Sicherheit kryptographischer Protokolle

Dissertation
zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Technischen Fakultät
der Universität des Saarlandes
von

Ralf Handl

Saarbrücken
1993

Inhaltsverzeichnis

Einleitung

In den 70er Jahren erlebte die theoretische Kryptographie einen neuen Aufschwung durch die Entwicklung des Konzeptes der Public-Key-Kryptosysteme. Die entscheidende Neuerung bei diesen Systemen war die Benutzung eines schwächeren als des bisher üblichen informationstheoretischen Sicherheitsbegriffs. Public-Key-Systeme sind zwar informationstheoretisch nicht sicher, ermöglichen aber eine Rückführung der Sicherheit auf die Schwierigkeit zahlentheoretischer Probleme wie etwa die Faktorisierung großer Zahlen. Für die kryptographische Praxis bedeutet dies, daß es keine effizienten Algorithmen zum Knacken eines Public-Key-Systems gibt, solange das entsprechende zahlentheoretische Problem schwierig bleibt.

0.1 Motivation

Die damit nötig gewordenen Komplexitätsbetrachtungen erfordern also Maschinen, die sowohl zufallsabhängig arbeiten als auch miteinander kommunizieren können.

Dieser Aspekt diente als Motivation für die ersten beiden Kapitel dieser Arbeit, in denen ein neues probabilistisches Modell vorgestellt wird, das sich von den bisherigen Ansätzen dadurch unterscheidet, daß weder Zufallsbänder verwendet noch spezielle Münzwurfzustände ausgezeichnet werden. Außerdem werden probabilistische Maschinenschemata entwickelt, mit deren Hilfe man aus einfachen Maschinen beliebige probabilistische Turingmaschinen konstruieren kann. Die Interaktionsfähigkeit der Maschinen wird durch spezielle Bausteine, sogenannte Kommunikationsprimitive, gewährleistet.

Die Möglichkeiten des neuen Sicherheitsverständnisses führten zu einer Vielzahl von kryptographischen Anwendungen wie elektronisches Geld, digitale Unterschriften, gleichzeitiger Austausch von Unterschriften, geheime Abstimmungen über Telefon, Bit-Hinterlegung und interaktive Beweise (eine Übersicht über zugehörige Literatur findet man z.B. in[GOLDR89]). Genauso vielfältig wie die vorgeschlagenen Verfahren sind die dafür entworfenen Sicherheitsbegriffe.

Im dritten Kapitel werden die wichtigsten dieser Sicherheitsbegriffe auf beliebige Protokolle verallgemeinert und ihre weitgehende Äquivalenz gezeigt.

Vorab eine informale Betrachtung des kryptographischen Szenarios und des benutzten Maschinenmodells.

0.2 Szenario

Leute kommunizieren miteinander, um ein gemeinsames Ziel zu erreichen.
Was kann dabei passieren?

- Leitungen können abgehört werden.
- Nachrichten können verändert werden.
- Leute können falsche Identitäten vortäuschen.
- Leute können bösartig sein.
- Jeder außer mir kann bösartig sein!
- Bösartige Leute können sich verschwören.

Was ist der schlimmste anzunehmende Fall?

- Ein böser Außenstehender greift ein und bestimmt, was wie schiefgeht.

0.3 Modell

Ein System ist eine probabilistische Turingmaschine, bestehend aus

- Teilnehmern, die Nachrichten empfangen und senden,
- Leitungen, die Nachrichten übermitteln,
- Angreifern, die Teile des Systems in ihre Gewalt bringen.

Die Leitungen unterteilen sich in zwei Typen: unangreifbare Kanäle zwischen zwei Teilnehmern und angreifbare Leitungen, die als zusätzliche Teilnehmer ohne Eingaben modelliert werden.
Das System hat eine Eingabe für jeden echten Teilnehmer und möglicherweise eine Eingabe für den Angreifer. Es generiert für jeden echten Teilnehmer und für den Angreifer eine Ausgabe.

KAPITEL 1

Maschinenmodell

Dieses erste Kapitel dient der Darstellung unseres Berechnungsmodells. Nach der Definition probabilistischer Turingmaschinen zeigen wir, wie man mit zwei Grundoperationen, nämlich dem Entwurf probabilistischer Maschinenschemata und der Parallelausführung von Maschinen, aus einfachen Bauelementen beliebig komplizierte probabilistische Turingmaschinen konstruieren kann. Zum Abschluß erweitern wir den probabilistischen Ansatz auf nichtuniforme Berechnungen.

1.1 Probabilistische Turingmaschinen

Wir beginnen mit der Definition des Turingmaschinenmodells, das wir in dieser Arbeit benutzen werden. Diese Turingmaschinen haben zweiseitig unendliche Bänder mit möglicherweise mehreren Köpfen pro Band.
Mit $[\![a, b]\!]$ bezeichnen wir die Menge ganzer Zahlen $\{a, \ldots, b\} \subset \mathbb{Z}$.

1.1. Definition *Eine* **probabilistische Turingmaschine** *mit* w *Köpfen auf* t *Bändern ist ein Quintupel* (K, Σ, p, s, τ). *Dabei ist*

- K *die endliche* **Zustandsmenge**, *die den* **Haltezustand** h *nicht enthält,*
- Σ *das endliche* **Bandalphabet**, *das insbesondere das Leerzeichen* $\#$, *nicht aber* L *und* R *enthält,*
- $p : K \times \Sigma^w \times (K \cup \{h\}) \times (\Sigma \cup \{L, R\})^w \to [0, 1]$ *die* **Übergangswahrscheinlichkeit**, *eine Funktion mit*

$$\bigvee_{(q,a)\in K\times\Sigma^w} \sum_{(q',a')\in(K\cup\{h\})\times(\Sigma\cup\{L,R\})^w} p(q, a, q', a') = 1, \qquad (1.1)$$

- $s \in K$ *der* **Startzustand**,
- $\tau : [\![1, w]\!] \to [\![1, t]\!]$ *eine surjektive Abbildung, die* **Kopfzuordnung**, *die beschreibt, welcher Kopf auf welchem Band steht:* $\tau(i) = j$ *heißt, daß sich Kopf* i *auf Band* j *befindet.*

Die Menge $K \times \Sigma^w \times (K \cup \{h\}) \times (\Sigma \cup \{L, R\})^w$ ist die **Menge aller Übergänge** von M.

Eine probabilistische Turingmaschine heißt **Münzwurfmaschine**, wenn die Übergangswahrscheinlichkeit p nur Werte in $\{k'2^{-k} | k' \in [\![0, 2^k]\!]\}$ annimmt für ein $k \in \mathbb{N}$, d.h. die Maschine darf k Münzen gleichzeitig werfen.

Eine **deterministische Turingmaschine** ist eine Münzwurfmaschine, die ihre Münzen nicht wirft, d.h. p nimmt nur die Werte 0 und 1 an. Die in der üblichen Definition deterministischer Turingmaschinen auftauchende Übergangsfunktion δ entspricht in dieser Definition dem Träger der Übergangswahrscheinlichkeit $\mathrm{supp}(p) = \{d \in \mathrm{Dom}(p) \mid p(d) > 0\}$.

Für den Rest dieses Abschnittes sei $M = (K, \Sigma, p, s, \tau)$ eine probabilistische Turingmaschine mit $w = |\mathrm{Dom}(\tau)|$ Köpfen und $t = |\mathrm{Ran}(\tau)|$ Bändern.

Der **Inhalt** eines zweiseitig unendlichen Bandes ist eine Funktion $c : \mathbb{Z} \to \Sigma$. Wir fordern, daß die Bänder mit Leerzeichen initialisiert sind, daher können in Berechnungen nur Bänder mit einer endlichen Anzahl nichtleerer Symbole auftreten. Die Menge dieser Bandinhalte bezeichnen wir mit T_Σ.

Eine **Bandbeschreibung** von M besteht aus den t Bandinhalten und den w **Kopfpositionen**, die **Menge aller Bandbeschreibungen** von M ist

$$D_M = T_\Sigma^t \times \mathbb{Z}^w.$$

Eine **Konfiguration** besteht aus dem aktuellen Zustand von M und der Bandbeschreibung, die **Menge aller Konfigurationen** von M ist

$$\Gamma_M = (K \cup \{h\}) \times D_M,$$

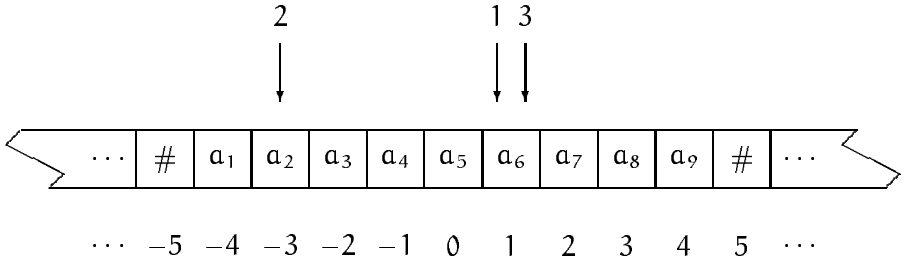
die **Menge aller Startkonfigurationen**

$$S_M = \{s\} \times D_M,$$

die **Menge aller Haltekonfigurationen**

$$H_M = \{h\} \times D_M.$$

1.2. Beispiel Eine Turingmaschine mit drei Köpfen auf einem Band hat in der Situation



den Bandinhalt

$$c(i) = \begin{cases} a_{i+5} & i \in [\![-4,4]\!], \\ \# & \text{sonst} \end{cases}$$

und die Bandbeschreibung

$$(c, 1, -3, 1).$$

◇

Seien $d = (c_1, \ldots, c_t, h_1, \ldots, h_w)$ und $d' = (c'_1, \ldots, c'_t, h'_1, \ldots, h'_w)$ Bandbeschreibungen aus D_M, $a = (a_1, \ldots, a_w) \in \Sigma^w$, $a' = (a'_1, \ldots, a'_w) \in (\Sigma \cup \{L, R\})^w$ und

$$H_{i,j} = \{k \in [\![1, w]\!] \mid \tau(k) = i, h_k = j, a'_k \in \Sigma\}$$

die Menge der Nummern der Köpfe, die auf Band i in Position j sind und ein Symbol schreiben. Dann ist (a, a') **kompatibel** mit (d, d'), falls $c_{\tau(i)}(h_i) = a_i$ für $i \in [\![1, w]\!]$,

$$h'_i = \begin{cases} h_i - 1 & a'_i = L, \\ h_i + 1 & a'_i = R, \\ h_i & \text{sonst, d.h. } a'_i \in \Sigma, \end{cases} \qquad i \in [\![1, w]\!]$$

und

$$c'_i(j) = \begin{cases} a'_{\min H_{i,j}} & H_{i,j} \neq \emptyset, \\ c_i(j) & \text{sonst.} \end{cases} \qquad i \in [\![1, t]\!],\ j \in \mathbb{Z}$$

Dabei bedeutet „min“ in der obigen Bedingung an $c'_i(j)$, daß bei einem Schreibkonflikt der Kopf mit der niedrigsten Nummer gewinnt.

Analog ist ein Übergang (q, a, q', a') kompatibel mit dem Paar von Konfigurationen $(\alpha, \alpha') \in \Gamma_M \times \Gamma_M$, wenn $\alpha = (q, d)$, $\alpha' = (q', d')$ und (a, a') kompatibel ist mit (d, d'). Die Menge aller Paare (a, a') kompatibel mit $(d, d') \in D_M \times D_M$ ist $\mathrm{comp}(d, d')$, die Menge aller mit $(\alpha, \alpha') \in \Gamma_M \times \Gamma_M$ kompatiblen Übergänge $\mathrm{Comp}(\alpha, \alpha')$.

Die Funktion $\wp_M : \Gamma_M \times \Gamma_M \to [0, 1]$ mit

$$\wp_M(\alpha, \alpha') = \sum_{\delta \in \mathrm{Comp}(\alpha, \alpha')} p(\delta)$$

ordnet jedem Paar von Konfigurationen (α, α') die Wahrscheinlichkeit zu, daß die probabilistische Turingmaschine M in genau einem Schritt von Konfiguration α zu Konfiguration α' gelangt. Für $\alpha, \alpha' \in \Gamma_M$ setzen wir

$$\wp_M^{(0)}(\alpha, \alpha') = \begin{cases} 1 & \alpha = \alpha', \\ 0 & \text{sonst,} \end{cases}$$

und für $n \geq 1$

$$\wp_M^{(n)}(\alpha, \alpha') = \sum_{\beta \in \Gamma_M} \wp_M^{(n-1)}(\alpha, \beta) \wp_M(\beta, \alpha').$$

Dann ist $\wp_M^{(n)}(\alpha, \alpha')$ die Wahrscheinlichkeit, in genau n Schritten von α nach α' zu gelangen.

1.3. Satz *Es gilt:*

$$\bigvee_{(\alpha,\alpha')\in\Gamma_M\times H_M} \sum_{n=0}^{\infty} \wp_M^{(n)}(\alpha,\alpha') < \infty \quad \textit{und} \quad \bigvee_{\alpha\in\Gamma_M} \sum_{\alpha'\in H_M} \sum_{n=0}^{\infty} \wp_M^{(n)}(\alpha,\alpha') \le 1.$$

Beweis: Sei $\overline{h} \notin K \cup \{h\}$ ein neuer Zustand. Wir setzen $\overline{\alpha} = (\overline{h}, d)$ für $\alpha = (h, d) \in H_M$, $\overline{H}_M = \{\overline{\alpha} | \alpha \in H_M\}$ und $\overline{\Gamma}_M = \Gamma_M \cup \overline{H}_M$. Wir weiten $\wp_M^{(n)}$ auf $\overline{\Gamma}_M \times \overline{\Gamma}_M$ aus mit

$$\wp_M(\alpha,\alpha') = \begin{cases} 1 & \alpha \in H_M,\ \alpha' = \overline{\alpha}, \\ 1 & \alpha \in \overline{H}_M,\ \alpha' = \alpha, \\ 0 & \text{sonst.} \end{cases}$$

Wir zeigen per Induktion, daß für alle $n \ge 0$ und $\alpha \in \Gamma_M$

$$\sum_{\alpha'\in\overline{\Gamma}_M} \wp_M^{(n)}(\alpha,\alpha') = 1 \tag{1.2}$$

ist, also eine obere Schranke existiert.

Im Induktionsanfang unterscheiden wir mehrere Fälle. Für $n = 0$ folgt die Gleichung aus der Definition von $\wp_M$. Für $n = 1$ und $\alpha \in H_M \cup \overline{H}_M$ gilt sie aus dem gleichen Grund. Für $n = 1$ und $\alpha = (q, c_1, \ldots, c_t, h_1, \ldots, h_w) \in \Gamma_M \setminus H_M$ ist

$$\bigcup_{\alpha'\in\Gamma_M} \mathrm{Comp}(\alpha,\alpha') = \{(q, c_{\tau(1)}(h_1), \ldots, c_{\tau(w)}(h_w))\} \times (K \cup \{h\}) \times (\Sigma \cup \{L, R\})^w.$$

Dann folgt aus Gleichung (1.1), daß

$$\sum_{\alpha'\in\overline{\Gamma}_M} \wp_M(\alpha,\alpha') = \sum_{\alpha'\in\Gamma_M} \wp_M(\alpha,\alpha') = \sum_{\delta\in\bigcup_{\alpha'\in\Gamma_M}\mathrm{Comp}(\alpha,\alpha')} p(\delta) = 1.$$

Für den Induktionsschritt betrachten wir

$$\begin{aligned} \sum_{\alpha'\in\overline{\Gamma}_M} \wp_M^{(n+1)}(\alpha,\alpha') &= \sum_{\alpha'\in\overline{\Gamma}_M} \sum_{\beta\in\overline{\Gamma}_M} \wp_M^{(n)}(\alpha,\beta)\wp_M(\beta,\alpha') \\ &= \sum_{\beta\in\overline{\Gamma}_M} \wp_M^{(n)}(\alpha,\beta) \underbrace{\sum_{\alpha'\in\overline{\Gamma}_M} \wp_M(\beta,\alpha')}_{=1} \\ &= \sum_{\beta\in\overline{\Gamma}_M} \wp_M^{(n)}(\alpha,\beta) = 1. \end{aligned}$$

Da beide Summen endlich sind, darf die Summationsreihenfolge im zweiten Schritt vertauscht werden. Darüberhinaus können wir zeigen, daß für $(\alpha,\alpha') \in \Gamma_M \times H_M$ und $n \ge 0$

$$\wp_M^{(n+1)}(\alpha,\overline{\alpha'}) = \sum_{i=0}^{n} \wp_M^{(i)}(\alpha,\alpha') \tag{1.3}$$

ist und damit $\wp_M^{(n)}(\alpha, \overline{\alpha'})$ monoton wächst in n. Für $\alpha' = \alpha$ ist $\wp_M^{(0)}(\alpha, \alpha') = 1 = \wp_M^{(1)}(\alpha, \overline{\alpha'})$, für $\alpha' \neq \alpha$ erhalten wir $\wp_M^{(0)}(\alpha, \alpha') = 0$ und $\wp_M^{(1)}(\alpha, \overline{\alpha'}) = \wp_M^{(0)}(\alpha, \overline{\alpha'}) = 0$. Dies beweist die Gleichung für $n = 0$. Nun sei Gleichung (1.3) richtig für n. Nach Definition von $\wp_M$ erhalten wir

$$\begin{aligned}\wp_M^{(n+1)}(\alpha, \overline{\alpha'}) &= \sum_{\beta \in \overline{\Gamma}_M} \wp_M^{(n)}(\alpha, \beta)\wp_M(\beta, \overline{\alpha'}) = \wp_M^{(n)}(\alpha, \alpha') + \wp_M^{(n)}(\alpha, \overline{\alpha'}) \\ &= \wp_M^{(n)}(\alpha, \alpha') + \sum_{i=0}^{n-1} \wp_M^{(i)}(\alpha, \alpha') = \sum_{i=0}^{n} \wp_M^{(i)}(\alpha, \alpha').\end{aligned}$$

Aus den Gleichungen (1.2) (Existenz einer oberen Schranke) und (1.3) (monotones Wachstum) folgt, daß für alle $(\alpha, \alpha') \in \Gamma_M \times H_M$

$$\lim_{n \to \infty} \wp_M^{(n)}(\alpha, \overline{\alpha'})$$

existiert.

Mit (1.2) und dem „Cauchyschen Doppelreihensatz“ [Heuser1, S. 258, °45.2] erhalten wir für alle $\alpha \in \Gamma_M$

$$\begin{aligned}\sum_{\alpha' \in H_M} \sum_{n=0}^{\infty} \wp_M^{(n)}(\alpha, \alpha') &= \sum_{\alpha' \in H_M} \lim_{n \to \infty} \wp_M^{(n)}(\alpha, \overline{\alpha'}) \\ &= \lim_{n \to \infty} \sum_{\alpha' \in H_M} \wp_M^{(n)}(\alpha, \overline{\alpha'}) \\ &\leq \lim_{n \to \infty} \sum_{\alpha' \in \overline{\Gamma}_M} \wp_M^{(n)}(\alpha, \alpha') = 1.\end{aligned}$$

□

1.4. Bemerkung Für alle $(\alpha, \alpha') \in \Gamma_M \times H_M$ existiert damit zum einen

$$\wp_M^*(\alpha, \alpha') = \sum_{n=0}^{\infty} \wp_M^{(n)}(\alpha, \alpha'),$$

also die Wahrscheinlichkeit, daß M aus Konfiguration α die Haltekonfiguration α' erreicht, und zum anderen

$$\wp_M^*(\alpha, \infty) = 1 - \sum_{\alpha' \in H_M} \wp_M^*(\alpha, \alpha'),$$

die Wahrscheinlichkeit, daß M aus Konfiguration α keine Haltekonfiguration erreicht. ◇

1.5. Definition *Sei M eine probabilistische Turingmaschine und $\alpha \in S_M$ eine Startkonfiguration von M. Wenn $\wp_M^*(\alpha, \infty) = 0$ ist und die Reihe*

$$\mathrm{er}_M(\alpha) = \sum_{\alpha' \in H_M} \sum_{n=1}^{\infty} n\wp_M^{(n)}(\alpha, \alpha')$$

konvergiert, nennen wir sie die **erwartete Laufzeit** *von* M *bei Startkonfiguration* α. *Wenn* $\wp^*_M(\alpha,\infty)=0$ *ist und die Zahl*

$$\mathrm{mr}_M(\alpha)=\max\left\{n\in\mathbb{N}\;\middle|\;\exists_{\alpha'\in H_M}\;\wp^{(n)}_M(\alpha,\alpha')>0\right\}$$

existiert, nennen wir sie die **maximale Laufzeit** *von* M *bei Startkonfiguration* α.

Existiert die obige Summe nicht, setzen wir $\mathrm{er}_M(\alpha)=\infty$, analog bei Nichtexistenz des Maximums $\mathrm{mr}_M(\alpha)=\infty$.

Schließlich ist eine **Berechnung** der Länge n einer probabilistischen Turingmaschine M eine Folge $\alpha_0,\ldots,\alpha_n$ von Konfigurationen aus Γ_M, wobei $\alpha_0\in S_M$ eine Startkonfiguration ist, $\alpha_n\in H_M$ eine Haltekonfiguration und $\wp_M(\alpha_{i-1},\alpha_i)>0$ für alle $i\in[\![1,n]\!]$.

Die Definition von probabilistischen Turingmaschinen ist damit zunächst abgeschlossen. Wir beschäftigen uns nun mit der Möglichkeit, zwei solche Maschinen zu vergleichen.

1.2 Äquivalenz probabilistischer Turingmaschinen

Der uns interessierende Aspekt einer probabilistischen Turingmaschine ist die Wahrscheinlichkeit, in einer gegebenen Zahl von Schritten aus einer gegebenen Anfangskonfiguration eine gegebene Endkonfiguration zu erreichen, dies bezeichnen wir als ihr *Verhalten*. Eine Umbenennung der Zustandsmenge ändert aber intuitiv nichts am Verhalten der untersuchten Maschine. Um dies zu beweisen, definieren wir zunächst, wann wir zwei probabilistische Turingmaschinen als äquivalent bezeichnen wollen, nämlich dann, wenn sie „dasselbe tun“.

1.6. Definition *Die probabilistischen Turingmaschinen* $M=(K,\Sigma,s,p,\tau)$ *und* $M'=(K',\Sigma,p',s',\tau)$ *sind* **äquivalent** *(*$M\sim M'$*), wenn*

$$\bigwedge_{d,d'\in D_M}\;\bigwedge_{n\in\mathbb{N}}\;\wp^{(n)}_M((s,d),(h,d'))=\wp^{(n)}_{M'}((s',d),(h,d')).$$

Offenbar ist „$\sim$“ reflexiv, transitiv und symmetrisch. Die Wahrscheinlichkeiten, bei gegebener Startbandbeschreibung eine bestimmte Endkonfiguration zu erreichen, sind identisch, damit auch maximale und erwartete Laufzeit von M und M'.

Maschinen sollten äquivalent sein, wenn ihre Zustände nur umbenannt wurden.

1.7. Definition *Die Zustandsmengen der probabilistischen Turingmaschinen* $M = (K, \Sigma, p, s, \tau)$ *und* $M' = (K', \Sigma, p', s', \tau)$ *mit* w *Köpfen sind* **isomorph**, *wenn es eine injektive Abbildung* $i : K \cup \{h\} \to K' \cup \{h\}$ *gibt mit* $i(s) = s'$, $i(h) = h$ *und*

$$\bigwedge_{(q,a,q',a') \in K \times \Sigma^w \times (K \cup \{h\}) \times (\Sigma \cup \{L,R\})^w} p(q, a, q', a') = p'(i(q), a, i(q'), a').$$

1.8. Lemma *Sind die Zustandsmengen der probabilistischen Turingmaschinen* M *und* M' *isomorph, dann ist* $M \sim M'$.

Beweis: Aus der Definition von $\wp_M$ und $\wp_{M'}$ folgt für alle $q, q' \in K \cup \{h\}$ und alle $d, d' \in D_M = D_{M'}$, daß $\wp_M((q,d),(q',d')) = \wp_{M'}((i(q),d),(i(q'),d'))$ ist. Dies überträgt sich auf $\wp^{(n)}$. Mit der Isomorphie der Start- bzw. Haltezustände folgt die Behauptung. □

Wenn nun die Zustandsmenge einer Turingmaschine durch redundante oder unerreichbare Zustände aufgebläht ist, wird das Rechenverhalten ebensowenig beeinflußt wie durch die Umbenennung von Zuständen.

Sei $M = (K, \Sigma, p, s, \tau)$ eine probabilistische Turingmaschine mit w Köpfen. Die Zustände $q, q' \in K$ sind **gleichwertig** ($q \doteq q'$), wenn

$$\bigwedge_{(a,q'',a') \in \Sigma^w \times (K \cup \{h\}) \times (\Sigma \cup \{L,R\})^w} p(q, a, q'', a') = p(q', a, q'', a')$$

ist, da Berechnungen von M aus q und q' auf die gleiche Art weitergehen. Die Relation „$\doteq$“ ist reflexiv, transitiv und symmetrisch. Ein Zustand $q \in K$ ist **überflüssig**, wenn

$$\bigwedge_{(q',a,a') \in K \times \Sigma^w \times (\Sigma \cup \{L,R\})^w} p(q', a, q, a') = 0$$

ist, da er nie erreicht wird.

Erweitern wir nun das Prinzip der Isomporphie von Zustandsmengen.

1.9. Definition *Die probabilistische Turingmaschine* $M = (K, \Sigma, p, s, \tau)$ *mit* w *Köpfen ist eine* **Ausdünnung** *von* $M' = (K', \Sigma, p', s', \tau)$, *wenn es eine injektive Abbildung* $i : K \cup \{h\} \to K' \cup \{h\}$ *gibt mit* $i(s) = s'$, $i(h) = h$, *alle* $q \in K'$ *sind entweder überflüssig oder gleichwertig mit einem* $q' \in i(K)$ *und*

$$\bigwedge_{(q,a,q',a') \in K \times \Sigma^w \times (K \cup \{h\}) \times (\Sigma \cup \{L,R\})^w} p(q, a, q', a') = \sum_{q'' \doteq i(q')} p'(i(q), a, q'', a').$$

1.10. Lemma *Ist* M *eine Ausdünnung von* M', *dann ist* $M \sim M'$.

Beweis: Per Induktion zeigen wir, daß für alle $(q,d),(q',d') \in \Gamma_M$

$$\wp_M^{(n)}((q,d),(q',d')) = \sum_{q'' \doteq i(q')} \wp_{M'}^{(n)}((i(q),d),(q'',d')) \tag{1.4}$$

ist. Daraus folgt dann

$$\wp_M^{(n)}((s,d),(h,d')) = \sum_{q'' \doteq i(h)} \wp_{M'}^{(n)}((i(s),d),(q'',d')) = \wp_{M'}^{(n)}((s',d),(h,d'))$$

und damit die Behauptung.

Es folgt nun der Beweis von Gleichung (1.4).

Der Induktionsanfang ergibt sich aus

$$\begin{aligned}
&\wp_M((q,d),(q',d')) \\
&= \sum_{(a,a') \in \mathbf{comp}(d,d')} p(q,a,q',a') \\
&= \sum_{(a,a') \in \mathbf{comp}(d,d')} \sum_{q'' \doteq i(q')} p'(i(q),a,q'',a') \\
&= \sum_{q'' \doteq i(q')} \sum_{(a,a') \in \mathbf{comp}(d,d')} p'(i(q),a,q'',a') \\
&= \sum_{q'' \doteq i(q')} \wp_{M'}((i(q),d),(q'',d')),
\end{aligned}$$

wobei wir im zweiten Schritt die Definition einer Ausdünnung benutzen, im dritten Schritt, daß beide Summen endlich sind. Der Induktionsschritt folgt aus

$$\begin{aligned}
&\wp_M^{(n)}((q,d),(q',d')) \\
&= \sum_{(q_1,d_1) \in \Gamma_M} \wp_M^{(n-1)}((q,d),(q_1,d_1)) \wp_M((q_1,d_1),(q',d')) \\
&= \sum_{(q_1,d_1) \in \Gamma_M} \wp_M^{(n-1)}((q,d),(q_1,d_1)) \sum_{q'' \doteq i(q')} \wp_{M'}((i(q_1),d_1),(q'',d')) \\
&= \sum_{(q_1,d_1) \in \Gamma_M} \sum_{q_2 \doteq i(q_1)} \wp_{M'}^{(n-1)}((i(q),d),(q_2,d_1)) \sum_{q'' \doteq i(q')} \wp_{M'}((i(q_1),d_1),(q'',d')) \\
&= \sum_{q'' \doteq i(q')} \sum_{d_1 \in D_M} \\
&\qquad \Bigg(\sum_{q_1 \in K} \sum_{q_2 \doteq i(q_1)} \wp_{M'}^{(n-1)}((i(q),d),(q_2,d_1)) \wp_{M'}((i(q_1),d_1),(q'',d'))
\end{aligned}$$

$$
\begin{aligned}
&\qquad + \sum_{q_2 \text{ überflüssig}} \underbrace{\wp_{M'}^{(n-1)}((i(q),d),(q_2,d_1))}_{=0} \wp_{M'}((q_2,d_1),(q'',d'))\Big) \\
&= \sum_{q''\doteq i(q')} \sum_{d_1\in D_{M'}} \sum_{q_2\in K'} \wp_{M'}^{(n-1)}((i(q),d),(q_2,d_1))\wp_{M'}((q_2,d_1),(q'',d')) \\
&= \sum_{q''\doteq i(q')} \wp_{M'}^{(n)}((i(q),d),(q'',d')).
\end{aligned}
$$

Im vorletzten Schritt nutzen wir, daß $q_2 \doteq i(q_1)$ ist und alle Zustände in K entweder überflüssig sind oder äquivalent zu Zuständen aus $i(K')$. Der letzte Schritt verwendet die Gleichheit von D_M und $D_{M'}$. Damit ist Gleichung (1.4) bewiesen. □

Wir haben gezeigt, daß die Äquivalenz von Turingmaschinen von ihrem Verhalten abhängt. In den folgenden beiden Abschnitten wird es nötig sein, die Isomorphie zweier Zustandsmengen zeigen zu können, in Abschnitt 1.5 wird das Prinzip der Ausdünnung Anwendung finden.

1.3 Schemata probabilistischer Turingmaschinen

Die Maschinenschemata nach [LewisPap, Definition 4.3.1] bilden eine elegante Möglichkeit zur Definition komplexer deterministischer Turingmaschinen aus einfacheren. Dieses Konzept verallgemeinern wir auf probabilistische Turingmaschinen.

1.11. Definition *Ein* **probabilistisches Maschinenschema** *ist ein Tripel* $(\mathcal{M},\eta,M_s)$ *aus*

- *einer endlichen Menge* $\mathcal{M}$ *probabilistischer Turingmaschinen mit gemeinsamem Bandalphabet* Σ*, disjunkten Zustandsmengen, identischer Kopfzuordnung* τ *und Kopfanzahl* $w = |\operatorname{Dom}(\tau)|$*,*
- *der* **Übergangsfunktion** $\eta : \mathcal{M}\times\Sigma^w\times\mathcal{M}\to[0,1]$ *mit*

$$\bigwedge_{(M,a)\in\mathcal{M}\times\Sigma^w} \sum_{M'\in\mathcal{M}} \eta(M,a,M') \le 1 \text{ und}$$

- *der* **Startmaschine** $M_s\in\mathcal{M}$.

Wir interpretieren ein probabilistisches Maschinenschema $S = (\mathcal{M},\eta,M_s)$ als eine probabilistische Turingmaschine MS(S) in folgender Weise:
Die Maschine MS(S) startet als M_s. Wenn die aktuelle Maschine $M\in\mathcal{M}$ hält und die Köpfe von M das Tupel von Bandsymbolen $a\in\Sigma^w$ sehen, arbeitet MS(S) mit Wahrscheinlichkeit $\eta(M,a,M')$ wie Maschine $M'\in\mathcal{M}$ weiter, oder MS(S) hält mit Wahrscheinlichkeit $1-\sum_{M'\in\mathcal{M}}\eta(M,a,M')$.

Die entstehende Maschine sei $\mathrm{MS}(S) = (K, \Sigma, p, s_{M_s}, \tau)$. Wir wählen zu jeder Maschine $M = (K_M, \Sigma, p_M, s_M, \tau)$ aus $\mathcal{M}$ einen neuen Zustand $q_M \notin \bigcup_{M' \in \mathcal{M}} K_{M'}$, insgesamt $|\mathcal{M}|$ paarweise verschiedene Zustände. Dann ist

$$K = \{q_M | M \in \mathcal{M}\} \cup \bigcup_{M \in \mathcal{M}} K_M$$

und für alle $a \in \Sigma^w$, $a' \in (\Sigma \cup \{L, R\})^w$ und $q \in K$, $q' \in K \cup \{h\}$ ist

$$p(q, a, q', a') = \begin{cases} p_M(q, a, q', a') & q, q' \in K_M, \\ p_M(q, a, h, a') & q \in K_M,\ q' = q_M, \\ \eta(M, a, M') p_{M'}(s_{M'}, a, q', a') & q = q_M,\ q' \in K_{M'}, \\ \eta(M, a, M') p_{M'}(s_{M'}, a, h, a') & q = q_M,\ q' = q_{M'}, \\ 1 - \sum_{M' \in \mathcal{M}} \eta(M, a, M') & q = q_M,\ q' = h,\ a = a', \\ 0 & \text{sonst.} \end{cases}$$

1.12. Lemma *Ist S ein probabilistisches Maschinenschema, dann ist die Maschine* $\mathrm{MS}(S)$ *eine probabilistische Turingmaschine.*

Beweis: Die einzige Schwierigkeit entsteht bei Übergängen zwischen zwei Maschinen aus $\mathcal{M}$. Die neuen Übergänge zu einem Zusatzzustand q_M, $M \in \mathcal{M}$, erfüllen Gleichung (1.1), da dieser neue Zustand nur den Haltezustand von Maschine M ersetzt. Also brauchen wir nur Übergänge zu betrachten, die die Zustände q_M, $M \in \mathcal{M}$, verlassen. Sei $a \in \Sigma^w$, dann ist

$$\begin{aligned} & \sum_{(q,a') \in (K \cup \{h\}) \times (\Sigma \cup \{L,R\})^w} p(q_M, a, q, a') \\ &= p(q_M, a, h, a) + \sum_{(q,a') \in K \times (\Sigma \cup \{L,R\})^w} p(q_M, a, q, a') \\ &= p(q_M, a, h, a) + \sum_{M' \in \mathcal{M}} \eta(M, a, M') \underbrace{\sum_{(q,a') \in (K_{M'} \cup \{h\}) \times (\Sigma \cup \{L,R\})^w} p_{M'}(s_{M'}, a, q, a')}_{=1} \\ &= 1 - \sum_{M' \in \mathcal{M}} \eta(M, a, M') + \sum_{M' \in \mathcal{M}} \eta(M, a, M') = 1. \end{aligned}$$

□

1.13. Bemerkung Bei der Konstruktion von Münzwurfmaschinen könnte man fordern, daß η nur die Werte 0 und 1 annimmt, und neben deterministischen Komponenten nur ein Münzwurfprimitiv $\mathtt{CoinToss}^{(i)}$ zulassen, das auf Arbeitsband i mit Wahrscheinlichkeit $1/2$ eine 0 bzw. eine 1 als „Münzwurfergebnis“ schreibt. Ausreichend ist aber, daß alle Teilmaschinen Münzwurfmaschinen sind und daß der Wertebereich von η eine Menge $\{k' 2^{-k} | k' \in [\![0, 2^k]\!]\}$ ist für ein $k \in \mathbb{N}$, ähnlich wie bei der Definition von Münzwurfmaschinen (vgl. Seite 4). ◇

Man beachte, daß die entstehende Maschine MS(S) nur eindeutig ist bis auf die Wahl der neuen Zustände. Nach Lemma 1.8 sind die entstehenden Turingmaschinen aber äquivalent, was unsere Notation rechtfertigt.

Wenn ein Maschinenschema aus Maschinen aufgebaut ist, die selbst aus Maschinenschemata entstanden sind, ist es anschaulich klar, daß man diese zweistufige Konstruktion auch in einem Schritt zusammenfassen kann.

Sei Σ ein Bandalphabet und τ eine Kopfzuordnung. Mit $0_{\Sigma,\tau}$ bezeichnen wir eine Maschine mit einem beliebigen Startzustand s, Zustandsmenge $K = \{s\}$ und Übergangswahrscheinlichkeit $p(s, a, h, a) = 1$ für alle $a \in \Sigma^{|\mathrm{Dom}(\tau)|}$ und 0 sonst. Damit macht $0_{\Sigma,\tau}$ immer genau einen deterministischen Schritt, ohne die Bandinhalte oder Kopfpositionen zu verändern. Da der einzige Zustand s frei wählbar ist, gibt es unendlich viele Versionen von $0_{\Sigma,\tau}$ zu jedem Σ, τ.

1.14. Satz *Sei* $S = (\mathcal{M}, \eta, M_s)$ *ein Maschinenschema mit Bandalphabet* Σ *und Kopfzuordnung* τ. *Sei* $M_u \in \mathcal{M}$, $M_u = \mathrm{MS}(\mathcal{M}', \eta', M'_s)$ *mit* $\mathcal{M} \cap \mathcal{M}' = \emptyset$. *Dann gibt es ein Schema* $S'' = (\mathcal{M}'', \eta'', M''_s)$ *mit* $\mathcal{M}'' = \mathcal{M} \setminus \{M_u\} \cup \{0_u\} \cup \mathcal{M}'$ *und* $\mathrm{MS}(S'') \sim \mathrm{MS}(S)$, *wobei* $0_u \notin \mathcal{M} \cup \mathcal{M}'$ *eine Version von* $0_{\Sigma,\tau}$ *ist.*

Beweis: Die Übergangsfunktion von S'' sei

$$\eta''(M, a, M') = \begin{cases} \eta(M, a, M') & M, M' \in \mathcal{M} \setminus \{M_u\}, \\ \eta(M, a, M_u) & M \in \mathcal{M}, M' = M'_s, \\ \eta'(M, a, M') & M, M' \in \mathcal{M}', \\ 1 - \sum_{M'' \in \mathcal{M}'} \eta'(M, a, M'') & M \in \mathcal{M}', M' = 0_u, \\ \eta(M_u, a, M') & M = 0_u, M' \in (\mathcal{M} \setminus \{M_u\}) \cup \{M'_s\}, \\ 0 & \text{sonst.} \end{cases}$$

Damit hat MS(S) die Zustandsmenge

$$\begin{aligned} K &= \{q_M | M \in \mathcal{M}\} \cup \bigcup_{M \in \mathcal{M}} K_M \\ &= \{q_M | M \in \mathcal{M} \setminus \{M_u\}\} \cup \bigcup_{M \in \mathcal{M} \setminus \{M_u\}} K_M \cup \{q_{M_u}\} \cup K_{M_u}. \end{aligned}$$

Da M_u aber aus einem Maschinenschema entstand, ist

$$K_{M_u} = \{q_M | M \in \mathcal{M}'\} \cup \bigcup_{M \in \mathcal{M}'} K_M,$$

also

$$K = \{q_M | M \in (\mathcal{M} \setminus \{M_u\}) \cup \mathcal{M}'\} \cup \bigcup_{M \in (\mathcal{M} \setminus \{M_u\}) \cup \mathcal{M}'} K_M \cup \{q_{M_u}\}$$

und

$$K'' = \{q_M | M \in (\mathcal{M} \setminus \{M_u\}) \cup \mathcal{M}'\} \cup \bigcup_{M \in (\mathcal{M} \setminus \{M_u\}) \cup \mathcal{M}'} K_M \cup \{q_{0_u}, s_{0_u}\}.$$

Wir wollen Lemma 1.10 benutzen. Die dafür benötigte Einbettung $i : K \cup \{h\} \to K'' \cup \{h\}$ ist

$$i(q) = \begin{cases} q_{0_u} & q = q_{M_u}, \\ q & \text{sonst.} \end{cases}$$

Damit ist $K'' \setminus i(K) = \{s_{0_u}\}$, s_{0_u} aber ist überflüssig: $p''(q, a, s_{0_u}, a') = 0$ für $q \notin K_{0_u} = \{s_{0_u}\}$ nach Definition von Maschinenschemata und $p''(s_{0_u}, a, s_{0_u}, a') = 0$ nach Definition von 0_u.

Bleibt also zu zeigen, daß $p(q, a, q', a') = p''(i(q), a, i(q'), a')$ gilt (gleichwertige Zustände treten hier nicht auf). Da η'' lokal wie η bzw. η' konstruiert ist, müssen nur Übergänge von bzw. zu q_{M_u} bzw. q_{0_u} verglichen werden. Übergänge von diesen Zuständen werden aber durch die Übergangsfunktionen bestimmt, und nach Definition ist $\eta(M_u, a, M) = \eta''(0_u, a, M)$ für alle relevanten M.

Übergänge zu q_{0_u} entsprechen Übergängen des Schemas S'' zu 0_u, deren Wahrscheinlichkeit ist die Haltewahrscheinlichkeit von $M_u = MS(\mathcal{M}', \eta', M'_S)$. Da q_{M_u} in S den Haltezustand von M_u ersetzt, sind die Voraussetzungen von Lemma 1.10 erfüllt und $MS(S) \sim MS(S'')$. □

1.15. Bemerkung Die Voraussetzung $\mathcal{M} \cap \mathcal{M}' = \emptyset$ ist nicht besonders hart, da wir aus jeder Maschine in $\mathcal{M} \cap \mathcal{M}'$ durch Umbenennen eines Zustandes zwei unterschiedliche, aber äquivalente Maschinen machen können. Diese Deformation erhält die Äquivalenz der aus den Maschinenschemata entstehenden Turingmaschinen. ◇

Maschinenschemata werden wir ähnlich wie in [LewisPap, Kapitel 4.3] als Graphen darstellen mit genau einem Knoten für jede Maschine $M \in \mathcal{M}$, die Startmaschine markiert mit „>“. Ist ein Übergang zwischen zwei Maschinen M und M' möglich, d.h. $\eta(M, a, M') > 0$ für ein $a \in \Sigma^w$, dann verbinden wir die Maschinen M und M' mit einem Pfeil, der mit a und der Wahrscheinlichkeit $\eta(M, a, M')$ des Übergangs markiert wird. Beträgt diese Wahrscheinlichkeit 1, geben wir sie nicht an. Für derartige deterministische Übergänge benutzen wir die in [LewisPap, Kapitel 4.4] beschriebenen intuitiven Vereinfachungen wie Zusammenfassen oder Weglassen von Pfeilen.

Neben diesem auf probabilistische Turingmaschinen verallgemeinerten Konzept von [LewisPap, Definition 4.3.1] gibt es auch die Möglichkeit, komplexe Turingmaschinen durch Parallelausführung zu definieren.

1.4 Parallelausführung probabilistischer Turingmaschinen

Als weiteres Bauprinzip führen wir die Parallelausführung von Maschinen ein, die mit verschiedenen Köpfen arbeiten. $M_1|M_2$ steht für eine Maschine, die die Teilma-

schinen M_1 und M_2 solange parallel ausführt, bis beide halten. Ist eine der Maschinen früher fertig, tritt sie solange auf der Stelle, bis auch die zweite hält.

1.16. Definition *Sei M eine probabilistische Turingmaschine mit w Köpfen und Übergangswahrscheinlichkeit p. Wir nennen $i \in [\![1, w]\!]$ einen* **agierenden Kopf** *von M, wenn*

$$\exists_{(q,a,q',a') \in \mathrm{supp}(p)} \quad a_i \neq a'_i,$$

d.h. es ist möglich, daß sich Kopf i bewegt oder ein Zeichen durch ein anderes überschreibt.

Die **Menge aller agierenden Köpfe** *von M bezeichnen wir mit A_M.*

1.17. Definition *Seien $M_1 = (K_1, \Sigma, p_1, s_1, \tau)$ und $M_2 = (K_2, \Sigma, p_2, s_2, \tau)$ probabilistische Turingmaschinen mit identischen Bandalphabeten und Kopfzuordnungen und disjunkten Mengen agierender Köpfe. Dann nennen wir $M_1|M_2$ die* **Parallelausführung** *von M_1 und M_2.*

Die Maschine $M_1|M_2$ führt in jedem Schritt je einen Schritt der beiden Teilmaschinen aus. Hält eine der Teilmaschinen vor der anderen, verharrt sie im Haltezustand. Die Gesamtmaschine hält, wenn beide Teilmaschinen in den Haltezustand übergegangen sind.
Formal ist $M_1|M_2 = (K, \Sigma, p, s, \tau)$ mit

$$K = ((K_1 \cup \{h\}) \times (K_2 \cup \{h\})) \setminus \{(h,h)\}, \qquad s = (s_1, s_2)$$

und für $q = (q_1, q_2) \in K$, $q' \in K \cup \{h\}$, $a \in \Sigma^w$, $a' \in (\Sigma \cup \{L, R\})^w$

$$p(q,a,q',a') = \begin{cases} p_1(q_1,a,q'_1,a'')p_2(q_2,a,q'_2,a''') \\ \quad q \in K_1 \times K_2, q' = (q'_1,q'_2) \in K \text{ oder } q' = q'_1 = q'_2 = h \\ \quad \text{und es gibt } a'', a''' \in (\Sigma \cup \{L,R\})^w, \text{ die zu } a, a' \text{ passen}, \\ p_1(q_1,a,q'_1,a') \\ \quad q \in K_1 \times \{h\}, q' = (q'_1,h) \in K \text{ oder } q' = q'_1 = h, \\ p_2(q_2,a,q'_2,a') \\ \quad q \in \{h\} \times K_2, q' = (h,q'_2) \in K \text{ oder } q' = q'_2 = h, \\ 0 \\ \quad \text{sonst.} \end{cases}$$

Dabei bedeutet „es gibt a'', a''', die zu a, a' passen“, daß sich a und a' nur für agierende Köpfe von M_1 oder M_2 unterscheiden und diese Kopfaktionen damit auf a'' für M_1 und a''' für M_2 aufgeteilt werden können, also $a_i = a'_i$ für $i \in [\![1, w]\!] \setminus (A_{M_1} \cup A_{M_2})$,

$$a''_i = \begin{cases} a'_i & i \in A_{M_1}, \\ a_i & \text{sonst} \end{cases} \qquad \text{und} \qquad a'''_i = \begin{cases} a'_i & i \in A_{M_2}, \\ a_i & \text{sonst.} \end{cases}$$

Damit sind a'' und a''' eindeutig durch a und a' festgelegt.

1.18. Lemma *Die Parallelausführung $M_1|M_2$ ist eine probabilistische Turingmaschine.*

Beweis: Es ist nur Gleichung (1.1) nachzuprüfen. Sei also $a \in \Sigma^w$ beliebig. Für $q = (q_1, h) \in K_1 \times \{h\}$ ist nach Definition von p

$$\sum_{(q',a')\in(K\cup\{h\})\times(\Sigma\cup\{L,R\})^w} p(q,a,q',a') = \sum_{(q_1',a')\in(K\cup\{h\})\times(\Sigma\cup\{L,R\})^w} p_1(q_1,a,q_1',a') = 1,$$

da M_1 eine probabilistische Turingmaschine ist. Der Fall $q = (h, q_2) \in \{h\} \times K_2$ geht analog. Für $q = (q_1, q_2) \in K_1 \times K_2$ schließlich ist

$$\sum_{(q',a')\in(K\cup\{h\})\times(\Sigma\cup\{L,R\})^w} p(q,a,q',a') = \sum_{\substack{(q_1',q_2',a')\in(K_1\cup\{h\})\times(K_2\cup\{h\})\times(\Sigma\cup\{L,R\})^w \\ \text{es gibt } a'',a'''\in(\Sigma\cup\{L,R\})^w \text{ passend zu } a,a'}} p_1(q_1,a,q_1',a'')p_2(q_2,a,q_2',a''').$$

Wegen der Voraussetzung $A_{M_1} \cap A_{M_2} = \emptyset$ durchlaufen a'' und a''' mit a' alle Werte mit $(q_1, a, q_1', a'') \in \mathrm{supp}(p_1)$ bzw. $(q_2, a, q_2', a''') \in \mathrm{supp}(p_2)$, d.h. die vorige Summe läßt sich umformen zu

$$\sum_{\substack{(q_1',a'')\in(K_1\cup\{h\})\times(\Sigma\cup\{L,R\})^w \\ (q_2',a''')\in(K_2\cup\{h\})\times(\Sigma\cup\{L,R\})^w}} p_1(q_1,a,q_1',a'')p_2(q_2,a,q_2',a''') = 1 \cdot 1.$$

□

Da $A_{M_1|M_2} = A_{M_1} \cup A_{M_2}$ ist, können wir die Parallelausführung $M_1|\cdots|M_n$ mehrerer Maschinen mit paarweise disjunkten Mengen agierender Köpfe induktiv definieren als $(M_1|\cdots|M_{n-1})|M_n$.

1.19. Lemma *Die Verknüpfung „|“ ist kommutativ und assoziativ bis auf Isomorphie der Zustandsmengen.*

Beweis: Die Kommutativität folgt aus der Isomorphie der entstehenden Zustandsmengen $((K_1 \cup \{h\}) \times (K_2 \cup \{h\})) \setminus \{(h,h)\}$ und $((K_2 \cup \{h\}) \times (K_1 \cup \{h\})) \setminus \{(h,h)\}$ und der Definition von p.

Für die Assoziativität überlegt man sich, daß die Zustandsmengen von $M_1|(M_2|M_3)$ und $(M_1|M_2)|M_3$ isomorph zu $((K_1 \cup \{h\}) \times (K_2 \cup \{h\}) \times (K_3 \cup \{h\})) \setminus \{(h,h,h)\}$ sind und daß sich wegen der paarweisen Disjunktheit der Mengen agierender Köpfe ein Übergang $((q_1,q_2,q_3),a,(q_1',q_2',q_3'),a')$ eindeutig aufspalten läßt in Übergänge $(q_i, a, q_i', a^{(i)})$, $i \in [\![1,3]\!]$. Die Wahrscheinlichkeit dieses Übergangs ist (im wesentlichen) $\prod_{i=1}^3 p_i(q_i, a, q_i', a^{(i)})$. □

Kombiniert man die Parallelausführung mit dem zuvor beschriebenen auf probabilistische Turingmaschinen verallgemeinerten Konzept der Maschinenschemata, so lassen sich aus einfachen deterministischen Turingmaschinen, die nur eine einzelne Aktion ausführen, beliebig komplexe probabilistische Turingmaschinen konstruieren.

1.5 Konstruierbarkeit beliebiger probabilistischer Turingmaschinen

In [LewisPap] werden Maschinenschemata benutzt, um Turingmaschinen mit *einem* Kopf und *einem* Band zu definieren. Diese Maschinen werden dort aus den Elementarmaschinen L, R und σ, $\sigma \in \Sigma$, konstruiert, die den Kopf eine Bandposition nach links bzw. rechts bewegen bzw. an der aktuellen Position das Zeichen σ schreiben.

Wir benutzen leicht erweiterte Elementarmaschinen $L^{(i)}_{\Sigma,\tau}$, $R^{(i)}_{\Sigma,\tau}$ und $\sigma^{(i)}_{\Sigma,\tau}$, wobei Σ das Bandalphabet und τ die Kopfzuordnung der Maschinen sind. Das Superskript (i) bedeutet, daß die entsprechende Aktion mit Kopf i ausgeführt wird. Zusätzlich zu diesen Maschinen benutzen wir die in Abschnitt 1.3 eingeführte Elementarmaschine $0_{\Sigma,\tau}$, die nichts tut und sofort hält. Wenn klar ist, welches Bandalphabet und welche Kopfzuordnung diese Elementarmaschinen (und damit die Zielmaschine) haben, lassen wir das Subskript weg.

Wir wollen nun zeigen, daß die Beschränkung auf diese Bauelemente nicht wesentlich ist, d.h. es ist immer möglich, zu einer gegebenen probabilistischen Turingmaschine eine äquivalente Maschine aus den angegebenen Elementarmaschinen zu konstruieren.

1.20. Definition *Eine probabilistische Turingmaschine M ist* **aus den Bausteinen $\mathcal{E}$ konstruierbar**, *wenn es ein Maschinenschema $S = (\mathcal{M}, \eta, M_s)$ gibt mit $M \sim MS(S)$ und jede Maschine $M' \in \mathcal{M}$ ist äquivalent zu einer Maschine $M'' \in \mathcal{E}$.*

1.21. Satz *Sei M eine probabilistische Turingmaschine mit Bandalphabet Σ und Kopfzuordnung τ. Dann kann mittels Maschinenschemata und Parallelausführungen aus den Elementarmaschinen $0_{\Sigma,\tau}$, $L^{(i)}_{\Sigma,\tau}$, $R^{(i)}_{\Sigma,\tau}$ und $\sigma^{(i)}_{\Sigma,\tau}$, $\sigma \in \Sigma$, $i \in Dom(\tau)$, eine zu $> 0_{\Sigma,\tau} \longrightarrow M$ äquivalente Maschine konstruiert werden.*

Beweis: Wir konstruieren aus den angegebenen Elementarmaschinen eine Maschine M', die zu $> 0_{\Sigma,\tau} \longrightarrow M$ ausdünnbar ist. Nach Lemma 1.10 ist sie dann auch äquivalent zu $> 0_{\Sigma,\tau} \longrightarrow M$.

Sei $w = |\operatorname{Dom}(\tau)|$ die Anzahl der Köpfe von M. Zu $(q,a) \in (K \cup \{h\}) \times (\Sigma \cup \{L,R\})^w$ ist

$$M_{q,a} = E_1 | \cdots | E_w \qquad \text{mit} \qquad E_i = \begin{cases} L^{(i)}_{\Sigma,\tau} & a_i = L, \\ R^{(i)}_{\Sigma,\tau} & a_i = R, \\ \sigma^{(i)}_{\Sigma,\tau} & a_i = \sigma \in \Sigma. \end{cases}$$

Eigentlich fordern wir lediglich $M_{q,a} \sim E_1 | \cdots | E_w$, da $M_{q,a}$ nur abhängig von a ist und $M_{q,a} \neq M_{q',a}$ sein muß für $q \neq q'$.

Für $q \in K$, $q' \in K \cup \{h\}$, $a \in \Sigma^w$ und $a', a'' \in (\Sigma \cup \{L,R\})^w$ setzen wir

$$\eta(0, a, M_{q',a'}) = p(s_M, a, q', a')$$

und

$$\eta(M_{q,a''}, a, M_{q',a'}) = p(q, a, q', a').$$

Nach Konstruktion hat $M_{q,a}$ nur einen Zustand, den Startzustand $s_{q,a}$, und macht genau einen deterministischen Schritt. Der zu $M_{q,a}$ gehörige neue Zustand sei $z_{q,a}$. Damit hat M' die Zustandsmenge

$$K' = \{s_0, z_0\} \cup \{s_{q,a}, z_{q,a} | q \in K \cup \{h\}, a \in (\Sigma \cup \{L,R\})^w\},$$

den Startzustand s_0 und die Übergangswahrscheinlichkeit

$$p'(z,a,z',a') = \begin{cases} 1 & z = s_0, z' = z_0, a = a' \\ & \quad \text{oder } z = z_{h,a''}, z' = h, a = a', \\ p(s_M, a, q', a') & z = z_0, z' = z_{q',a'}, \\ p(q, a, q', a') & z = z_{q,a''}, z' = z_{q',a'}, \\ 0 & \text{sonst.} \end{cases}$$

Die $> 0_{\Sigma,\tau} \longrightarrow M$ entsprechende Maschine M'' hat die Zustandsmenge

$$K'' = \{s_0, z_0, z_M\} \cup K,$$

den Startzustand s_0 und die Übergangswahrscheinlichkeit

$$p''(q,a,q',a') = \begin{cases} 1 & q = s_0, q' = z_0, a = a' \\ & \quad \text{oder } q = z_M, q' = h, a = a', \\ p(s_M, a, q', a') & q = z_0, q' \in K, \\ p(q, a, q', a') & q, q' \in K, \\ p(q, a, h, a') & q \in K, q' = z_M, \\ 0 & \text{sonst.} \end{cases}$$

Sei nun $a_0 \in \Sigma^w$ beliebig, aber fest und die Abbildung $i : K'' \to K'$ definiert als

$$i(s_0) = s_0, \quad i(z_0) = z_0, \quad i(z_M) = z_{h,a_0}, \quad i(q) = z_{q,a_0},$$

dann erfüllt sie alle Anforderungen von Definition 1.9, denn alle Zustände $z_{q,a}$ sind gleichwertig, alle Zustände $s_{q,a}$ sind überflüssig und die Summenbedingung gilt wegen $p'(z, a, z_{q',a''}, a') = 0$ für $z \in K'$ und $a'' \neq a'$. Damit ist M'' ausdünnbar zu M'. □

Die im Beweis durchgeführte Konstruktion war zweistufig, wobei die Parallelausführung nur in der untersten Ebene der Konstruktion benötigt wurde, die zweite Ebene war ein Maschinenschema. Wir können uns also auf eine einstufige Konstruktion beschränken, wenn wir die Menge der Elementarmaschinen passend erweitern.

1.22. Definition *Sei Σ ein Bandalphabet, τ eine Kopfzuordnung und $W \subset \mathrm{Dom}(\tau)$ eine Menge von Kopfnummern, dann ist die* **zu (Σ, τ, W) passende Elementarmaschinenmenge** *$\mathcal{E}(\Sigma, \tau, W)$ die Menge aller aus*

$$\left\{ 0_{\Sigma,\tau}, L^{(i)}_{\Sigma,\tau}, R^{(i)}_{\Sigma,\tau}, \sigma^{(i)}_{\Sigma,\tau} \;\middle|\; \sigma \in \Sigma, i \in W \right\}$$

durch Parallelausführung konstruierbaren Maschinen.

1.23. Bemerkung Die Menge $\mathcal{E}(\Sigma, \tau, W)$ ist zwar unendlich, hat aber nur endlich viele Äquivalenzklassen von Turingmaschinen, da die mit (i) indizierten Maschinen mit Kopf i agieren, also zu jedem $i \in W$ nur eine der $|\Sigma| + 2$ möglichen Maschinen parallel ausgeführt werden darf. Weiter ist $M \sim M|0_{\Sigma,\tau}$ für jedes M, so daß es nur $(|\Sigma| + 2)^{|W|}$ verschiedene Maschinentypen in $\mathcal{E}(\Sigma, \tau, W)$ gibt. ◇

Damit erhalten wir sofort das

1.24. Korollar *Sei M eine probabilistische Turingmaschine mit Bandalphabet Σ, Kopfzuordnung τ und agierenden Köpfen A_M. Dann kann aus den Elementarmaschinen in $\mathcal{E}(\Sigma, \tau, A_M)$ ein zu $> 0_{\Sigma,\tau} \longrightarrow M$ äquivalentes Maschinenschema konstruiert werden.*

1.25. Bemerkung Die Maschine $> 0_{\Sigma,\tau} \longrightarrow M$ unterscheidet sich von M nur durch jeweils einen deterministischen Anfangs- bzw. Endschritt, der unabhängig ist von den Bandinhalten und diese nicht verändert, wie man dem Beweis von Satz 1.21 entnehmen kann. Wenn die Aktionen von M im Startzustand unabhängig von den Bandzuständen sind, können wir die Äquivalenz zum Schema $> M$ beweisen, indem im Beweis die Maschine $0_{\Sigma,\tau}$ durch die der ersten Aktion von M ersetzt wird. Der zusätzliche Endschritt von $> M$ kann aber nicht vermieden werden. ◇

Die Definition und Konstruktion von probabilistischen Standard-Turingmaschinen ist damit abgeschlossen. Im folgenden beschäftigen wir uns mit einzelnen Aspekten ihrer Anwendung und führen dabei nichtuniforme probabilistische Turingmaschinen ein.

1.6 Berechnungen, Laufzeit und Nichtuniformität

Man sagt, eine deterministische Turingmaschine mit einem Kopf auf einem Band und Bandalphabet Σ hat bei Eingabe $x \in \Sigma^*$ die Ausgabe $y \in \Sigma^*$, wenn sie aus einer speziellen Startkonfiguration eine spezielle Endkonfiguration erreicht. Gewöhnlich fordert man, daß das (einzige) Band den String x bzw. y enthält und der (einzige) Kopf unmittelbar dahinter positioniert ist. Damit berechnet die Maschine M eine (partielle) Funktion $f_M : \Sigma^* \to \Sigma^*$ mit $f_M(x) = y$.

Bei probabilistischen Turingmaschinen wird diese Funktion ersetzt durch die Wahrscheinlichkeit, mit der eine Maschine einer gegebenen Eingabe eine bestimmte Ausgabe zuordnet. Wir definieren zunächst, wie die speziellen Start- bzw. Haltekonfigurationen aussehen sollen.

Sei Σ ein Alphabet, $x \in \Sigma^*$ ein String und $|x|$ die Länge von x, also $x = x_1 \cdots x_{|x|}$ mit $x_i \in \Sigma$ für $i \in [\![1, |x|]\!]$. Sei weiter $s \in \mathbb{Z}$, dann ist $c_{x,s} \in \mathsf{T}_\Sigma$ der Bandinhalt, der den String x ab Position s enthält, also

$$c_{x,s}(s + i - 1) = \begin{cases} x_i & i \in [\![1, |x|]\!], \\ \# & \text{sonst.} \end{cases}$$

Für $c_{x,0}$ schreiben wir kurz c_x, den leeren String bezeichnen wir mit ϵ.

Da wir meist mehrere Ein- und Ausgaben an eine Maschine geben wollen, identifizieren wir zur Vereinfachung der Notation ein Stringtupel $x = (x_1, \ldots, x_\ell) \in (\Sigma^*)^\ell$ mit dem String $\overline{x} = \$x_1\$ \cdots \$x_\ell\$ \in (\Sigma \cup \{\$\})^*$, dabei ist $\$ \notin \Sigma$ ein beliebiges Trennzeichen aus dem Alphabet der betrachteten Maschine. Die Länge eines derartigen Stringtupels x ist die Länge von $\overline{x}$, also

$$|x| = \ell + 1 + \sum_{k=1}^{\ell} |x_k|.$$

Eine Teilmenge $L \subset (\Sigma^*)^\ell$ nennen wir (in Erweiterung der üblichen Terminologie) eine **Sprache**.

Sei M eine probabilistische Turingmaschine mit w Köpfen auf t Bändern, Startzustand s und Bandalphabet $\Sigma_M \supset \Sigma \cup \{\$\}$. Die zu einem Stringtupel x gehörende Startkonfiguration ist

$$s_M(x) = (s, c_{\overline{x}}, \underbrace{c_\epsilon, \ldots, c_\epsilon}_{t-1}, \underbrace{0, \ldots, 0}_{w}),$$

die Menge aller zu y gehörenden Haltekonfigurationen ist

$$H_M(y) = \{h\} \times \{c_{\overline{y}}\} \times \mathsf{T}_{\Sigma_M}^{t-1} \times \mathbb{Z}^w \subset H_M.$$

Wir setzen beim Beginn einer Berechnung also alle Köpfe auf Position 0 und damit die auf Band 1 befindlichen Köpfe an den Anfang der Eingabe, da es von der Bauart der Maschine abhängt, wie viele und welche der Köpfe auf dem ersten Band stehen. Aus demselben Grund ignorieren wir die Kopfpositionen am Ende der Berechnung.

1.26. Definition *Sei* Σ *ein Alphabet,* $\ell, m \in \mathbb{N}$ *und* M *eine probabilistische Turingmaschine mit Bandalphabet* $\Sigma_M \supset \Sigma$*. Die Wahrscheinlichkeit, daß* M *bei Eingabe* $x \in (\Sigma^*)^\ell$ *die Ausgabe* $y \in (\Sigma^*)^m$ *hat, ist*

$$\pi_M(x, y) = \sum_{\alpha \in H_M(y)} \wp^*_M(s_M(x), \alpha),$$

die Wahrscheinlichkeit, keine wohlgeformte Ausgabe zu produzieren, ist

$$\pi_M(x, \infty) = 1 - \sum_{y \in (\Sigma^*)^m} \pi_M(x, y).$$

Nach Satz 1.3 und Bemerkung 1.4 existieren $\pi_M(x, y)$ und $\pi_M(x, \infty)$ für alle x und y. Weiter ist $\pi_M(x, \infty) \geq \wp^*_M(s_M(x), \infty)$ und genau dann gleich, wenn M bei Eingabe x nur wohlgeformte Ausgaben produziert oder unendlich lange läuft. Dies läßt sich immer erreichen, wenn man von M zu $> M \longrightarrow$ `CheckOutput` übergeht mit einer geeigneten Maschine `CheckOutput`, die ungültige Ausgaben „repariert". In Analogie zu Definition 1.5 erhalten wir

1.27. Definition *Sei* Σ *ein Alphabet,* $\ell, m \in \mathbb{N}$*,* M *eine probabilistische Turingmaschine mit Bandalphabet* $\Sigma_M \supsetneq \Sigma$ *und* $x \in (\Sigma^*)^\ell$*. Wenn* $\pi_M(x, \infty) = 0$ *ist, dann sind*

$$ER_M(x) = er_M(s_M(x)) \quad \textit{und} \quad MR_M(x) = mr_M(s_M(x))$$

die **erwartete Laufzeit** *und die* **maximale Laufzeit** *von* M *bei Eingabe* x*.*

Im deterministischen Fall erhält man eine größere Berechnungsmächigkeit, wenn man Turingmaschinen mit polynomieller Laufzeit durch Familien von Schaltkreisen mit polynomiellem Größenwachstum ersetzt. Die Berechnungsmächtigkeit derartiger Schaltkreisfamilien ist äquivalent zu der von Turingmaschinen, die einen eingabeabhängigen, polynomiell langen Hilfsstring als Zusatzeingabe erhalten: wähle einen Schaltkreisauswerter als Maschine und eine Schaltkreisbeschreibung als Hilfseingabe bzw. kodiere Maschine plus Hilfseingabe in einen Schaltkreis polynomieller Größe (siehe z.B. [Reischuk90, S.84 ff]). Diese *Nichtuniformität* der Berechnung möchten wir auch im Falle probabilistischer Turingmaschinen erlauben. Dazu geben wir einer Maschine neben der Eingabe auf dem ersten Band eine Zusatzeingabe auf dem zweiten Band.

1.28. Definition *Eine* **nichtuniforme probabilistische Turingmaschine** *ist ein Paar* (M, nu)*, wobei* M *eine probabilistische Turingmaschine mit mindestens zwei Bändern und Bandalphabet* Σ_M *ist und* $nu : \Sigma^*_M \to \Sigma^*_M$ *eine Abbildung.*

Ist $\Sigma \subset \Sigma_M$ *ein Alphabet,* $\ell \in \mathbb{N}$*,* $L \subset (\Sigma^*)^\ell$ *eine Sprache und* $f : L \to Ran(f)$ *eine beliebige Funktion, dann sagen wir, daß* (M, ν) *auf* L **nichtuniform in** f *ist, wenn für alle* $x, x' \in L$ *mit* $f(x) = f(x')$ *auch* $nu(x) = nu(x')$ *ist.*

Meist betrachtet man Maschinen, die nichtuniform in der Eingabelänge $|\cdot|$ sind.
Für eine nichtuniforme Maschine (M, nu) ist die zu einem Stringtupel x gehörende Startkonfiguration

$$s_{(M,nu)}(x) = (s, c_{\overline{x}}, c_{nu(x)}, \underbrace{c_\epsilon, \ldots, c_\epsilon}_{t-2}, \underbrace{0, \ldots, 0}_{w}).$$

Die Menge aller zu y gehörenden Haltekonfigurationen ist definiert wie im uniformen Fall, $\pi_{(M,nu)}(x, \cdot)$, $E_{(M,nu)}(x)$ und $R_{(M,nu)}(x)$ erhält man, indem man $s_M(x)$ durch $s_{(M,nu)}(x)$ ersetzt.
Führen wir zum Abschluß dieses ersten Kapitels noch folgende Definition ein, um in Kapitel 3 asymptotische Aussagen machen zu können.

1.29. Definition *Sei Σ ein Alphabet, $\ell \in \mathbb{N}$ und $L \subset (\Sigma^*)^\ell$ eine Sprache. Die Laufzeit der probabilistischen Turingmaschine M ist* **erwartet polynomiell** *auf L, wenn es Konstanten $k, k' > 0$ derart gibt, daß $E_M(x)$ existiert und kleiner gleich $|x|^k$ ist für alle $x \in L$ mit $|x| \geq k'$. Gilt dies auch für R_M, dann ist die Laufzeit von M* **polynomiell** *auf L.*

Bei einer nichtuniformen Turingmaschine (M, nu) fordern wir zusätzlich, daß $|nu(x)| \leq |x|^k$ ist für $|x| \geq k'$.

Die Menge aller auf L polynomiellen probabilistischen Turingmaschinen ist $\mathcal{PP}(L)$, Einschränkung auf erwartete oder maximale Laufzeit oder uniforme oder nichtuniforme Algorithmen kennzeichnen wir Subskripten e, m, u oder n.

Aufbauend auf den in diesem Kapitel gebildeten Begriffen wird im folgenden Kapitel die Möglichkeit der Interaktion zwischen probabilistischen Turingmaschinen behandelt.

KAPITEL 2

Interaktionsmodell

Ziel dieses Kapitels ist, zu klären, was ein Protokoll ist und woraus seine Bestandteile — interaktive Turingmaschinen, interaktive Systeme, Ein- und Ausgabefunktionen, Teilnehmer und Angreifer — bestehen. Wir geben Beispielprotokolle für typische kryptographische Problemstellungen und zeigen, wie man die Laufzeit eines Protokolls durch die Laufzeit der Teilnehmer abschätzen kann.

2.1 Interaktive Turingmaschinen

Zusätzlich zu w **Arbeitsköpfen** und t **Arbeitsbändern** besitzen interaktive Turingmaschinen c **Kommunikationskanäle**, die jeweils aus einem Paar von **Kommunikationsbändern** mit je einem **Kommunikationskopf** bestehen. Die Idee ist, daß eine interaktive Turingmaschine später mit anderen interaktiven Turingmaschinen verbunden wird. Sie schreibt Nachrichten für andere Maschinen auf das erste Band eines Kanals, das **Sendeband**, und liest Nachrichten von anderen Maschinen vom zweiten, dem **Empfangsband**.

Wir betrachten also nur Turingmaschinen der folgenden Bauart:

2.1. Definition *Eine* (w, c)-**Turingmaschine** *ist eine probabilistische Turingmaschine mit* $w + 2c$ *Köpfen und* $t + 2c$ *Bändern, deren Kopfzuordnung* τ *die Eigenschaften* $\tau([\![1, w]\!]) = [\![1, t]\!]$ *und* $\tau(w + i) = t + i$, $i \in [\![1, 2c]\!]$, *hat. Darüberhinaus enthält das Bandalphabet das* **Nachrichtenendesymbol** $\square$.

Wir werden später in diesem Abschnitt eine (w, c)-*interaktive* Turingmaschine als eine (w, c)-Turingmaschine definieren, die aus speziellen Elementarmaschinen aufgebaut ist. Die diesen Elementarmaschinen entsprechenden Maschinenschemata werden auf die in Abschnitt 1.3 beschriebene Weise dargestellt. Die verwendeten Bausteine $L^{(i)}_{\Sigma,\tau}$, $R^{(i)}_{\Sigma,\tau}$, $\sigma^{(i)}_{\Sigma,\tau}$ und $0_{\Sigma,\tau}$ wurden bereits in Abschnitt 1.5 eingeführt. Da Bandalphabet und Kopfzuordnung immer durch die zu konstruierende (w, c)-Turingmaschine festgelegt sind, lassen wir im folgenden die Subskripte Σ, τ weg.

Die Maschine

$$N_R^{(i)} = \; > 0 \xrightarrow{\overline{\#}^{(w+2i)}} R^{(w+2i)} \qquad i \in [\![1, c]\!]$$

bewegt den Empfangskopf von Kanal i nach rechts, falls er auf einem nichtleeren Symbol steht, die Maschine

$$W_\sigma^{(i)} = \; > 0 \xrightarrow{\#^{(w+2i-1)}} \sigma^{(w+2i-1)} \qquad \sigma \in \Sigma \setminus \{\#\},\ i \in [\![1, c]\!]$$

schreibt das Symbol σ auf die aktuelle Bandposition des Sendekopfes von Kanal i, falls er auf dem Leersymbol # steht, und die Maschine

$$N_S^{(i)} = \; > 0 \xrightarrow{\overline{\#}^{(w+2i-1)}} R^{(w+2i-1)} \qquad i \in [\![1, c]\!]$$

bewegt den Sendekopf von Kanal i nach rechts, falls er auf einem nichtleeren Bandsymbol steht.

Stärker eingeschränkte interaktive Turingmaschinen dürfen nur abwechselnd senden und empfangen. Dies wird durch komplexere Kommunikationsprimitive erreicht,

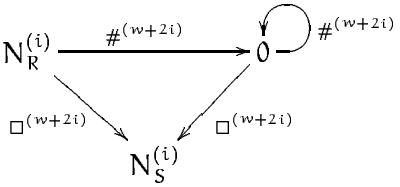
$$P_\square^{(i)} = W_\square^{(i)} \qquad i \in [\![1, c]\!]$$

und

$$P_\sigma^{(i)} = W_\sigma^{(i)} N_S^{(i)} \qquad \sigma \in \Sigma \setminus \{\#, \square\},\ i \in [\![1, c]\!]$$

schreiben ein nichtleeres Symbol auf das Sendeband von Kanal i und

$$G^{(i)} = \qquad i \in [\![1, c]\!]$$



bewegt den Empfangskopf zum nächsten nichtleeren Symbol auf dem Empfangsband von Kanal i.

Die ersten beiden Typen von Primitiven, die **Sendeprimitive**, garantieren, daß die Sendeköpfe auf einem □ stehen, nachdem dieses Zeichen gesendet und damit eine Nachricht beendet wurde, und daß sie auf einem # stehen, solange die Nachricht nicht beendet ist. Das **Empfangsprimitiv** $G^{(i)}$ garantiert, daß nach seinem Aufruf der Empfangskopf von Kanal i auf einem nichtleeren Symbol steht. Darüberhinaus wird der Sendekopf eines Kanals auf die nächste, leere Bandposition bewegt, wenn ein Nachrichtenendesymbol □ empfangen wird. Auf diese Weise wird zwischen dem „Empfangsmodus“ (Sendekopf auf □) und dem „Sendemodus“ (Sendekopf auf #) hin- und hergeschaltet. Man beachte, daß $N_S^{(i)}$ in $G^{(i)}$ den Sendekopf tatsächlich bewegt, da diese Stelle des Schemas nur erreicht wird, wenn der Sendekopf auf einem □ steht.

2.2. Definition *Eine* (w, c)*-Turingmaschine* M *mit Bandalphabet* Σ *und Kopfzuordnung* τ *wird* (w, c)**-interaktive Turingmaschine** *genannt, wenn aus den Maschinen in*

$$\mathcal{E}(\Sigma, \tau, [\![1, w]\!]) \cup \left\{ N_R^{(i)}, W_\sigma^{(i)}, N_S^{(i)} \;\middle|\; i \in [\![1, c]\!], \sigma \in \Sigma \setminus \{\#\} \right\}$$

ein zu M *äquivalentes Maschinenschema konstruiert werden kann. Sie wird* (w, c)**-alternierend-interaktive Turingmaschine** *genannt, wenn diese Konstruktion mit den Maschinen in*

$$\mathcal{E}(\Sigma, \tau, [\![1, w]\!]) \cup \left\{ G^{(i)}, P_\sigma^{(i)} \;\middle|\; i \in [\![1, c]\!], \sigma \in \Sigma \setminus \{\#\} \right\}$$

möglich ist.

Für $i \in [\![1, c]\!]$ *sei* $\Sigma_s(M, i)$ *die kleinste Menge aller Bandzeichen* $\sigma \in \Sigma$*, für die* $W_\sigma^{(i)}$ *bzw.* $P_\sigma^{(i)}$ *zur Konstruktion notwendig ist, sie heißt* **Sendealphabet** *von* M *auf Kanal* i*.*

2.3. Bemerkung Nach Konstruktion der $P_\sigma^{(i)}$ und $G^{(i)}$ sind alternierend-interaktive Turingmaschinen auch interaktive Turingmaschinen. ◇

2.4. Bemerkung Nach Korollar 1.24 und Satz 1.14 können wir jede beliebige auf den w Arbeitsbändern agierende probabilistische Turingmaschine als Baustein verwenden. ◇

Interaktive Turingmaschinen lassen sich zu interaktiven Systemen zusammensetzen. Dies ist Gegenstand des folgenden Abschnittes.

2.2 Interaktive Systeme

Die Grundidee von interaktiven Systemen ist wie folgt. Die Kommunikationskanäle zweier $(\cdot, 1)$-interaktiver Turingmaschinen werden verbunden, indem der Sendekopf der ersten mit dem Empfangskopf der zweiten Maschine auf ein Band gesetzt wird und der Empfangskopf der ersten zusammen mit dem Sendekopf der zweiten auf ein weiteres Band. So werden die vier Kommunikationsbänder auf zwei reduziert. Dann arbeiten die beiden Maschinen parallel, d.h. jeder Schritt der zusammengesetzten Maschine besteht aus je einem Schritt der ursprünglichen interaktiven Turingmaschinen. Durch geeignete Numerierung der Köpfe kann man erreichen, daß die Sendeköpfe einer jeden Teilmaschine gewinnen, wenn sie auf derselben Bandposition stehen sollten wie der Empfangskopf der anderen Maschine.

Was soll das interaktive System tun, wenn eine der Teilmaschinen hält, während die andere noch arbeitet? Die erste Idee ist, die bereits gehaltene Maschine in einem Pseudo-Halt-Zustand verharren zu lassen, ohne etwas zu tun, bis auch die zweite

Teilmaschine hält, wie es schon bei der Parallelausführung zweier gewöhnlicher Turingmaschinen gehandhabt wurde. Aber wenn die noch laufende Teilmaschine auf weitere Nachrichten von der bereits gehaltenen Teilmaschine wartet, hält das interaktive System nie. Dies passiert sogar dann, wenn beide Teilmaschinen beschränkte Laufzeit haben, da die Wartezeit nicht zur Laufzeit gezählt wird.

Dies ist nicht sehr zufriedenstellend, und es ist auch nicht sehr realistisch. Wenn ein Teilnehmer eines Telefongespräches auflegt, erhält der verbleibende Teilnehmer wiederholt die Nachricht „tüüüt". Wenn einer der beiden nur aufhört zu reden, kann man sagen, er/sie läßt aktiv Stille übermitteln.

Daher setzen wir interaktive Systeme aus *Paaren* interaktiver Turingmaschinen zusammen, den **Systemkomponenten**. Die erste Maschine eines Paares ist für die eigentliche Kommunikation verantwortlich, wenn sie hält, produziert die zweite Maschine das „tüüüt" (oder Stille). Diese zweite Maschine wird solange wiederholt ausgeführt, bis alle ersten Maschinen gehalten haben, dann ist die Interaktion beendet und das interaktive System hält.

Offenbar ist die zuerst erwähnte Variante der Zusammensetzung, in der bereits gehaltene Teilmaschinen in einem Pseudo-Halt-Zustand verharren, bis alle Maschinen halten, mit der von uns gewählten Art der Zusammensetzung realisierbar, indem man die spezielle Maschine 0 als zweite Maschine verwendet. Deren einziger Zustand ist dann der Pseudo-Halt-Zustand. Damit sind die in [Skript, BiBuMeThiThi] definierten interaktiven Systeme identisch mit Systemen alternierend-interaktiver Turingmaschinen mit Endmaschine 0.

2.5. Definition *Eine* **Systemkomponente** *ist ein Paar* $P = (B, E)$ *aus* (w_P, c_P)*-interaktiven Turingmaschinen mit jeweils gleichem Bandalphabet, identischen Kopfzuordnungen und disjunkten Zustandsmengen. Die* **Sendealphabete** *der einzelnen Kanäle einer Systemkomponente sind die Vereinigung der entsprechenden Sendealphabete seiner Teilmaschinen:* $\Sigma_s(P, i) = \Sigma_s(B, i) \cup \Sigma_s(E, i)$, $i \in [\![1, c_P]\!]$.

Da Bandalphabet und Kopfzuordnung der beiden Teilmaschinen einer Systemkomponente identisch sind, sind dies auch die in Abschnitt 1.1 definierten Mengen der Bandbeschreibungen D_B bzw. D_E der Maschinen B bzw. E. Wir bezeichnen daher die Menge der Bandbeschreibungen einer Komponente P mit D_P.

2.6. Definition *Ein* **interaktives System** *ist ein Paar* (C, V)*, wobei*

- C *die* **Menge der Systemkomponenten** *ist und*
- *die* **Kanalzuordnung** V *eine Bijektion auf der* **Menge der Kanäle**

$$\mathcal{K} = \{(P, j) \mid P \in C,\ j \in [\![1, c_P]\!]\}$$

mit $V(V(P, j)) = (P, j)$ *für alle* $(P, j) \in \mathcal{K}$.

Die Bandalphabete aller Komponenten müssen jeweils die Sendealphabete aller mit ihnen verbundener Kanäle anderer Komponenten enthalten, also

$$\bigwedge_{P \in C} \Sigma_P \supset \bigcup_{i=1}^{c_P} \Sigma_s(V(P,i)).$$

Ein interaktives System (C, V) wird in der folgenden Weise als eine gewöhnliche probabilistische Turingmaschine interpretiert:

Jede Systemkomponente besteht aus zwei interaktiven Turingmaschinen, der Anfangsmaschine und der Endmaschine. Wie schon erwähnt, simuliert die Systemmaschine alle Anfangsmaschinen parallel. Wenn eine Anfangsmaschine hält, simuliert die Systemmaschine an ihrer Stelle die zugehörige Endmaschine solange, bis auch die letzte Anfangsmaschine hält, danach hält die Systemmaschine.

Wir definieren nun die zusammengesetzte Maschine $IS_n(C, V) = (K, \Sigma, p, s, \tau)$ formal. Dabei ist n eine eindeutige (aber nicht eindeutig bestimmte) Numerierung der $k = |C|$ Komponenten, d.h. wir wählen eine Bijektion $n : C \to [\![1, k]\!]$ und bezeichnen die Anfangsmaschine einer Systemkomponente $P \in C$ mit $B_{n(P)}$, die Endmaschine mit $E_{n(P)}$. Wie wir in Abschnitt 2.3 zeigen werden, ist die tatsächliche Wahl der Numerierung für unser Modell unerheblich, die entstehenden Maschinen sind identisch bis auf Umnumerieren der Köpfe und Bänder und Umbenennen der Zustände. In der gewählten Numerierung sind $B_i = (K_i, \Sigma_i, p_i, s_i, \tau_i)$ und $E_i = (K_{i,E}, \Sigma_i, p_{i,E}, s_{i,E}, \tau_i)$ (w_i, c_i)-interaktive Turingmaschinen, $i \in [\![1, k]\!]$.

Mit den Abkürzungen $W_i = \sum_{\ell=1}^{i-1} w_\ell$, $C_i = \sum_{\ell=1}^{i-1} c_\ell$ und $T_i = \sum_{\ell=1}^{i-1} |\tau_\ell([\![1, w_\ell]\!])|$ für $i \in [\![1, k]\!]$, $W = W_k$, $C = C_k$ und $T = T_k$ ist $IS_n(C, V)$ eine probabilistische Turingmaschine mit $W + 2C$ Köpfen auf $T + C$ Bändern,

$$K = \bigtimes_{i=1}^{k} (K_i \cup K_{i,E}) \setminus \bigtimes_{i=1}^{k} K_{i,E}, \quad \Sigma = \bigcup_{i=1}^{k} \Sigma_i \quad \text{und} \quad s = (s_1, \ldots, s_k).$$

Die *Arbeitsköpfe* von Teilmaschine i entsprechen den Köpfen $W_i + 1, \ldots, W_{i+1}$ von $IS_n(C, V)$, die *Sendeköpfe* den Köpfen $W + C_i + 1, \ldots, W + C_{i+1}$ von $IS_n(C, V)$ und die *Empfangsköpfe* den Köpfen $W + C + C_i + 1, \ldots, W + C + C_{i+1}$ von $IS_n(C, V)$. Die resultierende Kopfzuordnung τ von $IS_n(C, V)$ ist damit

$$\tau(j) = \begin{cases} T_i + \tau_i(j - W_i) & j \in [\![W_i + 1, W_{i+1}]\!] \text{ für } i \in [\![1, k]\!] \\ T + j - W & j \in [\![W + 1, W + C]\!] \\ T + C_{i'} + j' & j \in [\![W + C + C_i + 1, W + C + C_{i+1}]\!] \text{ für } i \in [\![1, k]\!] \\ & \quad \text{und } (n^{-1}(i'), j') = V(n^{-1}(i), j - (W + C + C_i)) \end{cases}$$

für $j \in [\![1, W + 2C]\!]$. Eine Konsequenz dieser Kopfzuordnung ist, daß ein simulierter Sendekopf bei einem Schreibkonflikt mit einem simulierten Empfangskopf auf demselben simulierten Kommunikationsband gewinnt, weil die Nummern aller simulierten Sendeköpfe kleiner sind als die Nummern aller simulierten Empfangsköpfe.

Damit ist die Lösung von Schreibkonflikten insbesondere unabhängig von der Numerierung der Systemkomponenten.

Schließlich definieren wir die Übergangswahrscheinlichkeit p von $IS_n(C,V)$ aus den Übergangswahrscheinlichkeiten p_i und $p_{i,E}$ der Teilmaschinen B_i und E_i, $i \in [\![1,k]\!]$. Sei $K_i' = K_i \cup K_{i,E}$ und

$$p_i' : K_i' \times \Sigma_i^{w_i+2c_i} \times K_i' \times (\Sigma_i \cup \{L,R\})^{w_i+2c_i} \to [0,1]$$

mit

$$p_i'(q,a,q',a') = \begin{cases} p_i(q,a,q',a') & q,q' \in K_i \\ p_i(q,a,h,a') & q \in K_i,\ q' = s_{i,E} \\ p_{i,E}(q,a,q',a') & q \in K_{i,E},\ q' \in K_{i,E} \setminus \{s_{i,E}\} \\ p_{i,E}(q,a,h,a') & \\ \quad + p_{i,E}(q,a,s_{i,E},a') & q \in K_{i,E},\ q' = s_{i,E} \\ 0 & \text{sonst} \end{cases}$$

für alle $a \in \Sigma_i^{w_i+2c_i}$ und $a' \in (\Sigma_i \cup \{L,R\})^{w_i+2c_i}$. Das bedeutet, daß Übergänge von B_i und E_i in den Haltezustand h ersetzt werden durch Übergänge in den Startzustand $s_{i,E}$ der Endmaschine E_i. Man beachte, daß für alle $(q,a) \in K_i' \times \Sigma_i^{w_i+2c_i}$

$$\sum_{(q',a') \in K_i' \times (\Sigma_i \cup \{L,R\})^{w_i+2c_i}} p_i'(q,a,q',a') = 1 \tag{2.1}$$

ist nach Definition von p_i'. Wenn wir die Tupel der aktuellen Bandsymbole bzw. Kopfaktionen $a_i \in \Sigma_i^{w_i+2c_i}$ zerlegen in x_i, y_i und z_i für die w_i Arbeits-, die c_i Sende- und die c_i Empfangsköpfe, dann transformiert die Abbildung

$$\begin{aligned} \mu : & \bigtimes_{i=1}^{k} (K_i' \times \Sigma_i^{w_i+2c_i} \times K_i' \times (\Sigma_i \cup \{L,R\})^{w_i+2c_i}) \\ & \quad \to K \times \Sigma^{W+2C} \times (K \cup \{h\}) \times (\Sigma \cup \{L,R\})^{W+2C}, \\ & \mu((q_1,a_1,q_1',a_1'),\ldots,(q_k,a_k,q_k',a_k')) \\ & = \begin{cases} ((q_1,\ldots,q_k),x_1,\ldots,x_k,y_1,\ldots,y_k,z_1,\ldots,z_k, & \\ \quad h,x_1',\ldots,x_k',y_1',\ldots,y_k',z_1',\ldots,z_k') & q_i' \in K_{i,E},\ i \in [\![1,k]\!] \\ ((q_1,\ldots,q_k),x_1,\ldots,x_k,y_1,\ldots,y_k,z_1,\ldots,z_k, & \\ \quad (q_1',\ldots,q_k'),x_1',\ldots,x_k',y_1',\ldots,y_k',z_1',\ldots,z_k') & \text{sonst,} \end{cases} \end{aligned}$$

ein n-Tupel von Übergängen von „B_i und dann immer wieder E_i“ in einen Übergang von $IS_n(C,V)$. Damit ist

$$p(b) = \sum_{(b_1,\ldots,b_k) \in \mu^{-1}(b)} \prod_{i=1}^{k} p_i'(b_i).$$

2.7. Lemma *Ist* (C, V) *ein interaktives System, dann ist für jede Komponentennumerierung* n *die Maschine* $IS_n(C, V)$ *eine probabilistische Turingmaschine.*

Beweis: Wir müssen lediglich zeigen, daß $IS_n(C, V)$ mit dem so definierten p eine probabilistische Turingmaschine ist. Die mangelnde Injektivität von μ wird von p „repariert", daher haben wir für alle $q = (q_1, \ldots, q_k) \in K$ und $a \in \Sigma^{W+2C}$

$$\begin{aligned}
&\sum_{(q',a')\in(K\cup\{h\})\times(\Sigma\cup\{L,R\})^{W+2C}} p(q,a,q',a') \\
&= \sum_{\substack{(q',a')\\ \in(K\cup\{h\})\times(\Sigma\cup\{L,R\})^{W+2C}}} \sum_{\substack{((q_i,a_i,q_i',a_i')_{i\in[\![1,k]\!]})\\ \in\mu^{-1}(q,a,q',a')}} \prod_{i=1}^{k} p_i'(q_i,a_i,q_i',a_i') \\
&= \sum_{\substack{(q_i',a_i')\in K_i'\times(\Sigma_i\cup\{L,R\})^{w_i+2c_i}\\ i\in[\![1,k]\!]}} \prod_{i=1}^{k} p_i'(q_i,a_i,q_i',a_i') \\
&= \prod_{i=1}^{k} \sum_{(q_i',a_i')\in K_i'\times(\Sigma_i\cup\{L,R\})^{w_i+2c_i}} p_i'(q_i,a_i,q_i',a_i') = \prod_{i=1}^{k} 1 = 1,
\end{aligned}$$

wobei die letzte Gleichheit aus Gleichung (2.1) folgt. □

Aus dieser Definition der Übergangswahrscheinlichkeit eines interaktiven Systems als Produkt von Wahrscheinlichkeiten der Teilübergänge folgt die *statistische Unabhängigkeit* der Teilübergänge. Schließlich wollen wir modellieren, daß die einzelnen Maschinen nur aufgrund ihrer lokalen Information arbeiten und sich gegenseitig lediglich durch Nachrichtenaustausch beeinflussen.

Für interaktive Systeme sind etwas kompliziertere Konventionen notwendig, als dies für einzelne Turingmaschinen der Fall ist, wie wir im folgenden Abschnitt sehen werden.

2.3 Protokolle und ihre Ein- und Ausgaben

In Abschnitt 1.6 sagten wir, eine Turingmaschine mit Bandalphabet Σ hat bei Eingabe $x \in \Sigma^*$ die Ausgabe $y \in \Sigma^*$, wenn sie aus einer speziellen Startkonfiguration s_x eine spezielle Endkonfiguration h_y erreicht.

Diese einfache Konvention ist bei interaktiven Systemen aus drei Gründen nicht möglich. Erstens ist die entstehende Turingmaschine nicht eindeutig festgelegt, da keine bestimmte Komponentennumerierung vorgeschrieben wurde (die Unabhängigkeit eines Protokolls von der Wahl dieser Numerierung bleibt zu zeigen). Zweitens

wollen wir Privatheitsaspekte betrachten, d.h. jeder Teilnehmer erhält eine gesonderte Eingabe, die zunächst nur ihm bekannt sein soll. Drittens ist es für die Berechnung eines interaktiven Systems von entscheidender Bedeutung, welchen Inhalt die Kommunikationsbänder haben und wo die Kommunikationsköpfe stehen. So entscheidet bei alternierend-interaktiven Turingmaschinen das vom Sendekopf gesehene Zeichen, ob die Maschine auf diesem Kanal im Sende- oder Empfangsmodus startet. Wir benötigen also eine Art von Anfangsschablone, die mit den privaten Eingaben der Teilnehmer gefüllt wird.

2.8. Definition *Seien* Σ_e, Σ_a *Alphabete und* $\ell_e, \ell_a \in \mathbb{N}$. *Eine* **Eingabefunktion** *für eine Systemkomponente* P *ist eine Funktion* $e : \mathrm{Dom}(e) \to D_P$ *mit* $\mathrm{Dom}(e) \subset (\Sigma_e^*)^{\ell_e}$, *die Eingabetupel auf Bandbeschreibungen von* P *abbildet.*

Eine **Ausgabefunktion** $a : \mathrm{Dom}(a) \to (\Sigma_a^*)^{\ell_a}$ *wiederum bildet Bandbeschreibungen aus* $\mathrm{Dom}(a) \subset D_P$ *auf Ausgabetupel aus ab.*

Ein **Teilnehmer** *ist ein Tripel* (P, e, a), *bestehend aus einer Systemkomponente, einer Ein- und einer Ausgabefunktion für diese Komponente.*

Wir wollen annehmen, daß Eingabefunktionen die Empfangsköpfe jeweils auf einem nichtleeren Symbol positionieren.

Natürlich sollen diese Funktionen nicht mächtiger sein als die untersuchten interaktiven Systeme:

Ein Bandinhalt $c \in \top_\Sigma$ über einem Alphabet $\Sigma = \Sigma_0 \dot\cup \{\#\}$ ist durch einen String z, der die endlich vielen nichtleeren Bandsymbole enthält, und dessen Startpunkt s auf dem Band beschreibbar als $c = c_{z,s}$. Den Startpunkt schreiben wir unär als $1^{|s|}$ für negatives s, 0^s für positives s, ϵ für $s = 0$. Damit können wir eine Bandbeschreibung $d = (b_1, \ldots, b_t, h_1, \ldots, h_w) \in D_P$ einer Komponente P repräsentieren durch ein Tupel

$$r(d) = (z_1, s_1, \ldots, z_t, s_t, p_1, \ldots, p_w)$$

mit $z_i \in (\epsilon | \Sigma_0(\Sigma^*\Sigma_0)^*)$, $s_i, p_j \in (0^*|1^*)$, $b_i = c_{z_i,(-1)^{s_{i,1}}|s_i|}$ und $h_j = (-1)^{h_{j,1}}|h_j|$ für $i \in [\![1, t]\!]$, $j \in [\![1, w]\!]$. Da unsere Inhaltsstrings leer sind oder mit nichtleeren Zeichen beginnen und enden, sind sie (und damit die Darstellung) eindeutig.

Eine Eingabefunktion $e_P : (\Sigma^*)^\ell \to D_P$ wird **berechnet** durch die deterministische Turingmaschine M_{e_P}, wenn $\pi_{M_{e_P}}(x, r(d)) = 1 \iff e_P(x) = d$ ist für alle $x \in \mathrm{Dom}(e_P)$ und $d \in D_P$. Dabei benutzen wir die in Abschnitt 1.6 beschriebene Konvention für Ein- und Ausgaben probabilistischer Turingmaschinen. Berechenbarkeit von Ausgabefunktionen wird analog definiert.

Wir sind nun endlich in der Lage, zu sagen, was ein Protokoll ist, nämlich ein interaktives System und dazu passende Ein- und Ausgabefunktionen.

2.9. Definition *Ein* **Protokoll** *ist ein Quadrupel* $S = (C, V, (e_P)_{P\in C}, (a_P)_{P\in C})$, *wobei* (C, V) *ein interaktives System ist und alle* (P, e_P, a_P) *Teilnehmer sind für* $P \in C$.

Die Eingabefunktionen müssen **verträglich** *mit V sein, d.h. Kommunikationsbändern, die von V identifiziert werden, müssen von beiden beteiligten Eingabefunktionen dieselben Bandinhalte zugewiesen werden.*
Wir nennen

$$\mathrm{Dom}(S) = \bigtimes_{P \in C} \mathrm{Dom}(e_P)$$

den **Definitionsbereich**,

$$\mathrm{Ran}(S) = \bigtimes_{P \in C} \mathrm{Ran}(a_P)$$

den **Wertebereich** *von S.*

Aus den Ergebnissen der Eingabefunktionen wird, abhängig von der gewählten Komponentennumerierung n, eine Bandbeschreibung für $IS_n(C,V)$ gebildet, nämlich

(Arbeitsbandinhalte von Maschine $n^{-1}(1), \ldots,$ Arbeitsbandinhalte von $n^{-1}(k)$,
Sendebandinhalte von $n^{-1}(1), \ldots,$ Sendebandinhalte von $n^{-1}(k)$,
Arbeitskopfpositionen von $n^{-1}(1), \ldots,$ Arbeitskopfpositionen von $n^{-1}(k)$,
Sendekopfpositionen von $n^{-1}(1), \ldots,$ Sendekopfpositionen von $n^{-1}(k)$,
Empfangskopfpositionen von $n^{-1}(1), \ldots,$ Empfangskopfpositionen von $n^{-1}(k)$).

Man kann also aus den Eingabefunktionen $(e_P)_{P \in C}$ eine von n abhängige Funktion

$$e_n : \mathrm{Dom}(S) \to D_{IS_n(C,V)}$$

bilden, die einer Eingabe eine Bandbeschreibung von $IS_n(C,V)$ zuordnet. Die Zerlegung einer Bandbeschreibung von $IS_n(C,V)$ in Bandbeschreibungen der Systemkomponenten und die Definition der Ausgabefunktion

$$a_n : D_{IS_n(C,V)} \to \mathrm{Ran}(S)$$

ist analog.

2.10. Satz *Sei $S = (C, V, (e_P)_{P \in C}, (a_P)_{P \in C})$ ein Protokoll, n eine Komponentennumerierung und s_n der Startzustand von $IS_n(C,V)$. Dann sind für alle $x \in \mathrm{Dom}(S)$, alle $y \in \mathrm{Ran}(S)$ und alle $\ell \in \mathbb{N}$ die Größen*

$$\pi_S^{(\ell)}(x,y) = \sum_{d \in a_n^{-1}(y)} \wp_{IS_n(C,V)}^{(\ell)}((s_n, e_n(x)), (h,d))$$

und

$$\pi_S(x,y) = \sum_{d \in a_n^{-1}(y)} \wp_{IS_n(C,V)}^{*}((s_n, e_n(x)), (h,d))$$

unabhängig von n. Weiter ist für alle $x \in \mathrm{Dom}(S)$

$$\sum_{y \in \mathrm{Ran}(S)} \pi_S(x,y) \leq 1.$$

Beweis (Skizze): Die Existenz der Summen über $d \in a^{-1}(y)$ und die Gültigkeit der Ungleichung folgen aus Satz 1.3 und Bemerkung 1.4.

Für je zwei System-Maschinen zu verschiedenen Komponentennumerierungen n und n' kann man aus den Komponentennumerierungen ein Paar bijektiver Abbildungen (ϕ, ψ) der Kopf- bzw. Bandnummern konstruieren mit $\tau(i) = j \iff \tau'(\phi(i)) = \psi(j)$. Diese lassen sich ausweiten zu Isomorphien $\overline{\phi}$ auf den Übergängen und $\overline{\psi}$ auf den Konfigurationen, d.h. $p(b) = p'(\overline{\phi}(b))$ für alle Übergänge b und $\wp(\alpha, \beta) = \wp'(\overline{\psi}(\alpha), \overline{\psi}(\beta))$ für alle Konfigurationen α und β.

Weiter sind für eine Eingabe x die mit verschiedenen Komponentennumerierungen gebildeten Startkonfigurationen isomorph, d.h. $\overline{\psi}(s, e_n(x)) = (s', e_{n'}(x))$, und isomorphe Haltekonfigurationen geben dieselbe Ausgabe y. Damit folgt die Behauptung. □

Dieses $\pi_S(x, y)$ (bzw. $\pi_S^{(\ell)}(x, y)$) ist die Wahrscheinlichkeit, daß das Protokoll S bei Eingabe x die Ausgabe y (in genau ℓ Systemschritten) erzeugt. Weiter ist

$$\pi_S(x, \infty) = 1 - \sum_{y \in \mathbf{Ran}(S)} \pi_S(x, y)$$

die Wahrscheinlichkeit, daß das Protokoll bei Eingabe x nicht hält. Da die Ein- bzw. Ausgabefunktionen deterministisch sind, ist π_S unabhängig von der tatsächlichen Realisierung der e_P und a_P durch Turingmaschinen.

Damit können wir den Index n bei e_n, a_n und $IS_n(C, V)$ weglassen. Erwartete und maximale Laufzeit eines Protokolls sind unabhängig von der Komponentennumerierung definierbar als die entsprechenden Größen von $IS(C, V)$.

Zur Laufzeit eines Protokolles werden wir die Zeit zur Berechnung der Startbandbeschreibung bzw. der Ausgabe hinzuzählen.

Protokolle können auf vielfältige Weise gestört werden. Das allgemeinste Fehlermodell wird nun besprochen.

2.4 Fehlermodell Angreifer

Nachdem wir wissen, was Protokolle sind und welche Struktur sie haben, brauchen wir nur noch eines, um Kryptographie betreiben zu können: Fehler. Darunter verstehen wir abweichendes Verhalten einiger Teilnehmer. Die bösartigsten Fehler, die man sich vorstellen kann, sind die, die ein Bösewicht sich ausdenkt. Daher ersetzen wir für eine Fehlerbetrachtung einen oder mehrere Teilnehmer eines interaktiven Systems durch *einen* neuen Teilnehmer, den Angreifer, und erhalten so ein gestörtes interaktives System. Zum Angreifer gehören auch passende Ein- und Ausgabefunktionen, die die Eingaben der korrumpierten Teilnehmer sowie eine Zusatzeingabe für den Angreifer auf Startbandbeschreibungen des Angreifers und Haltbandbeschreibungen auf Ausgaben für die korrumpierten Teilnehmer und den Angreifer abbilden.

2.11. Definition *Ein* **Störer** *ist ein Tripel* (C_A, P_A, ν_A)*, bestehend aus*

- *der Menge* C_A *der* **angegriffenen Systemkomponenten**,
- *einer* **angreifenden** *Komponente* P_A*, deren Bandalphabet die Bandalphabete aller angegriffenen Komponenten umfaßt, und*
- *der* **Angreifer-Kanalzuordnung** ν_A*, einer bijektiven Abbildung der* c_{P_A} *Kanäle der angreifenden Komponente auf die Menge*

$$\mathcal{K}_A = \{(P, j) \mid P \in C_A, j \in [\![1, c_P]\!]\}$$

der Kanäle der angegriffenen Systemkomponenten.

Ein interaktives System (C, V) *ist* **störbar** *durch den Störer* (P_A, C_A, ν_A)*, wenn* $C_A \subset C$ *und* $\Sigma_s(P_A, j) \subset \Sigma_s(\nu_A(j))$ *ist für* $j \in [\![1, c_{P_A}]\!]$.

Das bedeutet, daß die Struktur des Störers durch die Struktur der angegriffenen Teilnehmer bestimmt wird, denn die Anzahl der Kommunikationskanäle des Störers muß gleich der Summe der Kommunikationskanäle aller angegriffenen Systemkomponenten sein.

Aus einem interaktiven System (C, V) und einem Störer (P_A, C_A, ν_A) wird auf folgende Weise ein **gestörtes System** (C', V'):

Das entstehende gestörte System hat die Komponentenmenge $C' = \{P_A\} \cup C \setminus C_A$, die Kanalmenge

$$\mathcal{K}' = \{(P, j) \mid P \in C', j \in [\![1, c_P]\!]\}$$

und die Kanalzuordnung $V' : \mathcal{K}' \to \mathcal{K}'$ mit

$$V'(P, j) = \begin{cases} V(P, j) & (P, j) \in \mathcal{K} \setminus \mathcal{K}_A, V(P, j) \in \mathcal{K} \setminus \mathcal{K}_A, \\ (P_A, \nu_A^{-1}(V(P, j)) & (P, j) \in \mathcal{K} \setminus \mathcal{K}_A, V(P, j) \in \mathcal{K}_A, \\ V(\nu_A(j)) & P = P_A, V(\nu_A(j)) \in \mathcal{K} \setminus \mathcal{K}_A, \\ (P_A, \nu_A^{-1}(V(\nu_A(j)))) & P = P_A, V(\nu_A(j)) \in \mathcal{K}_A. \end{cases}$$

Diese vier Fälle entsprechen Verbindungen zwischen nichtangegriffenen Komponenten, von nichtangegriffenen zu angegriffenen, von angegriffenen zu nichtangegriffenen und zwischen angegriffenen Komponenten.

Um zu zeigen, daß $V'(V'(P, j)) = (P, j)$ ist, müssen wir diese vier Fälle getrennt betrachten:

Für $(P, j) \in \mathcal{K} \setminus \mathcal{K}_A$ und $V(P, j)) \in \mathcal{K} \setminus \mathcal{K}_A$ ist $V(V(P, j)) = (P, j) \in \mathcal{K} \setminus \mathcal{K}_A$, also $V'(V'(P, j)) = V'(V(P, j)) = V(V(P, j)) = (P, j)$.

Für $(P, j) \in \mathcal{K} \setminus \mathcal{K}_A$, aber $V(P, j) \in \mathcal{K}_A$ ist $V'(V'(P, j)) = V'(P_A, \nu_A^{-1}(V(P, j)))$. Wegen $V(\nu_A(\nu_A^{-1}(V(P, j)))) = V(V(P, j)) = (P, j) \in \mathcal{K} \setminus \mathcal{K}_A$ ist dann $V'(P_A, \nu_A^{-1}(V(P, j))) = V(\nu_A(\nu_A^{-1}(V(P, j)))) = (P, j)$.

Für $P = P_A$ und $V(\nu_A(j)) \in \mathcal{K} \setminus \mathcal{K}_A$ ist $V'(V'(P,j)) = V'(V(\nu_A(j)))$. Dann ist aber $V(V(\nu_A(j))) = \nu_A(j) \in \mathcal{K}_A$ und $V'(V(\nu_A(j))) = (P_A, \nu_A^{-1}(V(V(\nu_A(j))))) = (P_A, j) = (P, j)$.

Für $P = P_A$, aber $V(\nu_A(j)) \in \mathcal{K}_A$ endlich ist $V'(V'(P,j)) = V'(P_A, \nu_A^{-1}(V(\nu_A(j))))$. Mit $V(\nu_A(\nu_A^{-1}(V(\nu_A(j))))) = \nu_A(j) \in \mathcal{K}_A$ erhalten wir $V'(P_A, \nu_A^{-1}(V(\nu_A(j)))) = (P_A, \nu_A^{-1}(V(\nu_A(\nu_A^{-1}(V(\nu_A(j))))))) = (P_A, j) = (P, j)$.

2.12. Definition *Ein* **Angreifer** *ist ein Quintupel* $(C_A, P_A, \nu_A, e_{P_A}, a_{P_A})$, *wobei* (C_A, P_A, ν_A) *ein Störer und* (P_A, e_{P_A}, a_{P_A}) *ein Teilnehmer ist.*

Ein Protokoll $(C, V, (e_P)_{P \in C}, (a_P)_{P \in C})$ *ist* **angreifbar** *durch* $(C_A, P_A, \nu_A, e_{P_A}, a_{P_A})$, *wenn*

- (C, V) *durch* (C_A, P_A, ν_A) *störbar ist, das dabei entstehende interaktive System sei* (C', V'),
- *die neuen Eingabefunktionen* $(e_P)_{P \in C'}$ *mit der neuen Kanalzuordnung* V' *verträglich sind und*
- *es ein Alphabet* Σ_A *und* $\ell, m \in \mathbb{N}$ *gibt mit*

$$\mathrm{Dom}(e_{P_A}) = (\Sigma_A^*)^\ell \times \bigtimes_{P \in C_A} \mathrm{Dom}(e_P) \text{ und } \mathrm{Ran}(a_{P_A}) = (\Sigma_A^*)^m \times \bigtimes_{P \in C_A} \mathrm{Ran}(e_P).$$

Wir nennen $\mathrm{Dom}(A) = (\Sigma_A^*)^\ell$ *den* **eigentlichen Definitionsbereich**, $\mathrm{Ran}(A) = (\Sigma_A^*)^m$ *den* **eigentlichen Wertebereich** *des Angreifers* A.

Der Angreifer bekommt also die Eingaben aller angegriffenen Teilnehmer und eine zusätzliche Eingabe. Die Ausgabefunktion des Angreifers erzeugt Pseudo-Ausgaben für alle angegriffenen Teilnehmer und eine Ausgabe für den Angreifer. Diese zusätzliche Ein- bzw. Ausgabe kann natürlich leer sein.

Die **Menge aller Angreifer** bezeichnen wir mit $\mathcal{ADV}$, die **Menge der möglichen Angreifer** für ein Protokoll $\mathcal{P}$ ist

$$\mathrm{Adv}(\mathcal{P}) \;=\; \{A \in \mathcal{ADV} | \mathcal{P} \text{ ist angreifbar durch } A\},.$$

Wir können die vorhergehende Konstruktion als eine (partielle) Funktion

$$\mathrm{aProt} : \mathcal{PROT} \times \mathcal{ADV} \to \mathcal{PROT}$$

betrachten, die einem Paar aus einem Angreifer A und einem durch diesen angreifbaren Protokoll S ein gestörtes Protokoll $\mathrm{aProt}(S, A)$ zuordnet. Dabei bezeichnet $\mathcal{PROT}$ die Menge aller Protokolle, $\mathcal{ADV}$ die Menge aller Angreifer.

2.13. Bemerkung Ein Protokoll kann auch von mehreren Angreifern attackiert werden, vorausgesetzt, die Mengen der jeweils angegriffenen Teilnehmer sind paarweise disjunkt. Man überzeugt sich leicht, daß die Reihenfolge, in der man die Angreifer zum System hinzunimmt, unerheblich ist, Angreifen ist kommutativ und assoziativ. ◇

2.14. Definition *Ein Angreifer* A *auf ein Protokoll* S *heißt* **einflußlos**, *wenn er das Verhalten des Systems nicht wesentlich stört, d.h.*

$$\bigforall_{\substack{x\in \mathrm{Dom}(S)\\ y\in \mathrm{Ran}(S)}} \bigforall_{x_A\in \mathrm{Dom}(e_A)} \pi_S(x,y) = \sum_{y_A\in \mathrm{Ran}(e_A)} \pi_{\mathrm{aProt}(S,A)}((x,x_A),(y,y_A)).$$

Die **Menge aller möglichen einflußlosen Angreifer** *für ein Protokoll ist*

$$\mathrm{eAdv}(\mathcal{P}) \;=\; \{A \in \mathrm{Adv}(\mathcal{P}) | A \textit{ ist einflußlos auf } \mathcal{P}.$$

Damit hat ein einflußlos angegriffenes Protokoll zwar nicht mehr dasselbe Laufzeit, aber dasselbe Ein-/Ausgabeverhalten für die ursprünglichen Teilnehmer.

2.15. Bemerkung Einflußlose Angreifer sind nicht zu verwechseln mit **passiven** Angreifern, die man in der Literatur oft findet. Diese erhalten nicht nur das globale Ein-/Ausgabeverhalten des Protokolls, sondern erzeugen sogar die gleiche Nachrichtenverteilung wie die durch sie angegriffenen Teilnehmer. Sie können höchstens neugierig sein und versuchen, soviel Information wie möglich aus den ihnen zugänglichen Daten zu extrahieren. Offenbar sind alle passiven Angreifer auch einflußlos. Wenn ein Teilnehmer die von ihm empfangenen Nachrichten in seine Ausgabe einschließt, sind alle einflußlosen Angreifer auch passiv gegenüber diesem Teilnehmer. ◇

2.16. Bemerkung In der kryptographischen Literatur findet man oft den Begriff des *Transkripts* oder der *Sicht* (engl. „view“) eines Teilnehmers bzw. des Angreifers. Dieses Transkript besteht aus allen Nachrichten, die ein Teilnehmer während der Protokollausführung zu sehen bekommt, sowie aus einer Niederschrift seiner zufallsgesteuerten Entscheidungsprozesse („Münzwürfe“).

Wir betrachten stattdessen nur die Ausgabe der Teilnehmer, da man diese immer so umformen kann, daß sie etwas entsprechendes als Ausgabe produzieren.

Der Unterschied zwischen den beiden Ansätzen wird deutlich bei der Untersuchung von Privatheit in Kapitel 3. Beim ersten Ansatz untersucht man, was ein Angreifer hätte lernen können, beim zweiten, was er tatsächlich aus seiner Sicht gelernt hat. ◇

Zunächst wollen wir jedoch einige Protokolle darstellen.

2.5 Beispiele für Protokolle

In diesem Abschnitt geben wir Beispielprotokolle für die beliebtesten kryptographischen Fragestellungen, nämlich die für Authentikationsverfahren grundlegenden interaktiven Beweissysteme, den klassischen Anwendungsfall der geheimen Nachrichtenübermittlung zwischen zwei Teilnehmern und das sehr allgemeingültige Prinzip der privaten Funktionsauswertung, auf das sich viele andere kryptographische Probleme reduzieren lassen.

2.5.1 Interaktive Beweissysteme

Bei interaktiven Beweissystemen überzeugt ein *Prover* P einen *Verifier* V, daß ein String x Element einer Sprache L ist. Dabei sind P und V $(\cdot,1)$-interaktive Turingmaschinen. In [SKRIPT, BIBUMETHITHI] wird ausführlich beschrieben, wie P und V zu einer probabilistischen Turingmaschine $S(P,V)$ zusammengesetzt werden. Dabei werden die Maschinen mit verbundenen Kommunikationskanälen solange parallel ausgeführt, bis beide halten.

Dies entspricht einem interaktiven System (C,V_0) mit Teilnehmern $P_1=(P,0)$, $P_2=(V,0)$, $C=\{P_1,P_2\}$ und $V_0=\{((P_1,1),(P_2,1)),((P_2,1),(P_1,1))\}$. Die Endmaschinen 0 sind zu P bzw. V passend zu wählen.

Bei interaktiven Beweissystemen sind nur zwei Arten von Angriffen möglich: Ersetzen des Provers P durch einen bösartigen Prover P^* und Ersetzen des Verifiers V durch einen bösartigen Verifier V^*, die dabei entstehenden Beweissysteme wurden in [SKRIPT, BIBUMETHITHI] $S(P^*,V)$ und $S(P,V^*)$ genannt.

Dies entspricht zwei Typen von Störern (P_A,C_A,ν_A): mit $P_A=(P^*,0)$, $C_A=\{P_1\}$ und $\nu_A=\{(1,(P_1,1))\}$ erhalten wir die Teilnehmermenge $C'=\{P_A,P_2\}$ und die neue Kanalzuordnung V' mit $V'(P_A,1)=V_0(P_1,1)=(P_2,1)$ und $V'(P_2,1)=(P_A,\nu_A^{-1}(V_0(P_2,1)))=(P_A,\nu_A^{-1}(P_1,1))=(P_A,1)$. Im zweiten Fall ist $P_A=(V^*,0)$, $C_A=\{P_2\}$ und $\nu_A=\{(1,(P_2,1))\}$, also $C'=\{P_1,P_A\}$ und $V'(P_1,1)=(P_A,1)$, $V'(P_A,1)=(P_1,1)$. Damit haben die angegriffenen Systeme dieselbe Struktur wie das nichtangegriffene System, eben $S(P^*,V)$ und $S(P,V^*)$.

Die Eingaben für diese Systeme sind (x,x) für $x\in\Sigma^*\supset L$, die Eingabefunktionen sind

$$e_{P_1}(x)=(c_x,c_\epsilon,\ldots,c_\epsilon,c_\square,c_\square,|x|,0,\ldots,0,1,0)$$

und

$$e_{P_2}(x)=(c_x,c_\epsilon,\ldots,c_\epsilon,c_\square,c_\square,|x|,0,\ldots,0,0,0),$$

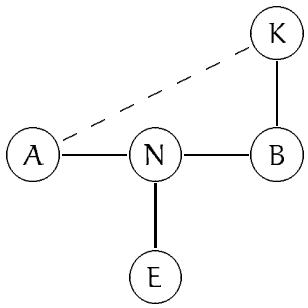
d.h. der Prover startet im Sendemodus, der Verifier im Empfangsmodus.

2.5.2 Kryptosysteme

Ein Kryptosystem besteht aus einem Kodier- und einem Dekodieralgorithmus sowie aus einer Schlüsselquelle. Diese ist in der Regel ein Algorithmus, der bei Eingabe eines Sicherheitsparameters, etwa der gewünschten Klartextlänge, einen Schlüssel ausgibt.

Sicherheit bedeutet hier Sicherheit gegen einen passiven Angriff auf das Transportmedium, also die Nachrichtenleitung. Damit muß die Leitung durch einen Protokollteilnehmer modelliert werden, der angegriffen werden kann. Weiter soll der Angreifer einflußlos sein, d.h. er darf die Übertragungseigenschaft des Netzknotens nicht verändern. Diese Einschränkung wird auf elegante Weise erfüllt, wenn neben der Senderin A(lice), dem Empfänger B(ob), dem Schlüsselgenerator K und dem N(etz) eine Lauscherin E(ve) hinzukommt, die ebenso wie A und B eine Verbindung mit N hat. Im ungestörten Fall ist $E = 0$, ansonsten beliebig. Das Netz gibt die durchzureichenden Nachrichten an die angreifbare Lauscherkomponente weiter. Damit ist die Passivität des Angreifers ins Netz eingebaut.

Eine Nachrichtenübertragung besteht aus drei Interaktionen (engl. „passes“) zwischen A und B. Zunächst schickt A an B den Sicherheitsparameter, den B an K weiterreicht. Bei symmetrischen Kryptosystemen schickt K den erzeugten Schlüssel an A und B, bei Public-Key-Kryptosystemen wird ein Schlüsselpaar erzeugt und an B gegeben, der den geheimen Teil behält und den öffentlichen Teil an A (und damit an E) schickt. In der dritten Interaktion schickt A den mithilfe des Schlüssels erzeugten Ziffertext an B, der mit seinem Schlüssel den Klartext rekonstruiert. Das zugehörige interaktive System mit alternierend-interaktiven Teilnehmern hat die Struktur



mit den Anfangsmaschinen

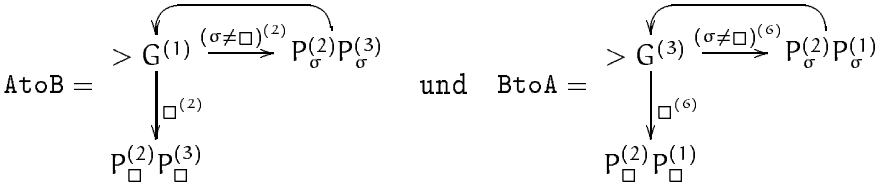
$$K = > \boxed{\texttt{GenerateKey}} \longrightarrow (\texttt{SendKeyToA} \longrightarrow)\texttt{SendKeyToB},$$

$$A = > \texttt{GetKeyFrom(K|B)} \longrightarrow \boxed{\texttt{Encode}} \longrightarrow \texttt{SendCodeToN},$$

$$B = > \texttt{GetKeyFromK} \longrightarrow \texttt{GetCodeFromN} \longrightarrow \boxed{\texttt{Decode}}.$$

Die Endmaschinen dieser drei Teilnehmer sind jeweils eine passende 0, die drei Kästen enthalten die drei Algorithmen eines Kryptosystems. Das Netz ist eine $(0,3)$-alternierend-interaktive Turingmaschine, Kanal 1 geht zu A, Kanal 2 zu E und Kanal

3 zu B. Mit den Maschinen



ist die Anfangsmaschine der Netzkomponente

$$\mathtt{N} \;=\; > \mathtt{AtoB} \longrightarrow \mathtt{BtoA} \longrightarrow \mathtt{AtoB}\,.$$

Die Lauscherin E hat Anfangs- und Endmaschine 0.

Ein Angreifer könnte nun das System allein dadurch blockieren, daß er auf mehr als die ihm zustehenden drei Nachrichten wartet. Dies tritt auch auf, wenn wir die Laufzeit des Angreifers beschränken, da wir Wartezeiten nicht zur Laufzeit zählen werden (siehe Abschnitt 2.7). Dieser unerwünschte Effekt kann mit einer geeigneten Endmaschine für das Netz verhindert werden, z.B. den „Sende-Nichts-und-Lies"-Maschinen, die eingehende Nachrichten überlesen und mit leeren Nachrichten antworten. Sie haben das Grundelement

$$\mathtt{SeNiL}^{(i)} = \;> \mathrm{P}_{\square}^{(i)} \longrightarrow \mathrm{G}^{(i)} \overline{\square}^{(w+2i)}\,,$$

das eine eine eventuell angefangene Nachricht beendet und eine eingegangene Nachricht überliest. Für c Kanäle führt man c Versionen dieser Maschine hintereinander aus, also

$$> \mathtt{SeNiL}^{(1)} \longrightarrow \cdots \longrightarrow \mathtt{SeNiL}^{(c)}\,.$$

Die Endmaschine $\mathtt{SeNiL}^{(2)}$ für das Netz versorgt einen Angreifer solange mit leeren Nachrichten bis dessen Rechenzeit verbraucht ist, dann hält das System. Damit ist jeder alternierend-interaktive Angreifer auf E mit beschränkter Laufzeit ein einflußloser Angreifer für dieses Protokoll, sogar passiv bezüglich A und B.

Die Eingabefunktionen müssen A im Sendemodus starten lassen, B, K und E im Empfangsmodus und N dazu passend. Die Senderin A bekommt als einzige eine nichtleere Eingabe, nämlich den zu sendenden Klartext m und den Sicherheitsparameter s. Wir werden dieses Protokoll als Grundlage der Untersuchung von Privatheit bei Kryptosystemen in Abschnitt 3.3.1 verwenden.

2.5.3 Private Funktionsauswertung

Bei privater Funktionsauswertung besteht die Kommunikationsstruktur aus je einem privaten Kanal für jedes Teilnehmerpaar und einem Broadcastkanal, d.h. es ist garantiert, daß alle Teilnehmer über diesen Kanal dieselbe Nachricht erhalten.

Dies ist notwendig, um die Fehlerschranke von $k/3$ für den verteilten Konsens zu durchbrechen, siehe auch [DOLEV82].

Da wir hier nur Angriffe auf „Endknoten“ und nicht auf die Kommunikationsstruktur betrachten, können wir der Einfachheit halber alle diese Kanalmaschinen *und* die privaten Kanäle zu einer einzigen Maschine zusammenfassen, dem Netz, mit der alle übrigen Maschinen über genau einen Kanal verbunden sind. Diese angreifbaren Endknoten, die eigentlichen Teilnehmer, senden Nachrichten einer speziellen Form, z.B. (Kanalnummer, Nachrichtentext), an das Netz, das diese entsprechend zustellt. Abweichungen von dieser Nachrichtenform können je nach Modell ignoriert oder an andere Maschinen gemeldet werden.

Durch die Kanalzuordnung erhält jeder Teilnehmer eine netzinterne Teilnehmernummer, nämlich die Nummer des Netz-Kanals, mit dem er verbunden wird. Diese ist unabhängig von der „äußeren“ Teilnehmernummer, die er bei der Konstruktion der Systemmaschine IS erhält. Das Netz kann damit auch Authentizität von Absendern garantieren, sofern das gewünscht ist.

Das Netz kann aber noch eine weitere, wichtige Randbedingung der Kommunikation erfüllen, nämlich *Synchronität*. Man möchte erreichen, daß die Kommunikation zwischen Teilnehmern in *Runden* abläuft, d.h. jeder Teilnehmer empfängt von den anderen Teilnehmern Nachrichten und generiert daraufhin eigene Nachrichten an andere Teilnehmer, dann warten alle Teilnehmer, bis diese Nachrichten zugestellt sind. Dies läßt sich mit dem Netz einfach realisieren.

Wir konstruieren eine Netzkomponente für diese Protokollklasse, eine (k, k)-alternierend-interaktive Turingmaschine, die private Kanäle und einen Broadcast-Kanal zur Verfügung stellt. Eine Erweiterung auf andere Kanaltypen ist einfach. Wir benutzen Sonderzeichen $1, \ldots, k$ und b in Σ, um Nachrichten an einzelne bzw. alle Teilnehmer zu kennzeichnen. Das Alphabet der eigentlichen Nachrichten ist dann

$$\Sigma_M = \Sigma \setminus \{1, \ldots, k, b, \#, \square\},$$

die an das Netz geschickten Nachrichten haben die Form

$$(\{1, \ldots, k, b\}\Sigma_M^*)^* \square$$

und die vom Netz zugestellten Nachrichten die Form

$$((b|\epsilon)\{1, \ldots, k\}\Sigma_M^*)^* \square .$$

Als erstes kopiert das Netz alle eingehenden Nachrichten auf ein entsprechendes Arbeitsband:

$$\mathtt{GetMsg}^{(i)} = \; > \square^{(i)}\, R^{(i)} \longrightarrow G^{(i)} \xrightarrow{(\sigma \neq \square)^{(w+2i)}} \sigma^{(i)} R^{(i)} \;,$$

$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Big\downarrow \square^{(w+2i)}$$

$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \square^{(i)} \longrightarrow L^{(i)} \; \overline{\square}^{(i)}$$

$$\texttt{GetAllMsgs} = > \texttt{GetMsg}^{(1)} \longrightarrow \cdots \longrightarrow \texttt{GetMsg}^{(k)} .$$

Da das Netz eine Runde immer mit den Köpfen auf dem Symbol $\Box$ auf den Arbeitsbändern beendet, ist es unbedenklich, mit dem Schreiben eines $\Box$ zu beginnen. Andererseits ist es notwendig, dies zu tun, um bei der ersten eingehenden Nachricht den Nachrichtenanfang wiederzufinden.

Zwischen Empfang und Weiterleiten der Nachrichten kann noch geprüft werden, ob die Nachrichten dem vorgegebenen Schema entsprechen und gegebenenfalls eine Sonderaktion durchgeführt werden, z.B. Leeren der Nachricht oder Ersetzen durch eine Broadcastnachricht „Fehler“ (der „Absender“ dieses Broadcasts ist der fehlgeleitete Teilnehmer). Wenn man leere Nachrichten oder mehrere Nachrichten an den gleichen Empfänger zuläßt, dann sind alle Strings der Form $(\epsilon|\{1,\ldots,k,b\})(\Sigma \setminus \{\#,\Box\})^*\,\Box$ gültige Nachrichten. Nach Konstruktion alternierend-interaktiver Turingmaschinen enthält eine Nachricht ohnehin keine Leerzeichen und wird mit einem $\Box$ beendet, daher muß lediglich getestet werden, ob die Nachricht mit einem der Zeichen aus $\{1,\ldots,k,b\}$ beginnt. Mit

$$\texttt{CheckMsg}^{(i)} = \; > R^{(i)} \circlearrowright \Sigma_M^{(i)} \quad \text{bzw.} \quad > R^{(i)} \xrightarrow{\Sigma_M^{(i)}} R^{(i)} \circlearrowright \overline{\Box}^{(i)}$$

und

$$\texttt{CheckAllMsgs} = > \texttt{CheckMsg}^{(1)} \longrightarrow \cdots \longrightarrow \texttt{CheckMsg}^{(k)} .$$

werden falsch gebildete Nachrichten entweder bis zum Erreichen einer sinnvollen Restnachricht gekappt oder ganz ignoriert.

Zuletzt werden die Nachrichten zugestellt:

$$\texttt{PutOneMsg}^{(i)} = \begin{array}{l} > 0 \xrightarrow{b^{(i)}} P_b^{(1)} \ldots P_b^{(k)} \longrightarrow R^{(i)} \xrightarrow{(\sigma\in\Sigma_M)^{(i)}} P_\sigma^{(1)} \ldots P_\sigma^{(k)} \\ \quad \Big\downarrow (j\in\{1,\ldots,k\})^{(i)} \\ P_i^{(j)} \longrightarrow R^{(i)} \xrightarrow{(\sigma\in\Sigma_M)^{(i)}} P_\sigma^{(j)} \end{array} ,$$

$$\texttt{PutMsgs}^{(i)} = \; > 0 \xrightarrow{\overline{\Box}^{(i)}} \texttt{PutOneMsg}^{(i)} \circlearrowright \overline{\Box}^{(i)} ,$$

$$\texttt{PutAllMsgs} = > \texttt{PutMsgs}^{(1)} \; \cdots \; \texttt{PutMsgs}^{(k)} \longrightarrow P_\Box^{(1)} \ldots P_\Box^{(k)} .$$

Eine Runde des Netzes ist damit

$$> \texttt{GetAllMsgs} \longrightarrow \texttt{CheckAllMsgs} \longrightarrow \texttt{PutAllMsgs} .$$

Runden sind garantiert, da diese Maschine im Get-Modus ist, solange noch mindestens ein Teilnehmer im Put-Modus ist, d.h. nach GetAllMsgs warten alle Teilnehmer auf das Eintreffen der für sie bestimmten Nachrichten.

Meist ist die Anzahl der notwendigen Runden eines Protokolls für private Funktionsauswertung aus der Eingabelänge abzulesen, man könnte also das Netz die Runden zählen lassen. Dies würde aber die Rundenzahl als Eingabe für das Netz erfordern. Ebenso ist denkbar, daß alle Teilnehmer an das Netz melden, wenn sie die Kommunikation für abgeschlossen halten.

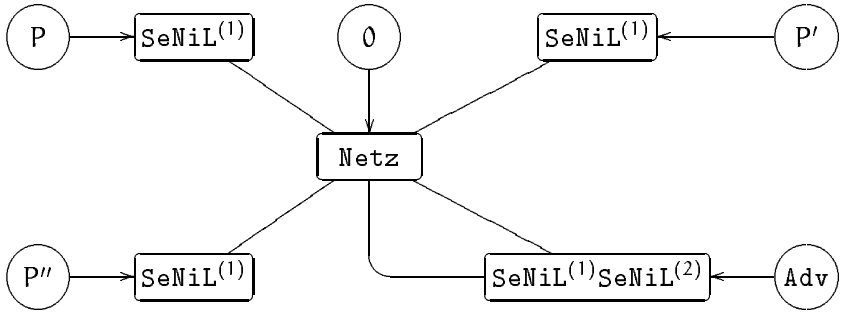
Eleganter ist es, die Abbrucheigenschaft eines interaktiven Systems als implizite Ende-Meldung zu verwenden: das System hält, wenn alle Endmaschinen erreicht sind. Wir verwenden die oben konstruierte Rundenmaschine als Endmaschine des Netzes mit 0 als Anfangsmaschine. Weiter fordern wir, daß alle übrigen Teilnehmer, einschließlich des Angreifers, die in Abschnitt 2.5.2 definierten SeNiL-Maschinen als Endmaschinen haben: ein gewöhnlicher Teilnehmer hat Endmaschine $\mathtt{SeNiL}^{(1)}$, ein Angreifer auf c Teilnehmer

$$> \mathtt{SeNiL}^{(1)} \longrightarrow \cdots \longrightarrow \mathtt{SeNiL}^{(c)} .$$

Das Netz erreicht so seine Endmaschine bereits im ersten Systemschritt und stellt solange Nachrichten zu, bis auch die letzte Teilnehmer-Anfangsmaschine beendet ist, dann hält das System.

Damit ist auch das Problem gelöst, daß ein Angreifer seine Nachrichten nicht abschließt und damit das System blockiert: wenn er nur endliche Rechenzeit zur Verfügung hat, wird er nach Verbrauch dieser Zeit von seiner Endmaschine abgelöst, die dann alle begonnenen Nachrichten abschließt, eingehende Nachrichten übergeht und genügend viele Leernachrichten produziert, damit der Rest des Systems weiterarbeiten und z.B. den Betrug feststellen kann.

Die eigentliche Kommunikationsstruktur wird also von den *Endmaschinen* gebildet, Teilnehmer- und Angreifer-Anfangsmaschinen sind nur Benutzer dieser Struktur, vergleichbar einer von Einflüssen des Angreifers geschützten Transportschicht.



Da das Netz auf allen Kanälen im Empfangsmodus beginnt, sollten es die übrigen Teilnehmer im Sendemodus tun.

Jedes Protokoll läßt sich so verändern, daß es — ähnlich wie das soeben beschriebene — eine sternförmige Struktur bildet. Dies wird im folgenden näher besprochen.

2.6 Sternförmige interaktive Systeme

Mit den aus zwei Bändern gebildeten Kommunikationskanälen haben Paare von interaktiven Turingmaschinen die Möglichkeit zur nichtabhörbaren, fehlerfreien Kommunikation, einen sogenannten **privaten Kanal**. Wie die vorangegangenen Beispiele zeigten, kann man andere Kanaltypen wie Broadcast-Kanäle, anonyme, abgehörte, gestörte oder verrauschte Kanäle über zusätzliche Teilnehmer realisieren, die z.B. eine eingehende Nachricht identisch an alle angeschlossenen Maschinen oder den eigentlichen Empfänger und den Abhörer weitergeben oder eine gestörte Version der Nachricht weiterleiten.

Den beiden letzten Beispielen ist die sternförmige Gestalt der Kommunikationsstruktur gemeinsam, das erste Beispiel kann man durch Einfügen einer zusätzlichen Komponente ebenfalls in diese Form bringen. Dies läßt sich mit beliebigen Protokollen tun, indem man alle nichtangreifbaren Komponenten in einer Netzkomponente zusammenfaßt, mit der die angreifbaren Teilnehmer über nur einen Kanal kommunizieren. Die verbleibenden Komponenten können wir mit alternierend-interaktiven Turingmaschinen realisieren:

2.17. Satz *Sei* S *ein beliebiges Protokoll, dann gibt es ein Protokoll* S' *mit alternierend-interaktiven Teilnehmer-Turingmaschinen und sternförmiger Kanalzuordnung, das* S *„simuliert", d.h.* $\pi_S = \pi_{S'}$ *und* $\pi_S^{(\ell)} = \pi_{S'}^{(k\ell)}$ *mit einer Konstanten* $k \in \mathbb{N}$.

Wir beweisen dies nicht explizit, die Grundidee ist, jeweils einen Schritt der B_i bzw. E_i zu simulieren und in einer Kommunikationsrunde die Inhalte der simulierten ursprünglichen Kommunikationsbänder zu aktualisieren. Sind B_i und E_i (w_i, c_i)-interaktive Turingmaschinen, dann sind B_i' und E_i' $(w_i+2c_i, 1)$-alternierend-interaktive Turingmaschinen mit jeweils zwei Köpfen auf den simulierten Kommunikationsbändern $w_i + j$, $j \in [\![1, 2c_i]\!]$.

Betrachten wir nun einen für Systeme immer wieder wichtigen Aspekt: die Laufzeit.

2.7 Laufzeit

Welche Laufzeit hat ein gegebenes interaktives System bzw. die ihm entsprechende probabilistische Turingmaschine? Und wie hängt diese Laufzeit von der Laufzeit der Teilnehmer ab? Was ist überhaupt die Laufzeit eines Teilnehmers? Diese Fragen sind nicht einfach zu beantworten. Zum einen wird die Endmaschine eines Teilnehmers bis zum Systemhalt wiederholt ausgeführt, und es ist a priori nicht klar, wie oft dies geschieht. Zum anderen wird das Verhalten einer interaktiven Turingmaschine von den Inhalten ihrer Empfangsbänder beeinflußt, diese wiederum werden von anderen Maschinen beschrieben, so daß die Laufzeit vom Verhalten des restlichen Systems mitbestimmt wird.

Bei alternierend-interaktiven Turingmaschinen aber wurde das Empfangsprimitiv $G^{(i)}$ so konstruiert, daß es wartet, bis der Empfangskopf i ein nichtleeres Symbol liest. Diese Warteschleifen sind zwar einerseits notwendig, weil die sendende Maschine möglicherweise mehr Schritte benötigt, um ein Nachrichtensymbol zu erzeugen, als die empfangende Maschine, um das zuvor gesendete Symbol zu verarbeiten. Andererseits wollen wir die so in Warteschleifen gemachten Schritte natürlich nicht zur Laufzeit einer interaktiven Turingmaschine hinzuzählen.

Da die Maschine *keine* Information darüber erhält, wieviele Warteschritte ausgeführt wurden, ist das Verhalten einer alternierend-interaktiven Turingmaschine unabhängig vom Zeitverhalten des übrigen Systems und nur abhängig von den empfangenen Nachrichten. Dies erlaubt uns, die Laufzeit einer alternierend-interaktiven Turingmaschine zu messen, indem wir die Empfangsbänder mit hinreichend langen Nachrichten vorbelegen, d.h. mit mehr Zeichen, als die Maschine konsumieren kann. Wenn wir über alle möglichen Empfangsbandinhalte maximieren, dann ist diese Laufzeit unabhängig vom interaktiven System, in dem die Maschine eingesetzt wird.

Sei also M eine (w, c)-interaktive Turingmaschine mit t Arbeitsbändern und Bandalphabet Σ. Für $\ell \in \mathbb{N}$ ist $D_M(\ell)$ die Menge aller Bandbeschreibungen von M, auf denen M noch mindestens ℓ Schritte machen kann, ohne ein $\#$ auf einem Empfangsband zu finden, d.h. die Menge aller $(b_1, \ldots, b_{t+2c}, h_1, \ldots, h_{w+2c}) \in D_M$, für die $b_{t+2i} = c_{x_i, s_i}$ ist mit $x_i \in (\Sigma \setminus \{\#\})^*$, $s_i \in \mathbb{Z}$ für alle $i \in [\![1, c]\!]$ und

$$\sum_{i \in [\![1,c]\!]} s_i + |x_i| - (h_{w+2i} + 1) \leq \ell.$$

Hier wurden die Tatsachen benutzt, daß der Inhalt eines Kommunikationsbandes, der von einer interaktiven Turingmaschine erzeugt wurde, keine leeren Symbole zwischen nichtleeren Symbolen enthält, und daß höchstens ein Empfangskopf pro Schritt bewegt werden kann, und zwar nur nach rechts.

Sind $c, c' \in T_\Sigma$ Bandinhalte von M, dann sagen wir, daß c **Bestandteil** von c' ist, wenn für alle $z \in \mathbb{Z}$ $c(z) = c'(z)$ ist oder $c(z) = \#$. Bei Bandbeschreibungen $d, d' \in D_M$ ist d Bestandteil von d', wenn d und d' gleich sind bis auf die Empfangsbandinhalte und die Empfangsbandinhalte von d Bestandteil der entsprechenden Empfangsbandinhalte von d' sind. Dann ist für $d \in D_M$

$$D_M(\ell, d) = \{d' \in D_M(\ell) | d \text{ ist Bestandteil von } d'\}.$$

2.18. Definition *Wir sagen, die (maximale)* **interaktive Laufzeit** *der (w, c)-alternierend-interaktiven Turingmaschine M bei Startkonfiguration (s, d) ist beschränkt durch $\ell \in \mathbb{N}$, kurz $IR_M((s, d)) \leq \ell$, wenn $R_M((s, d'))$ für alle $d' \in D_M(\ell, d)$ existiert und kleiner gleich ℓ ist.*

Wir wollen nun überlegen, was beim Ablauf eines interaktiven Systems passieren kann.

Alternierend-interaktive Teilnehmer bestehen aus zwei alternierend-interaktiven Turingmaschinen, von denen die erste genau einmal ausgeführt wird, die zweite nach Bedarf beliebig oft. Damit macht eine Laufzeitschranke nur Sinn für die Anfangsmaschine eines Teilnehmers.

Drei Typen von Teilnehmerschritten treten auf: Warteschritte, d.h. Schritte zur rechten 0-Maschine in $G^{(i)}$, „echte" Anfangsmaschinenschritte und „echte" Endmaschinenschritte. Das System macht in jedem Schritt einen Schritt pro Teilnehmer, es endet, wenn alle Endmaschinen erreicht werden. Damit gibt es folgende Typen von Systemschritten:

1. Mindestens eine Anfangsmaschine macht einen echten Schritt.
2. Es werden nur Warteschritte gemacht. Dann ändert sich die Bandbeschreibung nicht und es werden bis in alle Ewigkeit nur Warteschritte gemacht (Deadlock).
3. Alle noch aktiven Anfangsmaschinen warten und mindestens eine Endmaschine macht einen echten Schritt. Dann gibt es zwei Untermöglichkeiten:
   (a) Die noch laufenden Endmaschinen senden nie mehr ein Zeichen an eine der wartenden Anfangsmaschinen. Dann läuft das System ewig weiter.
   (b) Eine dieser Endmaschinen sendet irgendwann ein Zeichen, das auf irgendeinem Weg eine der wartenden Anfangsmaschinen erreicht.

Wenn wir garantieren können, daß die Möglichkeiten 2 und 3a nie erreicht werden, dann hält das System. Um eine Zeitschranke angeben zu können, müßten wir die maximale in Möglichkeit 3b auftauchende Zeit kennen. Da in diesem Fall alle Anfangsmaschinen warten, handelt es sich um eine Eigenschaft der Endmaschinen eines interaktiven Systems.

Wir definieren die **maximale Antwortzeit** $RT_{(C,V)}(\alpha)$ eines interaktiven Systems (C, V) bei Startkonfiguration α als die maximale Anzahl von Systemschritten vom Typ 3, die hintereinander auftreten kann.

Diese Antwortzeit kann sehr viel größer sein als die interaktive Laufzeit der Endmaschinen: eine Maschine, die einen binär dargestellten Zähler mit 0 vergleicht, alle Zählerstelen bei Gleichheit mit 1 initialisiert und ein Zeichen sendet, ihn ansonsten um 1 erniedrigt, hat eine interaktive Laufzeit, die linear in der Zählerlänge ist, als Endmaschine aber exponentielle Antwortzeit verursachen kann.

2.19. Satz *Sei* $S = (C, V, (e_P)_{P\in C}, (a_P)_{P\in C})$ *ein Protokoll mit alternierend-interaktiven Teilnehmern. Seien für ein* $x = (x_P)_{P\in C} \in \mathrm{Dom}(S)$ *die interaktiven Laufzeiten der Anfangsmaschinen aller* $P \in C$ *und die Antwortzeit der Endmaschinen endlich, dann sind die Längen aller endlichen Berechnungen von* $IS(C, V)$ *bei Eingabe* x *beschränkt durch*

$$RT_{(C,V)}(e(x)) \sum_{(B,E)\in C} IR_B(e_{(B,E)}(x_{(B,E)})).$$

Beweis: Da die Antwortzeit endlich ist, tritt der oben erwähnte Fall 3a nicht ein.

Die Empfangsprimitive verstecken Wartezeiten vor der Maschine, deren Aktionen sind also *unabhängig* von der Laufzeit der übrigen Maschinen, nur abhängig vom Nachrichteninhalt. Die interaktive Laufzeit der Anfangsmaschinen ist aber vom Nachrichteninhalt unabhängig.

Das System hält, wenn alle Endmaschinen erreicht sind, also wird in jedem Systemschritt mindestens ein Teilschritt einer Anfangsmaschine gemacht. Diese haben nur

$$\sum_{(B,E)\in C} IR_B(e_{(B,E)}(x_{(B,E)}))$$

echte Schritte (Fall 1).

Entweder es tritt ein Deadlock ein (Fall 2), dann endet die Berechnung nicht, oder es wird nach spätestens $RT_{(C,V)}(e(x))$ Schritten, in denen alle Anfangsmaschinen warten, wieder ein echter Schritt einer Anfangsmaschine ausgeführt (Fall 3b), dann gilt die im Satz genannte Schranke. □

Die Bestimmung der Antwortzeit ist nicht immer einfach, für zwei spezielle Typen von interaktiven Systemen wollen wir es hier tun.

2.20. Satz *Sei* $S = (C, V, (e_P)_{P\in C}, (a_P)_{P\in C})$ *ein interaktives System alternierend-interaktiver Teilnehmer mit Endmaschinen vom Typ* `SeNiL` *und azyklischer Kanalzuordnung. Sind die interaktiven Laufzeiten der Anfangsmaschinen für ein* $x = (x_P)_{P\in C} \in \mathrm{Dom}(S)$ *endlich, dann ist die Laufzeit von* $IS(C, V)$ *bei Eingabe* x *beschränkt durch*

$$k \sum_{(B,E)\in C} IR_B(e_{(B,E)}(x_{(B,E)}))$$

mit einer systemabhängigen, aber eingabeunabhängigen Konstanten $k \in \mathbb{N}$.

Beweis: Zwei Dinge sind zu zeigen: die Antwortzeit ist beschränkt und es tritt kein Deadlock auf.

Letzteres kann nicht passieren, da V azyklisch ist. Zu jedem Zeitpunkt muß jede Kette von „P′-wartet-auf-P“ enden, d.h. es gibt immer einen Teilnehmer, der nicht in einer Warteschleife ist und einen echten Schritt macht.

Eine Maschine vom Typ $\texttt{SeNiL}^{(i)}$ schreibt auf Kanal i ein □ und liest eine eingegangene Nachricht bis zum Ende. Sie benötigt dazu $k_1 + k_2\mathrm{length(Nachricht)}$ viele Schritte. Eine Endmaschine vom Typ `SeNiL` besteht aus je einem Exemplar von $\texttt{SeNiL}^{(i)}$ pro Kanal der Endmaschine, diese werden reihum ausgeführt. Damit laufen maximal $|V|$ Exemplare von $\texttt{SeNiL}^{(i)}$ gleichzeitig.

Eine wartende Anfangsmaschine erhält also auf einem Kanal von einer Endmaschine spätestens dann ein □, wenn alle im Netz laufenden Endmaschinen und damit höchstens $|V|$ Exemplare von $\texttt{SeNiL}^{(i)}$ einmal ausgeführt wurden, also nach höchstens

$$k_1|V| + k_2 \sum_{\text{Kanäle}} \text{length(Nachricht)}$$

Schritten. Nachrichten von Endmaschinen haben Länge 1, das ergibt eine maximale Antwortzeit von $(k_1 + k_2)|V|$. Nachrichten, die von Anfangsmaschinen geschrieben wurden, verlängern die Antwortzeit, aber um nicht mehr als insgesamt

$$k_2 \sum_{(B,E)\in C} IR_B(e_{(B,E)}(x_{(B,E)}))$$

Schritte während der gesamten Systemlaufzeit, da alle Anfangsmaschinen zusammen nicht mehr Zeichen senden können als sie echte Schritte machen. Daraus ergibt sich die im Satz genannte maximale Laufzeit mit $k = (k_1 + k_2)|V| + k_2$. □

2.21. Satz *Das in Abschnitt 2.5.3 beschriebene System für private Funktionsauswertung hat ebenfalls die Laufzeitschranke*

$$k \sum_{(B,E)\in C} IR_B(e_{(B,E)}(x_{(B,E)})).$$

Beweis: Hier wartet eine Anfangsmaschine höchstens nach dem Senden ihrer Nachrichten für die nächste Runde, und höchsten solange, bis das Netz alle Nachrichten eingesammelt, verarbeitet und wieder zustellt hat.

Wieder ist die Länge der vom Netz während der gesamten Laufzeit des Systems zu lesenden Nachrichten beschränkt durch die Summe der Laufzeiten der Anfangsmaschinen. Die Gesamtlänge der zuzustellenden Nachrichten kann um einen Faktor $|C|$ größer sein, da Broadcastnachrichten ja vervielfältigt werden müssen.

Das Netz selbst wartet auf höchstens $|V| = |C|$ Exemplare von $\texttt{SeNiL}^{(i)}$ in den Endmaschinen bereits fertiger Teilnehmer, die wieder über die gesamte Laufzeit summiert nur ein konstantes Vielfaches der Gesamtnachrichtenlänge an Schritten machen.

Damit gilt die im Satz genannte Abschätzung, die Konstante k ist ein Vielfaches der Teilnehmerzahl $|C|$. □

2.22. Definition *Ein Teilnehmer* (P, e_P, a_P) *mit* $P = (B, E)$ *heißt* **polynomiell** *auf* $L \subset \text{Dom}(e_P)$, *wenn es deterministische Turingmaschinen* M_{e_P} *und* M_{a_P} *und Konstanten* $k, k' > 0$ *gibt mit*

- M_{e_P} *berechnet* e_P *in polynomieller Zeit auf* L,
- M_{a_P} *berechnet* a_P *in polynomieller Zeit auf* D_P *und*

- $IR_B(e_P(x)) \leq |x|^k$ *für alle* $x \in L$ *mit* $|x| \geq k'$.

Ein Angreifer $(C_A, P_A, \nu_A, e_{P_\Lambda}, a_{P_\Lambda})$ *heißt polynomiell, wenn der angreifende Teilnehmer* $(P_A, e_{P_\Lambda}, a_{P_\Lambda})$ *polynomiell ist.*

2.23. Korollar *Das in Abschnitt 2.5.3 beschriebene Protokoll für private Funktionsauswertung ist polynomiell, wenn alle Teilnehmer polynomiell sind. Wird es von einem polynomiellen Angreifer attackiert, dann ist das angegriffene Protokoll immer noch polynomiell.*

Beweis: Man beachte, daß die Abschätzung in Satz 2.21 auch für angegriffene Systeme gilt, da die Angreiferendmaschine aus je einem Exemplar von $\texttt{SeNiL}^{(i)}$ pro angegriffenem Teilnehmer besteht. □

In diesem Kapitel wurde gezeigt, wie Protokolle aufgebaut sind, wie sie arbeiten und auf welche Weise sie gestört werden können. Die Sicherheit von Protokollen gegen Störungen wird im dritten Kapitel behandelt.

KAPITEL 3

Sicherheit

Nachdem wir im vorhergehenden Kapitel geklärt haben, was unter einem Protokoll zu verstehen ist, wollen wir nun einige wichtige Eigenschaften von Protokollen definieren. Ein Protokoll soll das tun, was man von ihm möchte (Korrektheit), dabei soll den Teilnehmern nicht mehr verraten werden, als zum Erreichen des Protokollzieles notwendig ist (Privatheit). Dies soll auch bei Störungen der Fall sein (Robustheit).

Es sind also zwei Aspekte zu betrachten: zum einen soll die *Information*, die ein Angreifer erhält, begrenzt sein, zum anderen soll er nur begrenzten *Einfluß* haben. Die Grundidee des hier präsentierten Sicherheitsansatzes ist es, zunächst die *Privatheit gegen einflußlose Angreifer* zu untersuchen und dann zu fordern, daß *alle Angreifer im wesentlichen einflußlos* sind.

Korrektheit ist eine Eigenschaft des ungestörten Protokolls: ein Protokoll kommt seiner Zielvorgabe hinreichend nahe.

Privatheit ist eine Eigenschaft des harmlos gestörten Protokolls: ein einflußloser Angreifer erfährt nicht mehr als die von ihm angegriffenen Teilnehmer, das Protokoll verrät im gestörten Fall nicht mehr als im ungestörten. Wir vergleichen hier also den gestörten Fall *nicht* mit dem Idealfall, d.h. ein Protokoll ist *für sich* privat, nicht privat für ein Ziel. Im letzten Fall ist ein Protokoll nur dann wirklich privat, wenn es für das vorgegebene Ziel korrekt ist. Im ersten Fall tut es das, was es tut, *privat*, unabhängig davon, worum es sich dabei handelt.

Robustheit ist eine Eigenschaft des beliebig gestörten Protokolls: alle Angreifer sind im wesentlichen einflußlos.

Die jeweilige Länge der vorhergehenden Beschreibungen der drei Eigenschaften spiegelt den Umfang der entsprechenden Abschnitte wider. Während Korrektheit und Robustheit relativ einfach zu fassen sind, gibt es für Privatheit zum Teil stark differierende Ansätze.

Wir gehen von verschiedenen Privatheitsansätzen aus, die im Laufe der letzten Jahre für unterschiedliche kryptographische Anwendungen entwickelt wurden, und zeigen, daß die Übertragungen dieser Definitionen auf die Privatheit allgemeiner Protokolle im wesentlichen äquivalent sind.

Im nachhinein ist dies nicht verwunderlich, da sich alle diese Ansätze auf äquivalente Spielarten von Sicherheit in der klassischen Informationstheorie zurückführen lassen.

Diese Varianten werden am Beispiel von Kryptosystemen in Unterabschnitt 3.3.1 vorgestellt, ihre Äquivalenz zeigen wir in Unterabschnitt 3.3.2.

Die algorithmischen Varianten dieser Privatheitsaspekte werden, nach einigen Vorüberlegungen, in Abschnitt 3.4 entwickelt und miteinander in Beziehung gesetzt.

Für die algorithmischen Sicherheitsbegriffe ist der Begriff der Ununterscheidbarkeit von Ensembles, d.h. Folgen von Wahrscheinlichkeitsverteilungen, von zentraler Bedeutung, wir werden ihn im ersten Abschnitt dieses Kapitels vorstellen.

3.1 Ensembles und Ununterscheidbarkeit

Eine **Quelle** oder **Wahrscheinlichkeitsverteilung** ist ein Paar (M, p), wobei M die abzählbare, meist endliche **Menge der Quellworte** ist und $p : M \to [0,1]$ ein **Wahrscheinlichkeitsmaß** auf M, d.h. $\sum_{x \in M} p(x) = 1$. Die **Menge aller Wahrscheinlichkeitsmaße** auf einer Menge M bezeichnen wir mit $\mathcal{W}(M)$.

Ein zumindest in der Stochastik weitverbreitetes Maß für den Unterschied zweier Wahrscheinlichkeitsverteilungen $U = (M, u)$ und $V = (M, v)$ auf derselben Menge M ist der **statistische Unterschied**

$$\Delta_s(U, V) = \tfrac{1}{2} \sum_{x \in M} |u(x) - v(x)|,$$

also die halbe $\|\cdot\|_1$-Norm der Differenz der Vektoren $(u(x))_{x \in M}$ und $(v(x))_{x \in M}$.

In der Kryptographie betrachtet man den **algorithmischen Unterschied** (siehe z.B. [GOLDR89]): man nimmt einen Algorithmus A, d.h. eine probabilistische Turingmaschine mit der in Abschnitt 1.6 beschriebenen Eingabe-Ausgabe-Konvention, und betrachtet sein Verhalten, also seine Ausgabeverteilung. Wenn diese weitgehend unabhängig davon ist, welche der beiden Verteilungen als Eingabeverteilung verwendet wurde, nennt man die Verteilungen A-ununterscheidbar, sie sind für A gleichwertig.*

3.1. Definition *Sei A ein Algorithmus, der bei Eingabe eines Strings $x \in M$ (o.B.d.A. $M \subset \Sigma^*$) mit Wahrscheinlichkeit $\pi_A(x, y)$ eine Ausgabe $y \in \{0,1\}$ erzeugt, dann ist der durch A gemessene* **algorithmische Unterschied** *zwischen den Quellen $U = (M, u)$ und $V = (M, v)$*

$$\Delta_A(U, V) = \Big| \sum_{x \in M} (u(x) - v(x)) \pi_A(x, 1) \Big|.$$

* Dieses Messen der Wirkung auf Dritte hat eine Entsprechung im deutschem Strafrecht: der Besitz täuschend imitierter Waffen ist ebenso strafbar wie der Besitz der entsprechenden echten Waffen, denn die beiden Waffentypen sind für einen damit Bedrohten ununterscheidbar, die Droh-Wirkung ist dieselbe.

Dies scheint zunächst eher unmotiviert. Betrachten wir aber die Quellen U_A und V_A auf der Menge $\{0,1\}$, wobei

$$u_A(y) = \sum_{x\in M} u(x)\pi_A(x,y) \quad \text{und} \quad v_A(y) = \sum_{x\in M} v(x)\pi_A(x,y)$$

ist, dann ist der statistische Unterschied dieser beiden Verteilungen

$$\Delta_s(U_A,V_A) \;=\; \tfrac{1}{2}\sum_{y\in\{0,1\}} |u_A(y)-v_A(y)| \;=\; \tfrac{1}{2}\sum_{y\in\{0,1\}} \Big|\sum_{x\in M}(u(x)-v(x))\pi_A(x,y)\Big|.$$

Nun ist

$$\begin{aligned}
&\Big|\sum_{x\in M}(u(x)-v(x))\pi_A(x,0)\Big| \\
&\quad= \Big|\sum_{x\in M}(u(x)-v(x))(1-\pi_A(x,1))\Big| \\
&\quad= \Big|-\sum_{x\in M}(u(x)-v(x))\pi_A(x,1) + \underbrace{\sum_{x\in M}u(x)}_{=1} - \underbrace{\sum_{x\in M}v(x)}_{=1}\Big| \\
&\quad= \Big|\sum_{x\in M}(u(x)-v(x))\pi_A(x,1)\Big|,
\end{aligned}$$

also $\Delta_s(U_A,V_A) = \Delta_A(U,V)$.

Die Gleichheit der beiden Summenterme können wir ein weiteres Mal nützen: angenommen, wir haben einen Algorithmus mit

$$\pi_A(x,1) = \begin{cases} 1 & u(x)\geq v(x), \\ 0 & \text{sonst}, \end{cases}$$

dann ist

$$\begin{aligned}
&\tfrac{1}{2}\sum_{x\in M}|u(x)-v(x)| \\
&\quad= \tfrac{1}{2}\Big(\sum_{\substack{x\in M\\ u(x)\geq v(x)}}(u(x)-v(x)) - \sum_{\substack{x\in M\\ u(x)<v(x)}}(u(x)-v(x))\Big) \\
&\quad= \tfrac{1}{2}\Big(\Big|\sum_{x\in M}(u(x)-v(x))\pi_A(x,1)\Big| + \Big|\sum_{x\in M}(u(x)-v(x))\pi_A(x,0)\Big|\Big) \\
&\quad= 2\tfrac{1}{2}\Big|\sum_{x\in M}(u(x)-v(x))\pi_A(x,1)\Big|,
\end{aligned}$$

also sogar $\Delta_s(U,V) = \Delta_A(U,V)$ für diesen Algorithmus.

Im allgemeinen gilt nur $\Delta_s \leq \Delta_A$, wie man leicht mit der Dreiecksungleichung nachrechnet.

In der Kryptographie will man in der Regel die Unterscheidungsalgorithmen auf polynomielle Laufzeit beschränken, man macht also asymptotische Aussagen. Daher reicht der Quellenbegriff nicht mehr aus, man muß Folgen von Quellen betrachten, sogenannte Ensembles.

3.2. Definition *Seien* Σ_M, Σ_I *Alphabete und* $\ell_M, \ell_I \in \mathbb{N}$. *Ein* **Ensemble** *über* $M \subset (\Sigma_M^*)^{\ell_M}$ *mit Parameter* $I \subset (\Sigma_I^*)^{\ell_I}$ *ist ein Paar* $U = (M, (p_i)_{i\in I})$, *wobei* $p_i \in \mathcal{W}(M)$ *ist für* $i \in I$.

Damit ist $U_i = (\{i\} \times M, \overline{p}_i)$ *mit* $\overline{p}_i(i, m) = p_i(m)$ *für jedes* $i \in I$ *eine Quelle.*

Ein Ensemble $U = (M, (p_i)_{i\in I})$ mit $M = M_1 \times \cdots \times M_m$ schreiben wir auch als $(U_1, \ldots, U_m)$. Ein solches m-Ensemble wird zu einem m'-Ensemble mit $m' < m$, wenn man über die Argumente $\{a_1, \ldots, a_{m-m'}\} \subset [\![1, m]\!]$ der Wahrscheinlichkeitsmaße p_i summiert und die entsprechenden Quellwortmengen $M_{a_1}, \ldots, M_{a_{m-m'}}$ wegläßt.

Aus zwei Ensembles $U = (M, (p_i)_{i\in I})$ und $V = (M', (p'_i)_{i\in I})$ mit demselben Parameter I kann man ein neues Ensemble $U \times V = (M \times M', (p_i \cdot p'_i)_{i\in I})$ bilden, wobei

$$p_i \cdot p'_i : M \times M' \to [0, 1] : (m, m') \mapsto p_i(m)p'_i(m')$$

das Produkt der Wahrscheinlichkeitsmaße ist. Man rechnet leicht nach, daß $p_i \cdot p'_i$ ein Wahrscheinlichkeitsmaß auf $M \times M'$ ist.

3.3. Definition *Ein Ensemble* $(M, (p_i)_{i\in I})$ *heißt* **polynomiell**, *wenn es Konstanten* $k, k' > 0$ *gibt mit*

$$\bigforall_{\substack{i\in I \\ |i|\geq k'}} \quad \bigforall_{m\in \mathbf{supp}(p_i)} \quad |m| \leq |i|^k.$$

Das heißt, die (in Abschnitt 1.6 definierte) Länge von Quellworten mit positiver Wahrscheinlichkeit ist polynomiell in der Länge des Quellenindexes.

3.4. Definition *Sei* $I \subset (\Sigma^*)^\ell$. *Eine Funktion* $f : I \to \mathbb{R}_0^+$ *heißt* **subpolynomiell**, *wenn*

$$\bigforall_{c>0} \quad \bigexists_{n_c\in\mathbb{N}} \quad \bigforall_{\substack{i\in I \\ |i|\geq n_c}} \quad f(i) \leq |i|^{-c}$$

ist. Dies schreiben wir kurz als $f \leq {}^1\!/_{\mathcal{P}}$. *Indizierung in* $\mathbb{N}$ *ist gleichbedeutend mit der Indexmenge* $I = \{1\}^*$.

3.5. Definition *Zwei Ensembles* $U = (M, (p_i)_{i\in I})$ *und* $V = (M, (p'_i)_{i\in I})$ *heißen* **statistisch ununterscheidbar**, *wenn die Funktion*

$$\Delta_s(U, V) : I \to \mathbb{R}_0^+ : i \mapsto \Delta_s(U_i, V_i)$$

subpolynomiell ist, sie heißen **algorithmisch ununterscheidbar**, *wenn für alle probabilistischen Algorithmen* $A \in \mathcal{PP}$ *die Funktion*

$$\Delta_A(U, V) : I \to \mathbb{R}_0^+ : i \mapsto \Delta_A(U_i, V_i)$$

subpolynomiell ist.

Manchmal läßt man für die algorithmische Ununterscheidbarkeit nur uniforme Algorithmen zu und bezeichnet das nichtuniforme Analogon als Schaltkreisununterscheidbarkeit. Man beachte, daß hier den Unterscheidern nur ein Quellwort zugänglich gemacht wird. Wenn man den Unterscheidern eine längere „Beobachtungszeit" zur Verfügung stellt und sie polynomiell viele Quellworte sehen läßt, erhöht das ihre Unterscheidungsfähigkeit nicht wesentlich.

3.6. Lemma *Wenn die Ensembles* $U = (M, (p_i)_{i\in I})$ *und* $V = (M, (p'_i)_{i\in I})$ *ununterscheidbar im Sinne von Definition 3.5 sind, dann können sie auch nicht durch einen Algorithmus unterschieden werden, der polynomiell lange Tupel von Quellworten erhält.*

Beweis: Sei A ein Algorithmus, der $\ell(i) \le |i|^{k_A}$ viele unabhängig gezogene Quellworte von U oder V erhält, dann ist der von A gemessene Unterschied

$$\begin{aligned}
&\Big| \sum_{x\in M^{\ell(i)}} (p_i(x) - p'_i(x))\pi_A((i,x),1) \Big| \\
&= \Big| \sum_{x_1,\ldots,x_{\ell(i)}\in M} \Big(\prod_{j=1}^{\ell(i)} p_i(x_j) - \prod_{j=1}^{\ell(i)} p'_i(x_j)\Big)\pi_A((i,x_1,\ldots,x_{\ell(i)}),1) \Big| \\
&= \Big| \sum_{x_1,\ldots,x_{\ell(i)}\in M} \sum_{k=1}^{\ell(i)} \prod_{j=1}^{k-1} p_i(x_j)(p_i(x_k) - p'_i(x_k)) \prod_{j=k+1}^{\ell(i)} p'_i(x_j)\pi_A((i,x_1,\ldots,x_{\ell(i)}),1) \Big| \\
&\le \sum_{k=1}^{\ell(i)} \Big| \sum_{x_1,\ldots,x_{\ell(i)}\in M} \prod_{j=1}^{k-1} p_i(x_j)(p_i(x_k) - p'_i(x_k)) \prod_{j=k+1}^{\ell(i)} p'_i(x_j)\pi_A((i,x_1,\ldots,x_{\ell(i)}),1) \Big|.
\end{aligned}$$

Jeder dieser $\ell(i)$ Summanden ist schreibbar als

$$\Big| \sum_{x_1\in M} p_i(x_1) \cdots \sum_{x_{k-1}\in M} p_i(x_{k-1}) \sum_{x_{k+1}\in M} p'_i(x_{k+1}) \cdots \sum_{x_{\ell(i)}\in M} p'_i(x_{\ell(i)}) \sum_{x_k\in M} (p_i(x_k) - p'_i(x_k))\pi_A((i,x_1,\ldots,x_{\ell(i)}),1) \Big|.$$

Wenn wir ein Tupel $h_{i,k} = (x_1^{(k)}, \ldots, x_{k-1}^{(k)}, x_{k+1}^{(k)}, \ldots, x_{\ell(i)}^{(k)})$ wählen, das die letzte Summe maximiert, und den Betragsstrich bis zur letzten Summe schieben, dann fallen die ersten $\ell(i) - 1$ Summen weg und der k-te Summand ist kleiner gleich

$$\Big| \sum_{x_k \in M} (p_i(x_k) - p_i'(x_k)) \pi_{A_k}((i, x_k, h_{i,k}), 1) \Big|.$$

Dabei sortieren die Algorithmen A_k, $k \in [\![1, \ell(i)]\!]$, nur ihre Eingaben um und arbeiten dann wie A.

Sei $\mathrm{kmax}(i)$ dasjenige $k \in [\![1, \ell(i)]\!]$, für das der vorstehende Ausdruck maximal ist, und $\mathrm{hmax}(i) = h_{i,\mathbf{kmax}(i)}$ das zugehörige maximierende Tupel, dann ist der von A gemessene Unterschied kleiner gleich

$$\ell(i) \Big| \sum_{x \in M} (p_i(x) - p_i'(x)) \pi_{A_{\mathbf{kmax}(i)}}((i, x, \mathrm{hmax}(i)), 1) \Big| \;\leq\; |i|^{k_A} \Delta_{(A', \mathbf{hmax})}(U_i, V_i),$$

wobei (A', hmax) ein in i nichtuniformer Algorithmus ist. Der von (A', hmax) gemessene Unterschied ist nach Voraussetzung subpolynomiell. Da k_A unabhängig von i ist, ist der von A gemessene Unterschied ebenfalls subpolynomiell. □

3.7. Bemerkung Dieses Lemma legt nahe, daß „subpolynomiell" der größtmögliche Begriff von Kleinheit bzw. Vernachlässigbarkeit ist, der für polynomiell beschränkte Algorithmen sinnvoll ist.

Wir werden in Lemma 3.21 (siehe Abschnitt 3.4.8) zeigen, daß man einen Algorithmus, der einen Unterschied von ε zwischen zwei Verteilungen mißt, in einen Algorithmus umbauen kann, der mit Erfolg $1/2 + \varepsilon/2$ vorhersagt, aus welcher Verteilung ein ihm präsentiertes Quellwort stammt. Einen nicht subpolynomiellen Vorhersageerfolg eines polynomiellen Algorithmus, d.h. asymptotisch größer als $|i|^{-c}$ für ein festes c, könnte man mit einem polynomiellen Mehrheitsentscheid zum konstanten Erfolg machen und diesen mit einem weiteren polynomiellen Mehrheitsentscheid exponentiell nahe an 1 bringen, d.h. die Verteilungen könnten von einem polynomiellen Algorithmus mit fast völliger Sicherheit unterschieden werden.

Nach Lemma 3.6 ist „subpolynomiell" stabil gegen *jede* derartige Konstruktion und damit brauchbar; jedes Kleinheitsmaß, das nicht mindestens subpolynomiell ist, ist nach der vorstehenden Überlegung unbrauchbar. ◇

3.2 Korrektheit

Wenn ein Protokoll das tut, was es soll, sollte man es sicherlich *korrekt* nennen. Dabei stellt sich die Frage, was ein Protokoll tut. Zunächst einmal ordnet es jeder Eingabe eine Verteilung auf den Ausgaben zu. Maximal kann es daher eine vorgegebene Zuordnung von Ausgabe-Verteilungen zu Eingaben realisieren. Da wir

nur asymptotische Aussagen machen wollen, muß diese Verteilung nicht exakt realisiert werden, sondern darf einen vernachlässigbaren statistischen Unterschied von der Zielverteilung haben. Da *wir* wollen, daß das Protokoll wie vorgeschrieben handelt, fordern wir hier einen vernachlässigbaren echten Fehler. Bei der anschließenden Privatheitsuntersuchung werden wir ausschließlich vernachlässigbare algorithmische Unterschiede fordern, da diese für den *Angreifer*, der ja ein Algorithmus ist, eben keine Unterschiede sind.

In der Literatur findet man durchaus ähnliche Korrektheitsbegriffe für (deterministische) Algorithmen, vergleiche z.B. [WaldWalt, Abschnitt 4.5.10]. Dort ist eine Problemspezifikation eine Menge $Z \subset X \times Y$ von Ein-/Ausgabepaaren. Ein Algorithmus ist korrekt auf einer Menge von Eingaben X, wenn alle Paare (x, y) von Eingabe $x \in X$ und zugehöriger Ausgabe $y \in Y$ in Z liegen. Üblicherweise soll zu einer Eingabe genau eine Ausgabe korrekt sein, die Spezifikation Z ist also eine Funktion von der Eingabe- in die Ausgabemenge.

3.8. Definition *Sei* $S_0 = (C, V, (e_P)_{P\in C}, (a_P)_{P\in C})$ *ein Protokoll. Eine* **Eingabefamilie** *mit* **Sicherheitsparameter** I *für* S_0 *ist ein Paar* $X_I = (M_X, (X_i)_{i\in I})$, *dabei sind* I *und* $M_X = \bigtimes_{P\in C} M_P$ *Sprachen, für alle* $i \in I$ *ist* $X_i \subset M_X$ *nichtleer und für alle* $x = (x_P)_{P\in C} \in X_i$ *ist* $((i, x_P))_{P\in C} \in \mathrm{Dom}(S_0)$.

Ein Element $(i, x) \in I \times M_X$ *mit* $x \in X_i$ *ist eine* **eigentliche Eingabe** *für* S_0, *die* **Menge aller eigentlichen Eingaben** *zur Eingabefamilie* X_I *ist* $\hat{X}_I$.

Es werden also die $|C|$ Komponenten des eigentlichen Eingabewertes $x = (x_P)_{P\in C}$ mit dem Wert i des Sicherheitsparameters verknüpt zu einem Element $((i, x_P))_{P\in C}$ des Definitionsbereiches der Eingabefunktion des Protokolls.

3.9. Definition *Sei* S_0 *ein Protokoll. Eine* **Zielspezifikation** *für* S_0 *auf einer Eingabefamilie* X_I *ist eine Funktion* $z : \hat{X}_I \to \mathcal{W}(\mathrm{Ran}(S_0))$, *die jedem* (i, x) *aus* $\hat{X}_I$ *ein Wahrscheinlichkeitsmaß* $z(i, x)$ *auf* $\mathrm{Ran}(S_0)$ *zuordnet.*

Bis auf einen subpolynomiellen statistischen Unterschied soll das Protokoll diese Zielverteilung erreichen.

3.10. Definition *Ein Protokoll* $S_0 = (C, V, (e_P)_{P\in C}, (a_P)_{P\in C})$ *ist* **korrekt** *für die Zielspezifikation* z *auf der Eingabefamilie* $X_I = (M_X, (X_i)_{i\in I})$, *wenn für alle* $c > 0$, *alle hinreichend großen* $|i|$ *und alle* $(x_P)_{P\in C} \in X_i$

$$\sum_{y\in \mathrm{Ran}(S_0)} |\pi_{S_0}(((i, x_P))_{P\in C}, y) - z(i, (x_P)_{P\in C})(y)| \;\leq\; |i|^{-c}$$

ist.

Die „bewiesene Korrektheit“ eines Protokolls für eine Aufgabe kann also nur bedeuten, daß man eine Zielspezifikation vorgibt, von der man glaubt, sie bedeute das, was man von einem Protokoll haben will, und daß man dann nachrechnet, daß ein gegebenes Protokoll diese Spezifikation erfüllt. Ob eine Zielspezifikation wirklich das bedeutet, was man sich von ihr erhofft, ist nicht beweisbar, der Schritt von der wirklichen Welt zum Modell ist nicht durch Beweise, sondern durch überzeugende Argumente zu machen.

3.3 Informationstheoretische Privatheit

Privatheit ist einfach zu beschreiben: kein Angreifer soll mehr herausbekommen als unbedingt nötig ist bzw. als ihm zusteht. Die Formalisierung dieser Aussage ist nicht ganz so einfach, sie ist das Ziel dieses und des folgenden Abschnitts.

Für diese Formalisierung gibt es in der Kryptographie viele Ansätze, die meisten basieren auf der Ununterscheidbarkeit oder Identität von Protokolltranskripten auf bestimmten Eingaben [Beaver89-3, ChorKush2, FeiSha90, Kush], der Unmöglichkeit, auf speziellen Eingabeverteilungen mit probabilistischen Algorithmen die Eingabe aus dem Transkript vorhersagen zu können [MicRackSlo88], der Unberechenbarkeit von Funktionen der Eingabe aus dem Angreifertranskript [MicRackSlo88, Yao82, Yao86, Yao88] oder der ununterscheidbaren Simulierbarkeit der Transkriptverteilungen aus unvollständiger Information [Beaver89, FeiSha90, GalilHabYung87]. (Auf das Verhältnis von Transkript und Angreiferausgabe gingen wir schon in Bemerkung 2.16 ein.)

Neben diesen aus der Anschauung motivierten Ansätzen gibt es für klassische Kryptosysteme die Privatheitsdefinition von C.E. Shannon und ihre algorithmische Adaption von A.C.-C. Yao [Yao82, Yao88], die fordern, daß die (algorithmische) Entropie der Klartextverteilung (weitgehend) identisch ist mit der (algorithmischen) Entropie der bedingten Klartextverteilung, wenn man die jeweils zugehörigen Ziffertexte kennt,

Wir zeigen in diesem Abschnitt, daß für Kryptosysteme die informationstheoretischen Konzepte hinter den gängigen Ansätzen äquivalent zum Privatheitsbegriff von Shannon sind. Danach beschäftigen wir uns mit dem Konzept der Enthüllung von Information, auf das wir uns bei der Erweiterung der Privatheitsdefinition auf allgemeine Protokolle in Abschnitt 3.4 stützen werden.

3.3.1 Privatheit von Kryptosystemen

Bei Kryptosystemen betrachtet man sinnvollerweise lediglich passive Angreifer gegen die Nachrichtenleitung, da ein Angreifer gegen Teilnehmer den zu übermittelnden Klartext und den verwendeten Schlüssel ohnehin erführe. Ein aktiver Angriff gegen die Nachrichtenleitung führt in das Gebiet fehlerkorrigierender Kodes.

Ein **konventionelles Kryptosystem** besteht aus der Klartextverteilung M, der Schlüsselverteilung K, den deterministischen Kodier- bzw. Dekodieralgorithmen E und D mit $D(E(m,k),k) = m$ für alle Klartexte m und Schlüssel k. Daraus leitet sich die Ziffertextverteilung $C = E(M,K)$ ab, deren Wahrscheinlichkeitsmaß ist

$$p(c) = \sum_{\substack{m,k \\ E(m,k)=c}} p(m)p(k).$$

Ein **Public-Key-Kryptosystem** besteht aus der Klartextverteilung M, probabilistischen Kodier- bzw. Dekodieralgorithmen E und D mit $D(c) = m$ für alle $m \in M$ und alle $c \in \mathrm{supp}(E(m))$. Die hiervon abgeleitete Ziffertextverteilung $C = E(M)$ hat das Wahrscheinlichkeitsmaß

$$p(c) = \sum_{m} p(m)\pi_E(m,c).$$

Das zugehörige interaktive System wurde bereits in Abschnitt 2.5.2 beschrieben. Teilnehmer A(lice) erhält als Eingabe einen zu übermittelnden Klartext, dazu im Falle eines symmetrischen Kryptosystems einen Schlüssel von einer Schlüsselquelle. Denselben Schlüssel erhält Teilnehmer B(ob), die Lauscherin E(ve) erhält keine Eingabe. Bei einem Public-Key-Kryptosystem erhält B von der Schlüsselquelle ein Paar aus öffentlichem und privatem Schlüssel, den öffentlichen Schlüssel sendet B an A (und damit an E).

Privatheit kann hier nur eines bedeuten: *Der Angreifer kann den Klartext nicht herausfinden.* Warum fordern wir nicht, daß es unmöglich ist, den Schlüssel herauszufinden? Ganz einfach, weil bei Kenntnis des Schlüssels der Klartext offenbart wird, also Geheimhaltung des Klartextes Geheimhaltung des Schlüssels beinhaltet. Außerdem ist es denkbar, daß der Schlüssel geheim bleibt und die Klartexte trotzdem aufgedeckt werden: bei digitalen Unterschriften sind (fehlerhafte) Verfahren bekannt, bei denen zwar der Unterschriftenschlüssel geheim bleibt, Unterschriften aber dennoch gefälscht werden können, siehe z.B. [Goldr89-2].

Wie könnte nun eine Formalisierung dieses Wunsches aussehen? Es gibt im wesentlichen folgende Ansätze:

Ansatz 1 Es ist nicht entscheidbar, aus welchem von zwei gegebenen Klartexten ein gegebener Ziffertext entstand.

Ansatz 2 Jede Funktion des Klartextes ist nur aus dem Ziffertext nicht besser berechenbar als ohne den Ziffertext.

Ansatz 3 Die Verteilung der Ziffertexte ist simulierbar.

Ansatz 4 Jeder Ziffertext ist für alle Klartexte gleich wahrscheinlich.

Ansatz 5 Die Entropie der Klartextverteilung ist identisch mit der Entropie der Klartextverteilung relativ zur Ziffertextverteilung.

Die **Entropie** einer Quelle ist ein Maß für den mittleren Informationsgehalt eines Quellwortes. Hinter der Definition der klassischen Entropie steht folgender Gedanke: Teile M in gleichwahrscheinliche Mengen M_0, M_1, diese in gleichwahrscheinliche Mengen M_{00}, M_{01}, M_{10}, M_{11} usw. bis jede Menge nur noch ein Quellwort enthält. Dies ergibt eine präfixfreie Binärkodierung $c : M \to \{0,1\}^*$ vermöge $m \in M_{c(m)}$. Dann ist $p(m) \approx 2^{-|c(m)|}$, und die erwartete Kodelänge

$$\sum_{m \in M} p(m)|c(m)| \approx -\sum_{m \in M} p(m) \log_2 p(m) = H(M)$$

ist die erwartete Anzahl von aHa-Effekten bis das Quellwort festgelegt ist, eben die Entropie H.

Die Anwendung des gleichen Verfahrens auf M^k, d.h. Blöcke von k Quellworten, ergibt eine genauere Approximation von $H(M^k) = kH(M)$. Dies ist die Aussage des **1. Satzes von** SHANNON:

$$\lim_{k \to \infty} \min_{\substack{c \text{ binärer} \\ \text{Kode für } M^k}} \frac{1}{k} \sum_{m \in M^k} p(m)|c(m)| = H(M).$$

Von **bedingter Entropie** spricht man, wenn man eine Paarquelle (M, C, p) betrachtet, also Mengen M und C und ein Wahrscheinlichkeitsmaß $p : M \times C \to [0,1]$. Mit $p(c) = \sum_{m \in M} p(m,c)$ ist $p(m|c) = p(m,c)/p(c)$ für alle $c \in C$ ein Wahrscheinlichkeitsmaß auf M, also $(M, p(\cdot|c))$ eine Quelle M_c. Die bedingte Entropie $H(M|C)$ ist die erwartete Entropie der Quellen M_c, gemittelt über die Wahrscheinlichkeit der $c \in C$,

$$\begin{aligned} H(M|C) &= \sum_c p(c) H(M_c) = -\sum_c p(c) \sum_m p(m|c) \log p(m|c) \\ &= -\sum_{m,c} p(m,c) \log p(m|c). \end{aligned}$$

3.3.2 Relation der Privatheitsansätze

Die fünf genannten Privatheitsansätze sind nicht so verschieden, wie sie durch ihre Formulierungen zunächst erscheinen.

Die Entropiedifferenz in Ansatz 5 schreibt sich ausführlich als

$$\begin{aligned} H(M) - H(M|C) &= -\sum_{m,c} p(m,c)\Big(\log p(m) - \log p(m|c)\Big) \\ &= \sum_{m,c} p(m,c) \log\Big(\frac{p(m,c)}{p(m)p(c)}\Big) \end{aligned}$$

und ist genau dann gleich 0, wenn alle Argumente des Logarithmus gleich 1 sind, also $p(m,c) = p(m)p(c)$ für alle m und c. Dies ist gleichbedeutend mit Ansatz 4, $p(c|m) = p(c)$ für alle m und c: jeder Ziffertext ist für alle Klartexte gleich wahrscheinlich.

Daraus folgt, daß $p(c|m) = \pi_E(m,c)$ unabhängig ist von der Klartextverteilung M, H-Privatheit auf M bedeutet H-Privatheit auf allen Klartextverteilungen M' mit $\mathrm{supp}(M') = \mathrm{supp}(M)$. Also ist die Ziffertextverteilung simulierbar (Ansatz 3), da eine Kenntnis des Klartextes nicht notwendig ist.

Damit wiederum ist jede Funktion des Klartextes aus dem Ziffertext nicht besser berechenbar als ohne den Ziffertext (Ansatz 2), man koppelt den Simulator für die Ziffertextverteilung und einen Berechner, der einen Ziffertext benötigt, und erhält einen Berechner, der ohne Ziffertextkenntnisse arbeitet.

Ist der erste Ansatz nicht erfüllt, dann gibt es Klartexte m_0 und m_1 mit unterscheidbaren Ziffertexten, d.h. der Vorhersage-Erfolg eines Algorithmus ist größer als $1/2$. Betrachtet man eine spezielle Klartextverteilung mit $p(m_0) = 1/2 = p(m_1)$, dann ist $f(m) = m$ auf dieser Verteilung mit Ziffertextkenntnis besser berechenbar als ohne.

Formalisieren wir den ersten Ansatz, erhalten wir

$$\forall_A \; \forall_{m_0,m_1} \; \sum_{i\in\{0,1\}} \tfrac{1}{2} \sum_c \pi_E(m_i,c)\pi_A(c,m_i) = \tfrac{1}{2},$$

d.h. der Vorhersage-Erfolg von A ist $1/2$. Damit ist die Danebenrate-Wahrscheinlichkeit ebenfalls $1/2$, die Differenz dieser Wahrscheinlichkeiten ist

$$\begin{aligned} 0 &= \sum_{i\in\{0,1\}} \tfrac{1}{2} \sum_c \pi_E(m_i,c)\pi_A(c,m_i) \; - \sum_{i\in\{0,1\}} \tfrac{1}{2} \sum_c \pi_E(m_i,c)\pi_A(c,m_{1-i}) \\ &= \sum_{i\in\{0,1\}} \tfrac{1}{2} \sum_c \pi_E(m_i,c)\Big(\pi_A(c,m_i) - \pi_A(c,m_{1-i})\Big) \\ &= \tfrac{1}{2} \sum_c \Big(\pi_E(m_0,c) - \pi_E(m_1,c)\Big)\Big(\pi_A(c,m_0) - \pi_A(c,m_1)\Big). \end{aligned}$$

Die Terme der Summe sind 0, wenn A keinem der beiden Klartexte den Vorzug geben kann und rät (zweite Klammer gleich 0) oder wenn ein Ziffertext für beide Klartexte gleichwahrscheinlich ist (erste Klammer gleich 0). Ist A_c ein Algorithmus, der für $c' \neq c$ rät und bei c immer m_0 ausgibt, dann muß c gleichwahrscheinlich sein für beide Klartexte, damit die Summe 0 sein kann. Dies gilt aber für alle Algorithmen A, insbesondere für alle A_c, $c \in C$, es folgt Ansatz 4.

Damit sind alle fünf Sicherheitsansätze äquivalent.

3.3.3 Enthüllen von Information

Daß nicht immer alle Information, die man geheimhalten möchte, auch geheimgehalten werden kann, ist allgemein bekannt. So liest man in [SPOERL63, Seite 352]:

> Frage: Soll Trude geopfert werden? Elisabeth ... sammelt die Zettel ... und öffnet mit zitternder Hand.
>
> Der erste: Ja. — Allgemeine Entrüstung.
> Der zweite: Ja. — Zweite Entrüstung.
> Der dritte: Ja. — Dritte Entrüstung.
> Weiter Ja und weiter Entrüstung. Bis zum letzten Ja und zur letzten Entrüstung.
>
> Das hat man von der geheimen Abstimmung! An diese Möglichkeit hat niemand gedacht. Jeder hat auf ein paar Nein-Stimmen der anderen gehofft, hinter der man sich hätte verstecken können. Eine einzige Nein-Stimme hätte genügt, um jeden zu decken.

Die in [SPOERL63] genannte „Lösung", daß jeder drei Stimmen abgibt, zwei entsprechend, eine entgegen seiner Entscheidung, ist mit demselben, nicht zu behebenden Mangel behaftet und eröffnet nur die Möglichkeit zum Betrug.

Der in [ABAFEIKIL89] definierte Begriff der Informationsenthüllung (engl.: „information leaking") versucht, den Umfang der unvermeidlich enthüllten Information zu fassen:

Sei $(X,Y) = (M_X \times M_Y, p(\cdot,\cdot))$ eine Paarquelle und $E : M_X \to M_E = E(M_X)$ eine beliebige Funktion. Daraus wird eine 3-Quelle $(X,Y,E) = (M_X \times M_Y \times M_E, p_E)$ mit $p_E(x,y,e) = p(x,y)\delta(E(x),e)$, wobei $\delta(x,y) = 1 \iff x = y$ die KRONECKERsche Deltafunktion ist.

3.11. Definition *Wir sagen,* Y **enthüllt höchstens** E *über* X*, wenn für alle* $e \in M_E$ *die bedingten Wahrscheinlichkeitsverteilungen* X_e *und* Y_e *unabhängig sind, d.h.*

$$\bigvee_{e \in M_E} \bigvee_{(x,y) \in M_X \times M_Y} p(x,y|e) = p(x|e)p(y|e).$$

Eine in [ABAFEIKIL89] nicht erwähnte Eigenschaft macht das Prinzip der Informationsenthüllung besonders natürlich: *Enthüllen ist monoton.* Wir sagen, eine Funktion E ist **informationsärmer** als eine Funktion E', kurz $E \preceq E'$, wenn es eine Funktion g gibt mit $E = g \circ E'$, d.h. der Wert von E ist aus dem Wert von E' ablesbar, E' hat nicht mehr Information über den Eingabewert verloren als E. Offenbar ist „$\preceq$" reflexiv und transitiv, konstante Funktionen sind informationsärmer, die Identität und alle bijektiven Funktionen informationsreicher als jede andere Funktion.

3.12. Satz *Enthüllen ist monoton, d.h. wenn* Y *höchstens* E *über* X *enthüllt und* $E \preceq E'$ *ist, dann enthüllt* Y *höchstens* E' *über* X*.*

Beweis: Sei wieder $p_E(x,y,e) = p(x,y)\delta(E(x),e)$ die von E induzierte Wahrscheinlichkeitsverteilung für (X,Y,E). Dann ist $p_E(x,e) = p(x)\delta(E(x),e)$, $p_E(y,e) = \sum_{x\in E^{-1}(e)} p(x,y)$ und $p_E(e) = \sum_{x\in E^{-1}(e)} p(x)$, dasselbe für E' und e'. Weiter ist

$$p_E(x,y|e) = \begin{cases} \frac{p(x,y)}{p_E(e)} & x \in E^{-1}(e) \\ 0 & \text{sonst} \end{cases} \quad \text{und} \quad p_E(x|e) = \begin{cases} \frac{p(x)}{p_E(e)} & x \in E^{-1}(e) \\ 0 & \text{sonst} \end{cases}.$$

Nach Voraussetzung ist $E = g \circ E'$ und

$$p_E(x,y|e) = p_E(x|e)p_E(y|e) \qquad \text{für alle } x,y \text{ und } e. \tag{3.1}$$

Für $e' \in g^{-1}(e)$ und $x \in E'^{-1}(e')$ ist dann $E(x) = g(E'(x)) = g(e') = e$ und

$$p_{E'}(x,y|e') = \frac{p_E(e)}{p_{E'}(e')}p_E(x,y|e) = \frac{p_E(e)}{p_{E'}(e')}p_E(x|e)p_E(y|e) = p_{E'}(x|e')p_E(y|e),$$

bleibt also $p_E(y|e) = p_{E'}(y|e')$ zu zeigen für $e' \in g^{-1}(e)$.
Aus (3.1) erhalten wir für alle $x \in E^{-1}(e)$

$$p_E(y|e) = \frac{p_E(x,y|e)}{p_E(x|e)} = \frac{p_E(x,y,e)p_E(e)}{p_E(x,e)p_E(e)} = \frac{p(x,y)\delta(E(x),e)}{p(x)\delta(E(x),e)} = p(y|x)$$

und

$$\begin{aligned} p_{E'}(y,e') &\overset{\text{Def}}{=} \sum_{x\in E'^{-1}(e')} p(x,y) \overset{\text{Def}}{=} \sum_{x\in E'^{-1}(e')} p(y|x)p(x) \\ &= \sum_{x\in E'^{-1}(e')} p_E(y|e)p(x) = p_E(y|e) \sum_{x\in E'^{-1}(e')} p(x) \\ &\overset{\text{Def}}{=} p_E(y|e)p_{E'}(e'), \end{aligned}$$

d.h. $p_{E'}(y|e') = p_E(y|e)$. Damit enthüllt Y auch höchstens E' über X. □

Das informationsärmste E, das enthüllt wird, ist also ein Maß für die Unabhängigkeit zweier Zufallsvariablen und damit für die Privatheit eines Protokolls.

Bei Kryptosystemen z.B. ist X die Klartextverteilung, Y die vom Angreifer abhörbare Ziffertextverteilung und $E(x)$ der leere String ϵ, d.h. es soll nichts entüllt werden. Das bedeutet, daß Klar- und Ziffertexte unabhängig verteilt sein müssen, also $p(m,c) = p(m)p(c)$, dies ist aber, wie im vorhergehenden Abschnitt gezeigt wurde, gleichwertig mit Ansatz 5. Damit erhalten wir den ebenfalls äquivalenten

Ansatz 0 Der Ziffertext enthüllt nichts über den Klartext.

Ausgehend von diesem Ansatz werden wir im nächsten Abschnitt überlegen, welche Information ein allgemeines Protokoll mindestens enthüllt, und die übrigen Privatheitsansätze dem anpassen.

3.4 Algorithmische Privatheit

Wenn wir die Privatheitsansätze aus dem vorigen Abschnitt auf allgemeine Protokolle übertragen wollen, haben wir drei Dinge zu beachten: zum einen muß die bereits erwähnte unvermeidlich enthüllte Information charakterisiert und einbezogen werden, zum zweiten wird nur noch algorithmische Ununterscheidbarkeit statt Gleichheit der auftretenden Wahrscheinlichkeitsverteilungen gefordert, und schließlich müssen wir uns auf probabilistische Algorithmen mit erwartet polynomieller Laufzeit beschränken.

Diese letzte Einschränkung bezieht zwei wesentliche Aspekte der realen Welt in das kryptographische Modell ein: die starke Abhängigkeit des Informationswertes von der Aktualität der Information und die hohen Kosten der Rechenzeit. Die Beschränkung auf polynomielle Laufzeit ist dabei vorsichtig genug, da ein Algorithmus mit asymptotischer Laufzeit von $O(n^{87})$ schon „quasi exponentiell", also unbrauchbar ist. Inwieweit eine asymptotische Betrachtungsweise in unserer bei 10^{80} endenden, endlichen Welt überhaupt sinnvoll ist, sei hier nicht weiter hinterfragt. In jedem Falle ermöglichen uns diese Einschränkungen, ganz neue Klassen beweisbar sicherer Protokolle zu finden.

Betrachten wir den Übergang von (symmetrischen) klassischen Kryptosystemen zu (asymmetrischen) Public-Key-Kryptosystemen. Hier wird die Schlüsselquelle K durch eine Quelle (K_o, K_p) ersetzt, die Paare von öffentlichen und privaten Schlüsseln generiert. Die Entropieformulierung von Privatheit nach SHANNON, die im symmetrischen Fall $H(M) - H(M|C) = 0$ lautet, wird im asymmetrischen Fall zu $H(M) - H(M|C, K_p) = 0$, da die Lauscherin hier neben dem Ziffertext auch den öffentlichen Schlüssel des Empfängers erhält. Ohne Einschränkungen an die Komplexität der verwendeten Algorithmen kann die Lauscherin durch vollständiges Durchsuchen des Klartextraumes immer einen zum öffentlichen Schlüssel und zum Ziffertext passenden Klartext ermitteln, die bedingte Entropie $H(M|C, K_p)$ ist gleich 0 und nur triviale Klartextverteilungen mit genau einem möglichen Klartext und damit $H(M) = 0$ sind privat.

Wir betrachten im folgenden ein 3-Ensemble (T, E, A), das sich aus einem Eingabe-Ensemble X und einem gestörten Protokoll S ableitet. Dabei steht T für die Ein- *und* Ausgaben aller (auch der angegriffenen) *Teilnehmer*, E für die dem Angreifer notwendig *enthüllte* Information und A für die *Angreiferausgabe*, also die Information, die er über die enthüllte Information hinaus erhält.

Für Public-Key-Kryptosysteme ist T der zu übertragende Klartext, E ist leer, denn eigentlich sollte einer Lauscherin nichts zugänglich sein, und A besteht aus dem, was die Lauscherin aus dem Ziffertext und dem öffentlichen Schlüssel berechnet.

Für private Funktionsauswertung, die in Abschnitt 2.5.3 vorgestellt wurde, enthält T die Ein- und Ausgaben aller Teilnehmer, E diejenigen der angegriffenen Teilnehmer und A die Ausgabe des Angreifers.

In der in Abschnitt 3.3.3 zitierten (mißglückten) geheimen Abstimmung besteht T aus den abgegebenen Stimmen aller Teilnehmer und aus dem Abstimmungsergebnis, das allen Teilnehmern mitgeteilt wird, und E besteht aus den Stimmen der angegriffenen Teilnehmer und dem Ergebnis. Die einzige Nein-Stimme, die alle gedeckt hätte, hätte dies nur für die Teilnehmer getan, die selbst mit Ja gestimmt haben, dem einzigen Nein-Stimmer wären wieder alle anderen offenbart worden.

Nach einer genauen Definition des schon erwähnten Ereignis-Ensembles (T, E, A) werden die in Abschnitt 3.3 erwähnten sechs Privatheitsansätze in jeweils einem Unterabschnitt für allgemeine Protokolle formalisiert, ihre gegenseitigen Abhängigkeiten und ihre daraus resultierende wesentliche Äquivalenz zeigen wir in den Unterabschnitten 3.4.8 und 3.4.9.

3.4.1 Eingaben und Ereignisse

Die in diesem und dem nächsten Unterabschnitt beschriebenen Bezeichnungskonventionen gelten für den gesamten restlichen Abschnitt.

Wir untersuchen Privatheit an einflußlos angegriffenen Protokollen. Sei also im folgenden immer $S_0 = (C, V, (e_P)_{P\in C}, (a_P)_{P\in C})$ das ungestörte Protokoll, das von einem Angreifer $A = (C_A, P_A, \nu_A, e_A, a_A) \in \mathrm{eAdv}(S_0)$ angegriffen wird. Das entstehende gestörte Protokoll ist $S = \mathrm{aProt}(S_0, A)$. Der Angreifer bekommt keine eigentliche Eingabe, seine Ausgabe enthält, was er während des Protokollablaufes gelernt hat.

Sei $X_I = (M_X, (X_i)_{i\in I})$ eine Eingabefamilie für das ungestörte Protokoll S_0 und sei $(i, x) \in \hat{X}_I$ eine eigentliche Eingabe für S_0. Dann ist die zugehörige *tatsächliche* Eingabe des ungestörten Protokolls $((i, x_P))_{P\in C}$. Da der Angreifer nur den Sicherheitsparameter erhält, entspricht jeder Eingabe des ungestörten Protokolls S_0 genau eine Eingabe für das gestörte Protokoll S, nämlich $(((i, x_P))_{P\in C}, (i, \epsilon))$. Daher müssen wir vom Angreifer zusätzlich $I \times \{\epsilon\} \subset \mathrm{Dom}(A)$ verlangen.

Sei $(i, x) \in \hat{X}_I$ eine eigentliche Eingabe, dann schreiben wir kurz

$$\pi_{S_0}((i, x), y)$$

für die Wahrscheinlichkeit, daß das ungestörte System S_0 bei eigentlicher Eingabe (i, x) die Ausgabe $y \in \mathrm{Ran}(S_0)$ erzeugt, anstelle des korrekten, aber unnötig längeren Ausdruckes $\pi_{S_0}(((i, x_P))_{P\in C}, y))$. Beim gestörten System S schreiben wir

$$\pi_S((i, x), (y, a))$$

anstelle von $\pi_S((((i, x_P))_{P\in C}, (i, \epsilon)), (y, a))$. Das ungestörte System S_0 ordnet damit jeder eigentlichen Eingabe $(i, x) \in \hat{X}_I$ eine Verteilung $\pi_{S_0}((i, x), \cdot) \in \mathcal{W}(\mathrm{Ran}(S_0))$ zu, das gestörte bildet (i, x) auf $\pi_S((i, x), \cdot) \in \mathcal{W}(\mathrm{Ran}(S_0) \times \mathrm{Ran}(A))$ ab.

Da wir uns nicht nur für die Eingaben, sondern auch für die Ausgaben eines Protokolles interessieren, nennen wir ein Tripel $(i, x, y) \in \hat{X}_I \times \mathrm{Ran}(S_0)$ ein **Ereignis**.

Eine **Ereignismenge** für S_0 auf X_I ist eine Teilmenge $L \subset \hat{X}_I \times \mathrm{Ran}(S_0)$, der **Abschluß**

$$\overline{L} = \{(i,x,y) \in \hat{X}_I \times \mathrm{Ran}(S_0) | \pi_{S_0}((i,x),y) > 0 \wedge \exists y' \in \mathrm{Ran}(S_0) : (i,x,y) \in L\}$$

von L ist die Menge aller verwandten Ereignisse mit positiver Wahrscheinlichkeit und

$$L_i = \{(x,y) | (i,x,y) \in L\}$$

ist die Menge der zum Sicherheitsparameter $i \in I$ gehörenden **Teilnehmersituationen** in L.

Eine Ereignismenge L ist **wesentlich** für S_0, wenn es Konstanten $c, c' > 0$ gibt mit

$$\bigwedge_{\substack{i \in I \\ |i| \geq c'}} \bigwedge_{(x,y) \in L_i} \pi_{S_0}((i,x),y) \geq |i|^{-c}.$$

Die Konstante c heißt **Wesentlichkeitsexponent** von L.

Ein **Eingabe-Ensemble** auf einer Eingabefamilie $X_I = (M_X, (X_i)_{i \in I})$ für S_0 ist ein Ensemble $X = (M_X, (p'_i)_{i \in I})$ mit $\mathrm{supp}(p_i) \subset X_i$ für alle $i \in I$, die zu einem Eingabe-Ensemble $X = (M_X, (p'_i)_{i \in I})$ für S_0 gehörende Ereignismenge ist

$$L(X) = \{(i,x,y) \in I \times M_X \times \mathrm{Ran}(S_0) \mid p'_i(x)\pi_{S_0}((i,x),y) > 0\}.$$

Das Eingabe-Ensemble X ist **auf** der Ereignismenge L, wenn $L(X) \subset L$ ist, es ist **konzentriert** auf L, wenn nur ein verschwindender Anteil möglicher Ereignisse nicht in L ist, d.h. wenn die Funktion

$$i \mapsto \sum_{(x,y) \notin L_i} p'_i(x)\pi_{S_0}((i,x),y)$$

subpolynomiell ist.

Wenn wir in den folgenden Abschnitten von der Privatheit des gestörten Protokolls $S = \mathrm{aProt}(S_0, A)$ reden, meinen wir, daß das ungestörte Protokoll S_0 privat ist gegen den Angreifer A.

3.4.2 Minimales Enthüllen

Was kann vor einem (einflußlosen) Angreifer A nicht versteckt werden? Die Ein- und Ausgaben der von ihm angegriffenen Teilnehmer $P \in C_A$, d.h. er erfährt immer den Anteil $((x_P)_{P \in C_A}, (y_P)_{P \in C_A})$ der von ihm angegriffenen Teilnehmer an einem Ereignis (i,x,y), also neben dem Sicherheitsparameter i eine vom Angreifer abhängige deterministische Funktion $E_A(x,y)$ der Teilnehmersituation (x,y). Was der Angreifer während des Protokollverlaufes zusätzlich erfährt, kondensiert er in der Angreifer-Ausgabe a.

Sei $X = (M_X, (p'_i)_{i\in I})$ wieder ein Eingabe-Ensemble für S_0, insbesondere ist also $M_X = \bigtimes_{P\in C} M_P$. Aus X bilden wir ein 3-Ensemble

$$(T, E, A) = (M_T \times M_E \times M_A, (p_i)_{i\in I})$$

mit

$$M_T = M_X \times \mathrm{Ran}(S_0), \quad M_E = \bigtimes_{P\in C_\Lambda} M_P \times \mathrm{Ran}(a_P) \quad \text{und} \quad M_A = \mathrm{Ran}(A).$$

Die erste Komponente T des Ensembles umfaßt die Ein- und Ausgaben aller, auch der angegriffenen *Teilnehmer*, die Komponente E die *enthüllte Information* und die Komponente A die Ausgabe des *Angreifers*. Das zugehörige Wahrscheinlichkeitsmaß ist

$$p_i(t, e, a) = p'_i(x)\pi_S((i, x), (y, a))\delta(E_A(t), e) \quad \text{für } t = (x, y),$$

dabei ist δ wieder die KRONECKER-Funktion.

Aus dem dreiargumentigen Wahrscheinlichkeitsmaß p_i erhalten wir durch Summieren über eines oder zwei der Argumente insgesamt sechs neue Wahrscheinlichkeitsmaße, die wir ebenfalls mit p_i bezeichnen, da diese durch die Benennung ihrer Argumente eindeutig identifiziert sind. Ebenso schreiben wir in Summenindices nur „b“ anstelle des korrekten „$b \in M_B$“ für $b \in \{t, e, a\}$, da klar ist, woher diese Größen stammen. Damit ist die Wahrscheinlichkeit einer Situation $t = (x, y)$ zum Sicherheitsparameter i

$$\begin{aligned} p_i(t) &= \sum_{e,a} p_i(t, e, a) = \sum_{e,a} p'_i(x)\pi_S((i, x), (y, a))\delta(E_A(t), e) \\ &= p'_i(x) \sum_a \pi_S((i, x), (y, a)) \sum_e \delta(E_A(t), e) = p'_i(x) \sum_a \pi_S((i, x), (y, a)) \\ &= p'_i(x)\pi_{S_0}((i, x), y), \end{aligned}$$

da die angreiferabhängige Enthüllungsfunktion E_A deterministisch ist und S einflußlos angegriffen wird.

In der informationstheoretischen Betrachtungsweise hieße „A enthüllt höchstens E_A über T“, daß die Verteilungen T_e und A_e unabhängig sind für jedes e, also $p(t, a|e) = p(t|e)p(a|e)$ ist. Im algorithmischen Fall fordern wir daher die Ununterscheidbarkeit der Ensembles $(T, A)_e$ und $T_e \times A_e$, gemittelt über e.

3.13. Definition *Das Protokoll S ist* **E-privat** *auf dem Eingabe-Ensemble X, wenn für alle Unterscheider $U \in \mathcal{PP}$, alle $c > 0$ und hinreichend großen $|i|$*

$$\sum_e p_i(e) \Big| \sum_{t,a} (p_i(t, a|e) - p_i(t|e)p_i(a|e))\pi_U((i, t, e, a), 1) \Big| \;\le\; |i|^{-c}$$

ist.

3.4.3 Ereignis-Unvorhersagbarkeit

Ereignis-Unvorhersagbarkeit findet sich unter anderem in [MicRackSlo88], die Idee dahinter ist: man nimmt zwei Situationen, die dem Angreifer das gleiche enthüllen sollten, wählt die Situationen mit gleicher Wahrscheinlichkeit und läßt den Vorhersager raten, welche der Situationen einer gegebenen Ausgabe des Angreifers zugrundeliegt.

Bei perfekten Kryptosystemen ist die Eingabe des Senders gleich der Ausgabe des Empfängers, also eine Situation bestimmt durch den zu sendenden Klartext m. Wählen wir einen von zwei Klartexten m_0, m_1 mit Wahrscheinlichkeit $1/2$, verschlüsseln diesen Klartext und geben den Ziffertext einem Vorhersager, dann sollte dessen Erfolgswahrscheinlichkeit

$$\sum_{j\in\{0,1\}} \tfrac{1}{2} \sum_{c} \pi_{\text{Encode}}(m_j, c)\pi_V((m_0, m_1, c), j)$$

nicht wesentlich von $1/2$ abweichen. Dieses $1/2$ ist die a priori Wahrscheinlichkeit der Klartexte und damit der maximale Vorhersage-Erfolg ohne Kenntnis des Ziffertextes.

Bei unperfekten Kryptosystemen oder allgemeinen Protokollen besteht eine Situation aber nicht nur aus der Eingabe, sondern aus einem Paar $t = (x, y)$ von Ein- und Ausgabe. Die Eingabe kann frei gewählt werden, die „Auswahl" der Ausgaben übernimmt das Protokoll. Um also eine von zwei Situationen gleichverteilt wählen zu können, benutzen wir das Zufallsexperiment

$$\mathrm{Ex}_S(i, x, y) = \begin{array}{l} > \texttt{Simuliere } S(i,x), \overbrace{\texttt{ Ausgabe ist } (y', a)}^{y'\neq y} \\ \qquad\qquad \Big\downarrow {\scriptstyle y'=y} \\ \texttt{Gib } a \texttt{ aus} \end{array},$$

das S solange simuliert, bis das gewünschte Ereignis eintritt. Die Wahrscheinlichkeit, daß dieses Experiment eine Angreiferausgabe a produziert, ist

$$\pi_{\mathrm{Ex}_S}((i, x, y), a) \;=\; \pi_S((i, x), (y, a)) + \Big(1 - \sum_a \pi_S((i, x), (y, a)\Big)\pi_{\mathrm{Ex}_S}((i, x, y), a),$$

da es für $y' \neq y$ von neuem beginnt.† Die betrachteten Angreifer sind einflußlos, deshalb ist $\sum_a \pi_S((i, x), (y, a)) = \pi_{S_0}((i, x), y)$ und

$$\pi_{\mathrm{Ex}_S}((i, x, y), a) = \frac{\pi_S((i, x), (y, a))}{\pi_{S_0}((i, x), y)} .$$

† Man überzeuge sich, daß das Bilden der entsprechenden unendlichen Reihe zum selben Ergebnis führt wie dieser rekursive Ansatz.

Dies wiederum ist identisch mit

$$p_i(a|t) \;=\; \frac{p_i(t,a)}{p_i(t)} \;=\; \frac{p'_i(x)\pi_S((i,x),(y,a))}{p'_i(x)\pi_{S_0}((i,x),y)}\,.$$

Auf wesentlichen Ereignissen ist deshalb die erwartete Laufzeit von Ex_S

$$\mathrm{ER}_{\mathrm{Ex}_S}(i,x,y) \;=\; \frac{1}{\pi_{S_0}((i,x),y)}\,\mathrm{ER}_S(i,x) \;\le\; |i|^c\,\mathrm{ER}_S(i,x)$$

erwartet polynomiell für (erwartet) polynomielles S.

3.14. Definition *Das Protokoll S ist* **V-privat** *auf der Ereignismenge L, wenn für alle Vorhersager $V \in \mathcal{PP}$ mit Ausgaben in $\{0,1\}$, für alle $c > 0$, alle hinreichend großen $|i|$ und alle $t_0, t_1 \in L_i$ mit $E_A(t_0) = e = E_A(t_1)$*

$$\sum_{j\in\{0,1\}} \tfrac{1}{2} \sum_a p_i(a|t_j,e)\pi_V((i,t_0,t_1,e,a),j) \;\le\; \tfrac{1}{2} + |i|^{-c}$$

ist.

Man beachte, daß in dieser Definition die Verteilung auf den Eingaben keine Rolle spielt (und deshalb auch nicht erwähnt wird), denn für $e = E_A(t)$ ist $p_i(a|t,e) = p_i(a|t) = \pi_{\mathrm{Ex}_S}((i,x,y),a)$ unabhängig von der Eingabeverteilung.

3.4.4 Semantische Privatheit

In [Yao82, Yao86, Yao88] (und in fehlerhafter Formalisierung in [MicRack-Slo88]) wird ein Kryptosystem als semantisch sicher bezeichnet, wenn keine Funktion des Klartextes aus dem Ziffertext besser berechnet werden kann als ohne Ziffertext, also durch raten.

Sei $(X,Y) = (M_X \times M_Y, (p_i)_{i\in I})$ ein 2-Ensemble und $D \in \mathcal{PP}$ ein Algorithmus mit Eingaben aus $I \times M_X$ und Ausgaben in Z. Ein **Rate-Algorithmus** für D mit Zugang zu Y ist ein Algorithmus $R \in \mathcal{PP}$ von M_Y nach Z, d.h. R versucht, aus dem möglicherweise von (i,x) abhängigen Teilereignis (i,y) auf den „Wert " $D(x)$ zu schließen. Die **Korrelation** zwischen D und R beim Ereignis (i,x,y) ist die Wahrscheinlichkeit, daß D und R zum selben Ergebnis kommen,

$$\mathrm{Korr}_{D,R}(i,x,y) = \sum_z \pi_D((i,x),z)\pi_R((i,y),z),$$

der **Rate-Erfolg** von R bei Parameter $i \in I$ ist der Erwartungswert der Korrelation

$$\mathrm{Succ}_{D,R}(i) = \sum_{x,y} p_i(x,y)\,\mathrm{Korr}_{D,R}(i,x,y).$$

3.15. Definition *Das Protokoll* S *ist* **R-privat** *auf dem Eingabe-Ensemble* X*, wenn es zu jedem Algorithmus* $D \in \mathcal{PP}$ *auf* $L(X)$ *und jedem Rate-Algorithmus* $R \in \mathcal{PP}$ *für* D *mit Zugang zu* (E, A) *einen Rate-Algorithmus* $R' \in \mathcal{PP}$ *mit Zugang nur zu* E *gibt, dessen Rate-Erfolg höchstens subpolynomiell kleiner ist, d.h. für alle* $c > 0$ *und hinreichend großen* $|i|$ *ist*

$$\mathrm{Succ}_{D,R}(i) \;\leq\; \mathrm{Succ}_{D,R'}(i) + |i|^{-c}.$$

3.4.5 Simulierbarkeit der Angreifersicht

Das Prinzip der ununterscheidbaren Simulierbarkeit von Angreiferausgaben bzw. -transkripten stammt aus dem Bereich der Zero-Knowledge-Beweissysteme, siehe etwa [FeiSha90], und wurde, z.B. in [Beaver89, GalilHabYung87] auf private Funktionsauswertung übertragen. Man glaubt, daß man alles, was der Angreifer aus seiner Sicht berechnen kann, auch ohne Interaktion mit dem Protokoll berechnen kann, wenn man die Angreifersicht ununterscheidbar simulieren kann. Dieser Glaube wird durch Satz 3.27 untermauert.

3.16. Definition *Das Protokoll* S *ist* **S-privat** *auf der Ereignismenge* L*, wenn es einen Simulator* $M \in \mathcal{PP}$ *gibt, der aus der dem Angreifer enthüllten Information die Angreiferausgabe algorithmisch ununterscheidbar simuliert, d.h. für alle Unterscheider* $U \in \mathcal{PP}$ *mit Ausgaben in* $\{0,1\}$*, für alle* $c > 0$*, alle hinreichend großen* $|i|$ *und alle* $t \in L_i$ *ist*

$$\left|\sum_a (p_i(a|t) - \pi_M((i, E_A(t)), a))\pi_U((i, t, E_A(t), a), 1)\right| \;\leq\; |i|^{-c}.$$

Wie die V-Privatheit ist diese Definition unabhängig von der Verteilung auf den Eingaben.

In [Beaver89] wird nur die Ununterscheidbarkeit der simulierten Transkripte von $p_i(a|e)$ gefordert, d.h. gemittelt über alle t mit $E_A(t) = e$. Diese abgeschwächte Forderung reicht *nicht* mehr aus, um Satz 3.27 zu beweisen, aus derartig simulierten Transkripten ist möglicherweise weniger ablesbar als aus den von uns geforderten.

Betrachtet man diese Definition von der Seite der Informationsenthüllung, dann heißt sie nichts anderes als daß die Angreiferausgabe A *informationsärmer* ist als die enthüllte Information E, da die erste aus der zweiten „berechnet“ werden kann.

3.4.6 Ereignis-Ununterscheidbarkeit

Eine weitere Anschauung von Privatheit, die man z.B. in [Beaver89-3, ChorKush2, FeiSha90, Kush] findet, ist, daß der Angreifer in für ihn gleichartigen Situationen, d.h. mit identischer enthüllter Information, dasselbe sehen sollte.

Dies entspricht wieder der Benutzung des Experiments Ex_S aus Abschnitt 3.4.3 mit Eingaben (i, t) bzw. (i, t'), $t, t' \in L_i$ mit $E_A(t) = E_A(t')$.

3.17. Definition *Das Protokoll* S *ist* **U-privat** *auf der Ereignismenge* L, *wenn Angreiferausgaben, die aus Situationen mit identischer enthüllter Information stammen, algorithmisch ununterscheidbar sind, d.h. für alle Unterscheider* $U \in \mathcal{PP}$ *mit Werten in* $\{0,1\}$, *für alle* $c > 0$, *alle hinreichend großen* $|i|$ *und alle* $t, t' \in L_i$ *mit* $E_A(t) = e = E_A(t')$ *ist*

$$\left|\sum_a (p_i(a|t,e) - p_i(a|t',e))\pi_U((i,t,t',e,a),1)\right| \;\leq\; |i|^{-c}.$$

Auch dieser Privatheitsbegriff ist unabhängig von der Verteilung auf den Eingaben.

3.4.7 Algorithmische Entropie und Entropie-Privatheit

Ein wesentlicher Schritt beim Übergang vom informationstheoretischen zum algorithmischen Szenario ist die Einschränkung der Komplexität der auftretenden Algorithmen. Wo aber sind in der Definition der klassischen Entropie Algorithmen beteiligt? Nach SHANNONS erstem Satz ist die Entropie einer Quelle identisch mit der minimalen erwarteten Länge einer Kodierung der Ereignisse über $\{0,1\}$. In Analogie dazu führt YAO [YAO82, YAO88] den Begriff der *algorithmischen Entropie* H_a ein. Dabei wird die erwartete Länge eines Kodes betrachtet, der von probabilistischen, polynomiell zeitbeschränkten Algorithmen erzeugt und mit vernachlässigbarer Fehlerwahrscheinlichkeit wieder entziffert werden kann.

Sei also $(X,Y) = (M_X \times M_Y, (p_i(\cdot,\cdot))_{i\in I})$ ein polynomielles 2-Ensemble. Wir setzen

$$p_i^{(n)}(x,y) = \prod_{\ell=1}^{n} p_i(x_\ell, y_\ell)$$

für $x = (x_1,\ldots,x_n) \in M_X^n$ und $y = (y_1,\ldots,y_n) \in M_Y^n$.

Eine **Quellenkodierung** von X relativ zu Y ist ein Tripel $(E,D,C) \in \mathcal{PP}_m^3$ probabilistischer Polynomzeitalgorithmen (Encoder, Decoder, Cutter) mit Ausgaben in $\{0,1\}^*$ derart, daß für ein $k \in \mathbb{N}$

1. die Dekodierwahrscheinlichkeit von E und D nur subpolynomiell kleiner als 1 ist, d.h. für alle $c > 0$ und alle hinreichend großen $|i|$ ist

$$\sum_{x\in M_X^{|i|^k}, y\in M_Y^{|i|^k}} p_i^{(|i|^k)}(x,y) \sum_{\alpha\in\{0,1\}^+} \pi_E((i,x,y),\alpha)\pi_D((i,\alpha,y),x) \;\geq\; 1 - |i|^{-c},$$

2. jede polynomielle Konkatenation von Kodeworten, die von E generiert wurden, von C mit nur subpolynomiellem Fehler in die korrekten Stücke zerlegt werden

kann, d.h. für alle $b \in \mathbb{N}$, alle $c > 0$ und alle hinreichend großen $|i|$ ist

$$\sum_{\substack{x_1,\ldots,x_{|i|^b} \in M_X^{|i|^k} \\ y_1,\ldots,y_{|i|^b} \in M_Y^{|i|^k}}} \prod_{j=1}^{n^b} p_i^{(|i|^k)}(x_j, y_j) \sum_{\alpha_1,\ldots,\alpha_{|i|^b} \in \{0,1\}^+} \prod_{j=1}^{|i|^b} \pi_E((i, x_j, y_j), \alpha_j)\pi_C((i, x_1 \cdots x_{|i|^b}), (x_1, \ldots, x_{|i|^b})) \;\geq\; 1 - |i|^{-c}.$$

Die **erwartete Kodelänge** von (E, D, C) ist

$$E_{(E,D,C)}(i) = |i|^{-k} \sum_{x \in (\Sigma^*)^{|i|^k}, y \in (\Sigma^*)^{|i|^k}} p_i(x, y) \sum_{\alpha \in \{0,1\}^+} \pi_E((i, x, y), \alpha)|\alpha|.$$

Wir können $E_{(E,D,C)}$ als Funktion von I nach $\mathbb{R}_0^+$ auffassen.

3.18. Definition *Sei (X, Y) ein 2-Ensemble. Die* **algorithmische Entropie** *von X relativ zu Y ist beschränkt durch $f : I \to \mathbb{R}$, kurz $H_a(X|Y) \leq f$, wenn es eine Quellenkodierung (E, D, C) von X relativ zu Y gibt mit $E_{(E,D,C)}(i) \leq f(i)$ für alle hinreichend großen $|i|$. Analog für $H_a(X) \leq f$.*

Wir schreiben $H_a(X) - H_a(Y) \leq 1/p$, wenn es für jede Quellenkodierung (E, D, C) von Y eine Quellenkodierung (E', D', C') von X derart gibt, daß die Funktion

$$E_{(E',D',C')} - E_{(E,D,C)} : I \to \mathbb{R}_0^+$$

subpolynomiell ist.

Da wir die Laufzeit der Kodierungsalgorithmen beschränken, scheint es klar, daß die algorithmische Entropie mindestens so groß ist wie die Shannonsche Entropie. Dies ist nicht notwendig so, da wir polynomielle Algorithmen zulassen und diesen eine kleine Fehlerwahrscheinlichkeit zubilligen. Es stellt sich aber heraus, daß diese zusätzlichen Möglichkeiten die erwartete Kodelänge höchstens subpolynomiell verkleinern können:

3.19. Satz *Ist X ein polynomielles Ensemble, dann ist $H(X) - H_a(X) \leq 1/p$.*

Beweis: Die Aussage des Satzes heißt ausführlich

$$\bigwedge_{(E,D,C)} \bigwedge_{c>0} \bigvee_{n_c \in \mathbb{N}} \bigwedge_{|i| \geq n_c} H(X_i) \leq E_{(E,D,C)}(i) + |i|^{-c}.$$

Idee: Man konstruiert aus jedem probabilistischen Kode (E, D) einen eindeutig dekodierbaren deterministischen Kode K für X. Dann ist nach dem ersten Shannonschen Satz $H(X_i) \leq E_K(i)$, der erwarteten Kodelänge von K. Zu zeigen bleibt dann $E_K(i) \leq E_{(E,d)}(i) + |i|^{-c}$ für $|i|$ groß genug.

Die Trenneigenschaft einer Quellenkodierung impliziert die eindeutige Dekodierbarkeit: bei einem (auftretenden) Kodestring α mit zwei (auftretenden) Trennungen $\beta_1, \ldots, \beta_u$ und $\beta'_1, \ldots, \beta'_{u'}$ ist jede Verlängerung von α (mindestens) zweifach auftrennbar, d.h. ein konstant wahrscheinlicher Anteil von Kodestrings ist zweideutig auftrennbar und die Trennerfolgswahrscheinlichkeit nach oben beschränkt durch $1 - p(\alpha)$ im Widerspruch zu Eigenschaft 2 von Quellenkodierungen.

Merke: Quellenkodierungen sind *immer* eindeutig auftrennbar, die Trennfehlerwahrscheinlichkeit rührt *nur* aus der polynomiellen Beschränkung der Trennalgorithmen.

Zwischenziel: Konstruktion einer eindeutigen Zuordnung von (einigen) Quellworten auf Kodeworte. Man beachte, daß $m \in X_i^{|i|^d}$ ist, da E Päckchen von $|i|^d$ Quellworten in Laufzeit $|i|^t$ kodiert.

Mit den Abkürzungen $p(m) = p_i^{(|i|^d)}(m)$, $p(m, \alpha) = \pi_E((i, m), \alpha)$ und $p(\alpha, m) = \pi_D((i, \alpha), m)$ ist die Gesamtdekodierungswahrscheinlichkeit

$$\bigwedge_{c>0} \bigwedge_{|i| \geq n_c} \sum_m p(m) \sum_\alpha p(m, \alpha) p(\alpha, m) \geq 1 - |i|^{-c}.$$

Sei $G_m = \{\alpha | p(\alpha, m) > 1/2\}$, dann ist $G_m \cap G_{m'} = \emptyset$ für $m \neq m'$. Sei weiter $G = \{m | G_m \neq \emptyset\}$ und α_m das kürzeste $\alpha \in G_m$ für $m \in G$. Sei

$$p_\emptyset = \sum_{m \notin G} p(m)$$

die Gesamtwahrscheinlichkeit aller *nicht* eindeutig dekodierbaren Quellworte, dann ist für alle $c > 0$ und alle $|i| \geq n_c$

$$\begin{aligned} 1 - |i|^{-c} &\leq \sum_m p(m) \sum_\alpha p(m, \alpha) p(\alpha, m) \\ &= \sum_{m \notin G} p(m) \sum_\alpha p(m, \alpha) p(\alpha, m) + \sum_{m \in G} p(m) \sum_\alpha p(m, \alpha) p(\alpha, m) \\ &\leq \sum_{m \notin G} p(m) \sum_\alpha p(m, \alpha) \cdot \tfrac{1}{2} + \sum_{m \notin G} p(m) \sum_\alpha p(m, \alpha) \cdot 1 \\ &= 1/2 p_\emptyset + (1 - p_\emptyset), \end{aligned}$$

also $p_\emptyset \leq 2|i|^{-c}$, d.h. die wichtigsten Quellwörter haben ein eindeutiges Kodewort, nur einem Anteil von Quellworten mit subploynomieller Gesamtwahrscheinlichkeit kann man kein eindeutiges Kodewort zuordnen.

Zwischenziel: Abschätzung die Entropie der Quelle $X_i^{|i|^d}$ durch die Entropie der Subquelle X'_i wichtiger Wörter $m \in G$ mit dem Wahrscheinlichkeitsmaß $p'(m) = 1/(1-p_\emptyset) p(m)$:

$$H(X_i^{|i|^d}) = -\sum_m p(m) \log p(m) = -\sum_{m \in G} p(m) \log p(m) - \sum_{m \notin G} p(m) \log p(m)$$

$$\begin{aligned} &= -(1-p_\emptyset)\sum_{m\in G}\frac{p(m)}{1-p_\emptyset}\log\Big(\frac{p(m)}{1-p_\emptyset}(1-p_\emptyset)\Big) - \sum_{m\notin G} p(m)\log p(m) \\ &= (1-p_\emptyset)H(X_i') - (1-p_\emptyset)\log(1-p_\emptyset) - \sum_{m\notin G} p(m)\log p(m). \end{aligned}$$

Mit der JENSENschen Ungleichung wird der *letzte Term*

$$\begin{aligned} -\sum_{m\notin G} p(m)\log p(m) &\le -|X_i^{|i|^d}\setminus G|\frac{p_\emptyset}{|X_i^{|i|^d}\setminus G|}\log\frac{p_\emptyset}{|X_i^{|i|^d}\setminus G|} \\ &\le -p_\emptyset\log\frac{p_\emptyset}{|X_i|^{|i|^d}} = -p_\emptyset\log p_\emptyset - p_\emptyset\log|X_i^{|i|^d}| \end{aligned}$$

Da X ein polynomielles Ensemble ist, gibt es c_X und c_X' mit $|x| \le |i|^{c_X}$ für $|i| \ge c_X'$ und $x \in \mathrm{supp}(p_i)$. Also ist $|X_i| \le |\Sigma|^{|i|^{c_X}}$, wobei Σ das Alphabet von X ist, und wir können weiter abschätzen durch

$$2|i|^{-c}c\log|i| + 2|i|^{-c}|i|^{c_X+d}\log|\Sigma| \le 4|i|^{-c+c_X+d+2}$$

für $|i| \ge \max\{c, c_X, c_X', \log|\Sigma|\}$.
Für den *mittleren Term* benutzen wir

$$-\log(1-x) = \sum_{i=1}^{\infty}\frac{x^i}{i} \le \sum_{i=1}^{\infty}x^i = x\sum_{i=0}^{\infty}x^i \le x\sum_{i=0}^{\infty}(\tfrac{1}{2})^i = 2x$$

für $x \le 1/2$. Damit ist $-(1-p_\emptyset)\log(1-p_\emptyset) \le 2p_\emptyset \le 4|i|^{-c}$.
Mit dem ersten SHANNONschen Satz ist der *erste Term*

$$(1-p_\emptyset)H(X_i') \le (1-p_\emptyset)H(X_i')\sum_{m\in G}\frac{p(m)}{1-p_\emptyset}|\alpha_m| = \sum_{m\in G}p(m)|\alpha_m|.$$

Sei p_S die Gegenwahrscheinlichkeit zu „wichtiges Wort tritt auf *und* wird gut dekodierbar kodiert“, dann ist

$$\begin{aligned} 1-|i|^{-c} &\le \sum_{m,\alpha}p(m)p(m,\alpha)p(\alpha,m) \\ &= \sum_{\substack{m,\alpha\\ \alpha\in G_m}}p(m)p(m,\alpha)p(\alpha,m) + \sum_{\substack{m,\alpha\\ \alpha\notin G_m}}p(m)p(m,\alpha)p(\alpha,m) \\ &\le \sum_{\substack{m,\alpha\\ \alpha\in G_m}}p(m)p(m,\alpha)\cdot 1 + \sum_{\substack{m,\alpha\\ \alpha\notin G_m}}p(m)p(m,\alpha)\cdot\tfrac{1}{2} \\ &= (1-p_S) + 1/2\,p_S, \end{aligned}$$

also $p_S \le 2|i|^{-c}$. Damit ist

$$\sum_{m\in G}p(m)|\alpha_m| = \sum_{m\in G}p(m)\sum_{\alpha}p(m,\alpha)|\alpha_m|$$

$$\begin{aligned}
&\leq \sum_{\substack{m,\alpha \\ \alpha \in G_m}} p(m)p(m,\alpha)|\alpha_m| + \sum_{\substack{m,\alpha \\ \alpha \notin G_m}} p(m)p(m,\alpha)|\alpha_m| \\
&\leq \sum_{\substack{m,\alpha \\ \alpha \in G_m}} p(m)p(m,\alpha)|\alpha| + \sum_{\substack{m,\alpha \\ \alpha \notin G_m}} p(m)p(m,\alpha)(|\alpha| + |\alpha_m| - |\alpha|) \\
&= \sum_{m,\alpha} p(m)p(m,\alpha)|\alpha| + \sum_{\substack{m,\alpha \\ \alpha \notin G_m}} p(m)p(m,\alpha)(|\alpha_m| - |\alpha|) \\
&\leq |i|^d E_{(E,D)}(i) + p_S \max|\alpha| \leq |i|^d E_{(E,D)}(i) + 2|i|^{-c+t}.
\end{aligned}$$

Setzen wir die drei Teilabschätzungen zusammen, dann ist für alle $c' > 0$ und alle $|i| \geq n_{c'} = \max\{n_c, c, c_X, c'_X, \log|\Sigma|, 10\}$

$$\begin{aligned}
H(X_i) &= |i|^{-d} H(X_i^{|i|^d}) \\
&\leq |i|^{-d}(4|i|^{-c+c_X+d+2} + 4|i|^{-c} + |i|^d E_{(E,D)}(i) + 2|i|^{-c+t}) \\
&\leq E_{(E,D)}(i) + 10|i|^{-c+c_X+t-d+2} \\
&\leq E_{(E,D)}(i) + |i|^{-c+c_X+t+3} = E_{(E,D)}(i) + |i|^{-c'},
\end{aligned}$$

wenn wir $c = c' + c_X + t + 3$ wählen. □

Wir werden diesen Satz noch im Beweis von Lemma 3.32 benötigen. Kommen wir schließlich zur Definition der Entropie-Privatheit.

3.20. Definition *Das Protokoll* S *ist* **H-privat** *auf dem Eingabe-Ensemble* X, *wenn*

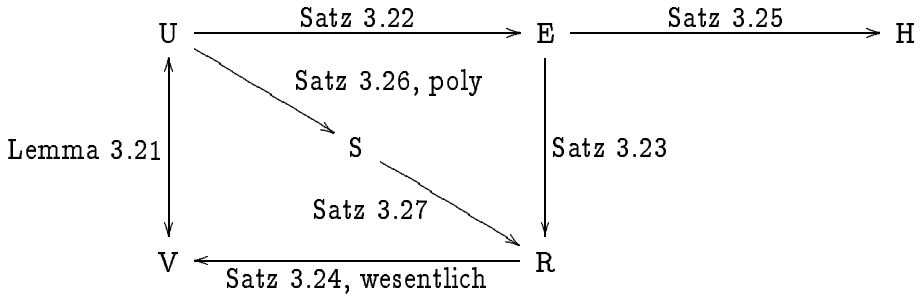
$$H_a(T|E) - H_a(T|E,A) \leq 1/p$$

ist.

Damit haben wir alle in Abschnitt 3.3 definierten Privatheitsansätze auf allgemeine Protokolle verallgemeinert und an das algorithmische Szenario angepaßt. In den nächsten beiden Unterabschnitten zeigen wir, daß diese Ansätze weitgehend äquivalent sind.

3.4.8 Relation der Privatheitsdefinitionen

In diesem Unterabschnitt werden folgende Zusammenhänge zwischen den sechs Privatheitsbegriffen für allgemeine Protokolle bewiesen:



3.21. Lemma *U-Privatheit auf der Ereignismenge* L *ist gleichbedeutend mit V-Privatheit auf* L.

Beweis: Wir zeigen, daß Unterscheider und Vorhersager ineinander transformiert werden können. Sei dazu M eine probabilistische Turingmaschine mit Ausgaben in $\{0,1\}$, dann ist für $i \in I$, Situationen $t_0, t_1 \in L_i$ und enthüllte Information e

$$
\begin{aligned}
& 2\tfrac{1}{2} \sum_{a \in \mathrm{Ran}(A)} \Big(p(a|t_0,e) - p(a|t_1,e)\Big)\pi_M((i,t_0,t_1,e,a),0) \qquad (3.2)\\
&= \tfrac{1}{2} \sum_{a \in \mathrm{Ran}(A)} \Big(p(a|t_0,e) - p(a|t_1,e)\Big)\pi_M((i,t_0,t_1,e,a),0)\\
&\quad + \tfrac{1}{2} \sum_{a \in \mathrm{Ran}(A)} \Big(p(a|t_0,e) - p(a|t_1,e)\Big)\Big(1 - \pi_M((i,t_0,t_1,e,a),1)\Big)\\
&= \tfrac{1}{2} \sum_{a \in \mathrm{Ran}(A)} \Big(p(a|t_0,e)\pi_M((i,t_0,t_1,e,a),0) + p(a|t_1,e)\pi_M((i,t_0,t_1,e,a),1)\\
&\qquad\quad -p(a|t_1,e)\pi_M((i,t_0,t_1,e,a),0) - p(a|t_0,e)\pi_M((i,t_0,t_1,e,a),1)\\
&\qquad\quad +p(a|t_0,e) - p(a|t_1,e)\Big)\\
&= \sum_{i \in \{0,1\}} \tfrac{1}{2} \sum_{a \in \mathrm{Ran}(A)} p(a|t_i,e)\pi_M((i,t_0,t_1,e,a),i)\\
&\quad - \sum_{i \in \{0,1\}} \tfrac{1}{2} \sum_{a \in \mathrm{Ran}(A)} p(a|t_i,e)\pi_M((i,t_0,t_1,e,a),1-i) + \tfrac{1}{2} - \tfrac{1}{2}\\
&= \text{Prob[richtig geraten]} - \text{Prob[falsch geraten]} = p_r - p_f \quad . \qquad (3.3)
\end{aligned}
$$

Angenommen, S_0 ist U-privat gegen A, dann ist Ausdruck (3.2) subpolynomiell für jeden Unterscheider und insbesondere für jeden Vorhersager M, also auch die Differenz (3.3) zwischen Erfolgs- und Irrtumswahrscheinlichkeit von M. Aber aus $p_r - p_f \leq \varepsilon$ und $p_r + p_f = 1$ erhalten wir $p_r \leq 1/2 + \varepsilon/2$, also weicht die Erfolgswahrscheinlichkeit von M nur subpolynomiell von $1/2$ ab und S_0 ist V-privat.

Sei S_0 nun V-privat, dann ist die Erfolgswahrscheinlichkeit eines jeden Vorhersagers (und eines jeden Unterscheiders) nur subpolynomiell verschieden von $1/2$, ebenso

die Irrtumswahrscheinlichkeit. Damit ist deren Differenz (3.3) und als Folge Ausdruck (3.2) subpolynomiell. Dasselbe ist der Fall für $\overline{M}$, der genau dann eine 1 ausgibt, wenn M eine 0 ausgibt. Aber (3.2) für $\overline{M}$ ist nur das Negative von (3.2) für M, also ist der Absolutbetrag von (3.2) subpolynomiell und S_0 ist U-privat. □

3.22. Satz *U-Privatheit auf der Ereignismenge* L *bedeutet E-Privatheit auf beliebigen Eingabe-Ensembles konzentriert auf* L.

Beweis: Sei U ein Unterscheider, der i, t, a, e als Eingabe bekommt, dann ist für alle e

$$\begin{aligned}
&\Big|\sum_{a,t}\Big(p_i(a,t|e) - p_i(a|e)p_i(t|e)\Big)\pi_U((i,t,e,a),1)\Big| \\
&= \Big|\sum_{a,t}\Big(p_i(t|e)p_i(a|t,e) - p_i(t|e)\sum_{t'}p_i(t'|e)p_i(a|t',e)\Big)\pi_U((i,t,e,a),1)\Big| \\
&= \Big|\sum_{t}p_i(t|e)\sum_{t'}p_i(t'|e)\sum_{a}\Big(p_i(a|t,e) - p_i(a|t',e)\Big)\pi_U((i,t,e,a),1)\Big| \\
&\le \sum_{t}p_i(t|e)\sum_{t'}p_i(t'|e)\Big|\sum_{a}\Big(p_i(a|t,e) - p_i(a|t',e)\Big)\pi_U((i,t,e,a),1)\Big|
\end{aligned}$$

Spalten wir die Summationen über t und t' auf in Situationen aus L_i und solche nicht aus L_i, erhalten wir

$$\begin{aligned}
&\sum_{t,t'\in L_i}\cdots + \sum_{t\in L_i,\,t'\notin L_i}\cdots + \sum_{t\notin L_i,\,t'\in L_i}\cdots + \sum_{t,t'\notin L_i}\cdots \\
&\le \underbrace{\sum_{t\in L_i}p_i(t|e)}_{\le 1}\underbrace{\sum_{t'}p_i(t'|e)}_{\le 1}\underbrace{\Big|\sum_{a}\Big(p_i(a|t,e) - p_i(a|t',e)\Big)\pi_U((t,e,a),1)\Big|}_{\le |i|^{-c}} \\
&\quad +3\sum_{t\notin L_i}p_i(t|e)\underbrace{\sum_{t'}p_i(t'|e)}_{\le 1}\underbrace{\Big|\sum_{a}\Big(p_i(a|t,e) - p_i(a|t',e)\Big)\pi_U((t,e,a),1)\Big|}_{\le 1} \\
&\le |i|^{-c} + 3\sum_{t\notin L_i}p_i(t|e).
\end{aligned}$$

Summieren über $\sum_e p(e)$ liefert

$$|i|^{-c} + 3\sum_{t\notin L_i}p_i(t) \;\le\; |i|^{-c} + 3|i|^{-c} \;\le\; |i|^{-c+1}$$

für $|i| \ge 4$, da das Eingabe-Ensemble im wesentlichen auf L ist und die Gesamtwahrscheinlichkeit von Ereignissen nicht aus L subpolynomiell. □

3.23. Satz *E-Privatheit auf Eingabe-Ensemble* X *bedeutet R-Privatheit auf* X.

Beweis: E-Privatheit bedeutet, daß für alle Unterscheider U, alle $c > 0$ und hinreichend große $|i|$

$$\sum_e p(e)\Big|\sum_{t,a}\big(p_i(t,a|e) - p_i(t|e)p_i(a|e)\big)\pi_U((i,t,e,a),1)\Big| \;\le\; |i|^{-c}$$

ist. Wenden wir die Dreiecksungleichung auf die linke Seite an, erhalten wir

$$\sum_e p(e)\sum_{t,a} p_i(t,a|e)\pi_U((i,t,e,a),1) \\ -\sum_e p(e)\sum_t p_i(t|e)\sum_a p_i(a|e)\pi_U((i,t,e,a),1) \;\le\; |i|^{-c}.$$

Sei D ein Algorithmus von $L(X)$ nach Z und R ein Rate-Algorithmus R für D mit Zugriff auf (E,A). Daraus bauen wir einen Unterscheider U mit $U(i,t,e,a) = 1 \iff D(i,t) = R(i,e,a)$. Dann ist $\pi_U((i,t,e,a),1) = \mathrm{Korr}_{D,R}(i,t,e,a)$ und für hinreichend große $|i|$

$$\begin{aligned}
\mathrm{Succ}_{D,R}(i) &= \sum_{t,e,a} p_i(t,e,a)\sum_z \pi_D((i,t),z)\pi_R((i,e,a),z) \\
&= \sum_e p_i(e)\sum_{t,a} p_i(t,a|e)\,\mathrm{Korr}_{D,R}(i,t,e,a) \\
&= \sum_e p_i(e)\sum_{t,a} p_i(t,a|e)\pi_U((i,t,e,a),1) \\
&\le \sum_e p_i(e)\sum_t p_i(t|e)\sum_a p_i(a|e)\pi_U((i,t,e,a),1) + |i|^{-c} \\
&\le \sum_e p_i(e)\sum_t p_i(t|e)\pi_U((i,t,e,\mathrm{amax}(i,e)),1)\underbrace{\sum_a p_i(a|e)}_{=1} + |i|^{-c}.
\end{aligned}$$

Mit $R'(i,e) = R(i,e,\mathrm{amax}(i,e))$ ist $\mathrm{Korr}_{D,R'}(i,t,e) = \pi_U((i,t,e,\mathrm{amax}(i,e))1)$ und

$$\mathrm{Succ}_{D,R}(i) \;\le\; \mathrm{Succ}_{D,(R',\mathrm{amax})}(i) + |i|^{-c}$$

für den in (i,e) nichtuniformen Algorithmus (R',amax). □

3.24. Satz *R-Privatheit auf allen Eingabe-Ensembles über* $\overline{L}$ *bedeutet V-Privatheit der wesentlichen Ereignismenge* L.

Beweis: Sei c_L der Wesentlichkeitsexponent von L. Angenommen, S ist nicht V-privat auf L, dann gibt es einen Vorhersager $V \in \mathcal{PP}$, ein $c_V > 0$, unendlich viele $i \in I$ und zugehörige Situationspaare $t_{i,0} = (x_{i,0}, y_{i,0})$, $t_{i,1} = (x_{i,1}, y_{i,1})$ mit

$$\pi_{S_0}((i,x_{i,j}),y_{i,j}) \ge |i|^{-c_L} \quad \text{und} \quad E_A(t_{i,j}) = e_i \qquad \text{für} \quad j \in \{0,1\} \tag{3.4}$$

und

$$\sum_{j\in\{0,1\}} \tfrac{1}{2}\sum_{a} p(a|t_{i,j})\pi_V((i,t_{i,0},t_{i,1},e_i,a),j) \;>\; \tfrac{1}{2}+|i|^{-c_V}. \tag{3.5}$$

Der Algorithmus $D(i,t)$ gibt j aus für $t = t_{i,j}$, ansonsten 0 oder 1 mit Wahrscheinlichkeit $\frac{1}{2}$. Damit ist er (nichtuniform in i) aus $\mathcal{PP}$.

Wir müssen zeigen, daß es ein Eingabe-Ensemble auf L und einen Rate-Algorithmus $R \in \mathcal{PP}$ für D mit Zugriff auf (E,A) gibt, der um einen polynomiellen Bruchteil besser ist als *jeder* Rate-Algorithmen R' mit Zugriff auf E. Nach Konstruktion von D ist für jedes R'

$$\begin{aligned}
\mathrm{Succ}_{D,R'}(i) &= \sum_{t,e} p_i(t,e)\,\mathrm{Korr}_{D,R'}(i,t,e)\\
&= \sum_{t} p_i(t)\sum_{e}\delta(E_A(t),e)\sum_{j\in\{0,1\}}\pi_D((i,t),j)\pi_{R'}((i,e),j)\\
&= p_i(t_{i,0})\pi_{R'}((i,e_i),0)+p_i(t_{i,1})\pi_{R'}((i,e_i),1)\\
&\quad+\sum_{t\neq t_{i,0},t_{i,1}} p_i(t)\tfrac{1}{2}\underbrace{\sum_{j\in\{0,1\}}\pi_{R'}((i,E_A(t)),j)}_{=1}\\
&= p_i(t_{i,0})(\pi_{R'}((i,e_i),0)-\tfrac{1}{2})+p_i(t_{i,1})(\pi_{R'}((i,e_i),1)-\tfrac{1}{2})+\tfrac{1}{2}\\
&= \tfrac{1}{2}+(p_i(t_{i,0})-p_i(t_{i,1}))(\pi_{R'}((i,e_i),0)-\tfrac{1}{2}).
\end{aligned}$$

Nun ist $p_i(t_{i,j}) = p'_i(x_{i,j})\pi_{S_0}((i,x_{i,j}),y_{i,j})$, und wenn wir

$$p'_i(x_{i,j}) = \frac{\pi_{S_0}((i,x_{i,1-j}),y_{i,1-j})}{\pi_{S_0}((i,x_{i,0}),y_{i,0})+\pi_{S_0}((i,x_{i,1}),y_{i,1})}$$

setzen, ist für diese Eingabeverteilung $p_i(t_{i,0}) = p_i(t_{i,1})$. Nach (3.4) sind beide Wahrscheinlichkeiten größer gleich $\frac{1}{2}|i|^{-2c_L}$, was wir später verwenden werden. Zunächst ist aber mit dieser Eingabeverteilung $\mathrm{Succ}_{D,R'}(i) = \frac{1}{2}$ für *jedes* R'.

Unser Rate-Algorithmus R mit Zugriff auf (E,A) wählt für $e \neq e_i$ ein $j \in \{0,1\}$ mit Wahrscheinlichkeit $\frac{1}{2}$, für $e = e_i$ gibt er $V(i,t_{i,0},t_{i,1},e_i,a)$ aus. Dann ist

$$\begin{aligned}
\mathrm{Succ}_{D,R}(i) &= \sum_{t,e,a} p_i(t,e,a)\,\mathrm{Korr}_{D,R}(i,t,e,a)\\
&= \sum_{t} p_i(t)\sum_{e}\delta(E_A(t),e)\sum_{a} p_i(a|t)\sum_{j\in\{0,1\}}\pi_D((i,t),j)\pi_R((i,e,a),j)\\
&= \sum_{j\in\{0,1\}} p_i(t_{i,j})\sum_{a} p_i(a|t)\pi_V((i,t_{i,0},t_{i,1},e_i,a),j)+\tfrac{1}{2}\sum_{t\neq t_{i,0},t_{i,1}} p_i(t).
\end{aligned}$$

Da wir $p_i(t_{i,0}) = p_i(t_{i,1})$ erreicht hatten, läßt sich die erste Summe schreiben als

$$2p_i(t_{i,0})\sum_{j\in\{0,1\}}\tfrac{1}{2}\sum_{a} p_i(a|t)\pi_V((i,t_{i,0},t_{i,1},e_i,a),j) \;>\; 2p_i(t_{i,0})(\tfrac{1}{2}+|i|^{-c_V})$$

nach (3.5). Insgesamt ist damit

$$\mathrm{Succ}_{D,R}(i) \;>\; \tfrac{1}{2} + 2p_i(t_{i,0})|i|^{-c_V} \;\geq\; \tfrac{1}{2} + |i|^{-(2c_L+c_V)},$$

was zu beweisen war. □

3.25. Satz *E-Privatheit auf Eingabe-Ensemble* X *bedeutet H-Privatheit auf* X.

Beweis: Die Beweisidee ist, eine Quellenkodierung für T relativ zu (E, A) zu nehmen und „gute" a's in die nichtuniforme Zusatzinformation zu stecken. Aufgrund der E-Privatheit sind diese „guten" a's gleichmäßig gut für alle t's mit $E_A(t) = e$, d.h. die entstehende Quellenkodierung für T relativ zu E ist nichtuniform in (i, e). Zu zeigen sind zwei Dinge: die Kodierung muß gut dekodieren können, und ihre erwartete Kodelänge darf nur subpolynomiell größer sein. Da der Trennalgorithmus C unabhängig von der relativen Information arbeitet, bleibt er unverändert gut. Man beachte, daß die Kodier- und Dekodieralgorithmen polynomiell lange Tupel von Situationen plus relativer Information bekommen. Nach Lemma 3.6 ändert dies nichts an der Ununterscheidbarkeitseigenschaft der E-Privatheit, wir schreiben der Einfachheit halber weiter t, e und a, obwohl es sich dabei um Tupel handelt.

Sei nun (E, D, C) eine Quellenkodierung für T relativ zu (E, A) und sei M ein Unterscheider mit $M(i, t, e, a) = 1 \iff D(iE(i, t, e, a), e, a) = t$.

Für beliebige i, e und a sei

$$P_i(e, a) = \sum_t p_i(t|e) \sum_\alpha \pi_E((i, t, e, a), \alpha)\pi_D((i, \alpha, e, a), t).$$

Dies ist der Dekodiererfolg für vorgegebenes i, e und a. Ein a ist ε-gut für (i, e), wenn $P_i(e, a) \geq \varepsilon$ ist. Betrachten wir also für festes i diejenigen e, die ein ε-gutes a besitzen. Wegen der E-Privatheit ist für alle $c > 0$ und hinreichend großen $|i|$

$$\begin{aligned}
&\sum_e p_i(e) \sum_a p_i(a|e) P_i(e, a) \\
&= \sum_e p_i(e) \sum_a p_i(a|e) \sum_t p_i(t|e) \sum_\alpha \pi_E((i, t, e, a), \alpha)\pi_D((i, \alpha, e, a), t) \\
&\geq \sum_e p_i(e) \sum_{t,a} p_i(t, a|e) \sum_\alpha \pi_E((i, t, e, a), \alpha)\pi_D((i, \alpha, e, a), t) \;-\; |i|^{-c}.
\end{aligned}$$

Der linke Term der letzten Zeile ist aber der Dekodiererfolg von (E, D, C), der für hinreichend großen $|i|$ größer als $1 - |i|^{-c}$ ist. Damit erhalten wir

$$\begin{aligned}
1 - 2|i|^{-c} \;&\leq\; \sum_e p_i(e) \sum_a p_i(a|e) P_i(e, a) \\
&\leq \sum_{e \text{ hat ein } \varepsilon\text{-gutes } a} p_i(e) \sum_a p_i(a|e) \cdot 1 \\
&\quad + \sum_{e \text{ hat kein } \varepsilon\text{-gutes } a} p_i(e) \sum_a p_i(a|e) \cdot \varepsilon \\
&= p_{\text{hat a}} + (1 - p_{\text{hat a}})\varepsilon \;=\; (1 - \varepsilon)p_{\text{hat a}} + \varepsilon,
\end{aligned}$$

oder $p_{\text{hat a}} \geq 1 - {}^{2}/_{1-\varepsilon}|i|^{-c}$, wobei $p_{\text{hat a}}$ die Gesamtwahrscheinlichkeit aller e mit ε-gutem a ist. Mit $\varepsilon = 1 - 2|i|^{-c/2}$ erhalten wir $p_{\text{hat a}} \geq 1 - |i|^{-c/2}$.

Bauen wir nun einen Kode (E', D', C'), der für jedes i und e ein vorher festgelegtes $a_{i,e}$ zum Kodieren verwendet, und zwar ein ε-gutes, falls vorhanden. Dann ist der Dekodiererfolg aller so gebildeten Kodes

$$\begin{aligned}\sum_e p_i(e)P_i(e, a_{i,e}) &\geq \sum_{e \text{ hat ein } \varepsilon\text{-gutes } a} p_i(e)\varepsilon \\ &= p_{\text{hat a}}\varepsilon \geq (1 - |i|^{-c/2})(1 - 2|i|^{-c/2}) \geq 1 - 4|i|^{-c/2},\end{aligned}$$

also nur subpolynomiell kleiner als 1.

Betrachten wir für festes i die Gesamtwahrscheinlichkeit guter Paare (e, a), dann erhalten wir wieder

$$\begin{aligned}1 - 2|i|^{-c} &\leq \sum_{e,a} p_i(e, a)P_i(e, a) \\ &\leq \sum_{(e,a) \text{ gut}} p_i(e, a) \cdot 1 + \sum_{(e,a) \text{ schlecht}} p_i(e, a) \cdot \varepsilon \\ &= p_{\text{gut}} + (1 - p_{\text{gut}})\varepsilon = (1 - \varepsilon)p_{\text{gut}} + \varepsilon,\end{aligned}$$

oder $p_{\text{gut}} \geq 1 - |i|^{-c/2}$, wir haben also viele gut dekodierbare Kodes mit Zugriff nur auf E zur Auswahl. Wählen wir den kürzesten, d.h. $\text{amin}(i, e)$ ist dasjenige ε-gute a für (i, e) mit minimaler erwarteter Kodelänge

$$\sum_t p_i(t|a, e) \sum_\alpha \pi_E((i, t, e, a), \alpha)|\alpha|.$$

Dieser Kode (E', D', C') ist nach den vorangegangenen Überlegungen gut dekodierbar. Die erwartete Kodelänge kann abgeschätzt werden durch

$$\begin{aligned}E_{(E',D',C')}(i) &= \sum_{e,a} p_i(e, a) \sum_\alpha \pi_E((i, t, e, \text{amin}(i, e)), \alpha)|\alpha| \\ &= \sum_{(e,a) \text{ gut}} p_i(e, a) \sum_t p_i(t|e, a) \sum_\alpha \pi_E((i, t, e, \text{amin}(i, e)), \alpha)|\alpha| \\ &\quad + \sum_{(e,a) \text{ schlecht}} p_i(e, a) \sum_t p_i(t|e, a) \sum_\alpha \pi_E((i, t, e, \text{amin}(i, e)), \alpha)|\alpha| \\ &\leq \sum_{e,a} p_i(e, a) \sum_t p_i(t|e, a) \sum_\alpha \pi_E((i, t, e, a), \alpha)|\alpha| \\ &\quad + \sum_{(e,a) \text{ schlecht}} p_i(e, a) \sum_t p_i(t|e, a) \sum_\alpha \pi_E((i, t, e, \text{amin}(i, e)), \alpha)|i|^t \\ &\leq E_{(E,D,C)}(i) + (1 - p_{\text{gut}})|i|^t \\ &\leq E_{(E,D,C)}(i) + |i|^{t-c/2}.\end{aligned}$$

Dabei benutzten wir, daß E polynomiell beschränkte Laufzeit hat. Die Konstante t ist fest, damit ist die erwartete Kodelänge von (E', D', C') höchstens subpolynomiell größer als die von (E, D, C), was zu zeigen war. □

3.26. Satz *U-Privatheit auf der wesentlichen Ereignismenge* L *bedeutet S-Privatheit auf* L*, wenn das gestörte Protokoll* S *erwartet polynomielle Laufzeit auf* L *hat.*

Beweis: Wir konstruieren einen Simulator M, der das gestörte System S benutzt und nichtuniform in (i, e) ist. Sei $nu(i, e) = t_e$ für ein beliebiges $t_e = (x_e, y_e) \in L_i$ mit $E_A(t_e) = e$. Unser Simulator $M(i, e) = Ex_S(i, x_e, y_e)$ führt das Experiment Ex_S mit der Eingabe $(i, nu(i, e))$ durch. Nach den Überlegungen in Unterabschnitt 3.4.3 ist die Laufzeit von Ex_S und damit die unseres Simulators M erwartet polynomiell auf L, weiter ist

$$\pi_M((i, e), a) \;=\; \pi_{Ex_S}((i, x_e, y_e), a) \;=\; p_i(a|t_e) \;=\; p_i(a|t_e, , e).$$

Nach Definition der U-Privatheit ist daher für alle Unterscheider U, alle $c > 0$, alle hinreichend großen $|i|$ und alle $t \in L_i$

$$\begin{aligned}
&\Big|\sum_a (p_i(a|t) - \pi_M((i, E_A(t)), a))\pi_U((i, t, E_A(t), a), 1)\Big| \\
&= \Big|\sum_a (p_i(a|t, E_A(t)) - p_i(a|t_{E_A(t)}, E_A(t)))\pi_U((i, t, E_A(t), a), 1)\Big| \\
&\leq |i|^{-c},
\end{aligned}$$

d.h. S ist S-privat auf L. □

3.27. Satz *S-Privatheit auf der Ereignismenge* L *bedeutet R-Privatheit auf allen Eingabe-Ensembles über* L.

Beweis: Sei M ein Simulator, sei $D \in \mathcal{PP}$ ein Algorithmus von $L(X) \subset L$ nach Z und R ein Ratealgorithmus für D mit Zugriff auf (E, A). Mit $R'(i, e) = R(i, e, M(i, e))$ ist R' ein Ratealgorithmus für D mit Zugriff nur auf E. Wegen der S-Privatheit auf L ist für jedes Eingabe-Ensemble auf L

$$\begin{aligned}
Succ_{D,R'}(i) &= \sum_{t,e} p_i(t, e) \sum_z \pi_D((i, t), z)\pi_{R'}((i, e), z) \\
&= \sum_{t,e} p_i(t, e) \sum_z \pi_D((i, t), z) \sum_a \pi_M((i, e), a)\pi_R((i, e, a), z) \\
&= \sum_{t,e} p_i(t, e) \sum_a \pi_M((i, e), a) \sum_z \pi_D((i, t), z)\pi_R((i, e, a), z) \\
&\overset{(*)}{\geq} \sum_{t,e} p_i(t, e) \sum_a p_i(a|t, e) \sum_z \pi_D((i, t), z)\pi_R((i, e, a), z) - |i|^{-c} \\
&= \sum_{t,e,a} p_i(t, e, a)\, Korr_{D,R}(i, t, e, a) - |i|^{-c} \\
&= Succ_{D,R}(i) - |i|^{-c}.
\end{aligned}$$

Für $(*)$ benutzen wir den Unterscheider $U(i,t,e,a) = 1 \iff D(i,t) = R(i,e,a)$, für den

$$= \pi_U((i,t,e,a),1) \;=\; \sum_z \pi_D((i,t),z)\pi_R((i,e,a),z)$$

ist, sowie die algorithmische Ununterscheidbarkeit von $\pi_M((i,e),a)$ und $p_i(a|t,e)$ für alle $t \in L_i$ mit $E_A(t) = e$. Für alle übrigen t ist $p_i(t,e) = p_i(t)\delta(E_A(t),e)$ gleich 0. Damit rät R' fast so gut wie R und S ist R-privat. □

3.4.9 Privatheit bei speziellen Zielen

Bei genauem Ansehen der im vorhergehenden Abschnitt bewiesenen Relationen zwischen den Privatheitsansätzen fällt auf, daß Privatheit auf einer Ereignismenge nur Privatheit auf allen Eingabe-Ensembles bedeutet, die auf diese Ereignismenge konzentriert sind, und daß nur aus der Privatheit auf allen Eingabe-Ensembles über einer wesentlichen Ereignismenge Privatheit auf dieser Ereignismenge folgt. Für eine große Klasse von Protokollen sind diese Einschränkungen aber unwesentlich, wie wir in diesem Unterabschnitt zeigen werden.

Einer Zielspezifikation z für die Eingabefamilie X_I entspricht eine Ereignismenge

$$L_z = \{(i,x,y) \in \hat{X}_I \times \mathrm{Ran}(S_0) | z(i,x)(y) > 0\}.$$

Diese Ereignismenge ermöglicht es, die Unstetigkeiten in den vorangegangenen Sätzen zu glätten.

3.28. Lemma *Ist ein Protokoll S_0 korrekt für eine Zielspezifikation z auf einer Eingabefamilie $X_I = (M_X,(X_i)_{i\in I})$, dann ist jedes Eingabe-Ensemble X auf X_I konzentriert auf L_z.*

Beweis: Nach Definition der Korrektheit ist für alle $c > 0$, alle hinreichend großen $|i|$ und alle $x \in X_i$

$$\begin{aligned} |i|^{-c} &\geq \sum_{y\in \mathrm{Ran}(S_0} |z(i,x)(y) - \pi_{S_0}((i,x),y)| \\ &= \sum_{y\in \mathrm{supp}(z(i,x))} |z(i,x)(y) - \pi_{S_0}((i,x),y)| \;+\; \sum_{y\notin \mathrm{supp}(z(i,x))} \pi_{S_0}((i,x),y) \\ &\geq \sum_{y\notin \mathrm{supp}(z(i,x))} \pi_{S_0}((i,x),y). \end{aligned}$$

Dann ist

$$\sum_{(x,y)\notin L_{z,i}} p'_i(x)\pi_{S_0}((i,x),y) \;=\; \sum_{x\in X_i} p'_i(x) \underbrace{\sum_{y\notin \mathrm{supp}(z(i,x))} \pi_{S_0}((i,x),y)}_{\leq |i|^{-c}} \;\leq\; |i|^{-c},$$

also X konzentriert auf L_z. □

Damit haben wir bereits eine passende Ereignismenge, die jetzt nur noch wesentlich sein muß.

3.29. Definition *Eine Zielspezifikation z für ein Protokoll S_0 auf einer Eingabefamilie $X_I = (M_X, (X_i)_{i\in I})$ heißt* **konzentriert**, *wenn sie nur wesentliche Ereignisse fordert, d.h. für eine Konstante $c_z > 0$, alle hinreichend großen $|i|$, alle $x \in X_i$ und alle $y \in \mathrm{supp}(z(i,x))$ ist $z(i,x)(y) \geq |i|^{-c_z}$.*

Alle deterministischen Zielspezifikationen sind damit konzentriert, d.h. insbesondere solche für Beweissysteme, Kryptosyteme und private Funktionsauswertung, aber auch Zielspezifikationen für **Oblivious Transfer**-Protokolle, bei dem ein Sender einem Empfänger eine Nachricht schickt, die nur mit Wahrscheinlichkeit $1/2$ ankommt.

3.30. Lemma *Ist ein Protokoll S_0 korrekt für eine konzentrierte Zielspezifikation z auf einer Eingabefamilie X_I, dann ist L_z wesentlich für S_0.*

Beweis: Aus der Korrektheit folgt für alle $c > 0$, alle $|i| \geq n_c$ und alle $(x,y) \in L_{z,i}$, daß $|z(i,x)(y) - \pi_{S_0}((i,x),y)| \leq |i|^{-c}$ ist. Andererseits ist für $z(i,x)(y) \geq |i|^{-c_z}$ für hinreichend großen $|i|$ und $(x,y) \in L_{z,i}$, also

$$\pi_{S_0}((i,x),y) \;\geq\; z(i,x)(y) - |z(i,x)(y) - \pi_{S_0}((i,x),y)| \;\geq\; |i|^{-c_z} - |i|^{-c} \;\geq\; |i|^{-2c_z}$$

für $|i| \geq \max\{n_{2c_z}, 2^{1/c_z}\}$. Damit ist L_z wesentlich für S_0. □

Damit erhalten wir

3.31. Satz *Ist ein Protokoll S_0 korrekt für ein konzentriertes Ziel z auf einer Eingabefamilie X_I, dann sind äquivalent:*

- *S ist V-privat auf L_z,*
- *S ist E-privat auf allen Ensembles auf X_I,*
- *S ist R-privat auf allen Ensembles auf X_I,*
- *S ist U-privat auf L_z.*

Für erwartet polynomielles S sind diese auch äquivalent zu

- *S ist S-privat auf L_z.*

Beweis: Nach Lemma 3.28 sind alle Eingabe-Ensembles auf X_I konzentriert auf L_z, nach Lemma 3.30 ist L_z wesentlich für S_0, weiter sind alle Eingabe-Ensembles auf $\overline{L_z}$ auch Eingabe-Ensembles auf X_I. □

Die algorithmische Version der Entropie-Privatheit ist auch in diesem eingeschränkten Fall schwächer als die übrigen Privatheitsbegriffe. Für den Fall von Public-Key-Kryptosystemen wurde in [MicRackSlo88] aber die Äquivalenz zwischen V-Privatheit und H-Privatheit gezeigt. Die dort gezeigten Beweise lassen sich nicht auf den allgemeinen Fall übertragen. Dies ist aber nicht weiter tragisch, denn nach Lemma 3.21, Satz 3.22 und Satz 3.25 impliziert V-Privatheit auch im allgemeinen Fall H-Privatheit. Für den Spezialfall deterministischer Ziele und korrekter Protokolle können wir eine Idee aus [MicRackSlo88] nutzen, um auch die Rückrichtung zu zeigen:

3.32. Lemma *Ist S_0 korrekt für ein deterministisches Ziel z auf der Eingabefamilie X_I, dann bedeutet H-Privatheit auf allen Eingabe-Ensembles auf X_I V-Privatheit auf L_z.*

Beweis: Angenommen, S ist nicht V-privat auf L_z, dann gibt es eine unendliche Zahl von Eingabepaaren $(x_{i,0}, x_{i,1})$ mit $E_A(x_{i,0}, z(i, x_{i,0})) = E_A(x_{i,1}, z(i, x_{i,1})$ und einen Vorhersager M, der mit Wahrscheinlichkeit größer $1/2 + |i|^{-k}$ rät, welches der beiden wesentlichen Ereignisse $(i, x_{i,j}, z(i, x_{i,j}))$ eine gegebene Angreiferausgabe erzeugt hat. Wesentlich deshalb, weil $\pi_{S_0}((i, x_{i,j}), z(i, x_{i,j}))$ wegen der Korrektheit von S_0 nur subpolynomiell von 1 abweicht.

Wählen wir die beiden Eingaben gleichverteilt, dann ist $p_i(x_{i,j}, z(i, x_{i,j}))$ nur subpolynomiell verschieden von $1/2$. Die klassische Entropie eines gleichverteilten Paares ist 1, d.h. ein Bit reicht aus, um mitzuteilen, welche Eingabe verwendet wurde. Wegen der Stetigkeit der Entropie ist die Entropie unserer Ereigispaare ohne Kenntnis der Angreiferausgaben auch nur subpolynomiell kleiner als 1.

Nach Satz 3.19 ist die algorithmische Entropie nur subpolynomiell kleiner als die klassische Entropie, wir müssen also eine Quellenkodierung konstruieren, die bei Benutzung der Angreiferausgaben einen polynomiellen Bruchteil weniger als ein Bit pro Eingabe benötigt.

Dies erreichen wir, indem wir nicht die Folge der entscheidenden Bits einer polynomiellen Folge von Ereignissen übertragen, sondern die Folge der Exklusiv-Oder von jeweils zwei aufeinanderfolgenden Bits. Damit sparen wir ein Bit in der Übertragung ein. Es gibt nun, je nach Wahl des fehlenden Bits, zwei mögliche Dekodierungen mit maximaler Hamming-Distanz, d.h. sie unterscheiden sich in jeder Stelle. Eine dritte Bitfolge wird vom Vorhersager erzeugt, der auf der Folge der Ausgaben arbeitet. Die mögliche Dekodierung mit der geringeren Hamming-Distanz zur Bitfolge des Vorhersagers wird als Dekodierung gewählt.

Da dies ein Mehrheitsentscheid ist, stimmt diese Dekodierung mit nur exponentiell (und damit subpolynomiell) kleinem Fehler, also ist

$$H_a(T|E,A)(i) \;<\; 1 - |i|^{-k} = H(T|E)(i) - |i|^{-k} \;<\; H_a(T|E)(i) - |i|^{-k}$$

für eine unendliche Zahl von i's im Widerspruch zur H-Privatheit von S auf allen Eingabe-Ensembles auf X_I. □

3.5 Robustheit

Robustheit bedeutet, daß das Protokoll bei Störungen nichts wesentlich anderes tut als ohne Störungen, also jeder Störungsversuch kaum mehr Auswirkungen auf den Protokollablauf hat als eine Abänderung der Ein- bzw. Ausgaben der vom Angreifer angegriffenen Teilnehmer und des Angreifers selbst. Diese Änderungen sind durch das Protokoll *nicht* zu verhindern, da die Eingabeänderung *vor* der ersten Kommunikation und die Ausgabeänderung *nach* der letzten Kommunikation stattfindet. Das Restsystem, d.h. die korrekt gebliebenen Teilnehmer, haben dann noch keinen Einfluß bzw. keinen Einfluß mehr auf den Angreifer. Maximale Robustheit bedeutet also, daß man jeden Angreifer ersetzen kann durch einen ein- und ausgabeverändernden einflußlosen Angreifer.

3.33. Definition *Ein Protokoll* S_0 *ist* **robust** *gegen den Angreifer* $A \in \mathrm{Adv}(S_0)$ *auf der Eingabefamilie* $X_I = (M_X, (X_i)_{i\in I})$, *wenn es einen einflußlosen Angreifer* $A' = (C_{A'}, P_{A'}, \nu_{A'}, e_{A'}, a_{A'}) \in \mathrm{eAdv}(S_0)$ *auf dieselben Teilnehmer und in probabilistischer Polynomzeit berechenbare Funktionen* e'_A *und* a'_A *gibt, so daß sich* $S = \mathrm{aProt}(S_0, A)$ *ungefähr wie* $S'' = \mathrm{aProt}(S_0, A'')$ *verhält mit* $A'' = (C_{A'}, P_{A'}, \nu_{A'}, e_{A'} \circ e'_A, a'_A \circ a_{A'})$, *d.h. für alle* $c > 0$, *alle hinreichend großen* $|i|$ *und alle* $x \in X_i$ *ist*

$$\sum_{(y,a)\in \mathrm{Ran}(S_0)\times \mathrm{Ran}(A)} |\pi_S((i,x),(y,a)) - \pi_{S''}((i,x),(y,a))| \;\leq\; |i|^{-c}.$$

Private Protokolle sind auch nach einer derartigen Störung noch maximal privat, als Anschauung diene die zweiargumentige Funktion f

$A\backslash B$	0	1
0	a	b
1	a	c

die durch folgendes Protokoll privat berechnet wird, siehe auch [Kush]:

$$\begin{aligned} B \to A &: x_B \\ A \to B &: f(x_A, x_B) \end{aligned}$$

Dieses Protokoll ist privat, da A *immer* die Eingabe von B aus dem Wert von f erkennen kann, B die Eingabe von A aber nur für $x_B = 1$.

Es gibt hier nur eine Betrugsmöglichkeit: B fälscht seine Eingabe und schickt immer eine 1 an A, damit erfährt B immer die Eingabe von A und den beabsichtigten Funktionswert, A erhält nur dann den korrekten Wert, wenn x_B ohnehin gleich 1 ist.

Dieser Betrug ist durch nichts zu verhindern, das Protokoll ist maximal privat, obwohl es alles verrät.

Der hier beschriebene Ansatz zur Robustheit ist weit verbreitet:

- [Yao86] fordert die Ununterscheidbarkeit des gestörten Protokolls von einer analog gestörten Zielspezifikation, dies ist aber äquivalent, wenn das ungestörte Protokoll korrekt ist.
- [GoldwLev90] haben den Begriff des *legalen* (aber unmoralischen) *Fehlers*, bei dem Ein- und Ausgaben der angegriffenen Teilnehmer beliebig verändert werden dürfen, ansonsten aber das ungestörte Protokoll abläuft. Robustheit bedeutet, daß es zu einer beliebigen angegriffenen Version eines Protokolls eine ununterscheidbare Version mit legalem Fehler gibt.
- [Beaver90, Beaver91-2] fordert die Existenz einer in polynomieller Laufzeit ausführbaren Reduktion von Angreifern für das tatsächliche Protokoll auf Angreifer für ein idealisiertes Protokoll, sogenannte *Interfaces*. Im idealisierten Protokoll dürfen nur Ein- und Ausgaben der angegriffenen Teilnehmer verändert werden dürfen. Allerdings soll dasselbe Interface für alle betrachteten Angreifer taugen, also eine härtere Forderung als bei der in dieser Arbeit geforderten Robustheit.
- [MicRog, Rog91] ersetzen das Interface auf ein idealisiertes Protokoll durch einen Simulator mit Zugriff auf ein Ein-/Ausgabe-Orakel, was im wesentlichen dasselbe ist.

Damit sind die drei wichtigsten Protokolleigenschaften definiert.

Schlußwort

Neben weitgehend identischen Definitionen von Korrektheit und Robustheit finden sich in der neueren kryptographischen Literatur Privatheitsdefinitionen, die in starkem Maße durch Anschauung motiviert sind. Trotz ihres zum Teil sehr unterschiedlichen Aussehens sind sie weitgehend äquivalent und lassen sich auf äquivalente Spielarten der klassischen informationstheoretischen Privatheit zurückführen.

Als wesentlich erwies sich dabei die Trennung von *Information* (Privatheit) und *Einfluß* (Robustheit). Die Einteilung in einflußlose und einflußnehmende Angreifer kann sich drastisch unterscheiden von der bisher üblichen Unterteilung in passive und aktive Angreifer, so sind z.B. bei Zero-Knowledge-Beweissystemen höchst aktive Angreifer auf den Verifier ohne Einfluß auf das Akzeptanzverhalten.

Durch die weitgehende Abstraktion der Sicherheitsdefinitionen läßt sich das zugrundeliegende Maschinenmodell einfach erweitern, ohne daß die einzelnen Sicherheitsdefinitionen wesentlich abgeändert werden müßten oder ihre Äquivalenz in Frage gestellt würde.

Zusammen mit dem vorgestellten Modell interaktiver Turingmaschinen und interaktiver Systeme lassen sich die meisten kryptographischen Fragestellungen einheitlich formulieren. Die vorliegende Arbeit bildet also eine Basis für die einheitliche Formalisierung aller kryptographischen Primitive und damit für die Untersuchung ihrer gegenseitigen Abhängigkeiten.

Auch die Erfassung weiterer Protokolleigenschaften wird durch diesen Formalismus vereinfacht, was wir abschließend am Beispiel der Betrugserkennung skizzieren.

Hier soll das Protokoll Versuche der Einflußnahme während des Protokollablaufes nicht nur unterbinden, es soll darüberhinaus aktiv auf jeden derartigen Versuch reagieren. Ein spezieller Eingabewert `Cheat!` emöglicht es, die Reaktion auf erkanntes abweichendes Verhalten festzulegen.

Ein Protokoll ist dann betrugserkennend, wenn es korrekt ist für ein entsprechendes Ziel und alle einflußlosen Angreifer passiv sind, d.h. jede Abweichung vom Protokoll erkannt und bestraft wird.

Literaturverzeichnis

[AbaFeiKil89] Martin Abadi, Joan Feigenbaum und Joe Kilian, *On hiding information from an oracle*, JCSS **39** (1989), 21–50.

[Beaver89] Donald Rozinak Beaver, *Multiparty protocols tolerating half faulty processors*, Crypto, 1989, 560–572.

[Beaver90] ———, *Security, fault tolerance, and communication complexity in distributed systems*, Dissertation, Harvard, May 1990.

[Beaver89-3] ———, *Perfect privacy for two party protocols*, Distributed Computing and Cryptography (Joan Feigenbaum und Michael Merritt, Hrsg.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Band 2, 1991, 65–78.

[Beaver91-2] ———, *Secure Multiparty Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority*, Journal of Cryptology 4 (1991), no. 2, 75–122.

[BiBuMeThiThi] Ingrid Biehl, Johannes Buchmann, Bernd Meyer, Christian Thiel und Christoph Thiel, *Tools for proving zero knowledge*, EuroCrypt, LNCS, Band 658, 1992, 356–365.

[Skript] Johannes Buchmann et al., *Theoretische Kryptographie*, Vorlesungsskript, Wintersemester 1990.

[ChorKush2] Benny Chor und Eyal Kushilevitz, *A Zero-One Law for Boolean Privacy*, STOC, 1989, 62–72.

[Dolev82] Danny Dolev, *The byzantine generals strike again*, Journal of Algorithms **3** (1982), 14–30.

[FeiSha90] Uriel Feige und Adi Shamir, *Witness indistinguishable and witness hiding protocols*, STOC, 1990, 416–426.

[GalilHabYung87] Zvi Galil, Stuart Haber und Moti Yung, *Cryptographic computation: secure fault-tolerant protocols and the public-key model*, Crypto, 1987, 135–155.

[GOLDR89-2] ODED GOLDREICH, *Foundations of Cryptography: List of References*, Tech. Report 89-573, Technion, Haifa, Israel, July 1989.

[GOLDR89] ———, *A Note on Computational Indistinguishability*, Tech. Report 89-051, Technion, Haifa, Israel, August 1989.

[GOLDWLEV90] SHAFI GOLDWASSER UND LEONID LEVIN, *Fair computation of general functions in presence of immoral majority*, Crypto, 1990, 75–84.

[HEUSER1] HARRO HEUSER, *Lehrbuch der Analysis, Teil 1*, B.G. Teubner Stuttgart, 1989.

[KUSH] EYAL KUSHILEVITZ, *Privacy and Communication Complexity*, FOCS, 1989, 416–421.

[LEWISPAP] HARRY R. LEWIS UND CHRISTOS H. PAPADIMITRIOU, *Elements of the theory of computation*, Prentice Hall, Inc., 1981.

[MICRACKSLO88] SILVIO MICALI, CHARLES RACKOFF UND BOB SLOAN, *The notion of security for probabilistic cryptosystems*, SIAM Journal of Computation **17** (1988), no. 2, 412–426.

[MICROG] SILVIO MICALI UND PHILLIP ROGAWAY, *Secure Computation*, Preprint des ersten Teiles, der Abstract erschien in Crypto, 1991.

[REISCHUK90] RÜDIGER K. REISCHUK, *Einführung in die Komplexitätstheorie*, Teubner, 1990.

[ROG91] PHILLIP ROGAWAY, *The round complexity of secure protocols*, Dissertation, Massachusetts Institute of Technology, 1991.

[SPOERL63] HEINRICH SPOERL, *Der Maulkorb*, Heinrich Spoerl's Gesammelte Werke, Lizenzausgabe des Deutschen Bücherbundes, 1963, 293–399.

[WALDWALT] E.H. WALDSCHMIDT UND H.K.-G. WALTER, *Grundzüge der Informatik I*, Reihe Informatik, Band 43, Bibliographisches Institut, 1984.

[YAO82] ANDREW CHI-CHIH YAO, *Theory and Applications of Trapdoor Functions*, FOCS, 1982, 80–91.

[YAO86] ———, *How to generate and exchange secrets*, FOCS, 1986, 162–167.

[YAO88] ———, *Computational Information Theory*, Complexity in Information Theory (Yaser S. Abu-Mostafa, Hrsg.), Springer, 1988, 1–15.

Stichwortverzeichnis