6.9 Polynomiale Schaltnetzfamilien

Ein Schaltnetz hat eine festgelegte Zahl von Eingängen, kann also (im Gegensatz zu einer TURING-Maschine) nur Eingaben bestimmter Länge verarbeiten. Bei Effizienzuntersuchungen will man aber meist das Wachstum des Aufwandes bei immer weiterer Vergrößerung der Eingabelänge abschätzen. Dazu unterstellt man eine ganze Familie $(C_n)_{n\in\mathbb{N}}$ von Schaltnetzen mit wachsender Zahl deterministischer Eingänge, deren Größe $\#C_n$ kontrolliert wächst; damit kann man dann das Wachstum des Aufwandes als Funktion der Größe der Eingabe ausdrücken. Genauer wird definiert: Eine **polynomiale (probabilistische) Schaltnetzfamilie (PPS)** ist eine Familie $C = (C_n)_{n\in\mathbb{N}}$,

$$C_n \colon \mathbb{F}_2^{r(n)} \times \mathbb{F}_2^{k(n)} \longrightarrow \mathbb{F}_2^{s(n)}, \tag{1}$$

von (probabilistischen) Schaltnetzen mit $r(n)$ deterministischen und $k(n)$ probabilistischen Eingängen, so dass es ein Polynom $f \in \mathbb{N}[X]$ (nichtnegative ganzzahlige Koeffizienten) gibt mit $\#C_n \leq f(n)$ für alle $n \in \mathbb{N}$. Insbesondere ist die Zahl der Eingänge aller Arten und die Zahl $s(n)$ der Ausgänge polynomial beschränkt.

Bemerkung. Das bedeutet nicht notwendig, dass die Funktionen r, k, s selbst Polynome sind.

Durch dieses Berechnungsmodell (im deterministischen Fall) könnten mehr Probleme berechenbar sein als durch das gebräuchliche Modell der TURING-Maschinen (und sind es tatsächlich), da für jede Eingabegröße ein anderer Algorithmus gewählt werden kann. Man spricht daher auch von einem „nicht-gleichmäßigen Berechnungsmodell“. Das erscheint auf den ersten Blick vielleicht als Nachteil dieses Berechnungsmodells, ist aber für die Kryptoanalyse sogar besonders realistisch: Nach Wahl der Inputgröße n hat der Kryptoanalytiker die Möglichkeit, einen passenden Algorithmus zu wählen. Das heisst, Aussagen über Nichteffizienz, die unter diesem Berechnungsmodell bewiesen werden, erlauben, die Inputgröße als öffentlich bekannt anzunehmen.

Ist eine TURING-Berechnung in polynomialer Zeit möglich, so gibt es für das gleiche Problem auch eine polynomiale Schaltnetzfamilie. Die Umkehrung davon gilt nicht; es sind allerdings nur „künstliche“ Beispiele bekannt. Gäbe es für irgendein NP-vollständiges Problem eine polynomiale Schaltnetzfamilie, so für alle. Nichtgleichmäßige Komplexität kann man allerdings auch mit TURING-Maschinen modellieren, indem man für jede Eingabelänge eine andere TURING-Maschine zulässt. Analog kann man natürlich auch probabilistische TURING-Maschinen zulassen.

Ein Problem soll **hart** heissen, wenn es kein PPS gibt, das es mit signifikanter Wahrscheinlichkeit löst. Die früher behandelten „harten zahlen-

theoretischen Probleme“ sind vermutlich in diesem Sinne hart, z. B. die Primzerlegung; präzisiert wird das später.

Wir wissen bereits, dass die Grundoperationen für ganze Zahlen mit polynomialen Schaltnetzfamilien berechenbar sind (sogar deterministisch).

Um die Ergebnisse aus 6.8 übertragen zu können, müssen auch die Begriffe des Vorteils und der Irrtumswahrscheinlichkeit auf Familien verallgemeinert werden. Gegeben sei dazu eine Menge $L \subseteq \mathbb{F}_2^*$, also eine Sprache über dem binären Alphabet, wobei wie üblich $L_n := L \cap \mathbb{F}_2^n$ gesetzt wird, sowie eine Abbildung

$$f\colon L \longrightarrow \mathbb{F}_2^* \quad \text{mit } f(L_{r(n)}) \subseteq \mathbb{F}_2^{s(n)}, \tag{2}$$

wobei $r(n)$ die monoton wachsende Folge der Indizes i mit $L_i \neq 0$ ist. Diese Abbildung soll mit einer PPS wie in (1) berechnet werden.

Beispiele.

1. Die Funktion $f(x, y, z) := xy \bmod z$ für n-Bit-Zahlen x, y, z lässt sich mit einem (deterministischen) Schaltnetz

   $$C_n\colon F_2^{3n} \longrightarrow \mathbb{F}_2^n$$

   der Größe $\#C_n = \mathrm{O}(n^3)$ (und Irrtumswahrscheinlichkeit 0) berechnen.

2. Sei L die Menge der ungeraden Zahlen ≥ 3 und $f : L \longrightarrow \mathbb{F}_2$ der „Primzahlanzeiger“ wie in Abschnitt 6.8. Die dort vorgestellte PPS für den strengen Pseudoprimzahltest hat die Größe $\mathrm{O}(n^3)$ und konstanten Vorteil $\frac{1}{4}$ sowie konstante Irrtumswahrscheinlichkeit $\frac{1}{4}$. Mit h Basen kommt man auf die Größe $\mathrm{O}(hn^3)$ und die Irrtumswahrscheinlichkeit $\frac{1}{4^h}$.

Definition 1. Sei $f : L \longrightarrow \mathbb{F}_2^*$ wie in (2). Sei C eine PPS, die bei der Berechnung von f auf $\mathbb{F}_2^{r(n)}$ eine Irrtumswahrscheinlichkeit von ε_n hat, und für jedes nichtkonstante Polynom $h \in \mathbb{N}[X]$ sei

$$\varepsilon_n \leq \frac{1}{h(n)} \quad \text{für fast alle } n \in \mathbb{N}.$$

Dann heißt C **effizienter Algorithmus** für f.

f heißt effizient berechenbar, wenn es einen effizienten Algorithmus für f gibt.

Für den strengen Pseudoprimzahltest kann man diese Forderung erfüllen, wenn man die Zahl der Basen mit n wachsen lässt; damit die Familie polynomial bleibt, nimmt man h als Polynom $h \in \mathbb{Z}[X]$. Dann hat C_n n deterministische und $h(n)$ probabilistische Eingänge, die Größe $\mathrm{O}(n^3 h(n))$ und die Irrtumswahrscheinlichkeit $\frac{1}{4^{h(n)}}$. Damit ist gezeigt:

Satz 1 *Der strenge Pseudoprimzahltest ist ein effizienter probabilistischer Algorithmus für die Bestimmung der Primzahleigenschaft.*

Etwas kniffliger ist die exakte Definition eines harten Problems. Es ist klar, dass die Forderung, das Problem solle für fast alle Eingaben nicht effizient lösbar sein, durch die Negierung der Eigenschaft „effizient“ nicht erfüllt wird. Näher kommt dem schon die Forderung, der Vorteil des Algorithmus solle mit wachsendem n gegen 0 gehen; aber auch das ist keine geeignete Definition, da der Vorteil nur eine untere Schranke ist. Der beste Ansatz ist, zu verlangen, dass es keinen Vorteil gibt, der „zu langsam“ gegen 0 geht. „Zu langsam“ soll bedeuten

$$\frac{1}{h(n)} \text{ mit einem beliebigen Polynom } h \in \mathbb{N}[X],$$

und es soll „fast keine“ Eingaben geben, die Ausnahmen sind – die Ausnahmemenge soll „dünn“ sein. Diese Vorstellung wird jetzt in eine exakte Definition umgesetzt.

Für $x \in L_{r(n)}$ betrachten wir die Wahrscheinlichkeit

$$p_x := P(\{\omega \in \Omega_{k(n)} \mid C_n(x,\omega) = f(x)\}),$$

ferner die Menge der Eingaben x, für die C_n einen ε-Vorteil hat:

$$L_{r(n)}(\varepsilon) := \{x \in L_{r(n)} \mid p_x \geq \frac{1}{2^{s(n)}} + \varepsilon\}.$$

Für ein Polynom $h \in \mathbb{N}[X]$ ist dann $L_{r(n)}(\frac{1}{h(n)})$ die Menge von Eingaben x, für die $f(x)$ von C mit Vorteil $\frac{1}{h(n)}$ berechnet wird. Die Ausnahmemenge für h ist damit

$$L^{[f,C,h]} := \bigcup_{n\in\mathbb{N}} L_{r(n)}(\frac{1}{h(n)}).$$

Wir bezeichnen sie als „**Vorteilsmenge für f, C und h**“. Ihre Bestandteile sollen mit wachsendem n immer unbedeutender werden, und das wird so definiert:

Definition 2. Eine Teilmenge $A \subseteq L$ heisst **dünn**, wenn für alle nichtkonstanten Polynome $f \in \mathbb{N}[X]$ mit $A_n = A \cap L_n$ gilt

$$\#A_n \leq \frac{\#L_n}{f(n)} \text{ für fast alle } n \in \mathbb{N}.$$

Bemerkungen und Beispiele.

1. Ist $\#A_n = c$ konstant und $L_n = \mathbb{F}_2^n$, so ist A dünn in L, denn die verlangte Ungleichung ist $c \leq \frac{2^n}{f(n)}$.

2. Wächst $\#A_n$ höchstens wie ein Polynom, aber $\#L_n$ schneller als jedes Polynom, so ist A dünn in L.

3. Ist $\#A_n = c \cdot \#L_n$ ein fester Anteil, so ist A nicht dünn in L.

4. Ist $L = \mathbb{N}$ und A die Menge der Primzahlen (beides binär codiert), so ist nach dem Primzahlsatz

$$\#A_n \approx \frac{2^{n-1}}{n \cdot \ln(2)} = \frac{\#L_n}{n \cdot \ln(2)}.$$

   Die Menge der Primzahlen ist also nicht dünn in $\mathbb{N}$.

5. Es ist kein effizienter Algorithmus bekannt, der mehr als eine dünne Teilmenge der Menge M aller Produkte von zwei Primzahlen, die sich in der Länge um höchstens ein Bit unterscheiden, faktorisieren kann.

Definition 3. Sei f wie in (2). Dann heißt f **hart**, wenn für jede PPS wie in (1) und für jedes Polynom $h \in \mathbb{N}[X]$ die Vorteilsmenge $L^{[f,C,h]}$ dünne Teilmenge von L ist.

Beispiele.

1. Nach Bemerkung 5 ist die Primzerlegung vermutlich hart.

2. **Quadratrest-Vermutung:** Sei B die Menge von Produkten zweier Primzahlen $\equiv 3 \pmod 4$ („Blum-Zahlen“),

$$L = \{(m, a) \mid m \in B, 1 \leq a \leq m-1\},$$

$$f : L \longrightarrow \mathbb{F}_2$$

   die Funktion

$$f(m,a) = \begin{cases} 1, & \text{wenn } a \text{ Quadratrest mod } m, \\ 0 & \text{sonst.} \end{cases}$$

   Dann ist f hart.

Damit haben wir auch die theoretische Grundlage, um Einwegfunktionen und starke symmetrische Chiffren exakt zu definieren:

Definition 4. Gegeben sei $f : L \longrightarrow \mathbb{F}_2^*$ wie in (2). Eine Rechtsinverse zu f ist eine Abbildung $g : f(L) \longrightarrow L \subseteq \mathbb{F}_2^*$ mit $f(g(y)) = y$ für alle $y \in f(L)$ – d. h., g findet Urbilder für f. f heißt **Einwegfunktion**, wenn jede Rechtinverse von f hart ist.

Nach dieser Definition ist die diskrete Exponentialfunktion in endlichen Primkörpern vermutlich Einwegfunktion.

Definition 5. Sei $f\colon L \longrightarrow \mathbb{F}_2^*$ eine Zusammensetzung von Abbildungen

$$f_n\colon \mathbb{F}_2^{r(n)} \times \mathbb{F}_2^{q(n)} \longrightarrow \mathbb{F}_2^{r(n)}$$

mit streng monoton wachsenden r und q, so dass $f_n(\bullet, k)$ für jedes $k \in \mathbb{F}_2^{q(n)}$ bijektiv ist und f sowie $(y, k) \mapsto f_n(\bullet, k)^{-1}(y)$ jeweils effizient berechenbar sind. Ein Angriff auf f mit bekanntem Klartext ist eine Abbildung g, zusammengesetzt aus Teilen

$$g_n\colon \mathbb{F}_2^{r(n)} \times \mathbb{F}_2^{r(n)} \longrightarrow \mathbb{F}_2^{q(n)}$$

mit

$$f_n(x, g_n(x, y)) = y \quad \text{für alle } x, y \in \mathbb{F}_2^{r(n)}.$$

f heißt **starke symmetrische Chiffre**, wenn jeder Angriff auf f mit bekanntem Klartext hart ist.

Die Definition einer Hash-Funktion ist etwas kniffliger und wird hier nicht ausgeführt.