Verifikation kryptographischer Protokolle

Mitschrift von www.kuertz.name

Hinweis: Dies ist **kein offizielles Script**, sondern nur eine private Mitschrift. Die Mitschriften sind teweilse **unvollständig, falsch oder inaktuell**, da sie aus dem Zeitraum 2001–2005 stammen. Falls jemand einen Fehler entdeckt, so freue ich mich dennoch über einen kurzen Hinweis per E-Mail – vielen Dank!

Klaas Ole Kürtz (klaasole@kuertz.net)

Inhaltsverzeichnis

1 Einführung

1.1 Asymetrische Verschlüsselung, Protokolle

Grundlegendes **Szenario**: Alice möchte einem bislang unbekannten Kommunikationspartner Bob eine Nachricht über einen unsicheren (abhörbaren) Kanal schicken.

Idee ist hier die Trennung zwischen Schlüssel k_b zum Verschlüsseln (wird veröffentlicht) und Schlüssel $\hat{k}_b$ zum Entschlüsseln (bleibt beheim) – Alice verschickt dazu $y = e_{k_b}(x)$ und Bob ermittelt $x = d_{\hat{k}_b}(y)$. Wichtig ist, daß es schwierig sein muß, $\hat{k}$ aus k zu berechnen. Ein Beispiel für ein solches *asymmetrisches Verfahren* ist RSA.

Nachteil dieser asymmetrischen Verschlüsselung ist, daß sie sehr rechenintensiv ist. Häufig wird daher ein *hybrides Verfahren* verwendet: Ein symmetrischer Schlüssel wird asymmetrisch verschlüsselt und dann ausgetauscht, dann wird der Rest symmetrisch verschlüsselt, was „billiger“ ist. Um den öffentlichen Schlüssel sicher auszutauschen, benötigt man ein *Schlüsselaustauschprotokoll*.

Beispiel für ein solches Protokoll ist eine vereinfachte Variante eines Protokolls von Needham und Schroeder aus dem Jahr 1978:

$$\begin{array}{lll} (1) & A \to B\colon & e_{k_B}(kA) \\ (2) & B \to A\colon & e_{k_A}(kN) \\ (3) & A \to B\colon & e_{k_B}(N) \end{array}$$

Dabei ist N ein Zufallszahl (Abkürzung für *Nonce*, number used once): Man schickt einen zufälligen Bitstring und erwartet, daß dieser wieder zurückgeschickt wird. Im Needham-Schroeder-Protokoll werden zwei Nonces benutzt zur gegenseitigen Authentifizierung; gleichzeitig wird ein Nonce als Sitzungsschlüssel benutzt.

Angriff gegen dieses Protokoll:

$$\begin{array}{llll} (1) & A \to I\colon & e_{k_I}(kA) & \\ (1') & & & I(A) \to B\colon \quad e_{k_B}(kA) \\ (2') & & & B \to I(A)\colon \quad e_{k_A}(kN) \\ (2) & I \to A\colon & e_{k_A}(kN) & \\ (3) & A \to I\colon & e_{k_I}(N) & \\ (3') & & & I(A) \to B\colon \quad e_{k_B}(N) \end{array}$$

Beide Protokollläufe sind ordnungsgemäß, aber Bob kann annehmen, daß k ein Sitzungsschlüssel ist, der nur Alice und ihm bekannt ist und daß er mit Alice kommuniziert.

Eine von Lowe vorgeschlagene Modifikation (NSL-Protokoll) fügt in der zweiten Nachricht Bobs Absender ein und verhindert damit den Angriff:

$$\begin{array}{lll} (1) & A \to B\colon & e_{k_B}(kA) \\ (2) & B \to A\colon & e_{k_A}(kNB) \\ (3) & A \to B\colon & e_{k_B}(N) \end{array}$$

NSL ist in einem sehr starken Sinn sicher.

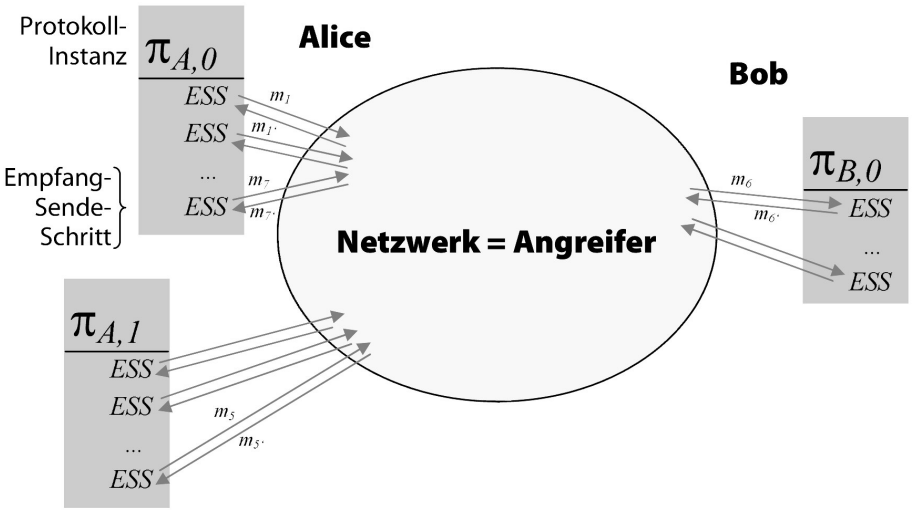
1.2 Inhalt der Vorlesung

Ziel der Vorlesung ist, Verfahren, Algorithmen und Tools kennenzulernen, um kryptographische Protokolle (automatisch) analysieren.

- Schlüsselaustauschprotokolle und Authentifizierungsprotokolle
  - Dolev-Yao-Modell
  - Sicherheit ist NP-vollständig
  - Constraint Solving: praktikable Algorithmen
  - Beispiele
- Faire Austauschprotokolle

2 Das Dolev-Yao-Modell

Idee des Modells von Dolev und Yao (1983):



Vergleich zur Kryptographie-Vorlesung:

	Kryptographie-VL	**Dolev-Yao-Modell**
Nachrichten	Bit-Strings	formale Terme
Angreifer	probabilistische Polynomzeit-Algorithmen	unbeschränkte nicht-deterministische Algorithmen, die eine bestimmte Menge von Operationen ausführen können
Instanzen	probabilistische Polynomzeit-Algorithmen	Folge von Termersetzungs-Regeln
Anzahl Instanzen	beliebig	beschränkt
Sicherheit	relativ zur Sicherheit kryptographischer Primitiven (Reduktion)	*Annahme*: Kryptographie funktioniert perfekt; Protokoll ist sicher, falls kein „schlechter“ Zustand erreicht wird.
Analyse	manuell	(semi-)automatisch

2.1 Das Protokollmodell

2.1.1 Terme und Nachrichten

Es sei A eine endliche Menge von *Konstanten* (Namen von Parteien (A, B, ...), Nonces (N_A, N_B, ...), Schlüssel (k_A, k_B, ...)). Weiter sei $K \subseteq A$ eine Menge von öffentlichen und privaten Schlüsseln. Es sei zudem $\cdot^{-1}$ eine bijektive Funktion auf K, die jedem öffentlichen Schlüssel $k \in K$ den zugehörigen privaten Schlüssel $k^{-1} \in K$ zuordnet.

Es sei V eine Menge von *Variablen* $x, y, z, \ldots$. Weiter bezeichne Σ_A die Signatur $A \cup \Sigma$ mit

$$\Sigma := \{\mathtt{enc}^{\mathrm{s}}_{(\cdot)}(\cdot), \mathtt{enc}^{\mathrm{a}}(\cdot,\cdot), \langle\cdot,\cdot\rangle\}$$

Dabei sind $\mathtt{enc}^{\mathrm{s}}$ und $\mathtt{enc}^{\mathrm{a}}$ symmetrische und asymmetrische Verschlüsselung[1], und $\langle\cdot,\cdot\rangle$ steht für die Konkatenation. Man könnte hier noch $\mathtt{hash}(\cdot)$ und $\mathtt{sig}(\cdot,\cdot)$ etc. dazunehmen, dies brauchen wir aber zunächst nicht. Die Mengen A, V, Σ seien paarweise disjunkt.

DEFINITION: Die Menge T der *Terme* (über Σ_A und V) sei definiert durch die folgende Grammatik:

$$T \ ::= \ A \mid V \mid \mathtt{enc}^{\mathrm{s}}_T(T) \mid \mathtt{enc}^{\mathrm{a}}(K,T) \mid \langle T,T\rangle$$

Zudem sei $\mathrm{Var}\,(t)$ die Menge der im Term t vorkommenden Variablen.

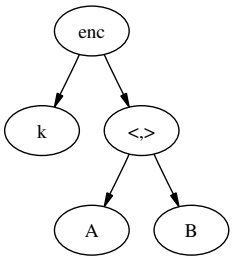
BEISPIEL: $t := \mathtt{enc}^{\mathrm{a}}_{k_B}(\langle x, A\rangle)$, $t \in T$, $\mathrm{Var}\,(t) = x$.

DEFINITION: Die Menge M der *Nachrichten* über Σ_A sei die Menge der Grundterme in T, d.h. die Terme ohne Variablen:

$$M := \{t \in T \mid \mathrm{Var}\,(t) = \emptyset\}$$

BEISPIEL: $t := \mathtt{enc}^{\mathrm{a}}_k(\langle A, B\rangle) \in M$.

Terme kann man sich als beschriftete Bäume vorstellen:



[1] statt $\mathtt{enc}^{\mathrm{a}}(k,x)$ wird auch oft $\mathtt{enc}^{\mathrm{a}}_k(x)$ verwendet, entsprechend für $\mathtt{enc}^{\mathrm{s}}$

2.1.2 Instanzen und Protokolle

DEFINITION: Ein *Empfang-Sende-Schritt (ESS)* ist definiert als eine Termersetzungsregel (TSR), d.h. es ist ein Tupel der Form $(t, t') \in T \times T$.

Wir schreiben $t \to t'$ statt (t, t').

BEISPIEL: Aus dem NEEDHAM-SCHROEDER-Protokoll:

$$\mathtt{enc}^{\mathrm{a}}_{k_B}(\langle x, A\rangle) \to \mathtt{enc}^{\mathrm{a}}_{k_A}(\langle x, N_B\rangle)$$

Zur Berechnung eines ESS:

DEFINITION: Eine Substitution σ ist eine Abbildung von V in M, so daß $\sigma(x) \neq x$ für nur endlich viele x aus V ist. σ wird homomorph erweitert auf T:

$$\sigma(f(x_1, \ldots, x_n)) := f(\sigma(x_1), \ldots, \sigma(x_n))$$

σ ist *Grundsubstitution*, falls $\sigma(x) \in M$ ist für alle x mit $\sigma(x) \neq x$.

BEISPIEL: Ist $\sigma = \{x \to N_A, y \to \mathtt{enc}^{\mathrm{a}}_{k_A}(B)\}$, so ist

$$\sigma(\langle y, \langle x, x\rangle\rangle) = \left\langle \mathtt{enc}^{\mathrm{a}}_{k_A}(B), \langle N_A, N_A\rangle \right\rangle$$

Ist $m \in M$ und $t \in T$, dann *matcht t mit m*, falls eine (Grund-)Substitution σ mit $\sigma(t) = m$ existiert. σ heißt dann *Matcher*.

Zur Berechnung eines ESSs $t \to t'$: Empfangen wird eine Nachricht m. Falls t nicht mit m matcht, wird nichts ausgegeben; falls jedoch ein Matcher σ existiert (d.h. $\sigma(t) = m$), so wird $\sigma(t')$ ausgegeben.

DEFINITION: Eine *Instanz* π ist eine endliche Folge $r_0 \to s_0, \ldots, r_{n-1} \to s_{n-1}$ von ESSs, so daß gilt

$$\mathrm{Var}\,(s_i) \subseteq \bigcup_{j \leq i} \mathrm{Var}\,(r_j)$$

Wir nehmen an, daß für verschiedene Instanzen die Menge der Variablen disjunkt sind.

DEFINITION: Ein *Protokoll* P ist ein Tupel $(\{\pi_i\}_{i<n}, W)$ mit einer endlichen Familie $\{\pi_i\}_{i<n}$ von Instanzen und einer endlichen Menge W von Nachrichten, das initiale Angreiferwissen.

Wir nehmen an, daß der Angreifer I seinen (konstanten) Namen kennt, d.h. $I \in W$. Im folgenden sei zudem A immer die Menge der im Protokoll vorkommenden Konstanten.

Beispiel: Das Needham-Schroeder-Protokoll (bzw. der Ablauf des Protokolls, der zum Angriff führt) kann beispielsweise mit drei Instanzen modelliert werden:

- π_0: Alice ist Initiator und spricht mit dem Angreifer
- π_1: Alice ist Initiator und spricht mit Bob
- π_2: Bob ist antwortende Partei und spricht mit Alice

Das formale Protokoll ist dann:

$$P = (\{\pi_0, \pi_1, \pi_2\}, \{I, A, B, N_I, k_A, k_B, k_I, k_I^{-1}\})$$

Die einzelnen Instanzen:

$$\begin{aligned}
\pi_0\colon \quad & I \to \mathtt{enc}^{\mathrm{a}}_{k_I}(\langle A, N_A\rangle) \\
& \mathtt{enc}^{\mathrm{a}}_{k_A}(\langle N_A, x\rangle) \to \mathtt{enc}^{\mathrm{a}}_{k_I}(x) \\
\pi_1\colon \quad & I \to \mathtt{enc}^{\mathrm{a}}_{k_B}(\langle A, N'_A\rangle) \\
& \mathtt{enc}^{\mathrm{a}}_{k_A}(\langle N'_A, y\rangle) \to \mathtt{enc}^{\mathrm{a}}_{k_B}(y) \\
\pi_2\colon \quad & \mathtt{enc}^{\mathrm{a}}_{k_B}(\langle A, z\rangle) \to \mathtt{enc}^{\mathrm{a}}_{k_A}(\langle z, N_B\rangle) \\
& \mathtt{enc}^{\mathrm{a}}_{k_B}(N_B) \to \mathtt{enc}^{\mathrm{s}}_{N_B}(\mathsf{secret})
\end{aligned}$$

2.2 Das Angreifermodell und der Sicherheitsbegriff

Der Angreifer hat vollständige Kontrolle über das Netzwerk, er **ist** das Netzwerk. Er leitet aus den von den Instanzen empfangenen Nachrichten neue ab, durch Anwenden bestimmter Operationen kann er die erzeugten Nachrichten in beliebiger Reihenfolge an die Instanzen senden; und er versucht schließlich, die Nachricht $\mathsf{secret} \in A$ zu erhalten.

Grundlegende Annahme ist hier also, daß die grundlegende Kryptographie **perfekt** funktioniert! D.h. es gibt hier keine Kryptanalyse etc. – ohne Schlüssel k kann der Angreifer beispielsweise nichts über eine Nachricht das secret aus $\mathtt{enc}^{\mathrm{s}}_k(\mathsf{secret})$ aussagen!

DEFINITION: Es sei $W \subseteq M$ eine endliche Menge von Nachrichten. Die (unendliche) Menge $d(W) \subseteq M$ von Nachrichten, die aus W abgeleitet werden kann, ist die kleinste Menge, die folgende Bedingungen erfüllt:

- $W \subseteq d(W)$
- Komposition: $m, m' \in d(W) \implies \langle m, m' \rangle \in d(W)$
- Dekomposition: $\langle m, m' \rangle \in d(W) \implies m, m' \in d(W)$
- symmetrische Entschlüsselung: $k, \mathtt{enc}^{\mathrm{s}}_k(m) \in d(W) \implies m \in d(W)$
- asymmetrische Entschlüsselung: $k^{-1}, \mathtt{enc}^{\mathrm{a}}_k(m) \in d(W) \implies m \in d(W)$
- symmetrische Verschlüsselung: $m, k \in d(W) \implies \mathtt{enc}^{\mathrm{s}}_k(m) \in d(W)$
- asymmetrische Verschlüsselung: $m \in d(W) \wedge k \in (d(W) \cap K) \implies \mathtt{enc}^{\mathrm{a}}_k(m) \in d(W)$

Wir sehen später, wie man effizient mit dieser unendlichen Menge umgehen kann.

DEFINITION: Die *Ausführungsordnung* ist eine partielle, injektive Funktion

$$\pi \colon P \dashrightarrow \{0, \ldots, |P| - 1\} \text{ mit } P := \{ (i, j) \mid i < n, j < n_i \}$$

Falls dabei $\pi(i, j)$ definiert ist, so folgt aus $j' < j < n_i$, daß $\pi(i, j')$ definiert ist und $\pi(i, j') < \pi(i, j)$ ist. Die *Größe* von π ist $p := |P|$.

Sei nun P ein Protokoll wie oben.

DEFINITION: Ein *Angriff* auf P ist ein Tupel (π, σ), wobei π eine Ausführungsordnung ist und σ eine Substitution auf den durch π referenzierten Variablen in P ist.

Sei p die Größe von π; sei zudem $r_l = r^i_j$ und $s_l = s^i_j$ für alle $\pi(i, j) = l$. Es muß gelten:

$$\sigma(r_l) \in d(W \cup \{\sigma(s_0), \ldots, \sigma(s_{l-1})\}) \; \forall \, l < p$$

DEFINITION: Ein Angriff (π, σ) ist *erfolgreich*, wenn gilt:

$$\mathsf{secret} \in d(W \cup \{\sigma(s_0), \ldots, \sigma(s_{p-1})\})$$

2.3 Sicherheit ist NP-schwer

Nun läßt sich ein Entscheidungsproblem formulieren:

2.3.1 Entscheidungsproblem INSECURE $\in$ NP

SATZ: Das folgende Entscheidungsproblem INSECURE ist NP-vollständig:

$$\text{INSECURE} := \{ P \mid \text{auf } P \text{ gibt es einen erfolgreichen Angriff} \}$$

BEWEIS: Hier nur für symmetrische Verschlüsselung – der asymmetrische Fall ist entsprechend einfacher, da nur atomare Schlüssel verwendet werden.

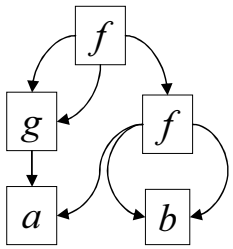
DEFINITION: Die Menge $\text{Sub}(t)$ der *Teilterme* eines Terms t ist induktiv definiert als

- $\text{Sub}(c) = \{c\}$ für $c \in A \cup V$
- $\text{Sub}(f(t_0, \ldots, t_{n-1})) = \{f(t_0, \ldots, t_{n-1})\} \cup \bigcup_{i=0}^{n-1} t_i$

BEISPIEL: Mit $t = f(g(a), g(a), f(b, b, a))$ gilt:

$$\text{Sub}(t) = \{a, b, g(a), f(b, b, a), t\}$$

Dies läßt sich auch als gerichteter, azyklischer Graph *(Directed Asyclic Graph, DAG)* darstellen:



DEFINITION: Die *DAG-Größe* von t ist $|t|_{\text{DAG}} = |\text{Sub}(t)|$; die DAG-Größe von $P = (\{\pi_i\}, W)$ ist

$$|P|_{\text{DAG}} = \left| \text{Sub}(W) \cup \bigcup_i \text{Sub}(\pi_i) \right|$$

NP-ALGORITHMUS: Eingabe ist ein Protokoll $P = (\{\pi_i\}, W)$.

1. Rate einen Angriff (π, σ):
   - Rate eine Ausführungsordnung π für P:

     $$r_0 \to s_0, \ldots, r_{p-1} \to s_{p-1}, r_p := \mathsf{secret}$$

   - Rate σ mit $|\sigma(x)|_{\mathrm{DAG}} \leq |P|_{\mathrm{DAG}}$ für alle $x \in \mathrm{Var}\,(P)$.
2. Teste, ob (π, σ) tatsächlich ein erfolgreicher Angriff ist:

   $$\sigma(r_i) \in d(W \cup \{\sigma(s_0), \ldots, \sigma(s_{i-1})\}) \text{ für alle } i \leq n$$

Ableiten von Nachrichten: Sei M die Menge aller Nachrichten.

2.3.2 Entscheidungsproblem DERIVE $\in \mathcal{P}$

LEMMA (1): Das folgende Entscheidungsproblem DERIVE ist in $\mathcal{P}$:

$$\mathrm{DERIVE} = \{\,(E, m) \mid E \subseteq M \text{ endlich}, m \in M, m \in d(E)\}$$

Zur Notation: E,t entspreche $E \cup \{t\}$. Die Menge $d(E)$ beschreiben wir durch Ableitungsregeln: Eine Regel L wird notiert als $L\colon E \to m$, wenn m aus den Nachrichten in E mittels L ableitbar ist – daher bezeichnen wir L als *m-Regel*. Diese Regeln induzieren eine Ableitungsrelation:

$$\begin{array}{ll} \text{sei} & \to_L \subseteq \mathcal{P}(M) \times \mathcal{P}(M) \\ \text{mit} & E' \to_L (E' \cup \{m\}) \;:\Longleftrightarrow\; E \subseteq E' \\ \text{für} & L\colon E \to m \end{array}$$

Sei nun $\mathcal{L}$ eine (unendliche) Menge von Ableitungsregeln. Dann sei

$$\to_{\mathcal{L}} := \bigcup_{L \in \mathcal{L}} \to_L$$

Die Komposition von Ableitungsrelationen ergibt sich als:

$$\to_L \circ \to_M := \{\,(E, E'') \mid (E \to_L E') \wedge (E' \to_M E'')\}$$

Entsprechend wird $\to_L^*$ definiert.

Um $d(E)$ zu beschreiben, betrachten wir folgende Ableitungsregeln:

1. *Dekompositionsregeln:*

$$\begin{array}{lll} L_{p_1}(\langle m,n\rangle) & : & \{\langle m,n\rangle\}\rightarrow m \\ L_{p_2}(\langle m,n\rangle) & : & \{\langle m,n\rangle\}\rightarrow n \\ L_{sd}(\mathtt{enc}^{\mathrm{s}}_n(m)) & : & \{\mathtt{enc}^{\mathrm{s}}_n(m), n\}\rightarrow m \end{array}$$

   Zusammengefaßt ist $L_d(m) = \{L_{p_1}(m), L_{p_2}(m), L_{sd}(m)\}$.

2. *Kompositionsregeln:*

$$\begin{array}{ll} L_{c_1}(\langle m,n\rangle) & : \{m,n\}\rightarrow\langle m,n\rangle \\ L_{c_2}(\mathtt{enc}^{\mathrm{s}}_n(m)) & : \{m,n\}\rightarrow\mathtt{enc}^{\mathrm{s}}_n(m) \end{array}$$

   Entsprechend zusammengefaßt ergibt sich $L_c(m) = \{L_{c_1}(m), L_{c_2}(m)\}$.

Zusammen definieren wir folgende Mengen:

$$\begin{array}{rcl} L_d & = & \{\, L_d(m) \mid m \in M\} \\ L_c & = & \{\, L_c(m) \mid m \in M\} \end{array}$$

BEISPIEL: Sei $m := \left\langle \mathtt{enc}^{\mathrm{s}}_{\langle a,a\rangle}(\mathsf{secret}), a \right\rangle$. Dann gilt $\mathsf{secret} \in d(m)$ wegen folgender Ableitung:

$$\begin{array}{lll} \{m\} & \rightarrow_{L_{p_2}} & \{m, a\} \\ & \rightarrow_{L_c(\langle a,a\rangle)} & \{m, a, \langle a,a\rangle\} \\ & \rightarrow_{L_{p_2}} & \{m, a, \langle a,a\rangle, \mathtt{enc}^{\mathrm{s}}_{\langle a,a\rangle}(\mathsf{secret})\} \\ & \rightarrow_{L_{sd}} & \{m, a, \langle a,a\rangle, \mathsf{secret}\} \end{array}$$

LEMMA (2): Die Menge $d(E)$ entsteht aus der Vereinigung aller möglichen Ableitungen aus E:

$$d(E) = \bigcup \{\, E' \mid E \rightarrow^*_{\mathcal{L}} E'\}$$

BEWEIS: als Übung

DEFINITION: Die *Ableitung* von m aus E ist eine Folge

$$E \rightarrow_{L_0} (E \cup \{m_0\}) \rightarrow_{L_1} \ldots \rightarrow_{L_{n-1}} (E \cup \{m_0, \ldots, m_{n-1}\})$$

Es sei $m = m_{n-1}$, und es gelte $m_i \notin E \cup \{m_0, \ldots, m_{i-1}\}$. Weiter ist $L_i \in \mathcal{L}$ die *i-te Regel*, zusammengefaßt ist $L := \bigcup_{i=0}^{n} L_i$. Die Länge der Ableitung ist n.

Ist $m \in d(E)$, so existiert nach Lemma (2) eine Ableitung von D von m aus E mit minimaler Länge, bezeichne $D_E(m)$ diese Ableitung.

LEMMA (3): Sei $m \in d(E)$. Dann gilt für alle Regeln $L \in D_E(m)$ und Nachrichten $m' \in M$:

1. Aus $L \in L_d(m')$ folgt $m' \in \mathrm{Sub}\,(E)$.
2. Aus $L \in L_c(m')$ folgt $m' \in \mathrm{Sub}\,(E, m)$.

BEWEIS: Sei folgende Ableitung gegeben:

$$E \to \ldots \to_{L_1} \ldots \to \ldots \to_{L_2} \ldots \to m$$

1. Sei dabei $L_2 \in L_d(m')$. Angenommen, $m' \notin \mathrm{Sub}\,(E)$. Dann existiert eine Regel $L_1 \in L_c(m')$, die m' erzeugt. Dies verletzt jedoch die Minimalität von $D_E(m)$, da $D_E(m) \setminus L_1, L_2$ eine kürzere Ableitung für m wäre.

   Somit existiert ein m'' mit $L_1 \in L_d(m'')$, wobei L_1 dann m' erzeugt. Damit ist m'' Teilterm von E und somit $m \in \mathrm{Sub}\,(m'') \subseteq \mathrm{Sub}\,(E)$, ein Widerspruch.

   Durch Iteration dieses Arguments von hinten nach vorne ergibt sich, daß Dekompositionsregeln nur auf Teilterme von E angewendet werden.

2. Sei nun obiges $L_1 \in L_c(m')$. Angenommen, $m' \notin \mathrm{Sub}\,(E)$. Dann ist entweder $L_2 \in L_d(m')$, womit wieder die Minimalität verletzt wäre. Oder es gilt $L_2 \in L_c(m'')$, wobei m' bei der Komposition verwendet wird. Dann wird der Schritt nur benötigt, falls m'' später benutzt wird – hieraus folgt $m'' \in \mathrm{Sub}\,(m)$ und somit $m' \in \mathrm{Sub}\,(m'') \subseteq \mathrm{Sub}\,(m)$.

   Dieses Argument kann nach hinten iteriert werden, woraus folgt, daß nur Elemente durch Komposition entstehen, die in E oder m verwendet werden.

FOLGERUNG: DERIVE $\in \mathcal{P}$

BEWEIS: Zunächst folgt aus obigem Lemma, daß die Länge der Ableitung $D_E(m)$ beschränkt ist durch $|\mathrm{Sub}\,(E \cup \{m\})| = |E \cup \{m\}|_{\mathrm{DAG}}$, d.h. polynomiell in der Eingabegröße.

Weiter gilt für alle m'-Regeln, daß $m' \in \mathrm{Sub}\,(E \cup \{m\})$ ist. Sei nun (E, m) gegeben. Dann kann $(E \to m) \in \mathcal{L}$ in polynomieller Zeit entscheiden werden: Wende alle Regeln ein mal auf E an und vergleiche das Ergebnis mit m.

Als Algorithmus: Untersuche alle Regeln $(E \to m)$ für $m \in \mathrm{Sub}(E)$, nimm entsprechende Teilterme zur Menge hinzu und iteriere dies $|\mathrm{Sub}(E \cup \{m\})|$ mal.

2.3.3 minimale Angriffe

DEFINITION: Der Angriff (π, σ) ist ein für P *minimaler Angriff*, falls (π, σ) ein erfolgreicher Angriff ist und $\sum_{x \in \mathrm{Var}(P)} |\sigma(x)|_{\mathrm{DAG}} \leq \sum_{x \in \mathrm{Var}(P)} |\sigma'(x)|_{\mathrm{DAG}}$ für alle erfolgreichen Angriffe (π', σ').

Falls ein erfolgreicher Angriff existiert, dann auch ein minimaler.

LEMMA (4): Ist (π, σ) minimaler Angriff auf P, so gilt:

$$|\sigma(x)|_{\mathrm{DAG}} \leq |P|_{\mathrm{DAG}} \ \forall\, x \in \mathrm{Var}(P)$$

BEWEIS: Um dieses Lemma zu zeigen, benötigen wir weitere Definitionen und Lemmatas:

DEFINITION: Sei t ein Term, m eine Nachricht und σ eine Grundsubstitution. Dann ist t ein *σ-Matching von m*, wenn t keine Variable ist und $\sigma(t) = m$ gilt. Wir schreiben $t \sqsubseteq_\sigma m$.

BEISPIEL: Sei $t = \mathtt{enc}^{\mathrm{s}}_x(y)$ und $m = \mathtt{enc}^{\mathrm{s}}_{\langle a,b \rangle}(b)$. Weiter sei $\sigma = \{x \mapsto \langle a, b \rangle, y \mapsto b\}$. Dann gilt $t \sqsubseteq_\sigma m$.

Wir benötigen zunächst noch ein weiteres Lemma für den späteren Beweis von Lemma (6):

LEMMA (5): Seien $m, m' \in d(E)$; die letzte Regel in $D_E(m')$ sei eine Komposistionsregel (d.h. aus L_c). Dann existiert eine Ableitung D von m aus E mit $L_d(m') \notin D$.

BEWEIS: Sei D eine Ableitung von m. Falls in dieser eine Regel $L_d(m')$ vorkommt, so ersetze diesen Schritt durch die obige Ableitung $D_E(m')$ (ohne den letzten Schritt). Dann existieren die Teilterme, die durch die Dekomposition von m' entstehen, bereits in der Menge der Nachrichten, da sie zu m' hätten komponiert werden können nach Wahl von $D_E(m')$.

LEMMA (6): Falls (π, σ) ein minimaler Angriff ist, dann existiert für alle $x \in \mathrm{Var}(P)$ ein $t \in \mathrm{Sub}(P)$ mit $t \sqsubseteq_\sigma \sigma(x)$.

BEWEIS: Wir nehmen an, es existiert $x \in \mathrm{Var}(P)$, so daß $t \not\sqsubseteq_\sigma \sigma(x)$ für alle $t \in \mathrm{Sub}(P)$. Wir zeigen: Dann ist auch (π, σ') ein erfolgreicher Angriff, wobei σ' aus σ dadurch entsteht, daß jeder Teilterm $\sigma(x)$ in σ durch I ersetzt wird. Intuitiv ergibt sich: Wenn die Belegung von x in keinem Teilterm von P vorkommt, d.h. auch in keiner linken Regelseite, so spielt die Belegung von x keine Rolle und kann durch etwas kürzeres (z.B. I) ersetzt werden. Das widerspräche jedoch der Minimalität des gewählten Angriffs.

Formal gilt (da $\sigma(x)$ keine Konstante ist, aber $\sigma'(x)$ eine ist):

$$\sum_{y \in \mathrm{Var}(P)} |\sigma'(y)|_{\mathrm{DAG}} < \sum_{y \in \mathrm{Var}(P)} |\sigma(y)|_{\mathrm{DAG}}$$

Sei die Ausführungsreihenfolge π wie folgt gegeben:

$$r_0 \to s_0, \ldots, r_{p-1} \to s_{p-1}, r_p = \mathsf{secret}$$

Sei $q \leq p$ minimal mit der Eigenschaft, daß $\sigma(x) \in \mathrm{Sub}(\sigma(r_q))$ ist. Das Angreiferwissen zu diesem Zeitpunkt ist $E_0 := W \cup \{\sigma(s_0), \ldots, \sigma(s_{q-1}))\}$.

Nun gilt $\sigma(x) \notin \mathrm{Sub}(E_0)$, da wegen $\sigma(x) \notin P$ auch $\sigma(x) \notin W$ gilt. Zusätzlich ist $\sigma(x) \notin \sigma(s_i)$ mit $i < q$, da dann ein $j \leq i$ existieren müßte mit $\sigma(x) \notin \sigma(r_j)$, was die Minimalität von q verletzt.

Weiter zeigen wir $\sigma(x) \in d(E_0)$: Wir wissen $\sigma(r_q) \in d(E_0)$. Also existiert eine Ableitung von $\sigma(r_q)$ mit folgender Form:

$$D_{E_0}(\sigma(r_q)) = E_0 \to_{L_0} E_1 \to_{L_1} \ldots \to_{L_{n-1}} E_n$$

wobei $\sigma(r_q) \in E_n$ gelte. Wie oben gezeigt gilt aber $\sigma(x) \notin \mathrm{Sub}(E_0)$. Also muß ein minimales $j > 0$ existieren, so daß $\sigma(x) \in \mathrm{Sub}(E_j)$ ist, denn $\sigma(x) \in \mathrm{Sub}(\sigma(r_q))$ nach Wahl von q. Wir unterscheiden die folgenden Fälle:

- $L_{j-1} \in L_c(m)$ für ein $m \neq \sigma(x)$ mit $\sigma(x) \in \mathrm{Sub}(m)$. Aus der Definition der Kompositionsregeln folgt, daß $\sigma(x) \in \mathrm{Sub}(E_{j-1})$ sein müßte, ein Widerspruch zur Minimalität von j.
- $L_{j-1} \in L_d(m')$ für ein m', wobei L_{j-q} eine m-Regel ist mit $\sigma(x) \in \mathrm{Sub}(m)$ mit $\sigma(x) \neq m$. Aber aus $m' \in E_{j-1}$ folgt, daß $\sigma(x) \in \mathrm{Sub}(() E_{j-1})$ ist, Widerspruch.

Damit existiert ein j mit $\sigma(x) \in \mathrm{Sub}(E_j)$.

Zusammengefaßt existiert eine Ableitung von $\sigma(x)$ aus E_0 – da jedoch $\sigma(x) \notin \mathrm{Sub}(E_0)$ ist, existiert solch eine Ableitung so, daß die letzte Regel eine Kompositionsregel ist.

Zu zeigen bleibt: (π, σ') ist ein Angriff auf P.

Wir müssen zeigen: $\sigma'(r_j) \in d(W \cup \{\sigma'(s_0), \ldots, \sigma'(s_{j-1})\})$ für alle j. Für jedes $j < q$ ist dies trivialerweise erfüllt, da dann – weil $\sigma(x)$ nach Wahl von q nicht vorkommt – gilt: $\sigma'(r_j) = \sigma(r_j)$.

Für $j \geq q$ sei D eine Ableitung von $\sigma'(r_j)$ mit σ. Ersetze in D jeden Teilterm $\sigma(x)$ durch I. Da – wie gerade gezeigt – eine Ableitung von $\sigma(x)$ existiert, in der die letzte Regel eine Kompositionsregel ist, muß nach Lemma (5) dann $\sigma(x)$ in D nie dekomponiert werden.

Damit ist leicht zu sehen, daß die so erhaltene Ableitung tatsächlich eine Ableitung ist für $\sigma'(R_j)$.

Damit folgt, daß der oben vorgestellt Algorithmus das Problem nichtdeterministisch in polynomieller Zeit löst, es gilt also:

SATZ: INSECURE liegt in NP.

2.3.4 NP-Vollständigkeit

Nun bleibt für die NP-Vollständigkeit noch zu zeigen:

SATZ: INSECURE ist NP-schwer.

BEWEIS: Wir geben eine Reduktion von $3-\mathrm{SAT}$ an:

- Repräsentation: Ein Literal x wird dargestellt durch die Variable x, ein negiertes Literal $\bar{x}$ durch $\mathtt{enc}^{\mathrm{s}}_k(x)$. Eine Klausel $(x \vee y \vee z)$ wird dargestellt durch $\langle x, \langle y, z\rangle\rangle$. Die Konjunktion der Klauseln wird ebenfalls als verschachtelte Liste dargestellt mit einem End-Element $\perp$, d.h. Klauseln $\bigwedge_{i=1}^{n} k_i$ werden dargestllt als

  $$\langle k_1, \langle k_2, \langle \ldots, \langle k_n, \perp\rangle\rangle\rangle\rangle$$

  Sei eine Formel φ gegeben mit

  $$\varphi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee z)$$

  Diese wird dann also repräsentiert durch

  $$t_\varphi = \langle\langle x, \langle \mathtt{enc}^{\mathrm{s}}_k(y), z\rangle\rangle, \langle\langle \mathtt{enc}^{\mathrm{s}}_k(x), \langle y, z\rangle\rangle, \perp\rangle\rangle$$

- Das initiale Angreiferwissen ist $W = \{0, 1\}$.

- Für drei Variablen x, y, z ergeben sich zunächst folgende Instanzen:

$$\begin{array}{ll}
\pi_1^1 & \mathtt{enc}_k^{\mathrm{s}}(\langle\langle 1, \langle x, y\rangle\rangle, z\rangle) \to \mathtt{enc}_k^{\mathrm{s}}(z) \\
\pi_2^1 & \mathtt{enc}_k^{\mathrm{s}}(\langle\langle x, \langle 1, y\rangle\rangle, z\rangle) \to \mathtt{enc}_k^{\mathrm{s}}(z) \\
\pi_3^1 & \mathtt{enc}_k^{\mathrm{s}}(\langle\langle x, \langle y, 1\rangle\rangle, z\rangle) \to \mathtt{enc}_k^{\mathrm{s}}(z) \\
\pi_1^0 & \mathtt{enc}_k^{\mathrm{s}}(\langle\langle \mathtt{enc}_k^{\mathrm{s}}(0), \langle x, y\rangle\rangle, z\rangle) \to \mathtt{enc}_k^{\mathrm{s}}(z) \\
\pi_2^0 & \mathtt{enc}_k^{\mathrm{s}}(\langle\langle x, \langle \mathtt{enc}_k^{\mathrm{s}}(0), y\rangle\rangle, z\rangle) \to \mathtt{enc}_k^{\mathrm{s}}(z) \\
\pi_3^0 & \mathtt{enc}_k^{\mathrm{s}}(\langle\langle x, \langle y, \mathtt{enc}_k^{\mathrm{s}}(0)\rangle\rangle, z\rangle) \to \mathtt{enc}_k^{\mathrm{s}}(z)
\end{array}$$

  Die Regel π_i^j überprüft, ob an Position 1 des Tripels $\langle x, \langle y, z\rangle\rangle$ das Zeichen j steht. Man benötigt von jeder dieser sechs Regel so viele Kopien, wie Klauseln in der Formel enthalten sind.

- Weitere Instanzen bilden den „Rahmen“:

$$\begin{array}{ll}
\pi_0 & \langle x, \langle y, z\rangle\rangle \to \mathtt{enc}_k^{\mathrm{s}}(t_\varphi) \\
\pi_\infty & \mathtt{enc}_k^{\mathrm{s}}(\bot) \to \mathsf{secret}
\end{array}$$

2.4 Constranint Solving

Die Constraint-Solving-Technik ist eine Heuristik, die es erlaubt, den Suchraum der möglichen Substitutionen σ eines Angriffes (π, σ) einzuschränken.

2.4.1 Constraint-Solving-Algorithmus

Eingabe: $P = (\{\pi_i\}, W)$ (sei hier $S_0 = W$).

1. Rate eine Ausführungsordnung π mit

$$r_1 \to s_1, \ldots, r_{p-1} \to s_{p-1}, r_p := \mathsf{secret}$$

2. Generier Constraint-System $(C_0, \sigma_\emptyset)$, d.h. eine Folge von einzelnen Constraints und eine leere Substitution $\sigma_\emptyset$.

$$\begin{array}{l}
r_1 \colon s_0 \\
r_2 \colon s_0, s_1 \\
r_3 \colon s_0, s_1, s_2 \\
\ldots \colon \ldots \\
r_p \colon s_0, s_1, s_2, \ldots, s_{p-1}
\end{array}$$

3. Löse das System C_0, d.h. finde σ, so daß jedes Constraint erfüllt ist, d.h.

$$\sigma(r_i) \in d(\{\sigma(s_0), \ldots, \sigma(s_{i-1})\})$$

Wende dazu nicht-deterministisch Reduktionsregeln auf das aktuelle Constraint-System (C, σ) an. Wende die Regeln so lange an, bis keine Regel mehr anwendbar ist oder das aktuelle Constraint-System in Normalform ist.

Ein Constraint-System ist in Normalform, falls bei allen Constraints des Systems die linke Seite eine Variable ist – dann ist das System lösbar, indem alle Variablen mit $I \in S_0$ belegt werden.

Die Anwendung von Regeln erfolgt nach folgendem Schema:

(a) Wähle das erste Constraint, bei der die linke Seite keine Variable ist – ein solches Constraint nennen wir *aktiv*. Es sei von der Form $t : T$

(b) Entferne alle Variablen $x \in T$ aus T. Dies ändert die Lösbarkeit des Constraint-Systems nicht, da jede Variable vorher auf einer linken Seite vorgekommen sein muß, d.h. (da wir ein aktives Constraint betrachten) die Variable in einem früheren Constraint in der Form $x : T'$ auftreten muß.

(c) Wende auf das aktive Constraint eines der Regeln aus der Liste[2] an.

Anmerkungen zur Liste: **Unifikation**: Seien zwei Terme mit t_1, t_2 gegeben, gesucht ist eine Substitution σ mit $\sigma(t) = \sigma(t')$. Gesucht ist oft auch ein „allgemeinster“ Unifikator, am Beispiel: Sei $t_1 = \langle x, A \rangle$, $t_2 = \langle x, A \rangle$; dann ist ein möglicher Unifikator:

$$\sigma = \{x \mapsto B, y \mapsto B\}$$

Ein allgemeinerer Unifikator ist (mit einer Variablen z):

$$\sigma = \{x \mapsto z, y \mapsto z\}$$

formale Definition: Ein allgemeinster Unifikator σ *(most general unificator, MGU)* für t_1, t_2 ist ein Unifikator für t_1 und t_2, so daß für alle Unifikatoren σ' von t_1, t_2 gilt: Es exsitiert eine Substitution τ, so daß $\tau \circ \sigma = \sigma'$, d.h. daß $\tau(\sigma(x)) = \sigma'(x)$.

THEOREM: Der Constraint-Algorithmus ist korrekt, ist vollständig und er terminiert.

[2] siehe Kopie

3 Faire Vertragsprotokolle

3.1 Szenario und Ziele

Alice und Bob wollen einen gemeinsamen Vertrag unterschreiben, es soll jedoch keine Partei im Vorteil sein, d.h. keine Partei soll einen von der Gegenseite unterschriebenen Vertrag vorhalten können, ohne daß er selbst den Vertrag unterschrieben haben muß. Das denkbar einfachste Protokoll erlaubt es Bob, Alices Unterschrift zu erhalten, ohne selbst unterzeichnen zu müssen:

$$\begin{array}{ll} (1) & A \to B\colon \quad \texttt{sign}_A(text) \\ (2) & B \to A\colon \quad \texttt{sign}_B(text) \end{array}$$

Definiere hierzu entsprechende Sicherheitsziele:

- *fairness*:
  - Wenn ein unehrlicher Teilnehmer einen Vertrag hat, dann muß es dem ehrlichen Teilnehmer möglich sein, auch einen solchen zu bekommen.
  - Wenn ein ehrlicher Teilnehmer ein `abort` hat, dann kann ein unehrlicher Teilnehmer keinen Vertrag mehr bekommen.
- *balance*: Zu keinem Zeitpunkt im Protokolllauf soll es möglich sein, daß eine Partei sowohl eine Strategie hat, die zur Vertragsunterzeichnung führt, als auch eine Strategie, die zum Abbruch der Unterzeichnung führt.
- *abuse freeness*: Es soll gewährleistet sein, daß Bob an keiner Stelle eine dritte Person Charlie davon überzeugen kann, daß Alice den Vertrag unterschreiben möchte, so daß Bob gleichzeitig noch den Ausgang des Protokolls (Unterzeichnung oder Abbruch) beeinflussen kann[3].

[3] Beispiel: Bob verkauft etwas, Alice gibt ein Gebot ab und der erste Schritt des Vertragsprotokolls läuft ab; dann kann Bob zu Charlie gehen und ihm sagen: „Wenn Du mir mehr bietest als Alice, dann verkaufe ich an Dich!“...

3.2 Vertragsprotokoll von Garay, Jakobsson und MacKenzie

Eine mögliche Lösungen dieses Problems setzen sich aus mehreren Protokollen zusammen. Zunächst das *exchange*-Protokoll für den Normalfall:

$$\begin{array}{lll} (1) & A \to B\colon & \mathtt{sign}_A(\text{„ich will den Vertrag“}) \\ (2) & B \to A\colon & \mathtt{sign}_B(\text{„ich will den Vertrag“}) \\ (3) & A \to B\colon & \mathtt{sign}_A(\mathit{text}) \\ (4) & B \to A\colon & \mathtt{sign}_B(\mathit{text}) \end{array}$$

Nun gibt es mehrere Zusatzprotokolle:

- *abort*: Alice ruft eine Trusted Third Party T an, um den Vertrag abzubrechen, falls Bob Nachricht (2) nicht schickt.
- *recover*: Alice oder Bob rufen die Trusted Third Party an, um den unterschriebenen Vertrag zu erhalten, falls sie Nachricht (3) oder (4) nicht erhalten.

Neuer kryptographischer Baustein ist eine *private contract signature (PCS)* $\mathtt{PCS}_A(m, B, T)$ mit folgenden Eigenschaften:

- Die Trusted Third Party T kann $\mathtt{PCS}_A(m, B, T)$ in eine allgemein verifizierbare Signatur $\mathtt{sign}_A(m)$ umwandeln.
- Bob kann eine falsche Version von $\mathtt{PCS}_A(m, B, T)$ erzeugen, die von anderen Teilnehmern als A, B und T nicht von der Originalversion von $\mathtt{PCS}_A(m, B, T)$ unterschieden werden kann.

Damit kann nun folgendes *exchange*-Protokoll (ein Protokoll von Garay, Jakobsson und MacKenzie) aufgebaut werden:

$$\begin{array}{lll} (1) & A \to B\colon & \mathtt{PCS}_A(\mathit{text}, B, T) \\ (2) & B \to A\colon & \mathtt{PCS}_B(\mathit{text}, A, T) \\ (3) & A \to B\colon & \mathtt{sign}_A(\mathit{text}) \\ (4) & B \to A\colon & \mathtt{sign}_B(\mathit{text}) \end{array}$$

Das *abort*-Protokoll für Alice:

$$(1)\quad A \to T\colon \quad \mathtt{sign}_A(\mathit{text}, A, B, \mathit{abort}) =: m$$

$$(2)\quad T \to A\colon \quad \begin{cases} \mathtt{sign}_B(\mathit{text}) & \text{falls } \mathtt{recovered}(A, B, \mathit{text}) = \mathtt{true} \\ \mathtt{sign}_T(m) & \text{falls } \mathtt{aborted}(A, B, \mathit{text}) = \mathtt{true} \end{cases}$$

Das *recover*-Protokoll für Alice und Bob:

$$(1)\quad X \to T\colon \quad \mathtt{PCS}_A(\mathit{text}, B, T), \mathtt{PCS}_B(\mathit{text}, A, T)$$

$$(2)\quad T \to A\colon \quad \begin{cases} \mathtt{sign}_Y(\mathit{text}) & \text{falls } \mathtt{recovered}(A, B, \mathit{text}) = \mathtt{true} \\ \mathtt{sign}_T(m) & \text{falls } \mathtt{aborted}(A, B, \mathit{text}) = \mathtt{true} \end{cases}$$

3.3 Alternating Time Temporal Logic

Die Eigenschaft *balance* kann durch Erreichbarkeit nicht mehr beschrieben werden. Wir brauchen hier eine *Logik*, um diese Sicherheitseigenschaft zu beschreiben – hier betrachten wir *ATL (alternating time temporal logic)*.

BEISPIEL: Ein Term in ATL, der die Eigenschaft *balance* modelliert:

$$\neg(\langle\langle R\rangle\rangle \diamond contract_R \wedge \langle\langle R\rangle\rangle \diamond (aborted \wedge \neg \langle\langle O_H\rangle\rangle \diamond contract_O))$$

Diese Formel besagt: Es ist nicht der Fall, daß R eine Strategie hat, um an den Vertrag zu kommen, und eine andere, in der zum einen abgebrochen wird und zum anderen O keine Möglichkeit mehr hat, an den Vertrag zu kommen.

3.3.1 Alternierendes Transitionssystem

DEFINITION: Alterniernde Transitionssysteme (ATS) haben die folgende Gestalt:

$$S = (\Pi, \Sigma, Q, \pi, \delta)$$

- Π ist die Menge von propositionalen Variablen (aussagenlogische Variablen mit Werten 0 oder 1)
- Σ ist die Menge der Agenten
- Q ist eine Menge von Zuständen
- $\pi\colon Q \to 2^{\Pi}$ ist eine Abbildung, die jedem Zustand q die Menge der Variablen in Π zuordnet, die in q wahr sind, d.h. den Wert 1 haben.
- $\delta\colon Q \times \Sigma \to 2^{2^Q}$ ist eine Abbildung der möglichen Folgezustände, abhängig von der Wahl der Variablen-Wahl eines Agenten, siehe Beispiel unten.

Zusätzlich gelte folgende Bedingung: Für $\Sigma = \{a_1, \ldots, a_n\}$ und für alle $q \in Q$ sowie $Q_1, \ldots, Q_n \subseteq Q$ mit $Q_i \in \delta(q, a_i)$ gilt, daß

$$Q_1 \cap \ldots \cap Q_n = \{q'\} \text{ für ein } q' \in Q$$

BEISPIEL: Variablen seien x und y. Jeder Zustand wird beschrieben durch eine Belegung der Variablen, schreibe q_{xy}, $q_{\bar{x}y}$, $q_{x\bar{y}}$ bzw. $q_{\bar{x}\bar{y}}$ für die vier möglichen Zustände. Angenommen, ein Agent a kann die Variable x beeinflussen, aber nicht die Variable y. Dann kann er aus einem Zustand q_{xy} übergehen zu zwei

möglichen Mengen – je nachdem, wie y beeinflußt wird:

$$\delta(q_{xy}, a) = \{\underbrace{\{q_{xy}, q_{x\bar{y}}\}}_{\text{setze } x}, \underbrace{\{q_{\bar{x}y}, q_{\bar{x}\bar{y}}\}}_{\text{setze } \bar{x}}\}$$

DEFINITION:

- Ein Zustand q' ist ein *a-Nachfolger* von q, falls ein Q' mit $q' in \mathbb{Q}'$ existiert, so daß $Q' \in \delta(q, a)$. Die Menge der a-Nachfolger von q wird beschrieben durch $\text{succ}(q, a)$.
- Ein Zustand q' ist ein *Nachfolger* von q, falls q' ein a-Nachfolger von q ist für alle $a \in \Sigma$. Die Menge der Nachfolger von q ist $\text{succ}(q)$.
- Eine *Berechnung in S* ist eine Sequenz $\lambda = q_0, q_1, q_2, \ldots$, so daß q_{i+1} Nachfolger von q_i ist. Eine *q-Berechnung* ist Berechnung, die mit q startet. Notationen:

$$\lambda[i] = q_i \qquad \lambda[i, \infty] = q_i, q_{i+1}, \ldots \qquad \lambda[i, j] = q_i, q_{i+1}, \ldots, q_j$$

3.3.2 Syntax von ATL

Lege nun die *Syntax* von ATL fest: Gegeben seien endliche Menge Π von Variablen und Σ von Agenten.

DEFINITION: *ATL-Formeln* sind induktiv definiert wie folgt:

1. Für alle $p \in \Pi$ ist p eine Formel.
2. Für Formeln φ, ψ sind $\neg\varphi$ und $\varphi \vee \psi$ (und auch $\varphi \wedge \psi$) Formeln.
3. Für $A \subseteq \Sigma$ und Formeln φ, ψ sind folgendes Formeln:

$\langle\langle A\rangle\rangle \circ \varphi$	$\langle\langle A\rangle\rangle \Box\varphi$	$\langle\langle A\rangle\rangle \varphi \cup \psi$	$\langle\langle A\rangle\rangle \diamond \varphi$
next	*always*	*until*	*some time*

3.3.3 Semantik von ATL

Definiere zunächst, was *Strategien* sind:

DEFINITION:

- Eine *Strategie* für Agent a ist eine Funktion f_a mit folgender Eigenschaft:

$$f_a \colon Q^+ \to 2^Q \quad \text{mit} \quad \forall\, \lambda \in Q^*, q \in Q \colon f_a(\lambda \circ q) \in \delta(q, a)$$

- Für eine Menge von Agenten $A \subseteq \Sigma$ sei $F_A = \{\, f_a \mid a \in A\}$ eine Menge von Strategien.

- $\mathrm{out}\,(q, F_A)$ ist die Menge der möglichen Berechnungen ausgehend von q, falls sich alle Agenten aus $a \in A$ gemäß ihrer Strategie $f_a \in F_A$ verhalten:

$$\mathrm{out}\,(q, F_A) = \left\{ q_0, q_1, \ldots \;\middle|\; \begin{array}{ll} & q_0 = q \\ \wedge & q_{i+1} \text{ Nachfolger von } q_i \\ \wedge & q_{i+1} \in \bigcap_{a \in A} f_a(\lambda[0, i]) \end{array} \right\}$$

Nun kann man die Semantik der oben definierten Formeln formalisieren:

DEFINITION: Bezeichne „φ gilt int q bezüglich S“ mit $S, q \models \varphi$.

$$\begin{aligned}
S, q \models p &\iff p \in \pi(q) \\
S, q \models \neg\varphi &\iff S, q \not\models \varphi \\
S, q \models \varphi_1 \vee \varphi_2 &\iff S, q \models \varphi_1 \text{ oder } S, q \models \varphi_2 \\
S, q \models \langle\!\langle A \rangle\!\rangle \circ \varphi &\iff \exists\, F_A\, \forall\, \lambda \in \mathrm{out}\,(q, F_A) : S, \lambda[1] \models \varphi \\
S, q \models \langle\!\langle A \rangle\!\rangle \Box\varphi &\iff \exists\, F_A\, \forall\, \lambda \in \mathrm{out}\,(q, F_A)\ \forall\, i \geq 0 \colon S, \lambda[i] \models \varphi \\
S, q \models \langle\!\langle A \rangle\!\rangle\, \varphi_1 \cup \varphi_2 &\iff \exists\, F_A\, \forall\, \lambda \in \mathrm{out}\,(q, F_A)\ \exists\, j \geq 0 \colon \\
&\qquad ((\,\forall\, i < 0 \colon S, \lambda[i] \models \varphi_1) \wedge (S, \lambda[j] \models \varphi_2))
\end{aligned}$$

Syntaktischer Zucker:

- Die Semantik von $\diamond$ (some time):

$$\diamond\neg\varphi \iff \neg\Box\varphi$$

- $[\![A]\!] \circ \varphi$ besagt, daß die Agenten in A nicht verhindern können, daß $\circ\varphi$ irgendwann auf jeden Fall falsch ist. Umschreibung:

$$[\![A]\!] \circ \varphi \iff \neg\, \langle\!\langle A \rangle\!\rangle \circ \neg\varphi$$

Es gilt folgender Zusammenhang:

$$S, q \vdash \langle\langle A \rangle\rangle \circ \varphi \Longrightarrow S, q \vdash [\![\Sigma \setminus A]\!] \circ \varphi$$

„$\Rightarrow$“ ist klar nach Definition

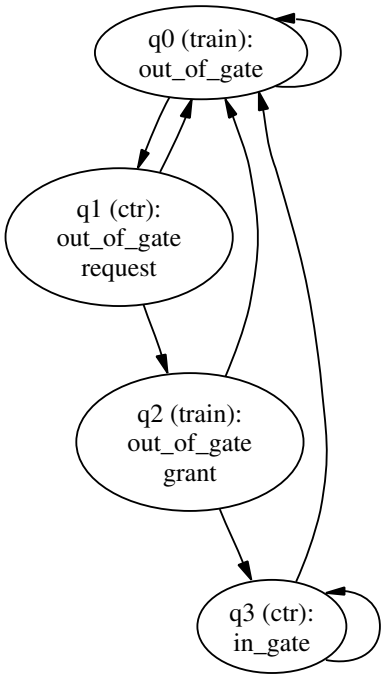
„$\not\Leftarrow$“ Sei $\Sigma = \{a, b\}$, $\pi(q_i) = \{p_i\}$ und

$$\begin{aligned} \delta(q, a) &= \{\{q_1, q_2\}, \{q_3, q_4\}\} \\ \delta(q, b) &= \{\{q_1, q_3\}, \{q_2, q_4\}\} \end{aligned}$$

Dann ist folgendes ein Gegenbeispiel für die Rückrichtung obiger Gleichung:

$$\begin{aligned} q &\not\models \langle\langle a \rangle\rangle \circ (p_1 \vee p_4) \\ q &\vdash \langle\langle b \rangle\rangle \circ (p_1 \vee p_4) = \neg \langle\langle b \rangle\rangle \circ (\neg p_1 \wedge \neg p_4) \end{aligned}$$

BEISPIEL: Seien zwei Agenten gegeben: $\Sigma = \{train, ctr\}$. Stelle die Zustände $Q = \{q_0, q_1, q_2, q_3\}$ und die Transitionen durch folgenden Transitionsgraphen dar:



Es gilt zum Beispiel im Zustand q_0:

$$\begin{aligned} \delta(q_0, train) &= \{\{q_0\}, \{q_1\}\} \\ \delta(q_0, ctr) &= \{Q\} \end{aligned}$$

Nun kann man Formeln angeben:

$$S, q_0 \not\models \langle\langle train \rangle\rangle \diamond \text{in\_gate}$$

Definiere...

DEFINITION: Eine *Einschränkung (constraint)* γ von δ ist:

$$\gamma \colon Q \times \Sigma \to 2^{2^Q} \text{ mit } \gamma \colon (q, a) \mapsto \{Q_1, \ldots, Q_n\} \in \delta(q, a)$$

Damit gilt $\gamma(q, a) \subseteq \delta(q, a)$.

- Gegeben sei $\lambda = q_0, q_1$. Dann ist γ *a-möglich* an der Stelle $i \geq 0$, falls $\gamma(q_i, a) \neq \emptyset$.
- γ ist zudem a-gewählt an der Stelle $i \geq 0$, falls $Q' \in \gamma(q_i, a)$ existiert mit $q_{i+1} \in Q$
- Die Berechnung λ ist *schwach* (γ, A)*-fair*, falls für alle Agenten $a \in A$ entweder
  - entweder unendlich viele Positionen in λ existieren, so daß γ nicht a-möglich ist,
  - oder unendlich viele Positionen in λ existieren, so daß γ a-gewählt ist.
- Die Berechnung λ ist *stark* (γ, A)*-fair*, falls für alle Agenten $a \in A$
  - entweder nur endlich viele Positionen in λ existieren, so daß γ a-möglich ist,
  - oder unendlich viele Positionen in λ existieren, so daß γ a-gewählt ist.

Starke Fairness bedeutet: Wenn unendlich oft so angefragt wird, daß ein Agent sich fair verhalten kann, dann muß er sich auch fair verhalten.
Sei $\gamma(q_1, ctr) = \{\{q_2\}\}$, für alle anderen q und a gelte $\gamma(q, a) = \emptyset$. Dann gilt für alle stark fairen Berechnungen:

$$S, q_0 \models \langle\langle train \rangle\rangle \diamond \text{in\_gate}$$

Für schwach faire Berechnungen gilt dies nicht!

3.3.4 Protokollmodellierung in ATL

[4]

Um ein Protokoll mit Hilfe von ATL zu analysieren, überprüft man eine Beziehung der folgenden Form:

$$S, q \vdash \varphi$$

Dabei ist das Protokoll in S kodiert, ebenso ein möglicher Angreifer etc. – die zu prüfende Bedingung, etwa *balance*, wird durch φ beschrieben. Um Protokolle effizient zu beschreiben, entwickelt man ein eigene Sprache.
BEISPIEL: Seien Variablen $Pi = \{p, q, r\}$, der Zustandsraum ergibt sich als $\{0, 1\}^3$. Sei $\Sigma = \{A, B, C\}$. Definiere nun *bewachte Befehle* der Form *Bedingung* $\rightarrow$ *Anweisung*:

```
A   true -> p := true
    true ->
B   p    -> q := true
    true ->
C   q    -> r := true
    true ->
```

Nun gelten folgende Formeln (mit (p, q, r) als Zustand):

$$\begin{aligned}
(0,0,0) &\vdash \langle\langle A, B, C\rangle\rangle \diamond r \\
(0,0,0) &\nvdash \langle\langle A, B\rangle\rangle \diamond r \\
(0,0,0) &\nvdash \langle\langle C\rangle\rangle \diamond r \\
(0,0,0) &\nvdash \langle\langle C\rangle\rangle \Box\neg r \\
(0,1,0) &\vdash \langle\langle C\rangle\rangle \diamond r
\end{aligned}$$

Für einzelne bewachte Befehle kann man nun *fairness contraints* angeben und die Formeln betrachten unter schwacher Fairness (WF) oder starker Fairness (SF) – sei Γ die Bedingung, daß C im ersten Befehl fair sein muß. Dann gilt:

$$(0,0,0), \Gamma \vdash_{WF} \langle\langle A, B\rangle\rangle \diamond r$$

Nun ist es möglich, Protokolle zu modellieren!

[4] KREMER, RASKIN, CSFW 2002

BEISPIEL: ASW:

$$
\begin{array}{lll}
(1) & A \to B: & m_1 = \mathtt{sign}_A(A, B, TTP, time, \mathtt{hash}(N_A)) \\
(2) & B \to A: & m_2 = \mathtt{sign}_B(\mathtt{hash}(m_1), \mathtt{hash}(N_B)) \\
(3) & A \to B: & m_3 = N_A \\
(4) & B \to A: & m_4 = N_B
\end{array}
$$

In der weiteren Beschreibung verwenden wir nun nur propositionale Variablen, d.h. in der weiteren Modellierung sind N_A etc. eine propositionale Variable.

ehrliche Alice (A_H):
$N_A \to h_{N_A} := \mathtt{true}$
$h_{N_A} \wedge A \wedge B \wedge TTP \wedge time \wedge \neg\, \mathtt{send}_{m_1} \wedge \neg \mathtt{stop}_A \to send_{m_1} := \mathtt{true}$

Kommunikationskanal:
$\mathtt{send}_{m_1} \wedge \neg m_1 \to m_1 := \mathtt{true}$
...

ehrlicher Bob (B_H):
$m_1 \wedge \neg \mathtt{stop}_B \wedge \neg A_B \wedge \neg B_B \wedge TTP_B$
$\quad \to A_B := \mathtt{true} \wedge B_B := \mathtt{true} \wedge TTP_B := \mathtt{true} \wedge \ldots$
$A_B \wedge B_B \wedge \ldots \to \mathtt{send}_{m_2} := \mathtt{true}$
...
$(N_A)_B \to contract_B := \mathtt{true} \wedge \mathtt{stop}_B := \mathtt{true} \wedge \mathtt{send}_{m_4} := \mathtt{true}$

unehrlicher Bob (B):
$\mathtt{true} \to \mathtt{send}_{m_4} := \mathtt{true}$

Wichtig ist hier die Fairness z.B. bei der Kommunikation mit der Trusted Third Party; normalerweise: Kommunikationskanal ist immer fair.
Resultate für das Protokoll von GARAY, JAKOBSSON und MACKENZIE bzw. für das von ASOKAN, SHOUP, WAIDNER:

- beide sind *fair*
- beide sind *timeliness*, soweit der Kommunikationskanal fair ist
- beide sind *balance*, falls der Kommunikationskanal zwischen der ehrlichen Partei und der Trusted Third Party nicht von einer unehrlichen Partei kontrolliert werden kann