunhofer SIT Institut Sichere Telekooperation

Übergang zur Analyse

- **Konkrete Agenten werden zu jeder Rolle angegeben. Das SHVT generiert automatisch das jeweilige Analysemodell.**

Übergang zur Analyse

- **Konkrete Agenten werden zu jeder Rolle angegeben. Das SHVT generiert automatisch das jeweilige Analysemodell.**

  role A : $\{Alice, Charly\}$

  role B : $\{Bob\}$

  role S : $\{Server\}$

Übergang zur Analyse

- **Konkrete Agenten werden zu jeder Rolle angegeben. Das SHVT generiert automatisch das jeweilige Analysemodell.**

  role A : $\{Alice, Charly\}$

  role B : $\{Bob\}$

  role S : $\{Server\}$

- **Initialzustand: wieviele Läufe führt welcher Agent in welcher Rolle durch.**

Übergang zur Analyse

- **Konkrete Agenten werden zu jeder Rolle angegeben. Das SHVT generiert automatisch das jeweilige Analysemodell.**

  role A : $\{Alice, Charly\}$

  role B : $\{Bob\}$

  role S : $\{Server\}$

- **Initialzustand: wieviele Läufe führt welcher Agent in welcher Rolle durch.**

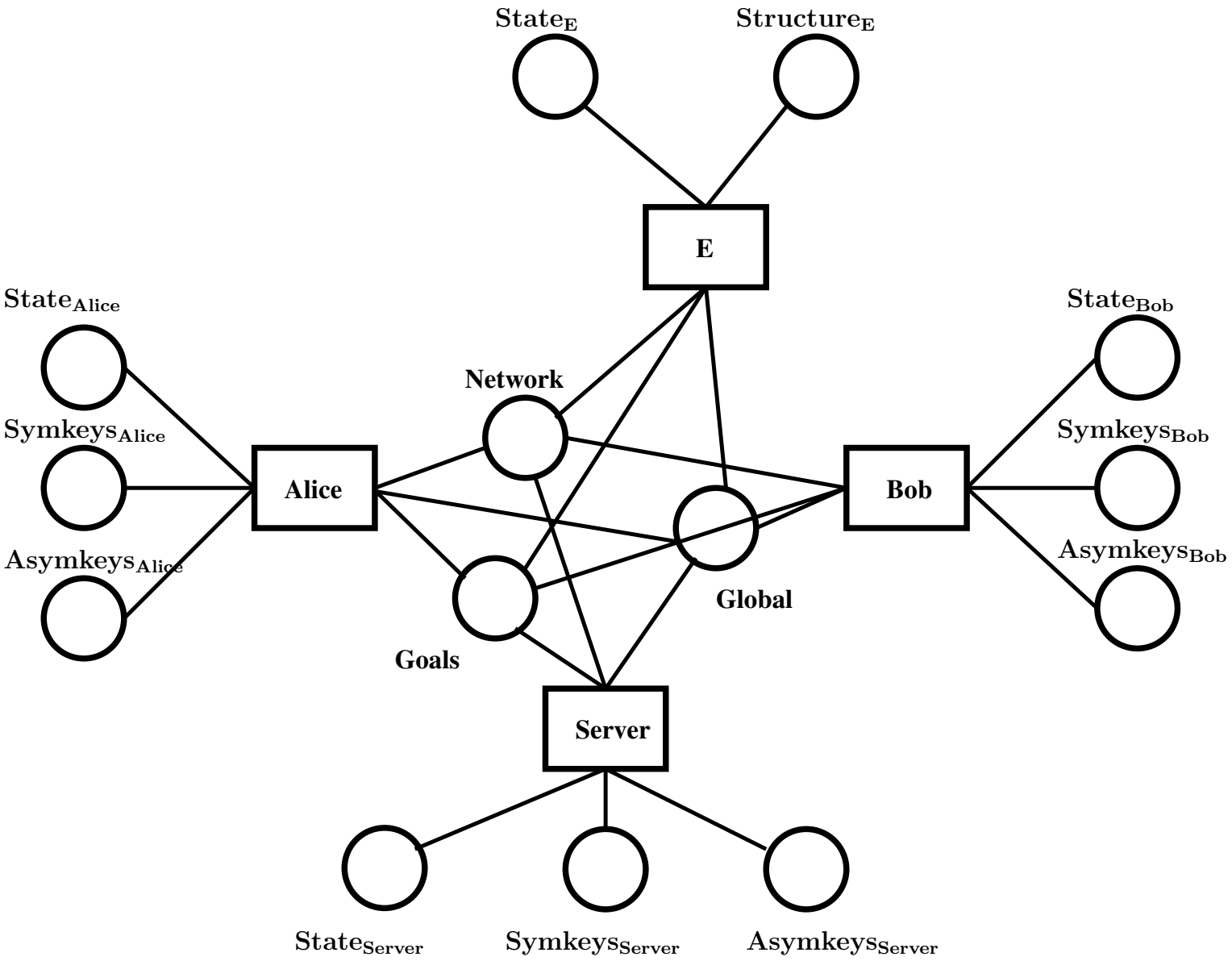
- **Angreiferverhalten muss hinzugefügt werden.**

VALIkrypt

APA Struktur Analysemodell

Fraunhofer Institut Sichere Telekooperation

APA Struktur Analysemodell



VALIkrypt

Angreifer

Fraunhofer Institut Sichere Telekooperation

Angreifer

Das Wissen des Angreifers:

Angreifer

Das Wissen des Angreifers:

- Der Angreifer kennt Daten in seinem Initialzustand.

Angreifer

Das Wissen des Angreifers:

- Der Angreifer kennt Daten in seinem Initialzustand.
- Der Angreifer kennt die Agentennamen, die zur Adressierung in Network verwendet wurden.

Angreifer

Das Wissen des Angreifers:

- Der Angreifer kennt Daten in seinem Initialzustand.
- Der Angreifer kennt die Agentennamen, die zur Adressierung in Network verwendet wurden.
- Der Angreifer kennt die Nachrichten aus Network und kann diese in Einzelteile zerlegen (ein Chiffretext wird als ein Teil angesehen).

Angreifer

Das Wissen des Angreifers:

- Der Angreifer kennt Daten in seinem Initialzustand.
- Der Angreifer kennt die Agentennamen, die zur Adressierung in Network verwendet wurden.
- Der Angreifer kennt die Nachrichten aus Network und kann diese in Einzelteile zerlegen (ein Chiffretext wird als ein Teil angesehen).
- Kennt der Angreifer den passenden Schlüssel, kann er den Klartext einer verschlüsselten Nachricht lernen.

Angreifer

- **Der Angreifer kann durch cons und append neue Nachrichten aus bekannten Nachrichten erzeugen.**

Angreifer

- **Der Angreifer kann durch cons und append neue Nachrichten aus bekannten Nachrichten erzeugen.**
- **Der Angreifer kann durch Anwendung symbolischer Funktionen neue Nachrichten erzeugen.**

Angreifer

- Der Angreifer kann durch cons und append neue Nachrichten aus bekannten Nachrichten erzeugen.
- Der Angreifer kann durch Anwendung symbolischer Funktionen neue Nachrichten erzeugen.
- Der Angreifer kann eigene Zufallszahlen und Schlüssel erzeugen.

Angriffsnachrichten

Das Wissen des Angreifers ist theoretisch unendlich und muss deshalb für die Berechnung endlicher Zustandsräume begrenzt werden:

Angriffsnachrichten

Das Wissen des Angreifers ist theoretisch unendlich und muss deshalb für die Berechnung endlicher Zustandsräume begrenzt werden:

Ein Angreifer schickt beliebige Nachrichten mit der richtigen *Struktur*.

Angriffsnachrichten

Die Struktur beinhaltet:

1. Die Anzahl der Nachrichtenteile

Angriffsnachrichten

Die Struktur beinhaltet:

1. **Die Anzahl der Nachrichtenteile**
2. **Das Format einzelner Nachrichtenteile**
   (a) **atomar (d.h. eine Zufallszahl, Agentenname, Schlüssel,. . .)**
   (b) **Chiffretext (symmetrisch, asymmetrisch)**

Angriffsnachrichten

Struktur der Nachrichten: im Initialzustand der Zustandskomponente $Structure$

Angriffsnachrichten

Struktur der Nachrichten: im Initialzustand der Zustandskomponente $Structure$

Schlüsselworte zur Typbestimmung der Nachrichtenteile (abhängig von der Prüfung beim Empfänger)

Angriffsnachrichten

Struktur der Nachrichten: im Initialzustand der Zustandskomponente $Structure$
Schlüsselworte zur Typbestimmung der Nachrichtenteile (abhängig von der Prüfung beim Empfänger)

Beispiel: $A \longrightarrow S : \ A, B, R_A$

(3,A,S,[agents,agents,atom])

Vorgehensweise Sicherheitsanalyse

Vorgehensweise Sicherheitsanalyse

1. **Spezifikation des Rollenverhaltens, Scenario und Angreifer.**

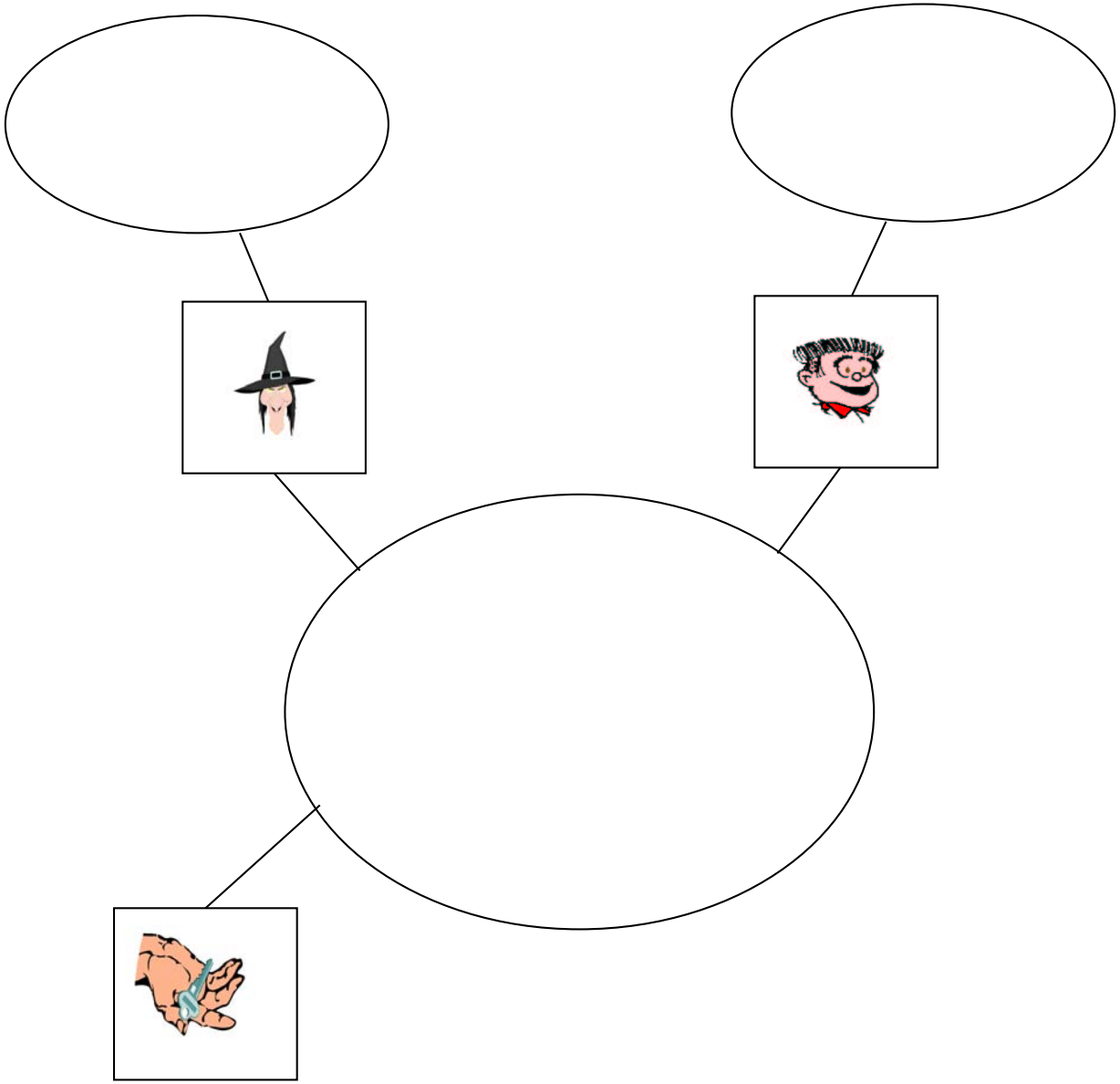
Vorgehensweise Sicherheitsanalyse

1. Spezifikation des Rollenverhaltens, Scenario und Angreifer.
2. SHVT berechnet automatisch Erreichbarkeitsgraph, bis alle erreichbaren Zustände berechnet sind oder die Abbruchbedingung gilt.

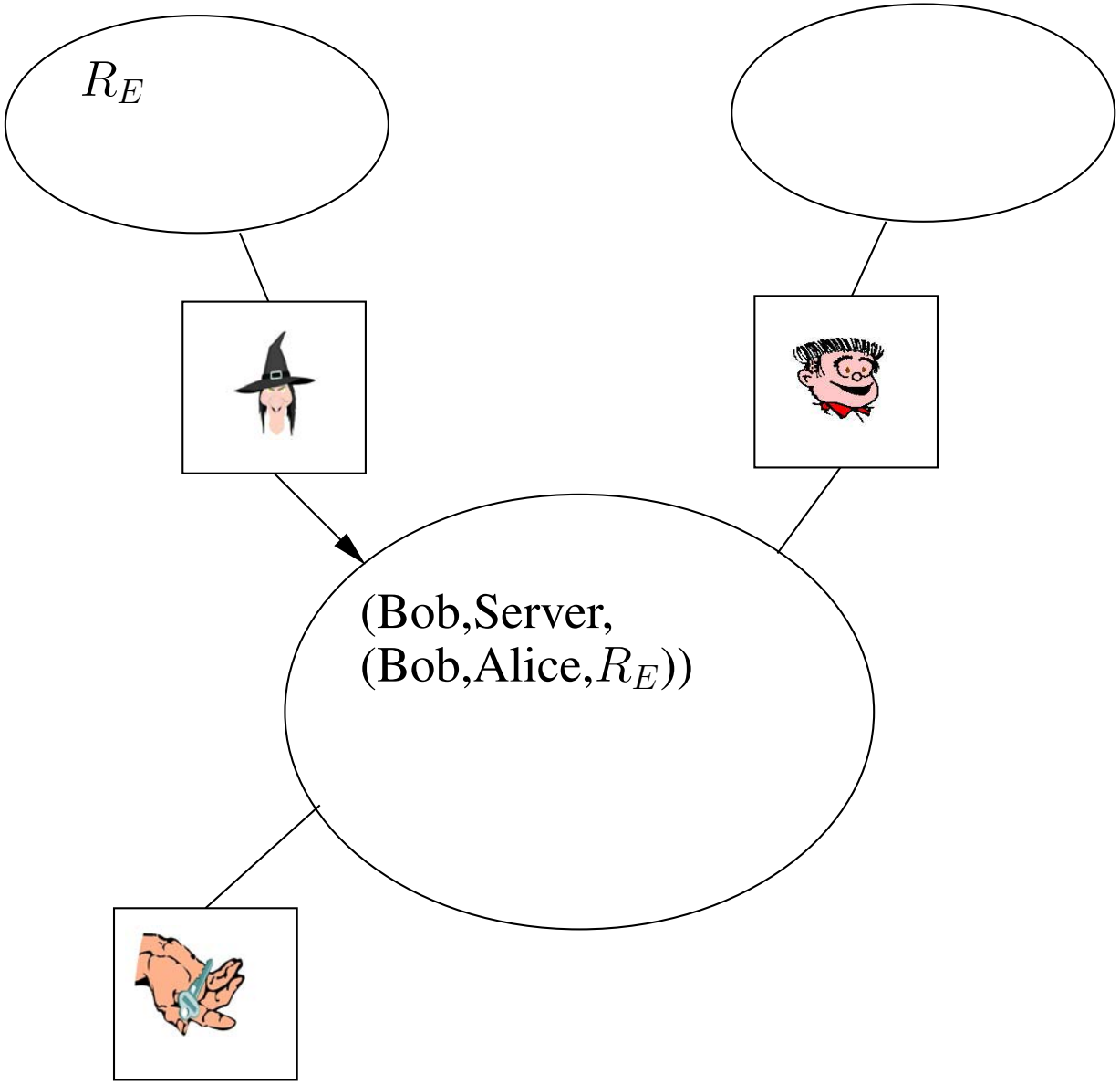
Vorgehensweise Sicherheitsanalyse

1. Spezifikation des Rollenverhaltens, Scenario und Angreifer.
2. SHVT berechnet automatisch Erreichbarkeitsgraph, bis alle erreichbaren Zustände berechnet sind oder die Abbruchbedingung gilt.
3. Bei Abbruch wird der letzte Zustandsübergang ausgegeben. Das SHVT berechnet einen Weg zum Initialzustand. Daran kann der Angriff nachvollzogen werden.

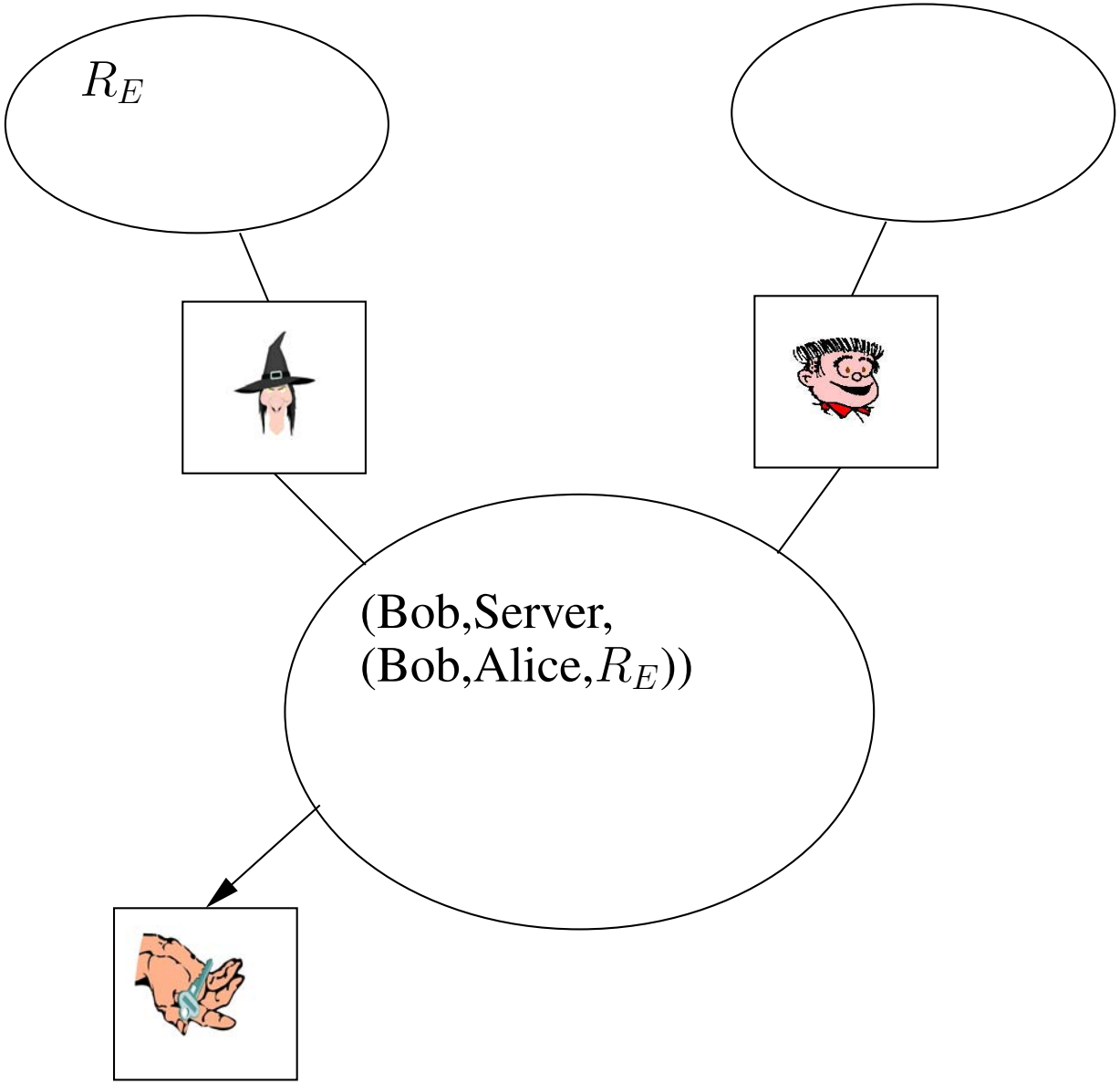
Ein Angriff



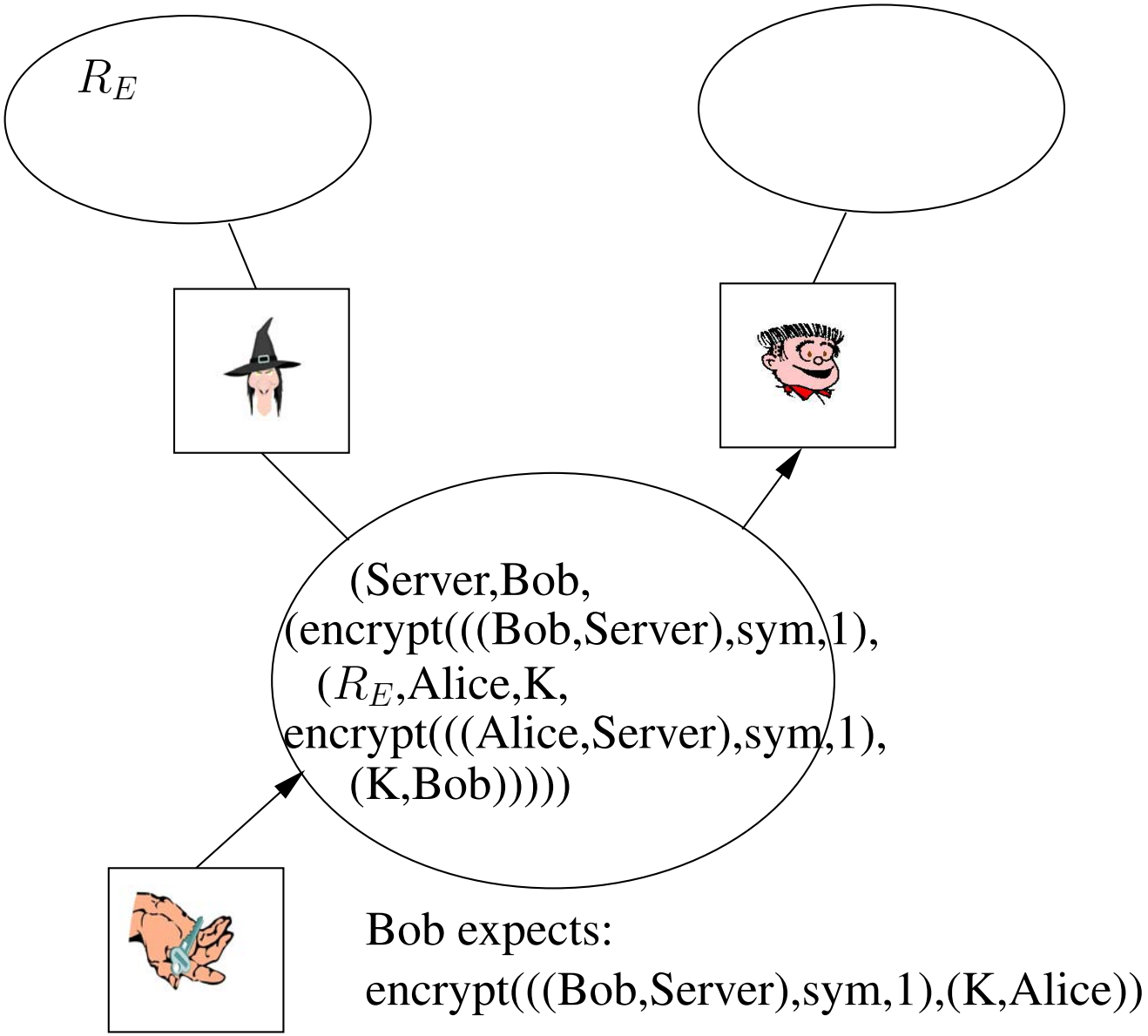
Ein Angriff



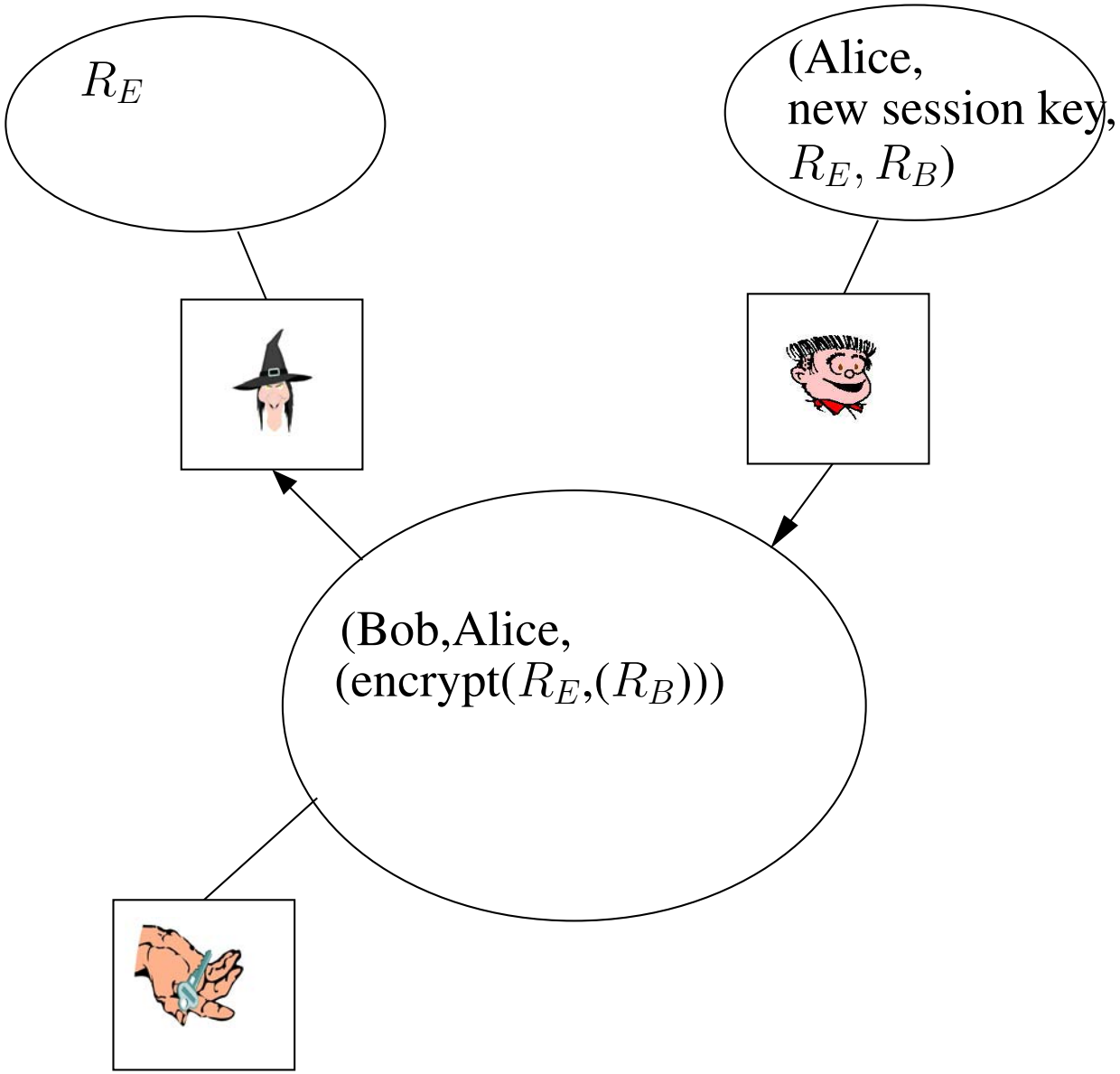
Ein Angriff



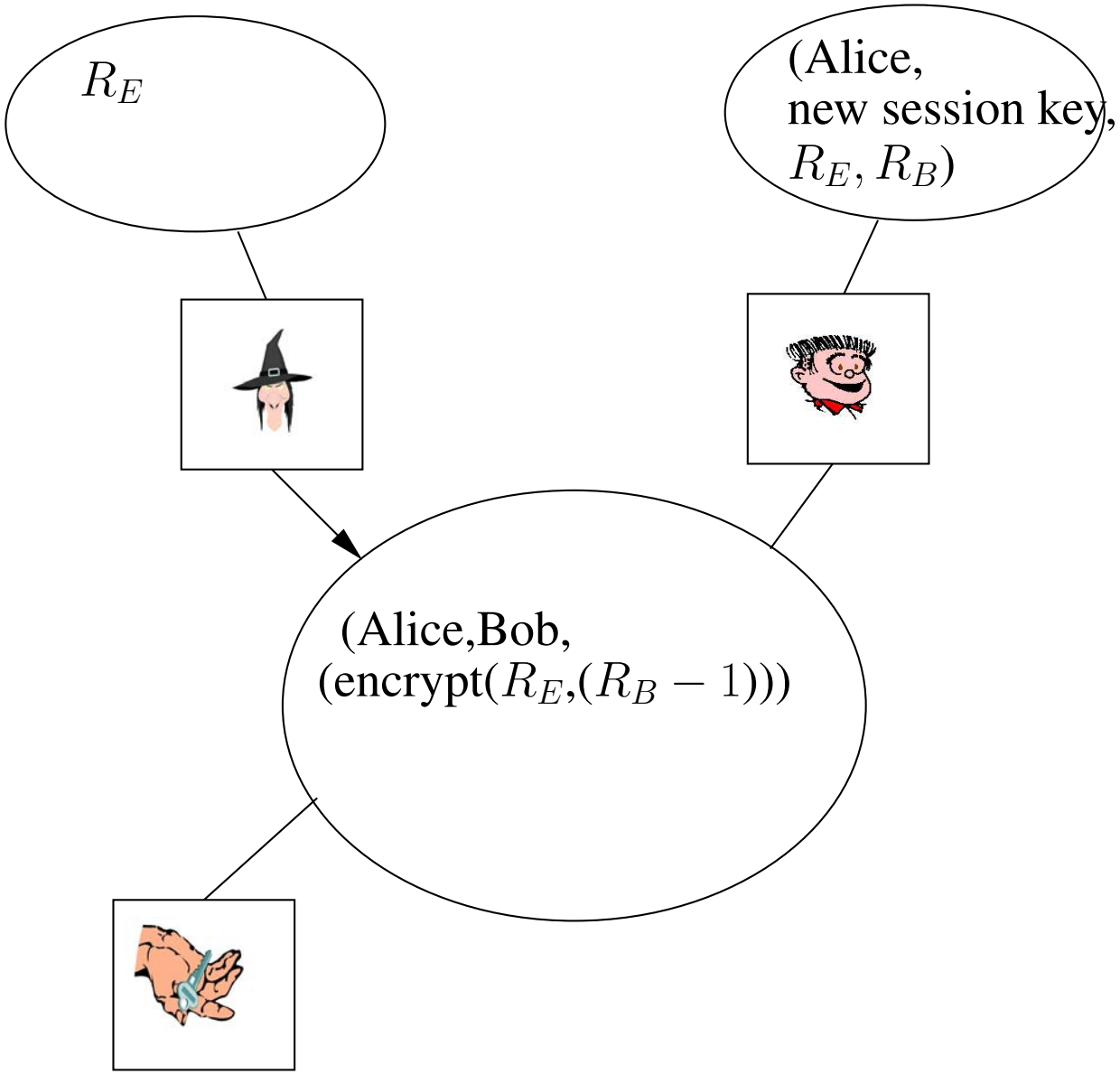
Ein Angriff


R_E
(Server,Bob,
(encrypt(((Bob,Server),sym,1),
(R_E,Alice,K,
encrypt(((Alice,Server),sym,1),
(K,Bob)))))
Bob expects:
encrypt(((Bob,Server),sym,1),(K,Alice))

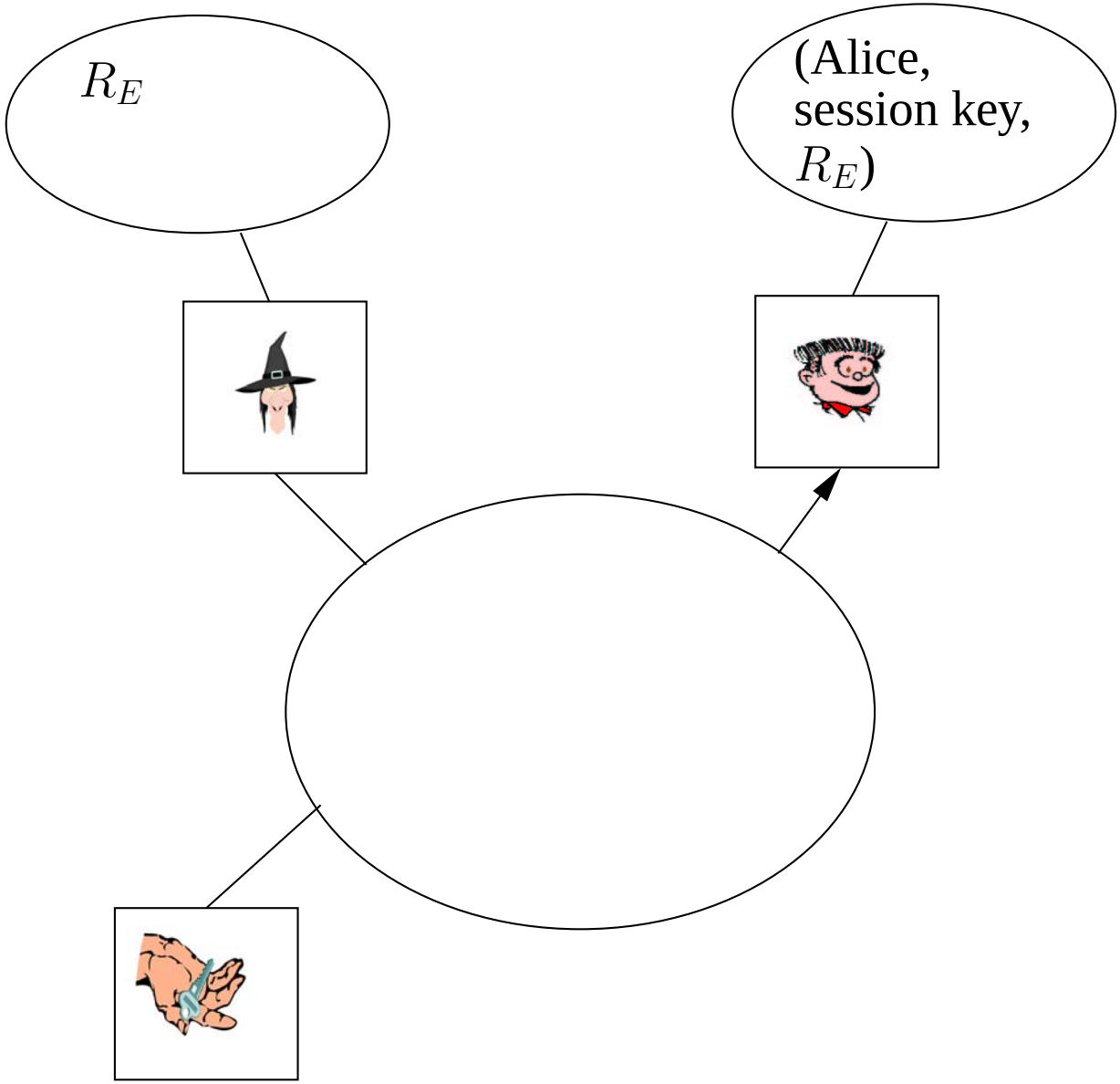
Ein Angriff



Ein Angriff



Ein Angriff



Ein Angriff

Wie kann man den Angriff vermeiden:

Ein Angriff

Wie kann man den Angriff vermeiden:

- **Typprüfung (nicht immer möglich)**

Ein Angriff

Wie kann man den Angriff vermeiden:

- **Typprüfung (nicht immer möglich)**
- **Prüfung der Länge des Chiffretextes (nicht trivial)**

Das SHVT

Das SHVT

Simple Homomorphism Verification Tool

VALIkrypt

Das SHVT

Fraunhofer Institut Sichere Telekooperation

Simple Homomorphism Verification Tool

Ein Werkzeug zur Verifikation kooperierender Systeme (entwickelt von SIT)

Das SHVT

Simple Homomorphism Verification Tool

Ein Werkzeug zur Verifikation kooperierender Systeme (entwickelt von SIT)

endliche Zustandsräume

Das SHVT

Simple Homomorphism Verification Tool
Ein Werkzeug zur Verifikation kooperierender Systeme (entwickelt von SIT)
endliche Zustandsräume
angepasst an Erfordernisse von Protokollvalidierung

Texteditor des SHVT

Texteditor des SHVT

Preamble: needham-sym

File Test Edit Options Pathnames Misc Help

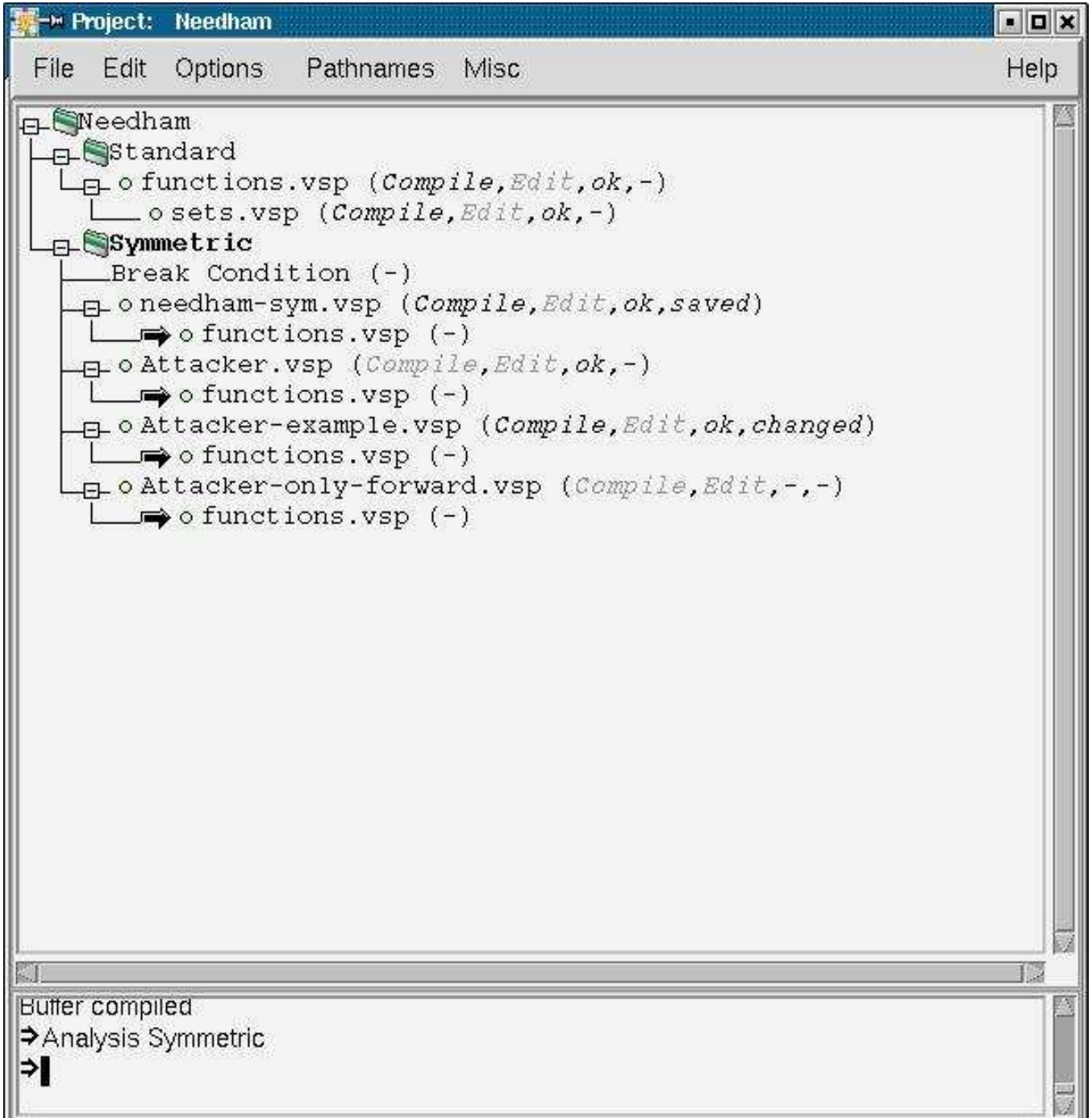
```
def_trans_pattern A step_5a
  (X,B,M_list,M,KAB,RB)
  /* X    ? Agents,
     B    ? Agents,
     M    ? Messages,
     M_list ? Messages,
     KAB ? Symkeys,
     RB ?  Nonce */
  (X,A,M_list) << Network,
   M:=elem(1,M_list),
  ['new_session_key',B,KAB] ? A_State,
  RB := elem(1,decrypt(KAB,M)),
  ['new_session_key',B,KAB] << A_State,
  ['session_key',B,KAB] >> A_State,
  (A,B,[encrypt(KAB,[subtract(RB)])]) >> Network;

def_trans_pattern B step_5b
  (X,A,M_list,M,KAB,RB)
  /* X    ? Agents,
     A    ? Agents,
     M    ? Messages,
     M_list ? Messages,
     KAB ? Symkeys,
     RB   ? Nonce */
  (X,B,M_list) << Network,
  M:=elem(1,M_list),
  ['new_session_key',A,KAB,RB] ? B_State,
  elem(1,decrypt(KAB,M)) = subtract(RB),
  ['new_session_key',A,KAB,RB] << B_State,
  ['session_key',A,KAB] >> B_State;
```

```
Buffer compiled
```

Projektverwaltung des SHVT

Fraunhofer Institut Sichere Telekooperation

Projektverwaltung des SHVT



Zusammenfassung

Zusammenfassung

- **sehr flexibler Ansatz**

Zusammenfassung

- sehr flexibler Ansatz
  - Abschächen der Annahme “perfekte Verschlüsselung”

Zusammenfassung

- sehr flexibler Ansatz
  - Abschächen der Annahme “perfekte Verschlüsselung”
  - unterschiedlich mächtige Angreifer

Zusammenfassung

- sehr flexibler Ansatz
  - Abschächen der Annahme “perfekte Verschlüsselung”
  - unterschiedlich mächtige Angreifer
  - unterschiedliche Prüfungen

Zusammenfassung

- sehr flexibler Ansatz
  - Abschächen der Annahme “perfekte Verschlüsselung”
  - unterschiedlich mächtige Angreifer
  - unterschiedliche Prüfungen
- minimal bzgl. impliziter Annahmen

Zusammenfassung

- sehr flexibler Ansatz
  - Abschächen der Annahme “perfekte Verschlüsselung”
  - unterschiedlich mächtige Angreifer
  - unterschiedliche Prüfungen
- minimal bzgl. impliziter Annahmen
- tool-unterstützt