VALIKRYPT Workshop

Verifikation kryptographischer Protokolle

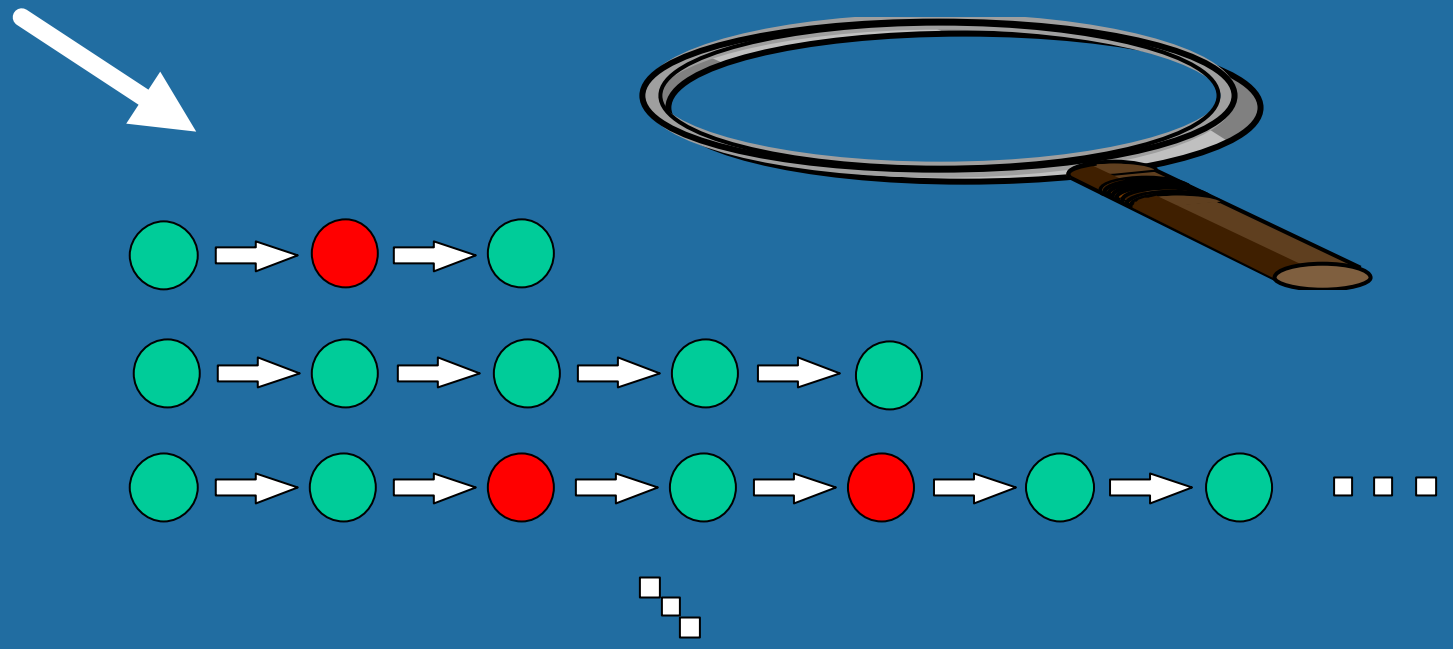
Lassaad Cheikhrouhou

Frankfurt, 30.09.2003

Source: Cheikhrouhou

Trace-basierte Analyse

1. $A \rightarrow S$: $A.B.Na$
2. $S \rightarrow A$: $\{Na.B.Kab.\{Kab.A\}K_{BS}\}K_{AS}$
3. $A \rightarrow B$: $\{Kab.A\}K_{BS}$
4. $B \rightarrow A$: $\{Nb\}Kab$
5. $A \rightarrow B$: $\{Nb.Nb\}Kab$



Source: Cheikhrouhou

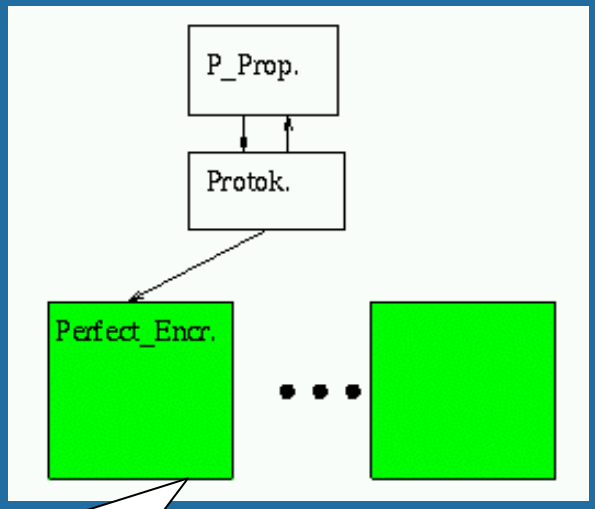
Spezifikation des Protokolls

1. $A \rightarrow S \;:\; A.B.Na$
2. $S \rightarrow A \;:\; \{Na.B.Kab.\{Kab.A\}K_{BS}\}K_{AS}$
3. $A \rightarrow B \;:\; \{Kab.A\}K_{BS}$
4. $B \rightarrow A \;:\; \{Nb\}Kab$
5. $A \rightarrow B \;:\; \{Nb.Nb\}Kab$

CAPSL

VSE–CAPSL

VSE-SL

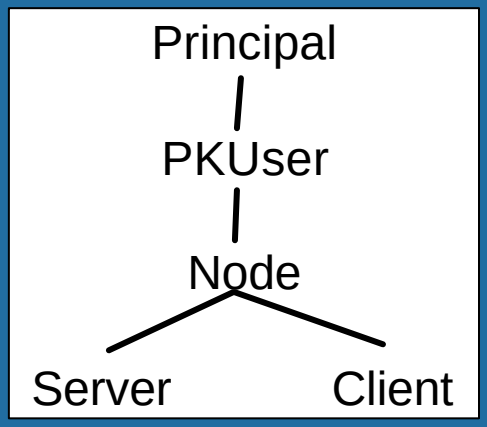


secureAg(.) ? friend(.) ? {spy}
agent(.) ? key(.) ? … ? crypt(.,.)
Says(.,.,.) ? Gets(.,.) ? Notes(.,.)
parts, analz, isSynth

Source: Cheikhrouhou

VSE-CAPSL (Datentypen)

- Nachrichten: (Nonce, Skey, …| cat(.,.), …)
- Protokollteilnehmer:



- Schlüssel: (csk(C,n) = ssk(S,C,n), msk(N,N',n) ? Skey | pk(A,n), sk(A,n) ? Pkey)
- Krypto. Funktionen: (se(Skey,Field), ped(Pkey,Field), …)

Source: Cheikhrouhou

Spezifikation in VSE-CAPSL

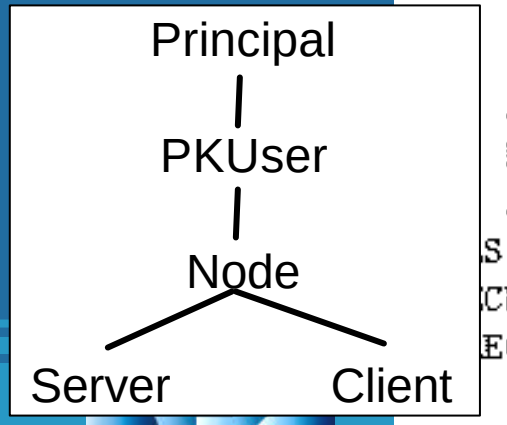
1. $A \rightarrow S$: A, B, Na
2. $S \rightarrow A$: $\{Na, B, Kab, \{Kab, A\}K_{BS}\}K_{AS}$
3. $A \rightarrow B$: $\{Kab, A\}K_{BS}$
4. $B \rightarrow A$: $\{Nb\}Kab$
5. $A \rightarrow B$: $\{Nb, Nb\}Kab$

Roles:

A: $-B, -S, compromised$
B: $-A, -S, compromised$
S: $-A, -B, 1$

?

```
PROTOCOL NSch_sym;
CONSTANTS
  S : Server;
VARIABLES
  A, B : Client;
  Na, Nb : Nonce, CRYPTO;
  Kab : Skey, CRYPTO, FRESH;
  F : Field;
ASSUMPTIONS
  HOLDS A: B;
  not(eqn(A,B));
MESSAGES
  1. A -> S: A, B, Na;
  2. S -> A: {Na, B, Kab,
                {Kab, A}ssk(S,B,0)%F}
               (ssk(S,A,0)%csk(A,0));
     A -> B: F%{Kab, A}csk(B,0);
     B -> A: {Nb}Kab;
     A -> B: {Nb, Nb}Kab;
S
CRET Kab;
ECEDES B: A | Nb, Kab;
```



Source: Cheikhrouhou

VSE-CAPSL ? VSE-SL


```
PROTOCOL NSch_sym;

...

MESSAGES
  1. A -> S: A, B, Na;
  2. S -> A: {Na, B, Kab,
               {Kab, A}ssk(S,B,0)%F}
              (ssk(S,A,0)%csk(A,0));
  3. A -> B: F%{Kab, A}csk(B,0);
  4. B -> A: {Nb}Kab;
  5. A -> B: {Nb, Nb}Kab;
GOALS

...

END;
```

```
THEORY T_NSch_sym
 USING TProtocol

 ...

 NSch_symNull : NSch_sym(nullEvent);
 NSch_symAdd : NSch_sym(addEvent(ev,evs)) <->
               (NSch_sym(evs) AND
                (NSch_sym_Says1(ev,evs) OR
                 NSch_sym_Says2(ev,evs) OR
                 NSch_sym_Says3(ev,evs) OR
                 NSch_sym_Says4(ev,evs) OR
                 NSch_sym_Says5(ev,evs) OR
                 NSch_sym_Oops1(ev,evs) OR
                 Gets_event(ev,evs) OR
                 Fake_event(ev,evs)));

 ...

THEORYEND
```

Source: Cheikhrouhou

VSE-CAPSL ? VSE-SL


```
PROTOCOL NSch_sym;
CONSTANTS
  S : Server;
VARIABLES
  A, B : Client;
  Na, Nb : Nonce, CRYPTO;
  Kab : Skey, CRYPTO, FRESH;
  F : Field;
ASSUMPTIONS
...
MESSAGES
  1. A -> S :  A, B, Na;
  2. S -> A :  {Na, B, Kab,
{Kab, A}ssk(S, B, 0)%F}ssk(S, A, 0)%csk(A, 0);
...
END;
```

```
THEORY T_NSch_sym

 ...

 /* Step2 : */
 NSch_symSays2 :
    NSch_sym_Says2(ev,evs) <->
    EX A, B, Na, Kab:
     (NOT msgIN(key(Kab), used(evs)) AND
      sessKey(Kab) AND
      eventIN(Gets(S,pair(agent(A),
                pair(agent(B),nonce(Na)))),evs) AND
      ev = Says(S,A,crypt(csk(A,0),pair(nonce(Na),
           pair(agent(B),pair(key(Kab),
           crypt(csk(B,0),pair(key(Kab),
           agent(A))))))))
     );

 ...


THEORYEND
```

Source: Cheikhrouhou

Protokolleigenschaften

- Vertraulichkeit (SECRET X)
- Authentifizierung (PRECEDES A : B | V1, V2, …)
  - Authentifizierung von A für B (PRECEDES A : B)
  - Agreement (aus der Sicht von B) mit A auf V1, V2, …
  - basiert auf Authentizität empfangener Nachrichtenteile

Source: Cheikhrouhou

Vertraulichkeit (Nonce, Session-Schlüssel)

- Ursprungs-Event von X: sayEv(X)
- A, B, ... erhalten X und sind nicht vom Typ Server
- A, B, ... kommen in sayEv(X) vor
- X kommt nur in mit long-term oder öffentlichen Schlüsseln verschlüsselten Nachrichtenteilen vor
- spy kann X nur über ein Oops-Event (oops(X)) erhalten

?

$$(P(\mathtt{evs})\ \mathtt{AND}\ sayEv(X) \in \mathtt{evs}\ \mathtt{AND}\ \mathtt{NOT}\ \mathtt{isBad}(A)\ \mathtt{AND}\ \mathtt{NOT}\ \mathtt{isBad}(B)\ \ldots\ \mathtt{AND}\ \mathtt{NOT}\ \mathtt{EX}\ x\text{:}\ oops(X) \in \mathtt{evs}) \rightarrow X \notin \mathtt{analz}(\mathtt{spies}(\mathtt{evs}))$$

Source: Cheikhrouhou

Vertraulichkeit (Kab)

```
PROTOCOL NSch_sym;
CONSTANTS ...
VARIABLES
  A, B : Client;
  Na, Nb : Nonce, CRYPTO;
  Kab : Skey, CRYPTO, FRESH;
...
MESSAGES
  1. A → S : A,B,Na;
  2. S → A : {Na,B,Kab,{Kab,A}csk(B,0)%F}csk(A,0);
  3. A → B : F%{Kab,A}csk(B,0);
  4. B → A : {Nb}Kab;
  5. A → B : {Nb,Nb}Kab;
GOALS
  SECRET Kab;
...
END;
```

$(P(\mathtt{evs})\ \mathtt{AND}\ sayEv(X) \in \mathtt{evs}\ \mathtt{AND}\ \mathtt{NOT}\ \mathtt{isBad}(A)$
$\mathtt{AND}\ \mathtt{NOT}\ \mathtt{isBad}(B)\ \ldots\ \mathtt{AND}\ \mathtt{NOT}\ \mathtt{EX}\ x\colon oops(X) \in \mathtt{evs})$
$\rightarrow\ X \notin \mathtt{analz}(\mathtt{spies}(\mathtt{evs}))$

?

$(\mathtt{NSch\_sym}(\mathtt{evs})\ \mathtt{AND}$
$\mathtt{Says}(\mathtt{S}, \mathtt{A}, \{\mathtt{Na}, \mathtt{B}, \mathtt{Kab}, \{\mathtt{Kab}, \mathtt{A}\}_{\mathtt{csk}(\mathtt{B},0)}\}_{\mathtt{csk}(\mathtt{A},0)}) \in \mathtt{evs}$
$\mathtt{AND\ NOT\ isBad}(\mathtt{A})\ \mathtt{AND\ NOT\ isBad}(\mathtt{B})\ \mathtt{AND}$
$\mathtt{NOT\ EX\ Nb}\ : \mathtt{Notes}(\mathtt{spy}, \{\mathtt{Na}, \mathtt{Nb}, \mathtt{Kab}\}) \in \mathtt{evs})$
$\rightarrow\ \mathtt{Kab} \notin \mathtt{analz}(\mathtt{spies}(\mathtt{evs}))$

Source: Cheikhrouhou

Unterstützung der Verifikation

- Lemmata:
  - Possibility Eigenschaft (Protokollmodell nicht restriktiv)
  - elementare Regularity-Lemmata
  - Unicity-Theoreme
  - Forwarding Lemmata
  - Compromise-Thm. für Session-Schlüssel
- Heuristiken basierend auf:
  - Beweisskizzen
  - typischen Fällen

Source: Cheikhrouhou

Forwarding Lemmata

- Weiterleiten eines Nachrichtenteils *X* aus einer empfangenen Nachricht M_x : Vor diesem Schritt ist X bereits in
  - parts(spies(evs))
  - analz(spies(evs)), falls *X* unverschlüsselt in M_x vorkommt

Bsp. Schritt 3 in NSch_sym:

$$2.\ S -> A :\ \{Na, B, Kab, F\}csk(A, 0);$$
$$3.\ A -> B :\ F$$

X ? parts(spies(Says(A,B,F)#evs))

$$(\mathtt{NSch\_sym(evs)}\ \mathtt{AND}\ \mathtt{Gets}(A, \{Na, B, Kab, F\}csk(A, 0)) \in \mathtt{evs}) \rightarrow F \in \mathtt{parts}(spies(evs))$$

X ? parts(spies(evs)) (Vorauss. der Ind.-Hyp.)

Source: Cheikhrouhou

Compromise-Theorem für Session-Schl.

- Ein Session-Schlüssel erlaubt dem Angreifer, keine weiteren Schlüssel zu erhalten.
- Voraussetzung: Schlüssel werden nicht mit Session-Schlüsseln verschlüsselt.

⇨ erlaubt Fallunterscheidung bei

$$K \in analz(\boldsymbol{K} \in spies(evs))$$

- $K \in \boldsymbol{K}$ (häufig durch Widerspruch)
- $K \in analz(spies(evs))$ (Vorauss. der Ind.-Hyp.)

Source: Cheikhrouhou

VSE-CAPSL ? VSE-SL

```
THEORY T_NSch_sym_Properties

 ...

 Forwarding2_3 :
   (NSch_sym(evs) AND
    eventIN(Gets(A,crypt(csk(A,0),pair(nonce(Na),
            pair(agent(B),pair(key(Kab),F)))))),evs)
   ) -> msgIN(F,parts(gotInfo(spy,evs)));

 ...

 SecretKab :
   (NSch_sym(evs) AND
    eventIN(Says(S,A,crypt(csk(A,0),pair(nonce(Na),
            pair(agent(B),pair(key(Kab),crypt(csk(B,0),
            pair(key(Kab),agent(A)))))))),evs) AND
    NOT isBad(A)  AND  NOT isBad(B)  AND
    NOT EX Nb : eventIN(Notes(spy,pair(nonce(Na),
                        pair(nonce(Nb),key(Kab)))),evs)
   ) -> NOT msgIN(key(Kab), analz(gotInfo(spy, evs)))

THEORYEND
```

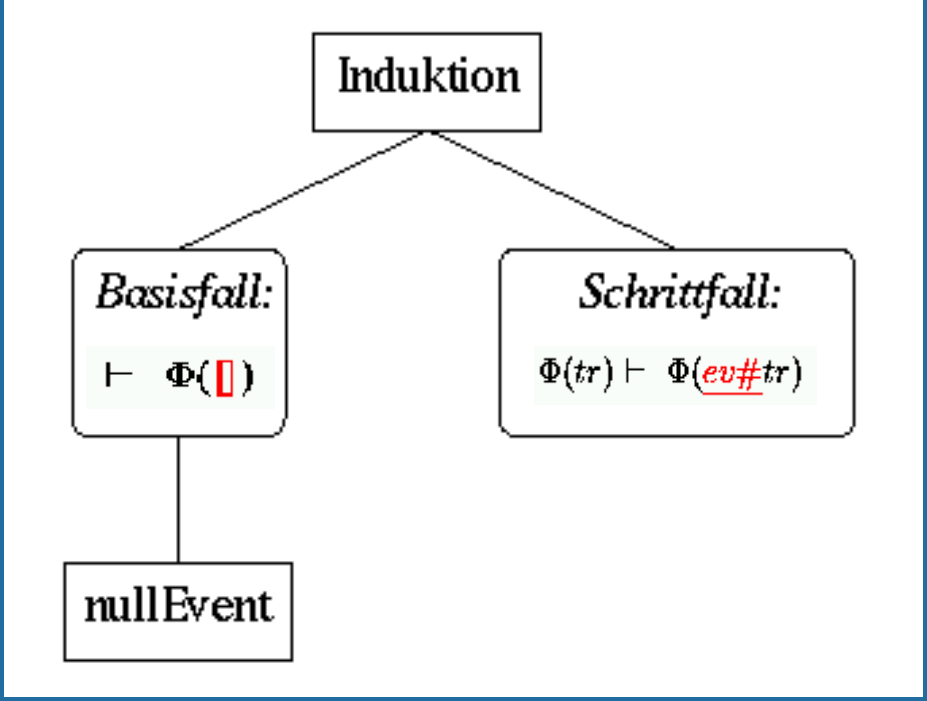
Source: Cheikh

Beweisheuristiken:

- Erste Schritte in Induktionsbeweisen:
  - Basis-Fall: kanonisch
  - Shritt-Fall: Fallunterscheidung bzgl. Protokollschritten
- Fallabhängige Beweistaktiken
- Heuristische Kontrolle: automatisches Beweisen typischer Eigenschaften (*Prove-Secrecy*)

Source: Cheikhrouhou

Induktion über Trace



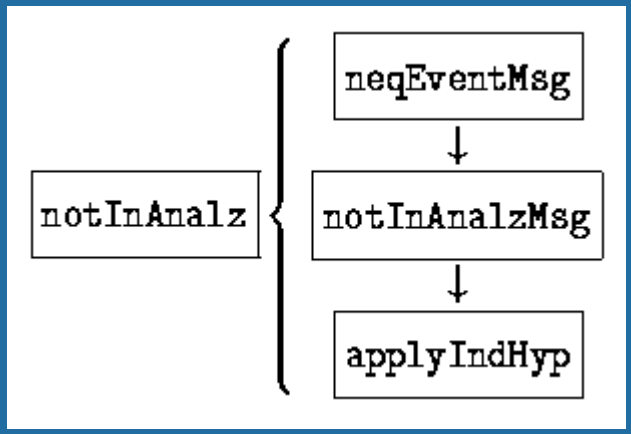
Bsp: $\mathtt{Says}(S, A, \{Na, B, Kab, \{Kab, A\}_{\mathtt{csk}(B)}\}_{\mathtt{csk}(A)}) \in []$

$$\frac{ev \in [], \overline{\Phi} \vdash \overline{\Psi}}{\vdash \bot}\ \mathtt{nullEvent}$$

Source: Cheikhrouhou

notInAnalz

```
PROTOCOL NSch_sym;
CONSTANTS ...
MESSAGES
  1. A → S : A,B,Na;
  2. S → A : {Na,B,Kab,{Kab,A}csk(B,0)%F}csk(A,0);
  3. A → B : F%{Kab,A}csk(B,0);
  4. B → A : {Nb}Kab;
  5. A → B : {Nb,Nb}Kab;
GOALS ...
END;
```



$$\frac{M \in \mathtt{analz}(\mathtt{spies}(ev(\overline{A}, M')\#tr)), \overline{\Phi} \vdash \overline{\Psi};\ M \not\sqsubseteq_{typ} M'}{M \in \mathtt{analz}(\mathtt{spies}(tr)), \overline{\Phi} \vdash \overline{\Psi}}\ \mathtt{notInAnalzMsg}$$

Bsp: $Kab \in \mathtt{analz}(\mathtt{spies}(\mathtt{Says}(A', S, \{A', B', Na'\})\#tr))$

Source: Cheikhrouhou

Zusammenfassung

- Spezifikation:
  - in VSE-CAPSL → Protokollmodell und Teil der Eigenschaften in VSE-SL
  - restliche Eigenschaften in VSE-SL (Methodik)
- Verifikation:
  - Heuristiken für typische Eigenschaften/Fälle
  - Interaktion für verbliebene Beweisziele

Source: Cheikhrouhou