Das Dolev und Yao Modell

-

Seminar Analyse kryptographischer Protokolle

Martin Stiel[1]

Humboldt Universität zu Berlin - Institut für Informatik

30. Dezember 2003

[1]stiel@informatik.hu-berlin.de

Zusammenfassung

Im Bemühen um die formale Analyse kryptographischer Protokolle wurden inzwischen eine Reihe von Modellen entwickelt. Ein relativ frühes Beispiel bildet der hier vorgestellte Versuch von *Danny Dolev* und *Andrew C. Yao*. Es stellt einen algebraischen Ansatz dar und beruht auf der Vorstellung von Nachrichten und Schlüsseln in Form von aus Buchstaben zusammengesetzten Wörtern. Wie bei anderen Verfahren auch werden lediglich Schwächen des kryptographischen Protokolls, nicht jedoch des zugrundeliegenden kryptographischen Systems untersucht. Das Modell versteht sich auch als Hilfestellung zum Design sicherer kryptographischer Protokolle.

Inhaltsverzeichnis

1 Einführung

Aufgrund der zunehmenden Nutzung von Protokollen mit assymetrischen Verschlüsselungsverfahren ist die Untersuchung deren Sicherheit von zunehmender Bedeutung. Das hierzu die informale Diskussion nicht (immer) ausreicht haben Fehler wie im Protokoll von Needham und Schröder in der Vergangenheit gezeigt. Oftmals übersteigt die Komplexität solcher Protokolle die Möglichkeiten informaler Analysen bei weitem.

2 Grundlagen

Assymetrische Verschlüsselungsverfahren (auch: *Public Key Encryption*) gehen von zwei, nicht auseinanderableitbaren Schlüsseln jedes Teilnehmers aus: Dem öffentlichen allgemein zugänglichen Schlüssel für die Ver- und dem privaten Schlüssel für die Entschlüsselung. Im weiteren Verlauf wird zur Analyse angenommen, daß das mit mit den Schlüsseln verwendete Verfahren sicher, d.h. nicht auflösbar ist. Zuerst zur Definition einiger Grundlagen die sich in der Notation größtenteils an [3] orientieren.

Definition 1 (Alphabet, Buchstaben, Wort)

Sei Σ eine endliche, nichtleere Menge. Ein *Wort* ist eine endliche Folge von Elementen aus Σ. Die Elemente von Σ werden auch *Buchstaben* oder *Symbole* genannt. $W(\Sigma)$ ist die Menge aller Wörter über dem *Alphabet* Σ.

Definition 2 (Wortlänge)

Sei Σ ein Alphabet. Die Abbildung:

$$l : W(\Sigma) \rightarrow I\!N$$

bezeichne die *Länge* eines Wortes über Σ.

Definition 3 (leeres Wort)

Das Wort λ mit $l(\lambda) = 0$ und $lt(\lambda) = \emptyset$ heißt das *leere Wort* und $W^+(\Sigma) = W(\Sigma) - \{\lambda\}$ ist die Menge der nichtleeren Wörter über dem Alphabet Σ.

Definition 4 (Wortsymbole)

Sei Σ ein Alphabet und $\gamma \in W(\Sigma)$, dann ist

$$lt(\gamma) \subseteq W(\Sigma)$$

die Menge der in γ vorkommenden Symbole.

Definition 5 (Konkatinationen, Präfix, Suffix)

Seien ϕ und τ Wörter über dem Alphabet Σ. Dann ist das Wort $\gamma = \phi\tau$ die

Konkatination (Verkettung) von ϕ und τ. $\phi = p(\gamma)$ nennt man dann auch *Präfix* und $\tau = s(\gamma)$ *Suffix* von $\phi\tau$. Die Menge $W(\Sigma)^\star$ ist dann die Menge aller Konkatinationen von Wörtern über dem Alphabet Σ.

Definition 6 (Ver- u. Entschlüsselungsfunktion)

Sei X ein Teilnehmer innerhalb eines kryptographisches Protokolls und Σ ein Alphabet, dann bezeichnet die Abbildung:

$$E_x : W(\Sigma) \rightarrow W(\Sigma)$$

die *Verschlüsselungsfunktion* von X und:

$$D_x : W(\Sigma) \rightarrow W(\Sigma)$$

dessen *Entschlüsselungsfunktion* mit

$$\forall_{M \in W(\Sigma)} : E_x(D_x(M)) = D_x(E_x(M)) = M$$

Seien E und D die Mengen der Ver- und Entschlüsselungsfunktionen aller beteiligten Teilnehmer.

Bemerkung Bei allen im folgenden eingeführten Operatoren und Funktionen, so auch bei den Ver- und Entschlüsselungsfunktionen, ist die Schreibweise $E_x M$ für X Teilnehmer und $M \in W(\Sigma)$ Klartext über Alphabet Σ identisch zu $E_x(M)$. Außerdem wird der Teilnehmer eines Protokolls und das Wort seines Namens synonym gebraucht.

Definition 7 (vollständig reduzierte Form)

Sei γ ein Wort über dem Alphabet Σ und X ein Teilnehmer. Dann ist $\gamma|_x$ die *gegenüber X vollständig reduzierte Form* von γ, dergestalt, daß iterativ alle Teilwörter $E_x(D_x(\sigma))$ respektive $D_x(E_x(\sigma))$ zu σ reduziert wurden.

Das Wort $\overline{\gamma}$ ist dann die gegenüber allen Nutzern des Systems *vollständig reduzierte Form* von γ.

Definition 8 (Komplement von Ver- u. Entschlüsselungsfunktionen)

Sei γ ein Wort über dem Alphabet Σ und X ein Teilnehmer. Das *Komplement* der Ent- und Verschlüsselungsfunktionen von X ist $E_x^c = D_x$ respektive $D_x^c = E_x$ Das Wort γ^c bezeichnet dann das Wort γ in dem alle Vorkommen von E_x und D_x durch ihr Komplement ersetzt wurden.

Für die im folgenden betrachteten Protokolle wird angenommen, daß für alle Teilnehmer X die Funktionen aus E frei zugänglich sind während D_x nur dem Teilnehmer X bekannt ist. Die Verschlüsselung wird wie bereits eingangs erwähnt als sicher vorausgesetzt, d.h. die Kenntnis von $E_x(M)$ für $M \in W(\Sigma)$ darf keinerlei Rückschlüsse auf M zulassen.

3 Cascade und Name-Stamp Protokolle

In der weiteren Betrachtung werden zwei Arten von Protokollen unterschieden. *Cascade Protokolle* sind die einfachste Form von *Public Key Protokollen*: Die beteiligten Nutzer tauschen schrittweise nicht weiter beschränkte Nachrichten aus und können diese vor dem Absenden beliebig mit den ihnen zur Verfügung stehenden Ver- und Entschlüsselungsfunktionen modifzieren. Ein gedachter Angreifer kann alle übertragenen Nachrichten abfangen, Kommunikation mit anderen initiieren und selber zum Empfänger von Nachrichten werden.

Name-Stamp Protokolle erweitern die einfachen *Cascade Protokolle* um die Möglichkeit den übertragenen Klartext um Namen der beteiligter Nutzer zu erweitern, diese zu extrahieren und zu löschen. Die erlaubten Operationen des Angreifers erweitern sich entsprechend. Die im voherigen Abschnitt gemachten Annahmen gelten weiterhin.

3.1 Beispiele

In den folgenden Beispielen seien die Nachrichten Tripel aus Sender, Empfänger und übertragenem Text. Wir werden die Notation von Nachrichten später noch präzisieren.

Beispiel 1

Seien X und Y Namen beliebiger Teilnehmer.

(i) X sendet $(X, Y, E_y(M))$

(ii) Y anwortet $(Y, X, E_x(M))$

Dieses einfache *Cascade* Protokoll kann durch Abfangen der von X gesendeten Nachricht durch einen Angreifer Z und dem Austausch des Senders durch Z gebrochen werden, da Y dann mit $(B, Z, E_Z(M))$ statt an X antwortet. Die Schwäche liegt in der fehlenden Möglichkeit für Y den ursprünglichen Absender der Nachricht eindeutig zu bestimmen. Eine Möglichkeit dies sicherzustellen wäre die Einbettung des Senders in die verschlüsselte Nachricht wie im folgenden Beispiel, einem *Name-Stamp* Protokoll.

Beispiel 2

Wiederum seien X und Y die Namen zweier verschiedener beliebiger Teilnehmer.

(i) X sendet $(X, Y, E_y(MX))$

(ii) Y anwortet $(X, Y, E_x(MY))$

Als drittes Beispiel sei ein Protokoll aufgeführt, welches ebenfalls den Sendernamen verschlüsselt, aber aufgrund der Kombination von Verschlüsselungsfunktionen unsicher ist.

Beispiel 3

X und Y sind Namen beliebiger Teilnehmer, dann:

(i) X sendet $(X, Y, E_y(E_y(M)X))$

(ii) Y anwortet $(X, Y, E_x(E_x(M)Y))$

Weiter unter wird die Unsicherheit dieses Protokoll bewiesen.

3.2 Ein Modell für Zwei-Parteien Cascade Protkolle

Definition 9 (Zwei-Parteien Cascade Protokoll)

Seien X und Y zwei Teilnehmer, Σ Alphabet, $M \in W(\Sigma)$ Klartext sowie $t, t' \in IN$ mit $t' = t$ oder $t' = t - 1$. $T = \{\alpha_i(X,Y), \beta_j(X,Y)\}$ ist dann ein *Zwei-Parteien Cascade Protokoll* mit:

$$\alpha_i(X,Y) \in W(\{E_x, E_y, D_x\}), \quad 1 \le i \le t$$
$$\beta_j(X,Y) \in W(\{E_x, E_y, D_y\}), \quad 1 \le j \le t'$$

und

$$\begin{aligned} N_1(X,Y) &= \alpha_1(X,Y) \\ N_{2j}(X,Y) &= \beta_j(X,Y)N_{2j-1}(X,Y), \ 1 \le j \le t' \\ N_{2i+1}(X,Y) &= \alpha_{i+1}(X,Y)N_{2i}(X,Y), \ 1 \le i \le t-1 \end{aligned}$$

Im i-ten Schritt wird zwischen X und Y die Nachricht $N_i(X,Y)M$. (Beide Teilnehmer senden also abwechselnd.)

Beispiel 4

Seien die ersten beiden Wörter der Wortmengen eines *Cascade Protkolls* für die Teilnehmer X und Y mit $\alpha_1 = E_y$ und $\beta_1 = E_x D_y$ gegeben und Σ ein Alphabet, $M \in W(\Sigma)$ Klartext. Im ersten Schritte sendet X die Nachricht

$$N_1(X,Y)M = \alpha_1(X,Y)M = E_y(M)$$

mit dem Klartext M an Y. Letzterer empfängt die Nachricht und antwortet mit

$$N_2(X,Y)M = \beta_1(X,Y)N_1(X,Y)M = E_x(D_y(E_y(M))) = E_x(M)$$

Dies entspricht im wesentlich dem bereits oben eingeführten *Beispiel 1* in der neuen Notation.

Wir suchen nun nach einer sinvollen Definition für ein sicheres Zwei-Parteien Cascade Protokoll.

Definition 10 (sicheres Cascade Protokoll)

Sei $T = \{\alpha_i, \beta_j\}$ ein Zwei-Parteien Cascade Protokoll mit den Teilnehmern X, Y und Z ein dritter Teilnehmer. Weiterhin seien

$$W_1 := \{\alpha_i(X, Y) \mid X \neq Y \wedge 1 \leq i \leq t\}$$
$$W_2 := \{\beta_j(X, Y) \mid X \neq Y \wedge 1 \leq j \leq t'\}$$

Dann ist T *sicher*, wenn kein

$$\gamma \in (W(E \cup \{D_z\}) \cup W_1 \cup W_2)^\star$$

existiert, so daß

$$\overline{\{\gamma, N_i(X, Y)\}^\star} = \lambda,\ 1 \leq i \leq t'$$

Anderenfalls ist T *unsicher*.

Angenommen Teilnehmer X möchte Teilnehmer Y eine Nachricht senden. So hat ein Angreifer Z die Möglichkeit alle gesendeten Nachrichten abzufangen und mit folgenden Wörtern beliebig zu verketten.

(i) Er kann seine eigenen oder öffentliche Verschlüsselungsfunktionen verwenden ($W(E \cup \{D_z\})$).

(ii) Z kann die im Cascade Protokoll verwendeten Wörter der Menge W_2 benutzen, indem er Teilnemer Y eine Nachricht $M \in W(\Sigma)$ beliebig unter der Vorgabe er ist X (wir haben keine *Name-Stamps* wie im folgenden Protokoll) eine Nachricht sendet. Teilnehmer Y antwortet gemäß dem *Zwei-Parteien Cascade Protokoll* mit $\beta_j(X, Y)M$ wobei j durch geeignete Wahl des Zeitpunkts durch Z beliebig ist.

(iii) In gleicher Weise kann Z die Wörter der Menge W_1 nutzen indem er eine Nachricht von Y an X abfängt und sich unter der Vorgabe er sei Y mit einer beliebigen Nachricht zu einem geeigneten Zeitpunkt an X wendet.

Kann nun bei einer diese möglichen Verkettungen ein leeres Wort als vollständig reduzierte Form erzielt werden, so ist es einem Angreifer Z möglich die eingesetzten und ihm unbekannten Verschlüsselungsfunktionen der kommunizierenden Teilnehmer X und Y 'aufzulösen'. Da im *Zwei-Parteien Cascade Protokoll* lt. Definition die Kommunikation immer von dem Teilnehmer X mit dem Wort

$\alpha_1(X,Y)$ als Suffix des Klartextes begonnen wird, muß Z zur Erlangung eines Wortes aus W_2 das Senden eine Nachricht durch X abwarten. Weil dies nicht notwendigerweise der Fall ist, spricht [1] von einer *konservativen* Definition für Protokollsicherheit.

3.2.1 Sichere Zwei-Parteien Cascade Protokolle

Im vorherigen Abschnitt haben wir eine intuitive Definition für die Sicherheit eines Zwei-Parteien Cascade Protokoll gesehen. Ziel dieses Abschnitts soll nun die Formulierung eines etwas differenzierteren Ansatzes zur Charakterisierung von sicheren Cascade Protokollen sein. Hierfür zunächst einige Definitionen.

Definition 11 (ausgeglichene Wörter)

Sei $\gamma \in W(E,D)$ ein Wort und X ein Teilnehmer. Das Wort γ ist *ausgeglichen gegenüber* X wenn gilt:

$$D_x \in lt(\gamma) \Rightarrow E_x \in lt(\gamma)$$

Definition 12 (ausgeglichenes Cascade Protokoll)

Seien X und Y Teilnehmer eines Zwei-Parteien Cascade Protokolls $T = \{\alpha_i, \beta_j\}$, sowie alle $\alpha_i \in T$ und alle $\beta_j \in T$ in vollständig reduzierter Form. T ist ein *ausgeglichenes Cascade Protokoll* wenn alle $\alpha_i \in T$ ausgeglichen gegenüber X und alle $\beta_j \in T$ ausgeglichen gegüber Y sind.

Auf Grundlage dieser beiden Definitionen läßt sich nun die im letzten Abschnitt vorgestellte intuitive Anschauung von sicheren Cascade Protkollen durch folgenden Satz erweitern.

Satz 1

Seien X und Y Teilnehmer eines Zwei-Parteien Cascade Protokolls $T = \{\alpha_i, \beta_j\}$. Das Protokoll T ist sicher genau dann wenn es ausgeglichen ist und es gilt:

$$lt(\alpha_1(X,Y)) \cap \{E_x, E_y\} \neq \emptyset$$

Definition 13 (doppelt geprüftes Cascade Protokoll)

Ein Cascade Protokoll $T = \{\alpha_i, \beta_j\}$ mit den Teilnehmern X und Y ist *doppelt geprüft* wenn gilt:

$$\exists_{i \in IN} : lt(\overline{N_i(X,Y)}) \subseteq \{E_x, D_y, D_x\}$$

und

$$\exists_{2 \leq j \in IN} : lt(\overline{N_j(X,Y)}) \subseteq \{E_y, D_y, D_x\}$$

Satz 2

Jedes doppelt geprüfte Zwei-Parteien Cascade Protokoll ist unsicher.

Bemerkung: Als direkte Folgerung aus *Satz 2* ergibt sich für ein sicheres Zwei-Parteien Cascade Protokoll folgende Aussage: Wenn der Empfänger einer verschlüsselten Nachricht den Klartext $M \in W(\Sigma)$ gewinnen kann, so darf es dem Sender nicht einfach möglich sein aus der Entschlüsselung einer an ihn zurückgesandten Nachricht ebenfalls M zu erhalten. Damit ist das Protokoll aus *Beispiel 1* offensichtlich unsicher.

3.3 Ein Modell für Name-Stamp Protokolle

In der Betrachtung von Cascade Protokollen war die innere Struktur der Nachrichten beliebig. Nun wird ein Modell betrachtet in dem ein Teil der übertragenen Nachricht für den Namen des Senders reserviert ist. Zuerst führen wir einige neue Operationen auf Wörtern über die bereits bekannte Ver- und Entschlüsselung hinaus ein:

Definition 14 (Anhängen und Matchen)

Seien Σ ein Alphabet und $x, y, z \in W(\Sigma)$, dann ist i_x das *Anhängen* mit $i_x(y) = yx$, d_x bezeichnet das *Matchen* mit:

$$d_x(y) = \begin{cases} z & : \quad y = zx \\ n.d. & : \quad sonst \end{cases}$$

Bemerkung: Offensichtlich gilt für zwei beliebige Wörter x und y:

$$d_x(i_x(y)) = y \quad (\star)$$

Um sicherzustellen, daß die Kommunikation zwischen zwei Teilnehmern X und Y nicht terminiert, weil die Anwendung von d_x in einem undefinierten Zustand verharrt fordern wir, daß alle d_x Operationen durch wiederholtes Anwenden von $(\star)$ beseitigt wurden.

Aufgrund der neu eingeführten Operationen erweitern wir nun die *Definition 7* der *vollständig reduzierten Form*:

Definition 15 (vollständig reduzierte Form, nicht weiter reduzierbar)

Sei Σ ein Alphabet und $\gamma \in W(\Sigma)$. Dann bezeichnet $\overline{\gamma}$ Die *vollständig reudzierte Form* von γ dergestalt, daß $\overline{\gamma}$ nach Defintion 7 vollständig reduziert ist und zusätzlich alle Teilwörter $d_x(i_x(y))$ von γ für beliebige x und y zu y reduziert wurden. Das Wort γ ist *nicht weiter reduzierbar*, wenn $\overline{\gamma} = \gamma$ gilt.

Definition 16 (Zwei-Parteien Name-Stamp Protokoll)

Seien X und Y zwei Teilnehmer sowie $t, t' \in IN$ mit $t' = t$ oder $t' = t - 1$. $T = \{\alpha_i(X,Y), \beta_j(X,Y)\}$ ist dann ein *Zwei-Parteien Name-Stamp Protokoll* mit:

$$\begin{aligned}\alpha_i(X,Y) &\in W(\{E_x, E_y, D_x, i_x, i_y, d_x, d_y\}), \quad 1 \leq i \leq t\\ \beta_j(X,Y) &\in W(\{E_x, E_y, D_y, i_x, i_y, d_x, d_y\}), \quad 1 \leq j \leq t'\end{aligned}$$

und

$$\begin{aligned}N_1(X,Y) &= \alpha_1(X,Y)\\ N_{2j}(X,Y) &= \beta_j(X,Y)N_{2j-1}(X,Y),\ 1 \leq j \leq t'\\ N_{2i+1}(X,Y) &= \alpha_{i+1}(X,Y)N_{2i}(X,Y),\ 1 \leq i \leq t-1\end{aligned}$$

Im i-ten Schritt wird zwischen X und Y die Nachricht $N_i(X,Y)M$ mit dem Klartext M ausgetauscht. (Beide Teilnehmer senden also abwechselnd.)

Als erstes wird nun analog zum Vorgehen bei den Cascade Protokollen eine Definition für die Sicherheit angeboten.

Definition 17 (Sicheres Name-Stamp Protokoll (1))

Sei $T = \{\alpha_i, \beta_j\}$ ein Zwei-Parteien Name-Stamp Protokoll mit den Teilnehmern X, Y und Z ein dritter Teilnehmer sowie

$$\begin{aligned}V_Z &:= \{\alpha_i(X,Y) \mid X \neq Y \wedge i \geq 2\}\\ &\cup\ \{\beta_j(X,Y) \mid X \neq Y \wedge j \geq 1\}\\ &\cup\ W(E \cup D_z)\\ &\cup\ W(\{i_a, d_a \mid A \in \{X,Y,Z\}\})\end{aligned}$$

Dann ist T unsicher, wenn

$$\exists_{\gamma \in V_Z} : \overline{\{\gamma, N_i(X,Y)\}^\star} = \lambda$$

Anderenfalls ist T sicher.

Beispiel 5

Das als *Beispiel 3* eingeführte Beispiel notiert man für zwei Teilnehmer X und Y gemäß der vorangegangen Definition wie folgt: X sendet Y

$$\begin{aligned}N_1(X,Y)M &= \alpha_1(X,Y)M\\ &= E_y i_x E_y M\\ &= E_y(E_y(M)X)\end{aligned}$$

woraufhin Y antwortet

$$\begin{aligned}
N_2(X,Y)M &= \beta_1(X,Y)N_1(X,Y)M \\
&= E_x i_y E_x D_y d_x D_y N_1(X,Y)M \\
&= E_x i_y E_x D_y d_x D_y \alpha_1(X,Y)M \\
&= E_x i_y E_x D_y d_x D_y E_y i_x E_y M \\
&= E_x i_y E_x D_y d_x i_x E_y M \\
&= E_x i_y E_x D_y E_y M \\
&= E_x i_y E_x M \\
&= E_x(E_x(M)Y))
\end{aligned}$$

Dieses Protokoll ist unsicher weil mit

$$\begin{aligned}
\beta_1(Z,X) &= E_z i_x E_z D_x d_z D_x \\
\gamma &= D_z d_x D_z \beta_1(Z,X) E_x i_z d_y D_z d_x D_z \beta_1(Z,X) E_x i_z
\end{aligned}$$

gilt:

$$\begin{aligned}
\overline{\gamma N_2(X,Y)} &= D_z d_x D_z \beta_1(Z,X) E_x i_z d_y D_z d_x D_z \beta_1(Z,X) E_x i_z E_x i_y E_x \\
&= D_z d_x D_z \beta_1(Z,X) E_x i_z d_y D_z d_x D_z E_z i_x E_z D_x d_z D_x E_x i_z E_x i_y E_x \\
&= D_z d_x D_z \beta_1(Z,X) E_x i_z d_y D_z d_x D_z E_z i_x E_z D_x d_z i_z E_x i_y E_x \\
&= D_z d_x D_z \beta_1(Z,X) E_x i_z d_y D_z d_x D_z E_z i_x E_z D_x E_x i_y E_x \\
&= D_z d_x D_z \beta_1(Z,X) E_x i_z d_y D_z d_x D_z E_z i_x E_z i_y E_x \\
&= D_z d_x D_z \beta_1(Z,X) E_x i_z d_y D_z d_x i_x E_z i_y E_x \\
&= D_z d_x D_z \beta_1(Z,X) E_x i_z d_y D_z E_z i_y E_x \\
&= D_z d_x D_z \beta_1(Z,X) E_x i_z d_y i_y E_x \\
&= D_z d_x D_z \beta_1(Z,X) E_x i_z E_x \\
&= D_z d_x D_z E_z i_x E_z D_x d_z D_x E_x i_z E_x \\
&= D_z d_x D_z E_z i_x E_z D_x d_z i_z E_x \\
&= D_z d_x D_z E_z i_x E_z D_x E_x \\
&= D_z d_x D_z E_z i_x E_z \\
&= D_z d_x i_x E_z \\
&= D_z E_z \\
&= \lambda
\end{aligned}$$

3.3.1 Algorithmus

In diesem Abschnitt soll eine Algorithmus beschrieben werden, der für einen gegebenes Name-Stamp Protokoll $T = \{\alpha_i, \beta_i\}$ und einer Eingabenlänge

$$n = \sum_i l(\alpha_i) + \sum_j l(\beta_j)$$

in $\mathcal{O}(n^8)$ bestimmt, ob das Protokoll sicher ist. Hierfür zunächst einige Festlegungen.

Definition 18 (nicht reduzierbare Wortmengen)

Seien X und Y Teilnehmer eines Name-Stamp Protokolls und

$$\eta \in W(\{E_a, D_a, i_a \mid A \in \{X, Y\}\})$$

nicht weiter reduzierbar. Dann ist $C(\eta)$ die Menge aller nicht reduzierbaren Wörter

$$\delta \in W(\{E_a, D_a, i_a, d_a \mid A - Teilnehmer\})$$

mit $\overline{\delta\eta} = \lambda$.

Definition 19 (Wortmenge S)

Seien X, Y Teilnehmer eines Name-Stamp Protokolls $T = \{\alpha_i, \beta_j\}$.

$$\begin{aligned} S &:= \{\alpha_i \mid A, B \in \{X, Y, Z\}, A \neq B, i \geq 2\} \\ &\cup \{\beta_j \mid A, B \in \{X, Y, Z\}, A \neq B, j \geq 1\} \\ &\cup \{E_a, i_a, d_a \mid A \in \{X, Y, Z\}\} \\ &\cup \{D_z\} \end{aligned}$$

Seien X und Y Teilnehmer eines Name-Stamp Protokolls. Dann leistet der folgende Algorithmus das gewünschte.

CheckSecurity$(T = \{\alpha_i, \beta_j\})$

$M := \{\overline{N_i(X,Y)} \mid 1 \leq i \leq 2 \cdot t\}$
  for $\rho \in M$ **do**
    for $\gamma \in S^\star$ **do**
      if $\overline{\gamma} \in C(\rho)$ **then return false**
    end
  end
  return true
end

In [1] ist ein Beweis angegeben. Für den vollständigen Beweis wird ein Teil des Problems also die Frage ($\rho \in M$ wie oben):

$$\exists_{\gamma \in S^\star} : \; \overline{\gamma} \in C(\rho)$$

unter Nutzung einer Grammatik in das *Worterweiterungsproblem* überführt. Zum Verständnis dieses hier nicht aufgeführten Beweises empfehle ich z.B. [3] zu Hilfe zu nehmen.

4 Der aktive Saboteur

Bisher haben wir Protokolle betrachtet in denen die Möglichkeit bestand, daß ein Dritter Empfänger von Kommunikation zwischen zwei Teilnehmern werden konnte ohne selbst zuvor Nachrichten versand zu haben. Im folgenden wird nun von einem Saboteur ausgegangen, der selbständig versucht Teilnehmer zur Kommunikation aufzufordern.

Für die Sicherheit von Name-Stamp Protokoll führen wir nun eine Abwandlung ein, die der neuen Saboteurrolle gerecht wird.

Definition 20 (Sicheres Name-Stamp Protokoll bei aktivem Saboteur)

Sei $T = \{\alpha_i, \beta_j\}$ ein Zwei-Parteien Name-Stamp Protokoll mit den Teilnehmern X, Y und Z ein dritter Teilnehmer (aktiver Saboteur) sowie

$$\begin{aligned} V_Z \;\; := & \;\; \{\beta_j(X,Y) \mid X \neq Y \wedge j \geq 1\} \\ \cup & \;\; W(E \cup D_z) \\ \cup & \;\; W(\{i_a, d_a \mid A \in \{X,Y,Z\}\}) \end{aligned}$$

Dann ist T unsicher, wenn

$$\exists_{\gamma \in V_Z} : \overline{\{\gamma, N_i(X,Y)\}^\star} = \lambda$$

Anderenfalls ist T sicher.

Analog ändert sich die Definition für sichere Cascade Protokolle.

Definition 21 (sicheres Cascade Protokoll bei aktiven Saboteur)

Sei $T = \{\alpha_i, \beta_j\}$ ein Zwei-Parteien Cascade Protokoll mit den Teilnehmern X, Y und Z ein dritter Teilnehmer (aktiver Saboteur). Weiterhin sei

$$W_1 := \{\beta_j(X,Y) \mid X \neq Y \wedge 1 \leq j \leq t'\}$$

Dann ist T *sicher*, wenn kein

$$\gamma = (W(E \cup \{D_z\}) \cup W_1)^\star$$

existiert, so daß
$$\overline{\{\gamma, N_i(X,Y)\}^\star} = \lambda,\ 1 \leq i \leq t'$$
Anderenfalls ist T *unsicher*.

Analog zu Satz 1 läßt sich nun wieder ein weiteres Kriterium für sichere Cascade Protokolle finden.

Satz 3

Seien X und Y Teilnehmer eines Zwei-Parteien Cascade Protokolls $T = \{\alpha_i, \beta_j\}$. Das Protokoll T ist sicher genau dann wenn gilt:
$$\forall_{k\geq 1}:\ lt(\overline{N_k(X,Y)}) \cap \{E_x, E_y\} \neq \emptyset$$
und für alle $k \geq 1$ das Wort $\beta_k(X,Y)$ ausgeglichen gegenüber Y ist.

5 Ein Automat für Name-Stamp Protokolle

Definition 22 (Nichtdeterministischer Automat (NFA))

Der Fünf-Tupel
$$M = (Z, \Sigma, \delta, S, E)$$
ist ein *nichtdeterministischer Automat (NFA)* mit Z Zustandsmenge, Σ Eingabealphabet
$$\delta:\ Z \times \Sigma\ \rightarrow\ \mathcal{P}(Z)$$
Zustandsübergangsfunktion, $S \subseteq Z$ die Menge der Anfangszustände und $E \subseteq Z$ die Menge der Endzustände.

Definition 23 (Akzeptierte Wortmenge eines NFA)

Sei $M = (Z, \Sigma, \delta, S, E)$ ein NFA, $W(\Sigma) \ni x = x_1...x_n$, $Z' \subseteq Z$ und
$$\hat{\delta}:\ Z \times W(\Sigma)\ \rightarrow\ \mathcal{P}(Z)$$
mit $\hat{\delta}(Z', x) \ni z'$ gdw wenn eine Folge von Zuständen $z_0, ..., z_{n-1}, z' \in Z$ existiert mit $z_0 \in Z_0$, $z_i \in \hat{\delta}(z_{i-1}, x_i)$ für $i = 1, ..., n$ und $z' \in \hat{\delta}(z_{n-1}, x_n)$. Die *akzeptierte Wortmenge* W_M *des NFA* M ist dann
$$W_M = \{x \in W(\Sigma) \mid \hat{\delta}(S, x) \cap E \neq \emptyset\}$$

Beide Definition nach [4].

Definition 24 (NFA für Name-Stamp Protokoll)

Sei $T = \{\alpha_i, \beta_j\}$ ein Name-Stamp Protokoll und $A = (Z, \Sigma, \delta, S, E)$ ein NFA. A ist ein *NFA für das Name-Stamp Protokoll* T, wenn $S = \{0\}$, $E = \{1\}$ und
$$\Sigma = E \cup D \cup \{i_b, d_b \mid B \in \{X, Y, Z\}\}$$

Weiterhin bildet die Zustandsübergangsfunktion Kanten von Zustand 0 nach 1 entsprechend den Buchstaben aus α_1, Kanten von Zustand 0 nach 0 für alle

$$x \in \Sigma_z = E \cup \{D_z\} \cup \{i_b, d_b \mid B \in \{X, Y, Z\}$$

und entsprechend den Buchstaben aller α_i sowie β_j für die Teilnehmer X,Y und Z.

Beispiel 6

Sei $T = \{\alpha_1, \beta_1\}$ mit

$$\alpha_1 = E_y$$

und

$$\beta_1 = E_x D_y$$

das als *Beispiel 1* oben eingeführte Name-Stamp Protokoll. Der zugehörige NFA $A = (Z, \Sigma, \delta, S, E)$ ist dann bestimmt durch:

$$\begin{aligned} Z &= \{0, 1, 2, 4, 5, 6, 7\} \\ \Sigma &= E \cup D \cup \{i_b, d_b \mid B \in \{X, Y, Z\}\} \\ S &= \{0\} \\ E &= \{1\} \end{aligned}$$

sowie der durch folgende Tabelle gegebenen Zustandsübergangsfunktion δ

δ	0	1	2	3	4	5	6	7
E_x	{0,2,3}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
E_y	{0,1,4,5}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
E_z	{0,6,7}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
D_x	{0}	$\emptyset$	$\emptyset$	$\emptyset$	{0}	$\emptyset$	{0}	$\emptyset$
D_y	{0}	$\emptyset$	{0}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	{0}
D_z	{0}	$\emptyset$	$\emptyset$	{0}	$\emptyset$	{0}	$\emptyset$	$\emptyset$
i_x	{0}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
i_y	{0}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
i_z	{0}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
d_x	{0}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
d_y	{0}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
d_z	{0}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Bemerkung: Offensichtlich gilt für die vom oben eingeführten NFA $A = (Z, \Sigma, \delta, S, E)$ für eine Name-Stamp Protokoll $T = \{\alpha_i, \beta_j\}$ mit den Teilnehmern X und Y sowie dem Angreifer Z akzeptierte Sprache:

$$W_A = (V_Z \cup N_i(X, Y))^{\star}$$

wobei V_Z die in der ersten Definition zu sicheren Name-Stamp Protokollen eingeführte Wortmenge ist. D.h. wir können nun vollkommen analog zu dieser Definition eine weitere auf Basis des eingeführten NFA finden.

Definition 25 (Sicheres Name-Stamp Protokoll (2))

Sei $T = \{\alpha_i, \beta_j\}$ ein Name-Stamp Protokoll und $A = (Z, \Sigma, \delta, S, E)$ der zugehörige NFA. T ist sicher gdw

$$\neg\exists_{\gamma \in W_A} : \overline{\gamma} = \lambda$$

5.1 Algorithmus

Um einen Algorithmus anzugeben, der das Problem eines sicheren Name-Stamp Protokolls $T = \{\alpha_i, \beta_j\}$ auf Grundlage des eingeführten NFA $A = (Z, \Sigma, \delta, S, E)$ löst konstruieren wir nun eine Menge von Zustandstupeln wie folgt:

$$C = \{(z_i, z_j) \in Z \times Z \mid \overline{x} = \lambda \ \wedge \ \hat{\delta}(\{z_i\}, x) \cap \{z_j\} \neq \emptyset\}$$

d.h. die Menge aller Zustandstupel für die ein Wort bestehend aus Buchstaben entlang des Pfades vom ersten zum zweiten Zustand des Tupels das Protokoll zerstört. Nach obiger Definition läßt sich die Frage nach der Sicherheit eines Name-Stamp Protokolls jetzt wie folgt formulieren.

Definition 26 (Sicheres Name-Stamp Protokoll (3))

Sei $T = \{\alpha_i, \beta_j\}$ ein Name-Stamp Protokoll, $A = (Z, \Sigma, \delta, S, E)$ der zugehörige NFA und C die wie oben hierzu konstruierte Menge. T ist sicher gdw

$$(S \times E) \cap C = \emptyset$$

Der folgende Algorithmus konstruiert die Menge C und bestimmt somit in $\mathcal{O}(n^3)$, ob ein gegebenes Name-Stamp Protokoll sicher ist. Die Eingabelänge n ist hierbei für ein gegebenes Name-Stamp Protokoll wie oben angegeben definiert.

CheckSecurity2($T = \{\alpha_i, \beta_j\}$,$A = (Z, \Sigma, \delta, S, E)$)
    $C := \{(z_i, z_i) \mid z_i \in Z\}$
    $Q := C$
    while $Q \neq \emptyset$ **do**
      (z_i, z_j) :=head(Q)
      if $(z_j, z_k) \in C$ **and** $(z_i, z_k) \notin C$ **then**
        $C := C \cup \{z_i, z_k\}$
        $Q := Q \cup \{z_i, z_k\}$
      end
      if $(z_k, z_i) \in C$ **and** $(z_k, z_j) \notin C$ **then**
        $C := C \cup \{z_k, z_j\}$
        $Q := Q \cup \{z_k, z_j\}$
      end
      if $\sigma\tau = \lambda$ **and** $z_i \in \delta(z_k, \sigma)$ **and** $z_l \in \delta(z_j, \tau)$ **and** $(z_k, z_l) \notin C$ **then**
        $C := C \cup \{z_k, z_l\}$
        $Q := Q \cup \{z_k, z_l\}$
      end
    end
    if $(S \times E) \cap C = \emptyset$ **return true** else **return false**
  end

Der Algorithmus terminiert, da die Menge der Zustandstupel endlich ist und jeder Zustandstupel nur einmal der Queue Q hinzugefügt wird. Beweise zu Korrektheit und Komplexität des Algorithmus sind in [2] zu finden.

Literatur

[1] Danny Dolev und Andrew C. Yao. *On the Security of Public Key Protocols.* IEEE Transactions on Information Theory, 29(2):198-208, March 1983.

[2] D.Dolev, S.Even und R.Karp. *On the Security of Ping-Pong Protocols.* Information and Control, S. 57-68, 1982.

[3] Peter H. Starke. *Logische Grundlagen der Informatik.* URL: http://www.informatik.hu-berlin.de/lehrstuehle/automaten /logik/skript.pdf.

[4] Johannes Köbler. *Theoretische Informatik II, Vorlesungsskript* URL: http://www.informatik.hu-berlin.de/Institut/struktur/algorithmenII /Lehre/WS2000-2001/Theo_Inf_2/skript.ps