

Modellierung kryptographischer Protokolle durch Induktion

Lukas Bulwahn

Fakultät für Informatik
Technische Universität München

22. Juli 2006

- 1 Das Otway-Rees Protokoll
- 2 Formales Modell
 - Agenten und Nachrichten
 - Events und Traces, Wissen des Spions
 - Formalisierung des Otway-Rees Protokolls
 - Operatoren auf Nachrichten
- 3 Eigenschaften eines korrekten Protokolls
 - possibility property und regularity lemmas
 - unicity theorems
 - secrecy theorem
 - authenticity guarantees
- 4 Angriff und Reparatur des Otway-Rees Protokolls
 - mögliche Angriff auf das Otway-Rees Protokoll
 - Reparatur des Otway-Rees Protokolls

Vorbemerkungen

- Kryptographie wird als sicher angenommen.
- Das Otway-Rees Protokoll arbeitet mit symmetrischen Schlüsseln.
- Alice besitzt den Schlüssel K_a , Bob besitzt den Schlüssel K_b .
- Der Server besitzt die Langzeitschlüssel aller Agenten.
- Ziel des Protokolls: Der Server verteilt einen Session Key an 2 Teilnehmer mit dem weitere Nachrichten sicher ausgetauscht werden können.

Vorbemerkungen

- Kryptographie wird als sicher angenommen.
- Das Otway-Rees Protokoll arbeitet mit symmetrischen Schlüsseln.
- Alice besitzt den Schlüssel K_a , Bob besitzt den Schlüssel K_b .
- Der Server besitzt die Langzeitschlüssel aller Agenten.
- Ziel des Protokolls: Der Server verteilt einen Session Key an 2 Teilnehmer mit dem weitere Nachrichten sicher ausgetauscht werden können.

Vorbemerkungen

- Kryptographie wird als sicher angenommen.
- Das Otway-Rees Protokoll arbeitet mit symmetrischen Schlüsseln.
- Alice besitzt den Schlüssel K_a , Bob besitzt den Schlüssel K_b .
- Der Server besitzt die Langzeitschlüssel aller Agenten.
- Ziel des Protokolls: Der Server verteilt einen Session Key an 2 Teilnehmer mit dem weitere Nachrichten sicher ausgetauscht werden können.

Vorbemerkungen

- Kryptographie wird als sicher angenommen.
- Das Otway-Rees Protokoll arbeitet mit symmetrischen Schlüsseln.
- Alice besitzt den Schlüssel K_a , Bob besitzt den Schlüssel K_b .
- Der Server besitzt die Langzeitschlüssel aller Agenten.
- Ziel des Protokolls: Der Server verteilt einen Session Key an 2 Teilnehmer mit dem weitere Nachrichten sicher ausgetauscht werden können.

Vorbemerkungen

- Kryptographie wird als sicher angenommen.
- Das Otway-Rees Protokoll arbeitet mit symmetrischen Schlüsseln.
- Alice besitzt den Schlüssel K_a , Bob besitzt den Schlüssel K_b .
- Der Server besitzt die Langzeitschlüssel aller Agenten.
- Ziel des Protokolls: Der Server verteilt einen Session Key an 2 Teilnehmer mit dem weitere Nachrichten sicher ausgetauscht werden können.

Das Otway-Rees Protokoll

Server



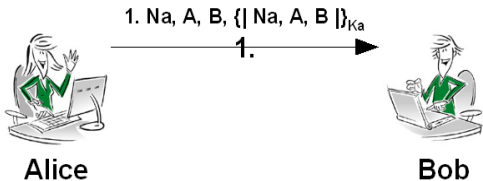
Alice



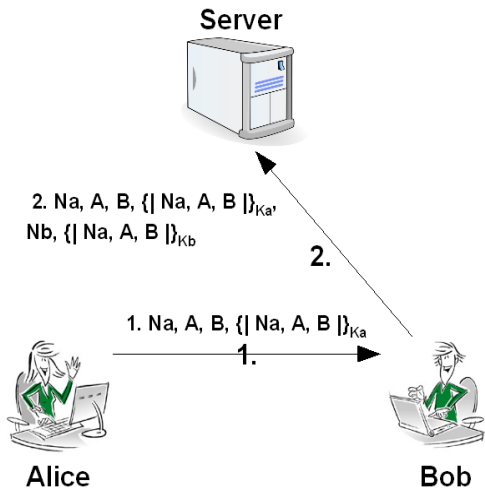
Bob

Das Otway-Rees Protokoll

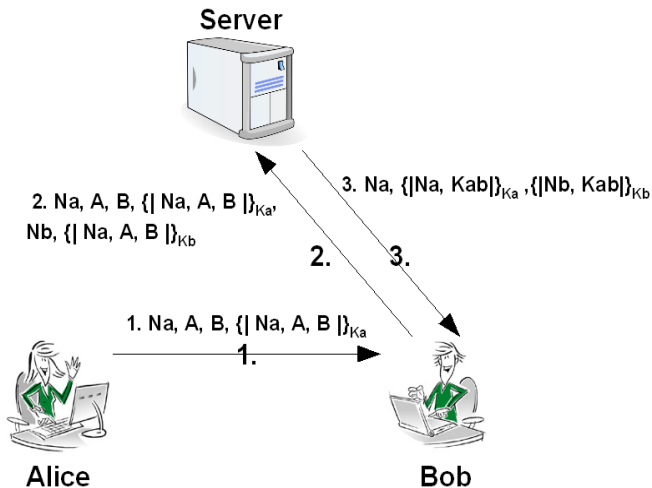
Server



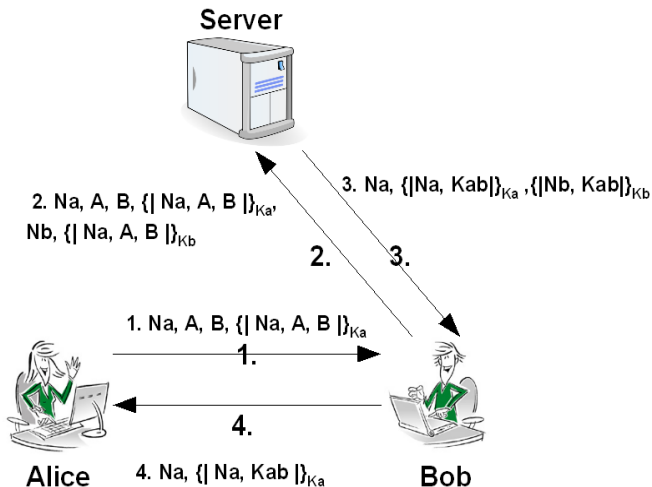
Das Otway-Rees Protokoll



Das Otway-Rees Protokoll



Das Otway-Rees Protokoll



- 1 Das Otway-Rees Protokoll
- 2 Formales Modell
 - Agenten und Nachrichten
 - Events und Traces, Wissen des Spions
 - Formalisierung des Otway-Rees Protokolls
 - Operatoren auf Nachrichten
- 3 Eigenschaften eines korrekten Protokolls
 - possibility property und regularity lemmas
 - unicity theorems
 - secrecy theorem
 - authenticity guarantees
- 4 Angriff und Reparatur des Otway-Rees Protokolls
 - mögliche Angriff auf das Otway-Rees Protokoll
 - Reparatur des Otway-Rees Protokolls

Agenten und Nachrichten

Es gibt drei Arten von Agenten: den Server, die freundlichgesinnten Agenten, und den Angreifer. Agenten werden wie folgt formalisiert:

Definition von Agenten

```
datatype agent = Server | Friend nat | Spy
```

Agenten und Nachrichten

Nachrichten werden entsprechend definiert:

Definition von Nachrichten

```
msg = Agent agent
| Nonce nat
| Key key
| { | msg, msg | }
| msgkey
```

Events und Traces

Eine Kommunikation besteht aus einer Folge von Ereignissen (Senden, Empfangen, "Wahrnehmen").

Definition von Event und Traces

```
event = Says agent agent msg  
| Gets agent msg  
| Notes agent msg
```

Traces (Ereignisfolgen) sind nun einfach Listen von Ereignissen.

Events und Traces

Eine Kommunikation besteht aus einer Folge von Ereignissen (Senden, Empfangen, "Wahrnehmen").

Definition von Event und Traces

```
event = Says agent agent msg  
| Gets agent msg  
| Notes agent msg
```

Traces (Ereignisfolgen) sind nun einfach Listen von Ereignissen.

SharedKeys und infiltrierte Agenten

- Jedem Agenten wird ein Schlüssel zugeordnet.
 $\text{shrK} : \text{agent} \rightarrow \text{key}$
 (ist injektiv, symmetrische Schlüssel)
 $K_a \equiv \text{shrK } A$ und $K_b \equiv \text{shrK } B$
- Agenten können von Spion infiltriert worden sein.
 $\text{bad} : \text{agent set}$
 (Spy \in bad, Server \notin bad)

SharedKeys und infiltrierte Agenten

- Jedem Agenten wird ein Schlüssel zugeordnet.
 $\text{shrK} : \text{agent} \rightarrow \text{key}$
(ist injektiv, symmetrische Schlüssel)
 $K_a \equiv \text{shrK } A$ und $K_b \equiv \text{shrK } B$
- Agenten können von Spion infiltriert worden sein.
 $\text{bad} : \text{agent set}$
($\text{Spy} \in \text{bad}$, $\text{Server} \notin \text{bad}$)

Weitere Anmerkungen

- Das Wissen eines Agenten wird durch eine Menge von Nachrichten repräsentiert.
- Der Spion kennt alle Nachrichten, die über das Netzwerk übertragen wurden.
 $\text{spies} : \text{event list} \rightarrow \text{msg set}$
- $\text{synth}(\text{analz}(\text{spies evs}))$ = Die Nachrichten, die der Spion aus analysierten Nachrichten erzeugen kann.
- used evs = alle bereits verbrauchten Noncen und Schlüssel

Formalisierung des Otway-Rees Protokolls

(siehe Beiblatt)

wichtige Punkte:

- Nil steht für die leere Ereignisfolge. OR1-4 sind die bereits erwähnten Schritte des Otway-Rees Protokolls.
- Fake steht für die betrügerischen Nachrichten, die der Angreifer erzeugen kann.
- Die Oops Regel steht für den unbeabsichtigten Verlust des Session Keys.
- Nachrichten werden nicht an sich selbst gesendet.
- die Noncen N_a und N_b sind unverbraucht. (freshness)
- die Überprüfung von Gleichheit von Noncen ist hier implizit gegeben.
- Ein Agent kann mehr als einmal auf eine Nachricht antworten.

Formalisierung des Otway-Rees Protokolls

(siehe Beiblatt)

wichtige Punkte:

- Nil steht für die leere Ereignisfolge. OR1-4 sind die bereits erwähnten Schritte des Otway-Rees Protokolls.
- Fake steht für die betrügerischen Nachrichten, die der Angreifer erzeugen kann.
- Die Oops Regel steht für den unbeabsichtigten Verlust des Session Keys.
- Nachrichten werden nicht an sich selbst gesendet.
- die Noncen N_a und N_b sind unverbraucht. (freshness)
- die Überprüfung von Gleichheit von Noncen ist hier implizit gegeben.
- Ein Agent kann mehr als einmal auf eine Nachricht antworten.

Formalisierung des Otway-Rees Protokolls

(siehe Beiblatt)

wichtige Punkte:

- Nil steht für die leere Ereignisfolge. OR1-4 sind die bereits erwähnten Schritte des Otway-Rees Protokolls.
- Fake steht für die betrügerischen Nachrichten, die der Angreifer erzeugen kann.
- Die Oops Regel steht für den unbeabsichtigten Verlust des Session Keys.
- Nachrichten werden nicht an sich selbst gesendet.
- die Noncen N_a und N_b sind unverbraucht. (freshness)
- die Überprüfung von Gleichheit von Noncen ist hier implizit gegeben.
- Ein Agent kann mehr als einmal auf eine Nachricht antworten.

Formalisierung des Otway-Rees Protokolls

(siehe Beiblatt)

wichtige Punkte:

- Nil steht für die leere Ereignisfolge. OR1-4 sind die bereits erwähnten Schritte des Otway-Rees Protokolls.
- Fake steht für die betrügerischen Nachrichten, die der Angreifer erzeugen kann.
- Die Oops Regel steht für den unbeabsichtigten Verlust des Session Keys.
- Nachrichten werden nicht an sich selbst gesendet.
- die Noncen N_a und N_b sind unverbraucht. (freshness)
- die Überprüfung von Gleichheit von Noncen ist hier implizit gegeben.
- Ein Agent kann mehr als einmal auf eine Nachricht antworten.

Formalisierung des Otway-Rees Protokolls

(siehe Beiblatt)

wichtige Punkte:

- Nil steht für die leere Ereignisfolge. OR1-4 sind die bereits erwähnten Schritte des Otway-Rees Protokolls.
- Fake steht für die betrügerischen Nachrichten, die der Angreifer erzeugen kann.
- Die Oops Regel steht für den unbeabsichtigten Verlust des Session Keys.
- Nachrichten werden nicht an sich selbst gesendet.
- die Noncen N_a und N_b sind unverbraucht. (freshness)
- die Überprüfung von Gleichheit von Noncen ist hier implizit gegeben.
- Ein Agent kann mehr als einmal auf eine Nachricht antworten.

Formalisierung des Otway-Rees Protokolls

(siehe Beiblatt)

wichtige Punkte:

- Nil steht für die leere Ereignisfolge. OR1-4 sind die bereits erwähnten Schritte des Otway-Rees Protokolls.
- Fake steht für die betrügerischen Nachrichten, die der Angreifer erzeugen kann.
- Die Oops Regel steht für den unbeabsichtigten Verlust des Session Keys.
- Nachrichten werden nicht an sich selbst gesendet.
- die Noncen N_a und N_b sind unverbraucht. (freshness)
- die Überprüfung von Gleichheit von Noncen ist hier implizit gegeben.
- Ein Agent kann mehr als einmal auf eine Nachricht antworten.

Formalisierung des Otway-Rees Protokolls

(siehe Beiblatt)

wichtige Punkte:

- Nil steht für die leere Ereignisfolge. OR1-4 sind die bereits erwähnten Schritte des Otway-Rees Protokolls.
- Fake steht für die betrügerischen Nachrichten, die der Angreifer erzeugen kann.
- Die Oops Regel steht für den unbeabsichtigten Verlust des Session Keys.
- Nachrichten werden nicht an sich selbst gesendet.
- die Noncen N_a und N_b sind unverbraucht. (freshness)
- die Überprüfung von Gleichheit von Noncen ist hier implizit gegeben.
- Ein Agent kann mehr als einmal auf eine Nachricht antworten.

Anfangszustand des Systems

- Der Anfangszustand wird folgendermaßen modelliert:
 $\text{initState} : \text{agent} \rightarrow \text{msg set}$
- Jeder normale Agent hat nur seinen eigenen Schlüssel.
 $\text{initState} (\text{Friend } i) = \{\text{Key} (\text{shrK}(\text{Friend } i))\}$
- Der Server hat alle Schlüssel.
 $\text{initState} \text{ Server} = \{\text{Key} (\text{shrK } a) \mid a \text{ ist Agent}\}$
- Der Spion hat alle Schlüssel der infiltrierten Agenten.
 $\text{initState} \text{ Spy} = \{(\text{Key} (\text{shrK } a) \mid a \in \text{bad})\}$

Anfangszustand des Systems

- Der Anfangszustand wird folgendermaßen modelliert:
 $\text{initState} : \text{agent} \rightarrow \text{msg set}$
- Jeder normale Agent hat nur seinen eigenen Schlüssel.
 $\text{initState} (\text{Friend } i) = \{\text{Key} (\text{shrK}(\text{Friend } i))\}$
- Der Server hat alle Schlüssel.
 $\text{initState} \text{ Server} = \{\text{Key} (\text{shrK } a) \mid a \text{ ist Agent}\}$
- Der Spion hat alle Schlüssel der infiltrierten Agenten.
 $\text{initState} \text{ Spy} = \{(\text{Key} (\text{shrK } a) \mid a \in \text{bad})\}$

Anfangszustand des Systems

- Der Anfangszustand wird folgendermaßen modelliert:
 $\text{initState} : \text{agent} \rightarrow \text{msg set}$
- Jeder normale Agent hat nur seinen eigenen Schlüssel.
 $\text{initState} (\text{Friend } i) = \{\text{Key} (\text{shrK}(\text{Friend } i))\}$
- Der Server hat alle Schlüssel.
 $\text{initState} \text{ Server} = \{\text{Key} (\text{shrK } a) \mid a \text{ ist Agent}\}$
- Der Spion hat alle Schlüssel der infiltrierten Agenten.
 $\text{initState} \text{ Spy} = \{(\text{Key} (\text{shrK } a) \mid a \in \text{bad})\}$

Anfangszustand des Systems

- Der Anfangszustand wird folgendermaßen modelliert:
 $\text{initState} : \text{agent} \rightarrow \text{msg set}$
- Jeder normale Agent hat nur seinen eigenen Schlüssel.
 $\text{initState} (\text{Friend } i) = \{\text{Key} (\text{shrK}(\text{Friend } i))\}$
- Der Server hat alle Schlüssel.
 $\text{initState} \text{ Server} = \{\text{Key} (\text{shrK } a) \mid a \text{ ist Agent}\}$
- Der Spion hat alle Schlüssel der infiltrierten Agenten.
 $\text{initState} \text{ Spy} = \{(\text{Key} (\text{shrK } a) \mid a \in \text{bad})\}$

Was kann der Spion lernen?

Der Spion kennt alle Nachrichten, die über das Netzwerk übertragen wurden.

Definition von spies

```

spies : event list → msg set
spies [] = initState Spy
spies ev#evs = case ev of
Says A B X ⇒ insert X (spies evs)
| Gets A X ⇒ spies evs
| Notes A X ⇒
if A ∈ bad then insert X (spies evs) else spies evs

```

Was kann der Spion lernen?

Der Spion kennt alle Nachrichten, die über das Netzwerk übertragen wurden.

Definition von spies

```

spies : event list → msg set
spies [] = initState Spy
spies ev#evs = case ev of
Says A B X ⇒ insert X (spies evs)
| Gets A X ⇒ spies evs
| Notes A X ⇒
if A ∈ bad then insert X (spies evs) else spies evs

```

Was kann der Spion lernen?

Der Spion kennt alle Nachrichten, die über das Netzwerk übertragen wurden.

Definition von spies

```

spies : event list → msg set
spies [] = initState Spy
spies ev#evs = case ev of
Says A B X ⇒ insert X (spies evs)
| Gets A X ⇒ spies evs
| Notes A X ⇒
if A ∈ bad then insert X (spies evs) else spies evs

```

Was kann der Spion lernen?

Der Spion kennt alle Nachrichten, die über das Netzwerk übertragen wurden.

Definition von spies

```

spies : event list → msg set
spies [] = initState Spy
spies ev#evs = case ev of
Says A B X ⇒ insert X (spies evs)
| Gets A X ⇒ spies evs
| Notes A X ⇒
if A ∈ bad then insert X (spies evs) else spies evs

```

Operatoren auf Nachrichten

Der parts Operator beschreibt die Menge aller Komponenten (parts) einer Nachricht, die aus einer Menge von Nachrichten möglicherweise wiederhergestellt werden können.

Definition von parts

parts : msg set \rightarrow msg set

$$\frac{X \in H}{X \in \text{parts } H}$$

$$\frac{\{|X,Y|\} \in \text{parts } H}{X \in \text{parts } H}$$

$$\frac{\{|X,Y|\} \in \text{parts } H}{Y \in \text{parts } H}$$

$$\frac{X_K \in \text{parts } H}{X \in \text{parts } H}$$

Operatoren auf Nachrichten

Der `analz` Operator definiert alle möglichen Komponenten von Nachrichten, die wirklich wiederhergestellt werden können.

Definition von `analz`

`analz` : msg set \rightarrow msg set

$$\frac{X \in H}{X \in \text{analz } H}$$

$$\frac{\{|X,Y|\} \in \text{analz } H}{X \in \text{analz } H}$$

$$\frac{\{|X,Y|\} \in \text{analz } H}{Y \in \text{analz } H}$$

$$\frac{X_K \in \text{analz } H \quad \text{Key}(\text{invKey } K) \in \text{analz } H}{X \in \text{analz } H}$$

Operatoren auf Nachrichten

Der synth Operator definiert welche Nachrichten aus einer Menge von Nachrichten erzeugt werden können.

Definition von synth

$\text{synth} : \text{msg set} \rightarrow \text{msg set}$

$$\frac{X \in H}{X \in \text{synth } H}$$

$$\frac{}{\text{Agent } \text{agt} \in \text{synth } H}$$

$$\frac{\begin{array}{l} X \in \text{synth } H \\ Y \in \text{synth } H \end{array}}{\{|X, Y|\} \in \text{synth } H}$$

$$\frac{\begin{array}{l} X \in \text{synth } H \\ \text{Key } (K) \in H \end{array}}{X_K \in \text{synth } H}$$

- 1 Das Otway-Rees Protokoll
- 2 Formales Modell
 - Agenten und Nachrichten
 - Events und Traces, Wissen des Spions
 - Formalisierung des Otway-Rees Protokolls
 - Operatoren auf Nachrichten
- 3 Eigenschaften eines korrekten Protokolls
 - possibility property und regularity lemmas
 - unicity theorems
 - secrecy theorem
 - authenticity guarantees
- 4 Angriff und Reparatur des Otway-Rees Protokolls
 - mögliche Angriff auf das Otway-Rees Protokoll
 - Reparatur des Otway-Rees Protokolls

possibility property

Die Einzelschritte greifen ineinander und ergeben ein Protokoll.

possibility property

- $B \neq \text{Server}$
- $K_{ab} \notin \text{used}[]$

$\implies \exists Na. \exists \text{evs} \in \text{otway}. \text{Says } B \ A \ \{|Na, \{|Na, Kab|\}_{K_a}|\} \in \text{set evs}$

possibility property

Die Einzelschritte greifen ineinander und ergeben ein Protokoll.

possibility property

- $B \neq \text{Server}$
- $K_{ab} \notin \text{used} []$

$\implies \exists Na. \exists \text{evs} \in \text{otway}. \text{Says } B \ A \ \{|Na, \{|Na, Kab|\}_{K_a}|\} \in \text{set evs}$

regularity lemmas

regularity lemmas sind Aussagen, welche Nachrichtenteile über das Netzwerk gesendet werden.

Beispiel: Kein guter Agent versendet seinen sharedKey.

Spy see shrK

- $evs \in \text{otway}$

$\implies \text{Key}(\text{shrK } A) \in \text{parts}(\text{spies } evs) \iff A \in \text{bad}$

regularity lemmas

regularity lemmas sind Aussagen, welche Nachrichtenteile über das Netzwerk gesendet werden.

Beispiel: Kein guter Agent versendet seinen sharedKey.

Spy see shrK

- $evs \in otway$

$\implies Key(shrK A) \in parts(spies\ evs) \iff A \in bad$

unicity theorems

unicity theorems sind Aussagen darüber, welche Nachrichten einzigartig sind.

Beispiel: Die Nachricht OR3 ist einzigartig und ein Session Key ist wird nur einmal vergeben.

unique session keys

- $evs \in otway$
- Says Server B $\{ | Na, X, \{ | Nb, Kab | \}_{Kb} | \} \in set\ evs$
- Says Server B' $\{ | Na', X', \{ | Nb', Kab | \}_{Kb'} | \} \in set\ evs$

$\implies X=X' \wedge B=B' \wedge Na=Na' \wedge Nb=Nb'$

unicity theorems

unicity theorems sind Aussagen darüber, welche Nachrichten einzigartig sind.

Beispiel: Die Nachricht OR3 ist einzigartig und ein Session Key ist
wird nur einmal vergeben.

unique session keys

- $evs \in \text{otway}$
- Says Server B $\{ | Na, X, \{ | Nb, Kab | \}_{Kb} | \} \in \text{set } evs$
- Says Server B' $\{ | Na', X', \{ | Nb', Kab | \}_{Kb'} | \} \in \text{set } evs$

$\implies X=X' \wedge B=B' \wedge Na=Na' \wedge Nb=Nb'$

unicity theorems

unicity theorems sind Aussagen darüber, welche Nachrichten einzigartig sind.

Beispiel: Die Nachricht OR3 ist einzigartig und ein Session Key ist nur einmal vergeben.

unique session keys

- $evs \in \text{otway}$
- Says Server B $\{ | Na, X, \{ | Nb, Kab | \}_{Kb} | \} \in \text{set } evs$
- Says Server B' $\{ | Na', X', \{ | Nb', Kab | \}_{Kb'} | \} \in \text{set } evs$

$$\implies X=X' \wedge B=B' \wedge Na=Na' \wedge Nb=Nb'$$

secrecy theorem

Das secrecy theorem besagt, dass bestimmte Nachrichtenteile geheim sind.

Beispiel: Der Session-Key ist geheim.

secrecy theorem

initial state

initial state

initial state

initial state

initial state

initial state

$\Rightarrow K_{ab} \notin \text{analz}(\text{spies evs})$

secrecy theorem

Das secrecy theorem besagt, dass bestimmte Nachrichtenteile geheim sind.

Beispiel: Der Session-Key ist geheim.

secrecy theorem

Let K_{ab} be the session key

Let $K_{ab} = \text{key}(A, B, \text{nonce})$

$\Rightarrow K_{ab} \notin \text{analz}(\text{spies evs})$

secrecy theorem

Das secrecy theorem besagt, dass bestimmte Nachrichtenteile geheim sind.

Beispiel: Der Session-Key ist geheim.

secrecy theorem

- $evs \in otway$
- $A \notin bad$
- $B \notin bad$
- Says Server B $\{|Na, \{|Na, Kab|\}_{Ka}, \{|Nb, Kab|\}_{Kb}|\} \in set\ evs$
- Notes Spy $\{|Na, Nb, Kab|\} \notin set\ evs$

$\implies Kab \notin analz\ (spies\ evs)$

secrecy theorem

Das secrecy theorem besagt, dass bestimmte Nachrichtenteile geheim sind.

Beispiel: Der Session-Key ist geheim.

secrecy theorem

- $evs \in otway$
- $A \notin bad$
- $B \notin bad$
- Says Server B $\{ |Na, \{ |Na, Kab| \}_{Ka}, \{ |Nb, Kab| \}_{Kb} \} \in set\ evs$
- Notes Spy $\{ |Na, Nb, Kab| \} \notin set\ evs$

$\implies Kab \notin analz\ (spies\ evs)$

secrecy theorem

Das secrecy theorem besagt, dass bestimmte Nachrichtenteile geheim sind.

Beispiel: Der Session-Key ist geheim.

secrecy theorem

- $evs \in otway$
- $A \notin bad$
- $B \notin bad$
- Says Server B $\{ |Na, \{ |Na, Kab| \}_{Ka}, \{ |Nb, Kab| \}_{Kb} \} \in set\ evs$
- Notes Spy $\{ |Na, Nb, Kab| \} \notin set\ evs$

$\implies Kab \notin analz\ (spies\ evs)$

secrecy theorem

Das secrecy theorem besagt, dass bestimmte Nachrichtenteile geheim sind.

Beispiel: Der Session-Key ist geheim.

secrecy theorem

- $evs \in otway$
- $A \notin bad$
- $B \notin bad$
- Says Server B $\{ |Na, \{ |Na, Kab| \}_{Ka}, \{ |Nb, Kab| \}_{Kb} \} \in set\ evs$
- Notes Spy $\{ |Na, Nb, Kab| \} \notin set\ evs$

$\implies Kab \notin analz\ (spies\ evs)$

secrecy theorem

Das secrecy theorem besagt, dass bestimmte Nachrichtenteile geheim sind.

Beispiel: Der Session-Key ist geheim.

secrecy theorem

- $evs \in \text{otway}$
- $A \notin \text{bad}$
- $B \notin \text{bad}$
- Says Server B $\{ |Na, \{ |Na, Kab| \}_{Ka}, \{ |Nb, Kab| \}_{Kb}| \} \in \text{set } evs$
- Notes Spy $\{ |Na, Nb, Kab| \} \notin \text{set } evs$

$\implies Kab \notin \text{analz (spies } evs)$

secrecy theorem

Das secrecy theorem besagt, dass bestimmte Nachrichtenteile geheim sind.

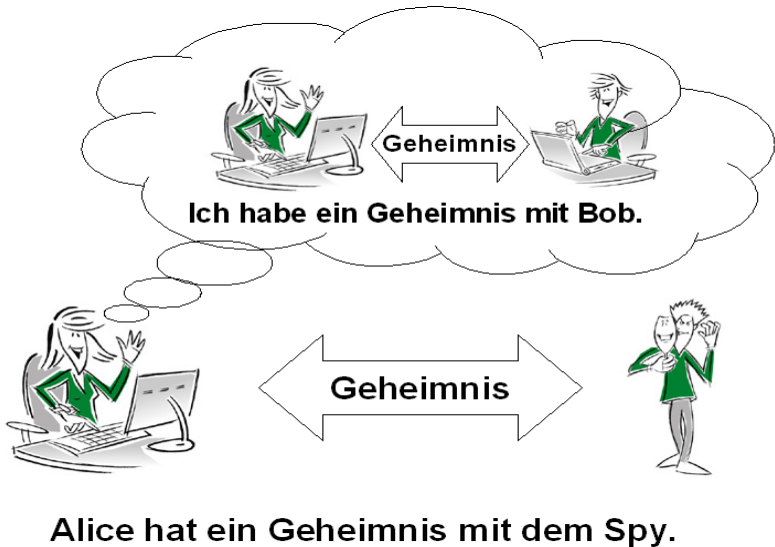
Beispiel: Der Session-Key ist geheim.

secrecy theorem

- $evs \in otway$
- $A \notin bad$
- $B \notin bad$
- Says Server B $\{|Na, \{|Na, Kab|\}_{Ka}, \{|Nb, Kab|\}_{Kb}|\} \in set\ evs$
- Notes Spy $\{|Na, Nb, Kab|\} \notin set\ evs$

$\implies Kab \notin analz\ (spies\ evs)$

Ist das Protokoll nun sicher?



authenticity guarantees

Die Echtheit besagt die Nachrichten, die ich bekomme, stammen von dem, von dem ich dies erwarte.

Beispiele:

- Authentizitätsgarantie für A: Wenn A OR4 erhält, wurde OR3 vom Server an B gesendet.
- Authentizitätsgarantie für B: Wenn B OR3 erhält, wurde OR3 vom Server an B gesendet.
- Die abschließende Garantie: Wenn A die Nachricht OR1 versendet hat und die Nachricht OR4 erhält, so muss die Nachricht OR3 vom Server an B gesendet worden sein.

authenticity guarantees

Die Echtheit besagt die Nachrichten, die ich bekomme, stammen von dem, von dem ich dies erwarte.

Beispiele:

- Authentizitätsgarantie für A: Wenn A OR4 erhält, wurde OR3 vom Server an B gesendet.
- Authentizitätsgarantie für B: Wenn B OR3 erhält, wurde OR3 vom Server an B gesendet.
- Die abschließende Garantie: Wenn A die Nachricht OR1 versendet hat und die Nachricht OR4 erhält, so muss die Nachricht OR3 vom Server an B gesendet worden sein.

authenticity guarantees

Die Echtheit besagt die Nachrichten, die ich bekomme, stammen von dem, von dem ich dies erwarte.

Beispiele:

- Authentizitätsgarantie für A: Wenn A OR4 erhält, wurde OR3 vom Server an B gesendet.
- Authentizitätsgarantie für B: Wenn B OR3 erhält, wurde OR3 vom Server an B gesendet.
- Die abschließende Garantie: Wenn A die Nachricht OR1 versendet hat und die Nachricht OR4 erhält, so muss die Nachricht OR3 vom Server an B gesendet worden sein.

Authentizitätsgarantie für A

- $evs \in \text{otway}$
- $A \notin \text{bad}$
- $\text{Says } A \ B \ \{ |Na, A, B, \{ |Na, A, B| \}_{K_a} | \} \in \text{set } evs$
- $\text{Says } B' \ A \ \{ |Na, \{ |Na, K_{ab}| \}_{K_a} | \} \in \text{set } evs$

$\implies \exists Nb. \text{Says Server } B$

$\{ |Na, \{ |Na, K_{ab}| \}_{K_a}, \{ |Nb, K_{ab}| \}_{K_b} | \} \in \text{set } evs$

Beweis für dieses Protokoll nicht möglich!

Authentizitätsgarantie für A

- $evs \in \text{otway}$
- $A \notin \text{bad}$
- $\text{Says } A \ B \ \{|Na, A, B, \{|Na, A, B|\}_{K_a}|\} \in \text{set } evs$
- $\text{Says } B' \ A \ \{|Na, \{|Na, K_{ab}|\}_{K_a}|\} \in \text{set } evs$

$\implies \exists Nb. \text{Says Server } B$

$\{|Na, \{|Na, K_{ab}|\}_{K_a}, \{|Nb, K_{ab}|\}_{K_b}|\} \in \text{set } evs$

Beweis für dieses Protokoll nicht möglich!

Authentizitätsgarantie für A

- $evs \in \text{otway}$
- $A \notin \text{bad}$
- $\text{Says } A \ B \ \{ |Na, A, B, \{ |Na, A, B| \}_{K_a}| \} \in \text{set } evs$
- $\text{Says } B' \ A \ \{ |Na, \{ |Na, K_{ab}| \}_{K_a}| \} \in \text{set } evs$

$\implies \exists Nb. \text{Says Server } B$

$\{ |Na, \{ |Na, K_{ab}| \}_{K_a}, \{ |Nb, K_{ab}| \}_{K_b}| \} \in \text{set } evs$

Beweis für dieses Protokoll nicht möglich!

Authentizitätsgarantie für A

- $evs \in \text{otway}$
- $A \notin \text{bad}$
- $\text{Says } A \ B \ \{ |Na, A, B, \{ |Na, A, B| \}_{K_a} | \} \in \text{set } evs$
- $\text{Says } B' \ A \ \{ |Na, \{ |Na, K_{ab}| \}_{K_a} | \} \in \text{set } evs$

$\implies \exists Nb. \text{Says Server } B$

$\{ |Na, \{ |Na, K_{ab}| \}_{K_a}, \{ |Nb, K_{ab}| \}_{K_b} | \} \in \text{set } evs$

Beweis für dieses Protokoll nicht möglich!

Authentizitätsgarantie für A

- $evs \in \text{otway}$
- $A \notin \text{bad}$
- $\text{Says } A \ B \ \{|Na, A, B, \{|Na, A, B|\}_{Ka}|\} \in \text{set } evs$
- $\text{Says } B' \ A \ \{|Na, \{|Na, Kab|\}_{Ka}|\} \in \text{set } evs$

$\implies \exists Nb. \text{Says Server } B$

$\{|Na, \{|Na, Kab|\}_{Ka}, \{|Nb, Kab|\}_{Kb}|\} \in \text{set } evs$

Beweis für dieses Protokoll nicht möglich!

Authentizitätsgarantie für A

- $evs \in otway$
- $A \notin bad$
- $Says\ A\ B\ \{|Na, A, B, \{|Na, A, B|\}_{Ka}|\} \in set\ evs$
- $Says\ B'\ A\ \{|Na, \{|Na, Kab|\}_{Ka}|\} \in set\ evs$

$\implies \exists Nb. Says\ Server\ B$

$\{|Na, \{|Na, Kab|\}_{Ka}, \{|Nb, Kab|\}_{Kb}|\} \in set\ evs$

Beweis für dieses Protokoll nicht möglich!

Authentizitätsgarantie für A

- $evs \in \text{otway}$
- $A \notin \text{bad}$
- $\text{Says } A \ B \ \{ |Na, A, B, \{ |Na, A, B| \}_{K_a} | \} \in \text{set } evs$
- $\text{Says } B' \ A \ \{ |Na, \{ |Na, K_{ab}| \}_{K_a} | \} \in \text{set } evs$

$\implies \exists Nb. \text{Says Server } B$

$\{ |Na, \{ |Na, K_{ab}| \}_{K_a}, \{ |Nb, K_{ab}| \}_{K_b} | \} \in \text{set } evs$

Beweis für dieses Protokoll nicht möglich!

- 1 Das Otway-Rees Protokoll
- 2 Formales Modell
 - Agenten und Nachrichten
 - Events und Traces, Wissen des Spions
 - Formalisierung des Otway-Rees Protokolls
 - Operatoren auf Nachrichten
- 3 Eigenschaften eines korrekten Protokolls
 - possibility property und regularity lemmas
 - unicity theorems
 - secrecy theorem
 - authenticity guarantees
- 4 Angriff und Reparatur des Otway-Rees Protokolls
 - mögliche Angriffe auf das Otway-Rees Protokoll
 - Reparatur des Otway-Rees Protokolls

Server



Alice



Bob



Spy

Server



Alice

1. $Na, A, B, \{ |Na, A, B| \}_{K_a}$

1. →

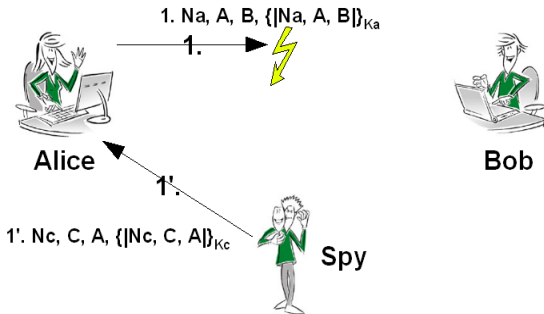


Bob

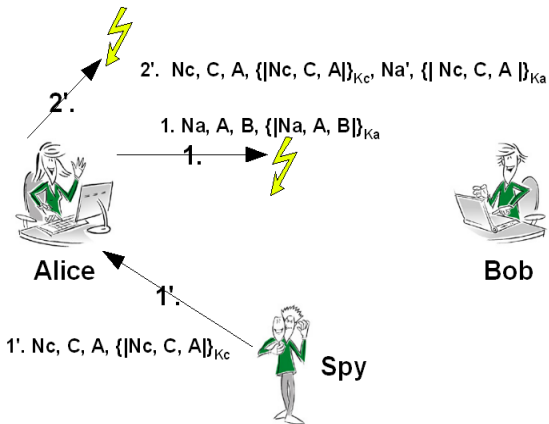


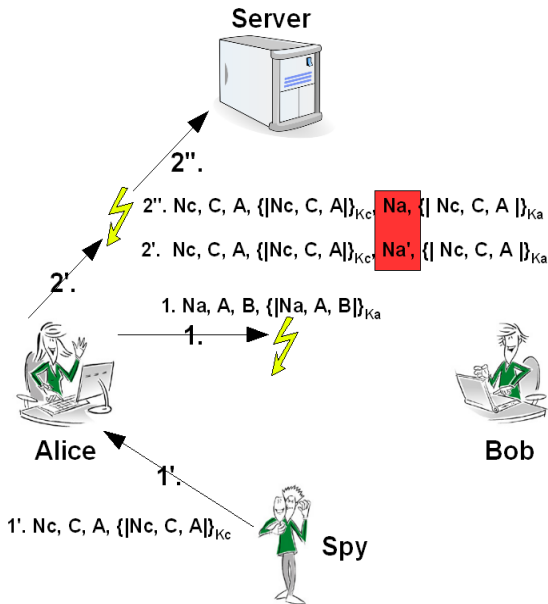
Spy

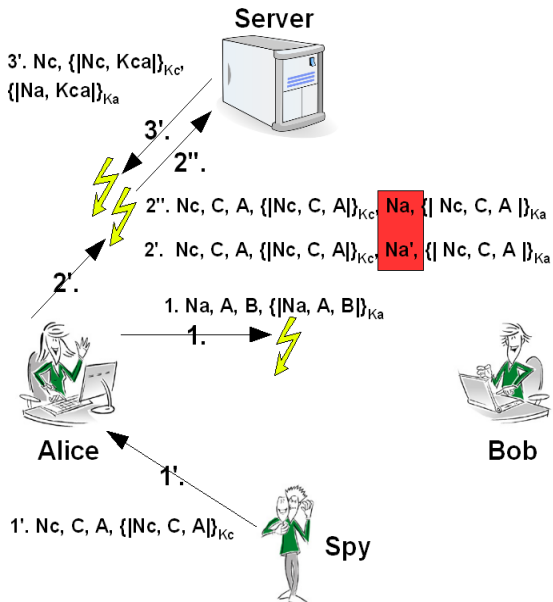
Server

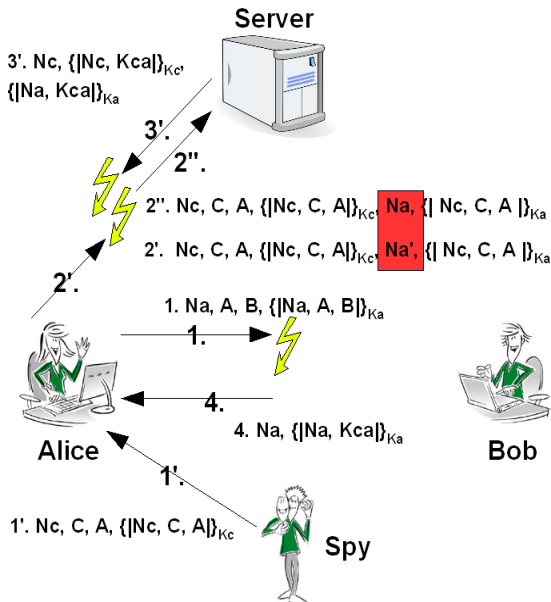


Server









Grund für die mögliche Angriffe

Warum war die Angriffe möglich?

- 1 die zweite Nonce wird in OR2 unverschlüsselt versendet.
- 2 die beiden Teile der Nachricht OR3 besitzen das gleiche Format.

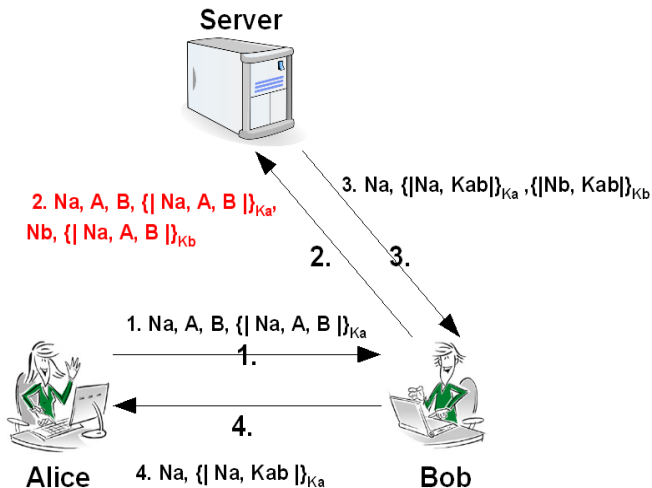
Dann werden dies nun reparieren...

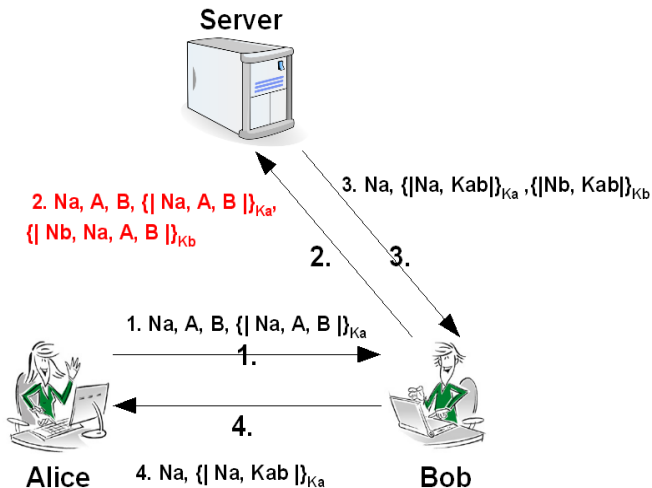
Grund für die mögliche Angriffe

Warum war die Angriffe möglich?

- 1 die zweite Nonce wird in OR2 unverschlüsselt versendet.
- 2 die beiden Teile der Nachricht OR3 besitzen das gleiche Format.

Dann werden dies nun reparieren...





Ausblick

- Es sind bereits weitere Protokolle verifiziert worden, z.B. TLS, Kerberos, Needham-Schroeder ...
- Die Formalisierung und die Beweise findet man unter HOL/Auth/ im Isabelle Sourcecode.
- Verifizierte Protokolle sind dennoch nicht immer sicher.

Ausblick

- Es sind bereits weitere Protokolle verifiziert worden, z.B. TLS, Kerberos, Needham-Schroeder ...
- Die Formalisierung und die Beweise findet man unter HOL/Auth/ im Isabelle Sourcecode.
- Verifizierte Protokolle sind dennoch nicht immer sicher.

Ausblick

- Es sind bereits weitere Protokolle verifiziert worden, z.B. TLS, Kerberos, Needham-Schroeder ...
- Die Formalisierung und die Beweise findet man unter HOL/Auth/ im Isabelle Sourcecode.
- Verifizierte Protokolle sind dennoch nicht immer sicher.