Verifikation von Ping-Pong Protokollen in Zeit $\mathcal{O}(n^2)$

Heiko Stamer
Universität Kassel, Fachbereich Mathematik/Informatik
Heinrich-Plett-Straße 40, D-34132 Kassel
http://www.theory.informatik.uni-kassel.de/~stamer
stamer@theory.informatik.uni-kassel.de

Abstract: Wir betrachten eine von Dolev, Even und Karp eingeführte Technik zur formalen Analyse von Ping-Pong Protokollen. Eine naheliegende Beobachtung zeigt, daß die Kürzungsregeln immer durch eine eindeutige kontextfreie Grammatik beschreibbar sind. Damit kann die asymptotische Laufzeit des ursprünglichen Verifikationsalgorithmus um einen linearen Faktor verbessert werden.

1 Einleitung

Der Entwurf kryptographischer Protokolle ist ausgesprochen kompliziert und deshalb oft fehlerbehaftet. Infolgedessen besteht ein großes praktisches Interesse an Techniken, welche die Sicherheit solcher Protokolle weitgehend automatisch verifizieren können.

Die erste Versuch einer formalen Analyse wurde 1982 von Dolev und Yao in ihrer wegweisenden Arbeit „On the Security of Public Key Protocols“ [DY83] unternommen. Darin wird ein Modell für einfache Zwei-Parteien Protokolle vorgestellt, welches idealisiert die perfekte Sicherheit der kryptographischen Primitive verlangt. Darüberhinaus gibt es aber nur wenige Einschränkungen hinsichtlich des Verhaltens der Angreifer. Weiterhin formalisieren die Autoren auch den Aufbau und die Verarbeitung der ausgetauschten Nachrichten mittels algebraischer Terme. Dabei beschreiben Operatoren, die auf empfangene oder initiale Nachrichten angewendet werden können, alle zugelassenen Teilnehmer- bzw. Angreiferaktionen (Ver- und Entschlüsselung, Hinzufügen, Überprüfen oder Löschen von Namenskennzeichen). Durch Kürzungsregeln werden Beziehungen zwischen den Operatoren ausgedrückt, z. B. ergibt die Verschlüsselung gefolgt von der Entschlüsselung die Identitätsfunktion. Dolev und Yao charakterisieren schließlich die Sicherheit für eine bestimmte Teilklasse vollständig und geben einen Verifikationsalgorithmus an. Kurz darauf verbesserten Dolev, Even und Karp [DEK82] dessen Laufzeit. Ihr Vorschlag konstruiert zuerst einen endlichen Automaten aus dem gegebenen Protokoll und berechnet dann in kubischer Zeit dessen *Collapsing Relation* bezüglich der Kürzungsregeln. Zusätzlich erwähnen die Autoren auch eine Verallgemeinerung dieses Ansatzes: Prinzipiell gilt es zur Beantwortung der Sicherheitsfrage zu entscheiden, ob der Schnitt einer speziellen kontextfreien Sprache mit der regulären Sprache der möglichen Angriffe leer ist oder nicht. Später beschreiben Book und Otto [BO85, BO93] eine etwas abgewandelte Technik. Bei ihrer Lösung wird die Sicherheitsfrage auf das *Common Descendant Problem* für mona-

dische Wortersetzungssysteme zurückgeführt. Die Laufzeitabschätzung zeigt jedoch auch hier mindestens kubisches Verhalten in der Größe des zu verifizierenden Protokolls.

Betrachtet man die erwähnten Kürzungsregeln etwas genauer, dann stellt sich heraus, daß fast ausschließlich Klammerstrukturen (Dyck-Sprachen) vorkommen. Deshalb kann die zur Verifikation benötigte kontextfreie Sprache in Form einer *eindeutigen* Grammatik angegeben werden. Diese Tatsache lässt sich ausnutzen, um den generischen Ansatz von Dolev, Even und Karp um einen linearen Faktor zu beschleunigen. Weiterhin wird deutlich, daß die erforderliche Sprache immer durch einen deterministischen Kellerautomaten erkennbar ist. Damit existiert aber auch immer eine erzeugende eindeutige Grammatik. Folglich gilt die Verbesserung allgemein für beliebige Mengen von Kürzungsregeln.

2 Ping-Pong Protokolle

Wir betrachten Ping-Pong Protokolle zwischen zwei Parteien im sogenannten Dolev-Yao Angriffsmodell [DY83] und in der Notation von Dolev, Even und Karp [DEK82].

Die Protokolle haben eine sehr einfache Struktur: Sei S der Sender und R der Empfänger einer geheimzuhaltenden Nachricht m. Zuerst wendet S eine endliche Anzahl ihm zur Verfügung stehender Funktionen iterativ auf m an und sendet anschließend das Ergebnis an R. Der Empfänger führt nun seinerseits eine Folge von Operationen auf dem erhaltenen Ergebnis aus und sendet das Resultat zurück an S. Je nach Protokollspezifikation wiederholt sich dieser „Ping-Pong“-Prozess endlich oft, wobei in den folgenden Schritten auch eine andere Kombination von Operatoren verwendet werden darf.

Sei $\mathcal{P}$ eine endliche Menge von Benutzernamen und seien X, Y Variablen über $\mathcal{P}$. Jeder Teilnehmer $X \in \mathcal{P}$ besitzt eine öffentliche Verschlüsselungsfunktion $E_X : \{0,1\}^* \to \{0,1\}^*$ sowie eine private Entschlüsselungsfunktion $D_X : \{0,1\}^* \to \{0,1\}^*$, wobei deren Komposition $E_X D_X$ bzw. $D_X E_X$ sinnvollerweise die Identität λ ergibt. Die öffentlichen Funktionen sind für alle Teilnehmer jederzeit verfügbar, z. B. durch Bereitstellung in einem vertrauenswürdigen Verzeichnis. Weiterhin nehmen wir an, daß das zugrundeliegende Public-Key Kryptosystem perfekt sicher ist, d. h. aus der Kenntnis von $E_X(m)$ kann keinerlei Wissen über $m \in \{0,1\}^*$ abgeleitet werden.

Weiterhin existieren noch folgende öffentliche Operatoren für alle Teilnehmer $X \in \mathcal{P}$:

- i_X hängt das eindeutige Namenskennzeichen von X an.
- d_X prüft, ob das Namenskennzeichen von X angehängt ist und entfernt es gegebenenfalls. Das Protokoll wird abgebrochen, wenn ein anderes Kennzeichen als X auftritt.
- d löscht ein beliebiges angehängtes Kennzeichen.

Sollten d_X oder d keine angehängten Namenskennzeichen vorfinden, wird das Protokoll ebenfalls abgebrochen. Außerdem gelten $d_X i_X = \lambda$ und $d i_X = \lambda$ für alle $X \in \mathcal{P}$.

Betrachten wir zunächst zwei Beispiele aus [DEK82]. Beide haben die Aufgabe, die geheime Nachricht m zu übertragen und gleichzeitig eine Emfangsbestätigung zurückzusenden.

Beispiel 1 (unsicher)

(1) S sendet $E_R(m)$.

(2) R antwortet $E_S(D_R(E_R(m)))$.

Beispiel 2 (sicher)

(1) S sendet $E_R(i_S(m))$.

(2) R antwortet $E_S(d_S(D_R(E_R(i_S(m)))))$.

Beispiel 1 ist unsicher, weil ein Angreifer Z die erste Protokollnachricht $E_R(m)$ abfangen und in einer neuen Sitzung selbst an R senden könnte. Als Antwort würde R dann $E_Z(D_R(E_R(m)))$ erzeugen, womit Z letztlich Kenntnis von m erlangt. Dieser Angriff wird im Beispiel 2 durch das Hinzufügen des Namenskennzeichens von S verhindert.

Wir wiederholen jetzt die formale Definition der Ping-Pong Protokolle [DEK82]. Sei Σ eine endliche Menge von Operatoren. Die Untermenge Σ_X (*Vokabular von* X) enthält diejenigen Operatoren, welche dem Teilnehmer X zur Verfügung stehen. *Kürzungsregeln* beschreiben Beziehungen zwischen Operatoren. Sie sind für alle Teilnehmer gleich und haben die Form $\sigma\tau = \lambda$, wobei σ und τ hier beliebige Elemente aus Σ sind. Man kann die Kürzungsregeln als spezielles Wortersetzungssystem [BO93] über dem Operatoralphabet Σ auffassen, wobei λ das leere Wort darstellt. Für ein gegebenes *Operatorwort* $\alpha \in \Sigma^*$ kann durch wiederholtes „Kürzen“ eine *reduzierte Form* $\overline{\alpha}$ berechnet werden, die dann selbst nicht mehr verkürzbar ist. Das zugrundeliegende Reduktionssystem hat immer die sogenannte Church-Rosser Eigenschaft [BO85], weshalb jede beliebige Reihenfolge von Anwendungen der Regeln letztendlich immer zu derselben Normalform $\overline{\alpha}$ führt.

Bei der Sicherheitsanalyse wird später vorausgesetzt, daß keine weiteren algebraischen Beziehungen zwischen den Operatoren bestehen. Es werden lediglich solche Abhängigkeiten beachtet, welche direkt oder implizit durch die Kürzungsregeln beschrieben sind.

Für das gerade eingeführte Szenario ergeben sich also die Operatormengen

$$\Sigma = \{d\} \cup \{E_Y, D_Y, i_Y, d_Y \mid Y \in \mathcal{P}\},$$
$$\Sigma_X = \{d, D_X\} \cup \{E_Y, i_Y, d_Y \mid Y \in \mathcal{P}\}$$

und die Kürzungsregeln

$$E_X D_X = \lambda, \qquad D_X E_X = \lambda, \qquad d_X i_X = \lambda, \qquad d i_X = \lambda$$

für alle Teilnehmer $X \in \mathcal{P}$.

Definition 1 *Ein* Ping-Pong Protokoll $P(S, R)$ *ist eine Folge* $\alpha_1, \alpha_2, \ldots, \alpha_\ell$ *von Operatorwörtern, wobei* $\alpha_i \in \Sigma_S^*$ *wenn* i *ungerade und andernfalls* $\alpha_i \in \Sigma_R^*$.

Das Protokoll aus Beispiel 1 können wir also durch die Operatorwörter $\alpha_1 = E_R$ und $\alpha_2 = E_S D_R$ beschreiben. Analog erhält man für das zweite Beispiel die Wörter $\alpha_1 = E_R i_S$ und $\alpha_2 = E_S d_S D_R$. In beiden Protokollen wird also zuerst α_1 auf m und später α_2 auf $\alpha_1(m)$ angewendet. Die reduzierte Form von $\alpha_2\alpha_1$ ist $\overline{\alpha_2\alpha_1} = E_S$. Mit $\alpha_i[X, Y]$ bezeichnen wir das Wort, welches entsteht, wenn wir die Operatoren des Senders durch die des Teilnehmers X und die Operatoren des Empfängers durch die von Y ersetzen.

Das bei der Sicherheitsanalyse untersuchte Schutzziel ist die Vertraulichkeit der initialen Nachricht, d. h. wir wollen für ein gegebenes Ping-Pong Protokoll entscheiden, ob ein Angreifer[1] Z unter den Voraussetzungen des Dolev-Yao Modells [DY83] in der Lage ist, die Nachricht m zu erfahren. Dabei nehmen wir natürlich an, daß Z weder der Sender noch der Empfänger in der fraglichen Protokollinstanz $P(S,R)$ ist. Ferner gehen wir davon aus, daß der Angreifer die Werte $\alpha_i(\tilde{m})$ für beliebige Nachrichten $\tilde{m} \in \{0,1\}^*$ und für alle $1 \leq i \leq \ell$ kennt. Beispielsweise könnte er durch geschickte Beeinflussung (Details sind in [DY83, DEK82] beschrieben) zwei beliebige Teilnehmer veranlaßt haben, eine entsprechende Sitzung des Protokolls zu initiieren.

Folglich wird die Menge der vom Angreifer Z kontrollierten Operatorwörter aus Σ^* durch die reguläre Sprache

$$\Delta = (\Sigma_Z \cup \{\alpha_i[X,Y] \mid 1 \leq i \leq \ell, X \in \mathcal{P}, Y \in \mathcal{P}, X \neq Y\})^*$$

beschrieben. Mit diesen Voraussetzungen können wir nun die Sicherheitseigenschaft eines Ping-Pong Protokolls formulieren.

Definition 2 *Sei $\alpha_1[S,R]$ das erste Operatorwort von $P(S,R)$ und sei $Z \notin \{S,R\}$. Das Ping-Pong Protokoll P ist* unsicher, *falls ein $\gamma \in \Delta$ existiert, so daß $\overline{\gamma\alpha_1} = \lambda$ gilt.*

Es ist nicht notwendig, alle übertragenen Nachrichten $\alpha_i\alpha_{i-1}\cdots\alpha_1(m)$ für $1 < i \leq \ell$ zu betrachten, denn für jedes $\gamma \in \Delta$ mit $\overline{\gamma\alpha_i\alpha_{i-1}\cdots\alpha_1} = \lambda$ existiert ein $\gamma' \in \Delta$, so daß $\overline{\gamma'\alpha_1} = \lambda$ gilt. Außerdem kann man sich bei der Analyse auf zwei ehrliche Teilnehmer, den Sender S sowie den Empfänger R, und genau einen Angreifer Z beschränken [DEK82].

Deshalb redefinieren wir die reguläre Sprache der Angriffswörter nunmehr als

$$\Delta = (\Sigma_Z \cup \{\alpha_i[X,Y] \mid 1 \leq i \leq \ell, X \in \{S,R,Z\}, Y \in \{S,R,Z\}, X \neq Y\})^*$$

und das Operatoralphabet als $\Sigma = \Sigma_S \cup \Sigma_R \cup \Sigma_Z$.

3 Verifikation von Ping-Pong Protokollen in Zeit $\mathcal{O}(n^2)$

Betrachten wir kurz die von Dolev, Even und Karp [DEK82] vorgestellte Lösung. Der Algorithmus konstruiert zuerst einen nicht-deterministischen endlichen Automaten $\mathcal{A} = (Q, \Sigma, \delta, q_0, \{q_f\})$ für die reguläre Sprache

$$L_Z = \{\gamma\alpha_1[S,R] \mid \gamma \in \Delta\}.$$

Anschließend wird dessen *Collapsing Relation* $C \subseteq Q \times Q$ berechnet. Ein Paar (q,q') ist genau dann in C, wenn es für $u,v \in \Sigma^*$ eine Berechnung $quv \vdash_{\mathcal{A}}^* q'v$ gibt, so daß u mit den bekannten Kürzungsregeln zum leeren Wort λ reduziert werden kann. Das zu untersuchende Ping-Pong Protokoll ist also genau dann unsicher, wenn das Paar (q_0, q_f)

[1] Für Zwei-Parteien Ping-Pong Protokolle reicht es aus, nur genau einen Angreifer zu betrachten. [DEK82]

in C liegt. Die Laufzeit für die Berechnung von C wird von Dolev, Even und Karp mit $\mathcal{O}(s^3)$ abgeschätzt, wobei $s = |Q|$ die Anzahl der Zustände des Automaten $\mathcal{A}$ ist.

Sei $L = \{w \in \Sigma^* \mid \overline{w} = \lambda\}$ die kontextfreie Sprache, welche gerade diejenigen Operatorwörter beinhaltet, die mit den gegebenen Kürzungsregeln zum leeren Wort reduzierbar sind. Der generische Ansatz von Dolev, Even und Karp [DEK82] beantwortet die Sicherheitsfrage für Ping-Pong Protokolle allgemein als folgendes Entscheidungsproblem:

Entscheidungsproblem 1

Instanz: *Die kontextfreie Sprache L und die reguläre Sprache L_Z.*

Fragestellung: *Gilt $L \cap L_Z \neq \emptyset$?*

Prinzipiell führt auch dieser Ansatz zu kubischen Laufzeitverhalten, weil die Schnittmengenkonstruktion entsprechend aufwendig ist. Die folgende Beobachtung hilft uns jedoch, die Laufzeit für spezielle Kürzungsregeln um einen linearen Faktor zu beschleunigen.

Beobachtung 1 *Die Kürzungsregeln der klassischen Ping-Pong Protokolle können durch eine* eindeutige *kontextfreie Grammatik beschrieben werden.*

Die Grammatik $\mathcal{G} = (\{A, B\}, \Sigma, A, P)$ mit den Produktionen

$$P = \left\{ \begin{array}{l} A \to AB \mid \lambda, \\ B \to D_X A E_X \mid E_X A D_X \mid d_X A i_X \mid d A i_X \quad \text{für alle } X \in \{S, R, Z\} \end{array} \right\}$$

erzeugt die gewünschte kontextfreie Sprache L. Offensichtlich ist $\mathcal{G}$ auch eindeutig, weil für jedes Wort $w \in L(\mathcal{G})$ nur genau eine Links-Ableitung existiert.

Beobachtung 2 *Es gibt Algorithmen, z. B. [Ear70, ERS00], die für eindeutige kontextfreie Sprachen L das Entscheidungsproblem 1 in quadratischer Zeit lösen.*

Zur effizienten Beantwortung der Sicherheitsfrage geht man nun folgendermaßen vor.

Algorithmus 1 (Verifikation von Ping-Pong Protokollen)

Eingabe: *Zwei-Parteien Ping-Pong Protokoll $P(S, R)$*

Ausgabe: *Antwort auf die Sicherheitsfrage bezüglich $P(S, R)$*

1. *Konstruktion und Vereinfachung eines nicht-deterministischen endlichen Automaten $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ für die reguläre Sprache L_Z (analog zu [DEK82]):*
   (a) *Der Startzustand ist q_0 und der einzige Endzustand ist q_1, d.h. $F = \{q_1\}$.*
   (b) *Für das Suffix $\alpha_1[S, R]$ wird der Pfad $(q_0, \alpha_1[S, R], q_1)$ in $\mathcal{A}$ eingefügt.*
   (c) *Für jeden Operator $\sigma \in \Sigma_Z$ wird eine Schleife (q_0, σ, q_0) in $\mathcal{A}$ eingefügt.*

   (d) *Für $1 \leq i \leq \ell$ und $X, Y \in \{S, R, Z\}$ mit $X \neq Y$ erzeugt jedes Operatorwort der Protokollbeschreibung einen Pfad $(q_0, \alpha_i[X, Y], q_0)$ in $\mathcal{A}$.*

2. *Konstruktion der eindeutigen Grammatik $\mathcal{G}$ für die kontextfreie Sprache L und deren Überführung in die erweiterte Chomsky-Normalform $\mathcal{G}'$ (d. h. Einheits- und λ-Produktionen sind erlaubt). Diese Transformation erhält die Eindeutigkeit.*

3. *Berechnung des endlichen Automaten $\mathcal{A}' = (Q, \Sigma', \delta', q_0, F)$ für $pre^*(L_Z)$ mit dem Algorithmus von Esparza, Rossmanith und Schwoon [ERS00, BEF+00].*
   Dabei ist $pre^*(L_Z)$ die Menge der Vorgänger aller Wörter aus L_Z bezüglich der von $\mathcal{G}'$ induzierten Ableitungsrelation. Für kontextfreie Grammatiken $\mathcal{G}'$ und reguläre Sprachen L_Z ist $pre^*(L_Z)$ stets regulär [BO93].

4. *Überprüfen, ob das Startsymbol A der Grammatik $\mathcal{G}'$ durch den Automaten $\mathcal{A}'$ akzeptiert wird. Das Protokoll ist genau dann unsicher, wenn $A \in pre^*(L_Z)$.*

Analyse der Laufzeit: Der erste Schritt konstruiert den nicht-deterministischen endlichen Automaten $\mathcal{A}$ in linearer Zeit. Die Anzahl seiner Zustände $s = |Q|$ ist linear in der Protokollgröße $n = \sum_{i=1}^{\ell} |\alpha_i|$. Die Anzahl der Produktionen $p = |P|$ der Grammatik $\mathcal{G}$ bleibt konstant, wenn man die Kürzungsregeln fest vorgibt. Die Überführung in die erweiterte Chomsky-Normalform $\mathcal{G}'$ ist deshalb ebenfalls konstant. Für eindeutige Grammatiken hat der Algorithmus von Esparza, Rossmanith und Schwoon eine Laufzeit von $\mathcal{O}(ps^2)$ [ERS00]. Schließlich kann man in linearer Zeit feststellen, ob das Startsymbol von $\mathcal{G}'$ durch den Automaten $\mathcal{A}'$ akzeptiert wird. Die Gesamtlaufzeit ist folglich aus $\mathcal{O}(n^2)$.

Korollar 1 *Für Zwei-Parteien Ping-Pong Protokolle mit klassischen Kürzungsregeln existiert ein Algorithmus, welcher die Sicherheitsfrage in quadratischer Zeit beantwortet.*

Interessanterweise gilt dieses Resultat auch für beliebige Kürzungsregeln. Im Allgemeinen ist es zwar unentscheidbar, ob für eine gegebene kontextfreie Grammatik eine äquivalente eindeutige Grammatik existiert, da aber das zu einem Ping-Pong Protokoll korrespondierende Reduktionssystem stets monadisch ist und außerdem die Church-Rosser Eigenschaft [BO85] besitzt, kann man immer einen *deterministischen* Kellerautomaten für die entsprechende Sprache L angeben. Ferner hat jede deterministische kontextfreie Sprache eine eindeutige $LR(1)$-Grammatik [HU79, AU72].

Satz 1 *Für Zwei-Parteien Ping-Pong Protokolle mit beliebigen Kürzungsregeln existiert ein Algorithmus, welcher die Sicherheitsfrage in quadratischer Zeit beantwortet.*

4 Anwendung

In diesem Abschnitt wollen wir den verbesserten Verifikationsalgorithmus am Beispiel 2 (vgl. [DEK82]) demonstrieren. Das entsprechende Ping-Pong Protokoll war durch die Operatorwörter $\alpha_1[S, R] = E_R i_S$ und $\alpha_2[S, R] = E_S d_S D_R$ gegeben.

Der erste Schritt konstruiert gemäß dem Verfahren von Dolev, Even und Karp den endlichen Automaten $\mathcal{A} = (Q, \Sigma, \delta, q_0, \{q_1\})$ (siehe Abbildung 1 im Anhang A) in vereinfachter Form, d. h. alle Pfade für Operatorwörter $\alpha_i[X, Y]$, die nur Symbole aus Σ_Z enthalten, werden entfernt, da diese Fälle bereits durch die Schleife (q_0, σ, q_0) behandelt sind.

Im zweiten Schritt wird die eindeutige Grammatik $\mathcal{G}$ in erweiterte Chomsky-Normalform überführt. Es entsteht die Grammatik $\mathcal{G}' = (\{A, B_1, \ldots, B_{14}, C_{D_S}, \ldots\}, \Sigma, A, P')$, wobei die veränderten Produktionen

$$P' = \left\{ \begin{array}{l} A \to AB_1 \mid AB_2 \mid AB_3 \mid AB_4 \mid \lambda, \\ B_1 \to C_{D_S}B_6 \mid C_{D_R}B_7 \mid C_{D_Z}B_8, B_2 \to C_{E_S}B_9 \mid C_{E_R}B_{10} \mid C_{E_Z}B_{11}, \\ B_3 \to C_{d_S}B_{12} \mid C_{d_R}B_{13} \mid C_{d_Z}B_{14}, B_4 \to C_dB_{12} \mid C_dB_{13} \mid C_dB_{14}, \\ B_6 \to AC_{E_S}, B_7 \to AC_{E_R}, B_8 \to AC_{E_Z}, \\ B_9 \to AC_{D_S}, B_{10} \to AC_{D_R}, B_{11} \to AC_{D_Z}, \\ B_{12} \to AC_{i_S}, B_{13} \to AC_{i_R}, B_{14} \to AC_{i_Z}, \\ C_{D_S} \to D_S, C_{D_R} \to D_R, C_{D_Z} \to D_Z, C_{E_S} \to E_S, \ldots, C_{i_Z} \to i_Z \end{array} \right\}$$

die Eindeutigkeit erhalten. Für die von $\mathcal{G}'$ erzeugte kontextfreie Sprache gilt $L(\mathcal{G}') = L$.

Jetzt wird mit Hilfe des Algorithmus von Esparza, Rossmanith und Schwoon ein endlicher Automat $\mathcal{A}' = (Q, \Sigma', \delta', q_0, \{q_1\})$ für die reguläre Sprache $pre^*(L_Z)$ konstruiert. Der Algorithmus bekommt als Eingabe die Überführungsrelation von $\mathcal{A}$ und die Produktionen von $\mathcal{G}'$ und berechnet in quadratischer Zeit den gesuchten Automaten (siehe Abbildung 2).

Schließlich kann man leicht feststellen, daß $\mathcal{A}'$ nicht das Startsymbol von $\mathcal{G}'$ akzeptiert, d. h. $A \notin pre^*(L_Z)$. Also gilt offensichtlich $L \cap L_Z = \emptyset$, und das gegebene Ping-Pong Protokoll ist im Sinne der Definition 2 sicher.

Natürlich ist hiermit lediglich die Sicherheit unter den Annahmen des Dolev-Yao Angriffsmodells gezeigt. Insbesondere kann der Algorithmus nur diejenigen algebraischen Beziehungen beachten, welche durch vorgegebene Kürzungsregeln impliziert worden sind.

5 Zusammenfassung

Ausgangspunkt für unsere Betrachtungen war der generische Ansatz von Dolev, Even und Karp, welcher die Sicherheitsfrage für Zwei-Parteien Ping-Pong Protokolle in kubischer Zeit beantwortet. Eine offensichtliche Beobachtung hat jedoch gezeigt, daß sich die klassischen Kürzungsregeln auch als eindeutige kontextfreie Grammatik beschreiben lassen. Deshalb kann beispielsweise der Algorithmus von Esparza, Rossmanith und Schwoon verwendet werden, um das Sicherheitsproblem in quadratischer Zeit zu lösen. Ferner wurde deutlich, daß dieser Ansatz auch für beliebige Kürzungsregeln funktioniert.

Danksagung: Der Autor dankt Prof. Dr. Friedrich Otto und den Gutachtern der Tagung SICHERHEIT 2006 für die wertvollen und konstruktiven Hinweise.

Literatur

[AU72] Alfred V. Aho und Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling*. Prentice-Hall, Englewood Cliffs, N.J., 1972.

[BEF+00] Ahmed Bouajjani, Javier Esparza, Alain Finkel, Oded Maler, Peter Rossmanith, Bernard Willems und Pierre Wolper. An efficient automata approach to some problems on context-free grammars. *Information Processing Letters*, 74(5-6):221–227, 2000.

[BO85] Ronald V. Book und Friedrich Otto. Cancellation rules and extended word problems. *Information Processing Letters*, 20(1):5–11, 1985.

[BO93] Ronald V. Book und Friedrich Otto. *String-Rewriting Systems*. Springer Texts and Monographs in Computer Science. Springer-Verlag, London, UK, 1993.

[DEK82] Danny Dolev, Shimon Even und Richard M. Karp. On the Security of Ping-Pong Protocols. *Information and Control*, 55(1–3):57–68, 1982.

[DY83] Danny Dolev und Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[Ear70] Jay Clark Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[ERS00] Javier Esparza, Peter Rossmanith und Stefan Schwoon. A Uniform Framework for Problems on Context-Free Grammars. *Bulletin of the EATCS*, 72:169–177, 2000.

[Har79] Michael A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, Reading, M.A., 1979.

[HU79] John E. Hopcroft und Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, M.A., 1979.

A Abbildungen und Tabellen

Abbildung 1: Vereinfachter endlicher Automat $\mathcal{A}$ für die reguläre Sprache L_Z

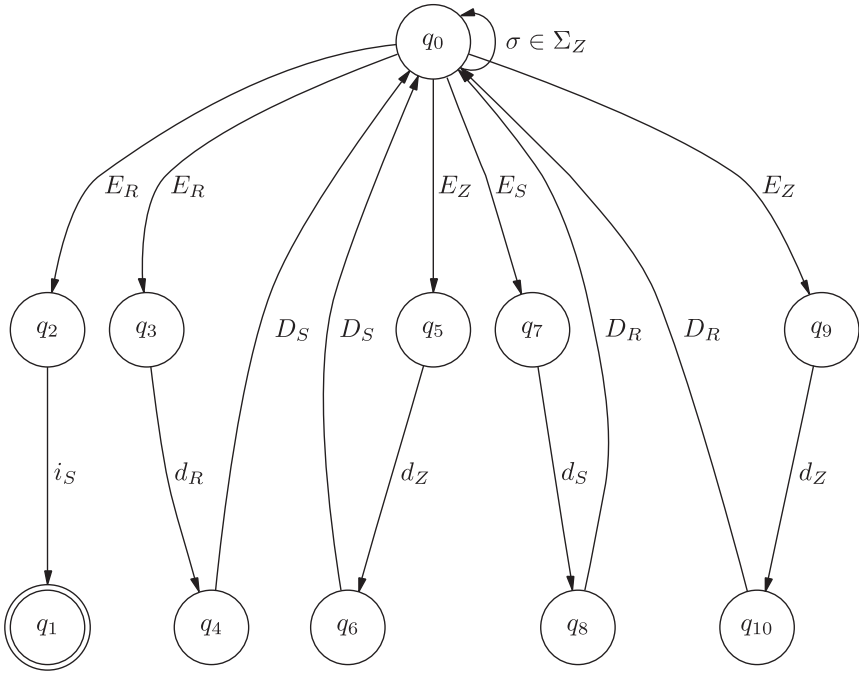
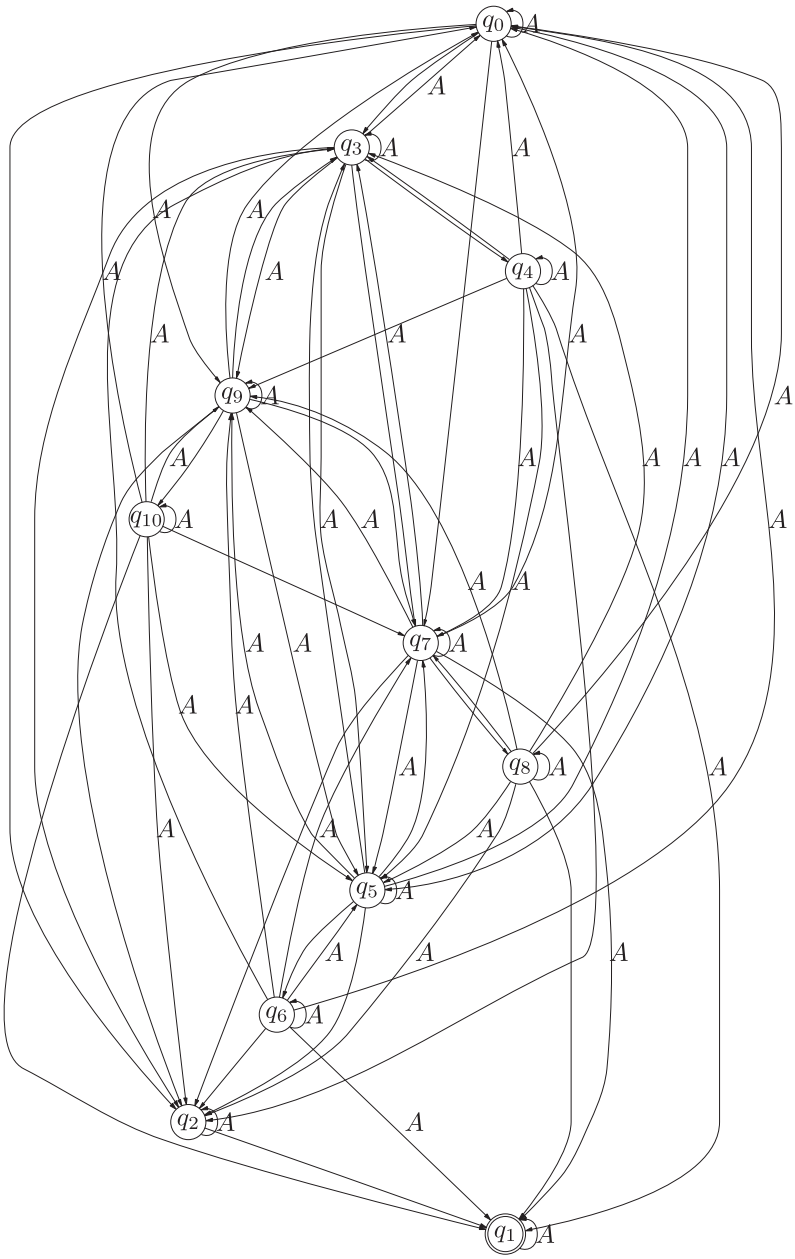


Tabelle 1: Überführungsrelation des Automaten $\mathcal{A}$

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}
E_R	$\{q_0, q_2, q_3\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
E_S	$\{q_0, q_7\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
E_Z	$\{q_0, q_5, q_9\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_{10}\}$	$\emptyset$
D_R	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_0\}$	$\emptyset$	$\{q_0\}$
D_S	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_0\}$	$\emptyset$	$\{q_0\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
D_Z	$\{q_0\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
i_R	$\{q_0\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
i_S	$\{q_0\}$	$\emptyset$	$\{q_1\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
i_Z	$\{q_0\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
d_R	$\{q_0\}$	$\emptyset$	$\emptyset$	$\{q_4\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
d_S	$\{q_0\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_8\}$	$\emptyset$	$\emptyset$	$\emptyset$
d_Z	$\{q_0\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_6\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
d	$\{q_0\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Abbildung 2: Endlicher Automat $\mathcal{A}'$ für die reguläre Sprache $pre^*(L_Z)$



Es wurden nur diejenigen Kanten mit A beschriftet, welche mindestens den Übergang (q, q') mit $q' \in \delta'(q, A)$ enthalten.

Tabelle 2: Überführungsrelation des Automaten $\mathcal{A}'$ (Auszug)

	q_0	q_1	q_2	q_3
A	$\{q_0, q_5, q_9\}$	$\{q_1\}$	$\{q_2\}$	$\{q_0, q_3, q_5, q_9\}$
B_1	$\{q_0, q_5, q_9\}$	$\emptyset$	$\emptyset$	$\emptyset$
B_2	$\{q_0\}$	$\emptyset$	$\emptyset$	$\emptyset$
B_3	$\{q_0\}$	$\emptyset$	$\emptyset$	$\{q_0\}$
B_4	$\{q_0\}$	$\emptyset$	$\emptyset$	$\emptyset$
B_5	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
B_6	$\{q_0, q_7\}$	$\emptyset$	$\emptyset$	$\{q_0, q_7\}$
B_7	$\{q_0, q_2, q_3\}$	$\emptyset$	$\emptyset$	$\{q_0, q_2, q_3\}$
B_8	$\{q_0, q_5, q_9\}$	$\emptyset$	$\emptyset$	$\{q_0, q_5, q_9\}$
B_9	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
B_{10}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
B_{11}	$\{q_0\}$	$\emptyset$	$\emptyset$	$\{q_0\}$
B_{12}	$\{q_0\}$	$\emptyset$	$\{q_1\}$	$\{q_0\}$
B_{13}	$\{q_0\}$	$\emptyset$	$\emptyset$	$\{q_0\}$
B_{14}	$\{q_0\}$	$\emptyset$	$\emptyset$	$\{q_0\}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

	q_4	q_5	q_6	q_7
A	$\{q_0, q_1, q_4, q_5, q_7, q_9\}$	$\{q_0, q_5, q_9\}$	$\{q_0, q_1, q_5, q_6, q_7, q_9\}$	$\{q_0, q_1, q_5, q_7, q_9\}$
B_1	$\{q_0, q_7\}$	$\emptyset$	$\{q_0, q_7\}$	$\emptyset$
B_2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
B_3	$\emptyset$	$\{q_0\}$	$\emptyset$	$\{q_0, q_1\}$
B_4	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
B_5	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
B_6	$\{q_0, q_7\}$	$\{q_0, q_7\}$	$\{q_0, q_7\}$	$\{q_0, q_7\}$
B_7	$\{q_0, q_2, q_3\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_2, q_3\}$
B_8	$\{q_0, q_5, q_9\}$	$\{q_0, q_5, q_9\}$	$\{q_0, q_5, q_9\}$	$\{q_0, q_5, q_9\}$
B_9	$\{q_0\}$	$\emptyset$	$\{q_0\}$	$\emptyset$
B_{10}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
B_{11}	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$
B_{12}	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$
B_{13}	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$
B_{14}	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

	q_8	q_9	q_{10}
A	$\{q_0, q_2, q_3, q_5, q_8, q_9\}$	$\{q_0, q_5, q_9\}$	$\{q_0, q_2, q_3, q_5, q_9, q_{10}\}$
B_1	$\{q_0, q_2, q_3\}$	$\emptyset$	$\{q_0, q_2, q_3\}$
B_2	$\emptyset$	$\emptyset$	$\emptyset$
B_3	$\emptyset$	$\{q_0, \}$	$\emptyset$
B_4	$\emptyset$	$\emptyset$	$\emptyset$
B_5	$\emptyset$	$\emptyset$	$\emptyset$
B_6	$\{q_0, q_7\}$	$\{q_0, q_7\}$	$\{q_0, q_7\}$
B_7	$\{q_0, q_2, q_3\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_2, q_3\}$
B_8	$\{q_0, q_5, q_9\}$	$\{q_0, q_5, q_9\}$	$\{q_0, q_5, q_9\}$
B_9	$\emptyset$	$\emptyset$	$\emptyset$
B_{10}	$\{q_0\}$	$\emptyset$	$\{q_0\}$
B_{11}	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$
B_{12}	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0, q_1\}$
B_{13}	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$
B_{14}	$\{q_0\}$	$\{q_0\}$	$\{q_0\}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$