3 Schieberegister

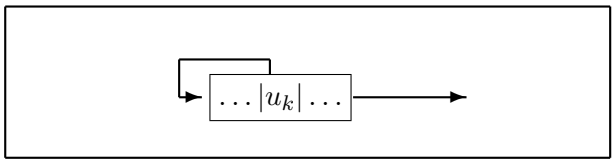
Der vorige Abschnitt hat gezeigt, dass lineare Schieberegister in Reinform kryptographisch völlig unbrauchbar sind. Auch für nichtlineare Schieberegister wurde mit Hilfe des NOETHERschen Prinzips ein effizientes Vorhersageverfahren konstruiert, das ihre Sicherheit gefährlich unterminiert.

In diesem Abschnitt wird die Vorhersagbarkeit von Bitfolgen aus einem anderen Blickwinkel heraus systematisch betrachtet: Wie kann man eine gegebene Bitfolge optimal durch ein lineares Schieberegister erzeugen? Die minimale Länge eines solchen Registers wird sich – bis auf wenige Ausreißer – als gutes Maß für die Vorhersagbarkeit der Folge erweisen.

3.1 Die lineare Komplexität von Bitfolgen

Wir betrachten – zunächst unendliche – Bitfolgen $u = (u_i)_{i\in\mathbb{N}} \in \mathbb{F}_2^{\mathbb{N}}$. Gesucht ist ein lineares Schieberegister möglichst geringer Länge, das die Folge produziert.

Gibt es ein solches Schieberegister, muss die Folge periodisch sein. Umgekehrt wird jede periodische Folge stets durch ein lineares Schieberegister erzeugt, dessen Länge die Summe der Längen von Vorperiode und Periode ist – nämlich durch das **zirkuläre Schieberegister**, das als Rückkopplung nur dasjenige Bit wieder einspeist, bei dem die Periode beginnt; ist $u_{l+i} = u_{k+i}$ für $i \geq 0$, so sind die Koeffizienten $a_{l-k} = 1$, $a_i = 0$ sonst.



Damit ist gezeigt:

Hilfssatz 1 *Eine Bitfolge $u \in \mathbb{F}_2^{\mathbb{N}}$ lässt sich genau dann von einem linearen Schieberegister erzeugen, wenn sie periodisch ist.*

Definition. Die **lineare Komplexität** $\lambda(u)$ einer Bitfolge $u \in \mathbb{F}_2^{\mathbb{N}}$ ist die minimale Länge eines linearen Schieberegisters, das u erzeugt.

Ist u konstant 0, wird $\lambda(u) = 0$, ist u nicht periodisch, wird $\lambda(u) = \infty$ gesetzt.

Es handelt sich hierbei also um einen Komplexitätsbegriff, der auf dem sehr speziellen Maschinen-Modell der linearen Schieberegister beruht.

Bemerkungen und Beispiele

1. Falls $\tau(u)$ die Summe aus der Länge der Periode und der Vorperiode von u ist und u von einem linearen Schieberegister der Länge l erzeugt wird, gilt
$$\lambda(u) \leq \tau(u) \leq 2^l - 1 \quad \text{und} \quad \lambda(u) \leq l.$$

2. Die periodisch wiederholte Folge $0, \ldots, 0, 1$ ($l-1$ Nullen) hat die Periode l und die lineare Komplexität l. Ein lineares Schieberegister der Länge $< l$ würde nämlich mit dem Nullvektor als Startwert gefüttert und könnte dann nur noch weitere Nullen produzieren.

Für endliche Bitfolgen $u = (u_0, \ldots, u_{N-1}) \in \mathbb{F}_2^N$ definiert man die lineare Komplexität analog. Insbesondere ist $\lambda(u)$ die minimale Zahl l, so dass es $a_1, \ldots, a_l \in \mathbb{F}_2$ gibt mit

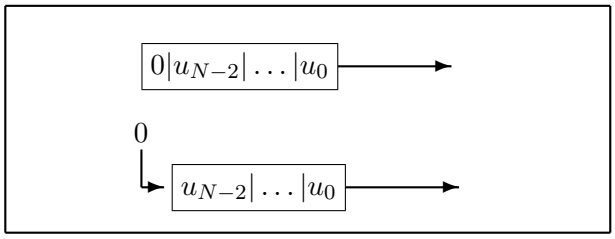
$$u_i = a_1 u_{i-1} + \cdots + a_l u_{i-l} \quad \text{für } i = l, \ldots, N-1.$$

Bemerkungen und Beispiele

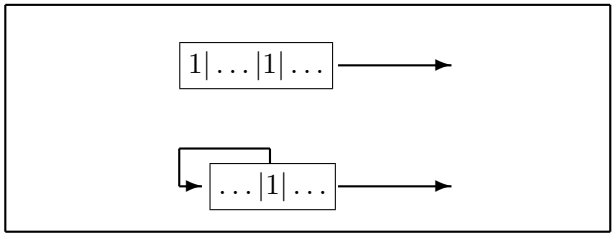
3. Für $u \in \mathbb{F}_2^N$ gilt $0 \leq \lambda(u) \leq N$.

4. $\lambda(u) = 0 \Longleftrightarrow u_0 = \cdots = u_{N-1} = 0$.

5. $\lambda(u) = N \Longleftrightarrow u = (0, \ldots, 0, 1)$. Die Implikation „$\Longleftarrow$“ folgt wie in Bemerkung 2. Für die Umkehrung kann u_{N-1} nicht 0 sein, denn sonst könnte man das lineare Schieberegister der Länge $N-1$ mit Rückkopplung konstant 0 nehmen; die beiden Schieberegister



haben den gleichen Output. Also muss $u_{N-1} = 1$ sein. Gäbe es vorher in der Folge schon eine 1, könnte man das Schieberegister der Länge $N-1$ nehmen, das genau diese Bitposition rückkoppelt; die beiden Schieberegister



haben den gleichen Output.

6. Sind die ersten $2\lambda(u)$ Bits der Bitfolge u bekannt, so lässt sich der Rest von u daraus vorhersagen. (Dem Kryptoanalytiker, der n Bits der Folge kennt, den Rest aber nicht, ist natürlich auch $\lambda(u)$ unbekannt, d. h., er weiß nicht, dass seine Vorhersage von nun an immer korrekt sein wird. Das hindert ihn aber nicht daran, Bit für Bit, und zwar korrekt, vorherzusagen!)

3.2 Synthese linearer Schieberegister

In diesem Abschnitt geht es darum, zu einer gegebenen endlichen Bitfolge ein lineares Schieberegister kürzester Länge zu finden. Der Ansatz aus Abschnitt 2 zur Vorhersage von Zufallsgeneratoren lieferte ein lineares Schieberegister unter der Annahme, dass man schon ein evtl. nichtlineares hat, lässt aber kaum erkennen, ob es vielleicht ein kürzeres gäbe. Mit einem etwas anderen Ansatz ist die gestellte Aufgabe aber überraschend einfach zu lösen: mit einem Algorithmus von Massey (1969), der in einem anderen Kontext vorher schon von Berlekamp (1968) angegeben worden war.

Da keine speziellen Eigenschaften des Körpers $\mathbb{F}_2$ benützt werden, wird hier ein beliebiger Körper K zu Grunde gelegt. Gesucht wird ein homogener Kongruenzgenerator möglichst geringer Rekursionstiefe l, der eine gegebene endliche Folge $u \in K^N$ erzeugt.

Für einen solchen Kongruenzgenerator mit Bildungsgestz

$$u_k = a_1 u_{k-1} + \cdots + a_l u_{k-l} \quad \text{für } k = l, \ldots, N-1$$

ist $(a_1, \ldots, a_l) \in K^l$ der Koeffizientenvektor; das Polynom

$$\varphi = 1 - a_1 T - \cdots - a_l T^l \in K[T]$$

heißt **Rückkopplungspolynom**. Es ist das reziproke Polynom zum charakteristischen Polynom der Begleitmatrix

$$A = \begin{pmatrix} 0 & 1 & \ldots & 0 \\ & \ddots & \ddots & \\ & & & 1 \\ a_l & a_{l-1} & \ldots & a_1 \end{pmatrix}.$$

Ist dieses $\chi = \mathrm{Det}(T \cdot 1 - A)$, so ist

$$\chi = T^l - a_1 T^{l-1} - \cdots - a_l, \quad \text{also} \quad \varphi = T^l \cdot \chi(\frac{1}{T}).$$

Hilfssatz 2 *Der homogene lineare Kongruenzgenerator mit Koeffizienten $(a_1, \ldots, a_l)$ erzeuge die Folge $u = (u_0, \ldots, u_{n-1}) \in K^n$, aber nicht die Folge $\hat{u} = (u_0, \ldots, u_n) \in K^{n+1}$. Dann hat jeder homogene lineare Kongruenzgenerator, der $\hat{u}$ erzeugt, eine Länge $m \geq n + 1 - l$.*

Beweis. **Fall 1:** $l \geq n$. Dann ist $l + m \geq n + 1$, außer wenn $l = n$, $m = 0$. In diesem Fall müsste aber wegen $m = 0$ notwendig $u_0 = \ldots = u_n$ sein, und der Generator würde auch $\hat{u}$ erzeugen, Widerspruch.

Fall 2: $l \leq n - 1$. *Annahme*: $m \leq n - l$. Es ist

$$u_j = a_1 u_{j-1} + \cdots + a_l u_{j-l} \quad \text{für } l \leq j \leq n-1.$$

Sei $(b_1, \ldots, b_m)$ der Koeffizientenvektor eines homogenen linearen Kongruenzgenerators, der $\hat{u}$ erzeugt; dann ist

$$u_j = b_1 u_{j-1} + \cdots + b_m u_{j-m} \quad \text{für } m \le j \le n.$$

Insgesamt folgt

$$\begin{aligned} u_n &\neq a_1 u_{n-1} + \cdots + a_l u_{n-l} \\ &= \sum_{i=1}^{l} a_i \cdot \underbrace{\sum_{k=1}^{m} b_k u_{n-i-k}}_{u_{n-i}} \quad [\text{da } n-l \ge m] \\ &= \sum_{k=1}^{m} b_k \cdot \underbrace{\sum_{i=1}^{l} a_i u_{n-k-i}}_{u_{n-k}} = u_n, \end{aligned}$$

Widerspruch. $\diamond$

Sei nun $u \in K^N$ eine Folge. Für $0 \le n \le N$ sei $\lambda_n(u) = \lambda_n$ die kleinste Rekursionstiefe eines Generators, der $(u_0, \ldots, u_{n-1})$ erzeugt.

Hilfssatz 3 *Für jede Folge $u \in K^N$ gilt:*

(i) $\lambda_{n+1} \ge \lambda_n$.
(ii) *Genau dann, wenn es einen Generator der Rekursionstiefe λ_n gibt, der $(u_0, \ldots, u_n)$ erzeugt, gilt $\lambda_{n+1} = \lambda_n$.*
(iii) *Gibt es einen solchen nicht, ist*

$$\lambda_{n+1} \ge n + 1 - \lambda_n.$$

Beweis. (i) Jeder Generator, der $(u_0, \ldots, u_n)$ erzeugt, erzeugt erst recht $(u_0, \ldots, u_{n-1})$.

(ii) folgt aus (i).

(iii) Die Voraussetzung von Hilfssatz 2 gilt für jeden Generator von $(u_0, \ldots, u_{n-1})$. $\diamond$

Satz 1 [Massey] *Sei $u \in K^N$ und $0 \le n \le N-1$. Sei ferner $\lambda_{n+1}(u) \neq \lambda_n(u)$. Dann ist*

$$\lambda_n(u) \le \frac{n}{2} \quad \textit{und} \quad \lambda_{n+1}(u) = n + 1 - \lambda_n(u).$$

Beweis. Der Fall $\lambda_n = 0$ ist besonders leicht: Es ist $u_0 = \ldots = u_{n-1} = 0$. Falls $u_n = 0$, ist $\lambda_{n+1} = \lambda_n = 0$, also nichts zu beweisen. Falls $u_n \neq 0$, ist $\lambda_{n+1} = n + 1 = n + 1 - \lambda_n$ nach Bemerkung 5 in 3.1.

Allgemein folgt die erste Aussage aus der zweiten, denn wegen $\lambda_n < \lambda_{n+1}$ ist $2\lambda_n < n+1$.

Die zweite Aussage wird nun durch Induktion über n bewiesen. Im Fall $n = 0$ ist $\lambda_0 = 0$ – dieser Fall ist schon erledigt.

Sei jetzt $n \geq 1$. Sei o. B. d. A. $l := \lambda_n \geq 1$. Sei

$$u_j = a_1 u_{j-1} + \cdots + a_l u_{j-l} \quad \text{für } j = l, \ldots, n-1;$$

das zugehörige Rückkopplungspolynom ist also

$$\varphi := 1 - a_1 T - \cdots - a_l T^l \in K[T].$$

Die „n-te Diskrepanz“ sei

$$d_n := u_n - a_1 u_{n-1} - \cdots - a_l u_{n-l}.$$

Ist $d_n = 0$, so erzeugt der Generator auch u_n, und es ist nichts zu beweisen. Sei also $d_n \neq 0$. Sei r die Länge der Folge vor der letzten Zunahme der linearen Komplexität, also

$$t := \lambda_r < l, \quad \lambda_{r+1} = l.$$

Wegen der Induktionsannahme ist $l = r + 1 - t$. Ist

$$u_j = b_1 u_{j-1} + \cdots + b_t u_{j-t} \quad \text{für } j = t, \ldots, r-1,$$

so ist das zugehörige Rückkopplungspolynom

$$\psi := 1 - b_1 T - \cdots - b_t T^t \in K[T],$$

und für die analog gebildete r-te Diskrepanz gilt

$$d_r := u_r - b_1 u_{r-1} - \cdots - b_t u_{r-t} \neq 0,$$

Ist $t = 0$, so $\psi = 1$ und $d_r = u_r$. Nun wird das Polynom

$$\eta := \varphi - \frac{d_n}{d_r} \cdot T^{n-r} \cdot \psi = 1 - c_1 T - \cdots - c_m T^m \in K[T]$$

gebildet mit $m = \operatorname{Grad} \eta$. Was macht der zugehörige homogene lineare Kongruenzgenerator? Es ist

$$\begin{aligned} u_j - \sum_{i=1}^{m} c_i u_{j-i} &= u_j - \sum_{i=1}^{l} a_i u_{j-i} - \frac{d_n}{d_r} \cdot \left[u_{j-n+r} - \sum_{i=1}^{t} b_i u_{j-n+r-i} \right] \\ &= 0 \quad \text{für } j = m, \ldots, n; \end{aligned}$$

für $j = 0, \ldots, n-1$ folgt das direkt, für $j = n$ kommt zunächst $d_n - [d_n/d_r] \cdot d_r$ heraus. Er erzeugt also $(u_0, \ldots, u_n)$. Nun ist

$$\lambda_{n+1} \leq m \leq \max\{l, n - r + t\} = \max\{l, n + 1 - l\}.$$

Wegen der Monotonie der linearen Komplexität ist $m > l$, nach Hilfssatz 2 ist $m \geq n + 1 - l$. Also folgt $m = n + 1 - l$ und $\lambda_{n+1} = m$. Damit ist die Behauptung bewiesen. $\diamond$

Korollar 1 *Ist $d_n \neq 0$ und $\lambda_n \leq \frac{n}{2}$, so ist*

$$\lambda_{n+1} = n + 1 - \lambda_n > \lambda_n.$$

Beweis. Nach Hilfssatz 2 ist $\lambda_{n+1} \geq n + 1 - \lambda_n$, also $\lambda_{n+1} \geq \frac{n}{2} + 1 > \lambda_n$. Nach Satz 1 folgt daraus $\lambda_{n+1} = n + 1 - \lambda_n$. $\diamond$

Bei dem sukzessiven Aufbau eines linearen Rekurrenzgenerators im Beweis des Satzes tritt also in jedem Iterationsschritt einer der folgenden Fälle ein:

- $d_n = 0$: Dann ist $\lambda_{n+1} = \lambda_n$.
- $d_n \neq 0$: Dann ist
  - $\lambda_{n+1} = \lambda_n$, falls $\lambda_n > \frac{n}{2}$,
  - $\lambda_{n+1} = n + 1 - \lambda_n$, falls $\lambda_n \leq \frac{n}{2}$.

Insbesondere gilt stets:

- Ist $\lambda_n > \frac{n}{2}$, so $\lambda_{n+1} = \lambda_n$.
- Ist $\lambda_n \leq \frac{n}{2}$, so $\lambda_{n+1} = \lambda_n$ oder $\lambda_{n+1} = n + 1 - \lambda_n$.

Nebenbei haben wir eine alternative Möglichkeit gefunden, lineare Schieberegister vorherzusagen:

Korollar 2 *Wird $u \in \mathbb{F}_2^n$ von einem linearen Schieberegister der Länge $\leq l$ erzeugt, so lässt sich ein solches aus $u_0, \ldots, u_{2l-1}$ bestimmen.*

Beweis. Wäre erstmals $d_n \neq 0$ für $n \geq 2l$, so $\lambda_n \leq l \leq \frac{n}{2}$, also $\lambda_{n+1} = n + 1 - \lambda_n \geq l + 1$, Widerspruch. $\diamond$

3.3 Der Berlekamp-Massey-Algorithmus

Der Beweis von Satz 1 ist konstruktiv: Er enthält einen Algorithmus, mit dem sukzessive ein linearer Rekurrenzgenerator aufgebaut wird. Beim Schritt von der Folgenlänge n zur Folgenlänge $n+1$ gibt es drei mögliche Fälle (1, 2a, 2b):

1. Fall $d_n = 0$, d. h. der Generator zum Rückkopplungspolynom φ erzeugt auch u_n: Dann bleiben φ und l ungeändert und ebenso ψ, t, r, d_r.

2. Fall $d_n \neq 0$, d. h. der Generator zum Rückkopplungspolynom φ erzeugt u_n nicht: Dann wird ein neues Rückkopplungspolynom η gebildet, dessen zugehöriger Generator $(u_0, \ldots, u_n)$ erzeugt. Es wird weiter unterschieden:

a) $l > \frac{n}{2}$: Dann ist $\lambda_{n+1} = \lambda_n$; es wird φ durch η ersetzt, l bleibt, und ebenso bleiben ψ, t, r, d_r.

b) $l \leq \frac{n}{2}$: Dann ist $\lambda_{n+1} = n + 1 - \lambda_n$; es wird φ durch η und l durch $n + 1 - l$ ersetzt, ferner ψ durch φ, t durch l, r durch n und d_r durch d_n.

Damit ist der Berlekamp-Massey-Algorithmus semiformal beschreibbar:

Input: Eine Folge $u = (u_0, \ldots, u_{N-1}) \in K^N$.

Output: Die lineare Komplexität $\lambda_N(u)$,

das Rückkopplungspolynom φ eines linearen Rekurrenzgenerators der Länge $\lambda_N(u)$, der u erzeugt.

Hilfsvariablen: n: aktueller Index, initialisiert mit $n := 0$,

l: aktuelle lineare Komplexität, initialisiert mit $l := 0$,

φ: aktuelles Rückkopplungspolynom $= 1 - a_1 T - \cdots - a_l T^l$, initialisiert mit $\varphi := 1$,

Invarianzbedingung: $u_i = a_1 u_{i-1} + \cdots + a_l u_{i-l}$ für $l \leq i < n$,

d: aktuelle Diskrepanz $= u_n - a_1 u_{n-1} - \cdots - a_l u_{n-l}$,

r: voriger Index, initialisiert mit $r := -1$,

t: vorige lineare Komplexität,

ψ: voriges Rückkopplungspolynom $= 1 - b_1 T - \cdots - b_t T^t$, initialisiert mit $\psi := 1$,

Invarianzbedingung: $u_i = b_1 u_{i-1} + \cdots + b_t u_{i-t}$ für $t \leq i < r$,

d': vorige Diskrepanz $= u_r - b_1 u_{r-1} - \cdots - b_t u_{r-t}$, initialisiert mit $d' := 1$,

η: neues Rückkopplungspolynom,

m: neue lineare Komplexität.

Iterationsschritte: Für $n = 0, \ldots, N-1$:

$d := u_n - a_1 u_{n-1} - \cdots - a_l u_{n-l}$
Falls $d \neq 0$
  $\eta := \varphi - \frac{d}{d'} \cdot T^{n-r} \cdot \psi$
  Falls $l \leq \frac{n}{2}$ [lineare Komplexität wächst]
    $m := n + 1 - l$
    $t := l$
    $l := m$
    $\psi := \varphi$
    $r := n$
    $d' := d$
  $\varphi := \eta$

Natürlich kann man sich auch gleich die ganze Folge (λ_n) ausgeben lassen.

Dieser Algorithmus wird jetzt auf das **Beispiel** der Folge 001101110 angewendet. Der Fall $d \neq 0$, $l \leq \frac{n}{2}$ wird durch „[!]“ bezeichnet.

Eingangsbedingungen				Aktionen
$n = 0$	$u_0 = 0$	$l = 0$	$\varphi = 1$	$d := u_0 = 0$
$r = -1$	$d' = 1$	$t =$	$\psi = 1$	
$n = 1$	$u_1 = 0$	$l = 0$	$\varphi = 1$	$d := u_1 = 0$
$r = -1$	$d' = 1$	$t =$	$\psi = 1$	
$n = 2$	$u_2 = 1$	$l = 0$	$\varphi = 1$	$d := u_2 = 1$ [!]
$r = -1$	$d' = 1$	$t =$	$\psi = 1$	$\eta := 1 - T^3$
				$m := 3$
$n = 3$	$u_3 = 1$	$l = 3$	$\varphi = 1 - T^3$	$d := u_3 - u_0 = 1$
$r = 2$	$d' = 1$	$t = 0$	$\psi = 1$	$\eta := 1 - T - T^3$
$n = 4$	$u_4 = 0$	$l = 3$	$\varphi = 1 - T - T^3$	$d := u_4 - u_3 - u_1 = -1$
$r = 2$	$d' = 1$	$t = 0$	$\psi = 1$	$\eta := 1 - T + T^2 - T^3$
$n = 5$	$u_5 = 1$	$l = 3$	$\varphi = 1 - T + T^2 - T^3$	$d := u_5 - u_4 + u_3 - u_2 = 1$
$r = 2$	$d' = 1$	$t = 0$	$\psi = 1$	$\eta := 1 - T + T^2 - 2T^3$

Von jetzt an unterscheiden sich die Ergebnisse je nach Charakteristik des Grundkörpers K. Sei zuerst $\operatorname{char} K \neq 2$. Dann geht es so weiter:

Eingangsbedingungen	Aktionen
$n = 6 \quad u_6 = 1 \quad l = 3$ $\varphi = 1 - T + T^2 - 2T^3$ $r = 2 \quad d' = 1 \quad t = 0 \quad \psi = 1$	$d := u_6 - u_5 + u_4 - 2u_3 = -2$ [!] $\eta = 1 - T + T^2 - 2T^3 + 2T^4$ $m := 4$
$n = 7 \quad u_7 = 1 \quad l = 4$ $\varphi = 1 - T + T^2 - 2T^3 + 2T^4$ $r = 6 \quad d' = -2 \quad t = 3$ $\psi = 1 - T + T^2 - 2T^3$	$d := u_7 - u_6 + u_5 - 2u_4 + 2u_3 = 3$ $\eta = 1 + \frac{1}{2}T - \frac{1}{2}T^2 - \frac{1}{2}T^3 - T^4$
$n = 8 \quad u_8 = 0 \quad l = 4$ $\varphi = 1 + \frac{1}{2}T - \frac{1}{2}T^2 - \frac{1}{2}T^3 - T^4$ $r = 6 \quad d' = -2 \quad t = 3$ $\psi = 1 - T + T^2 - 2T^3$	$d := u_8 + \frac{1}{2}u_7 - \frac{1}{2}u_6 - \frac{1}{2}u_5 - u_4 = -\frac{1}{2}$ [!] $\eta := 1 + \frac{1}{2}T - \frac{3}{4}T^2 - \frac{1}{4}T^3 - \frac{5}{4}T^4 + \frac{1}{2}T^5$ $m := 5$

Als Ergebnis erhalten wir die Folge der linearen Komplexitäten

$$\lambda_0 = 0, \lambda_1 = 0, \lambda_2 = 0, \lambda_3 = 3, \lambda_4 = 3, \lambda_5 = 3, \lambda_6 = 3, \lambda_7 = 4, \lambda_8 = 4, \lambda_9 = 5$$

und die Rekursionsvorschrift

$$u_i = -\frac{1}{2}u_{i-1} + \frac{3}{4}u_{i-2} + \frac{1}{4}u_{i-3} + \frac{5}{4}u_{i-4} - \frac{1}{2}u_{i-5} \quad \text{für } i = 5, \ldots, 8.$$

Im Falle $\operatorname{char} K = 2$ sehen die letzten drei Iterationen so aus:

Eingangsbedingungen	Aktionen
$n = 6 \quad u_6 = 1 \quad l = 3$ $\varphi = 1 - T - T^2$ $r = 2 \quad d' = 1 \quad t = 0 \quad \psi = 1$	$d := u_6 - u_5 - u_4 = 0$
$n = 7 \quad u_7 = 1 \quad l = 3$ $\varphi = 1 - T - T^2$ $r = 2 \quad d' = 1 \quad t = 0 \quad \psi = 1$	$d := u_7 - u_6 - u_5 = 1$ [!] $\eta = 1 - T - T^2 - T^5$ $m := 5$
$n = 8 \quad u_8 = 0 \quad l = 5$ $\varphi = 1 - T - T^2 - T^5$ $r = 7 \quad d' = 1 \quad t = 3 \quad \psi = 1 - T - T^2$	$d := u_8 - u_7 - u_6 - u_3 = 1$ $\eta := 1 - T^3 - T^5$

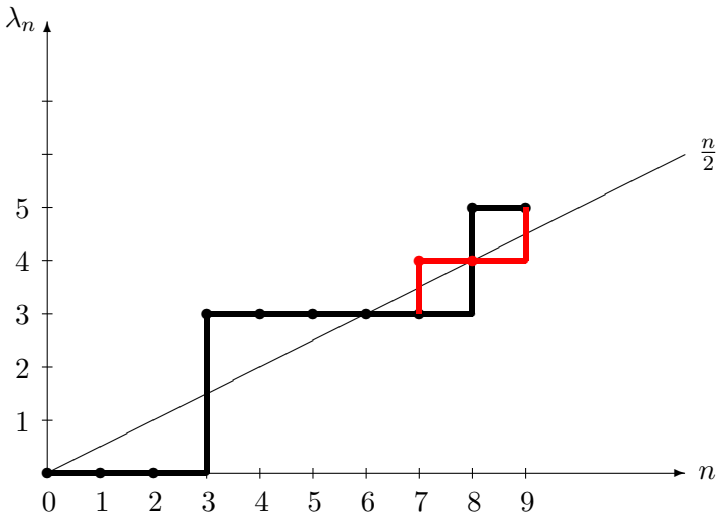
Die Folge der linearen Komplexitäten ist hier

$$\lambda_0 = 0, \lambda_1 = 0, \lambda_2 = 0, \lambda_3 = 3, \lambda_4 = 3, \lambda_5 = 3, \lambda_6 = 3, \lambda_7 = 3, \lambda_8 = 5, \lambda_9 = 5$$

und die Rekursionsvorschrift

$$u_i = u_{i-3} + u_{i-5} \quad \text{für } i = 5, \ldots, 8.$$

Die Entwicklung der linearen Komplexität wird auch noch grafisch dargestellt; die rote Linie kennzeichnet den Fall $\operatorname{char} K \neq 2$:



Der Aufwand für den Berlekamp-Massey-Algorithmus ist $O(N^2 \log N)$.

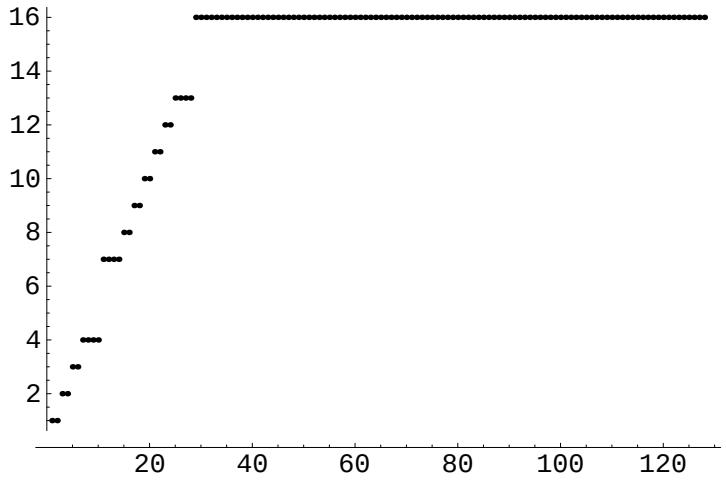
Die Folge $(\lambda_n)_{n\in\mathbb{N}}$ bzw. (für endliche Bitfolgen) $(\lambda_n)_{0\le n\le N}$ heißt das **Linearitätsprofil** der Bitfolge u.

Für die ersten 128 Bits der Folge, die in 1.10 von einem linearen Schieberegister erzeugt wurde, ist das Linearitätsprofil:

$$(0, 1, 1, 2, 2, 3, 3, 4, 4, 4, 4, 7, 7, 7, 7, 8, 8, 9, 9, 10, 10, 11, 11, 12,$$

$$12, 13, 13, 13, 13, 16, 16, 16, 16, \ldots),$$

und graphisch dargestellt sieht das so aus:

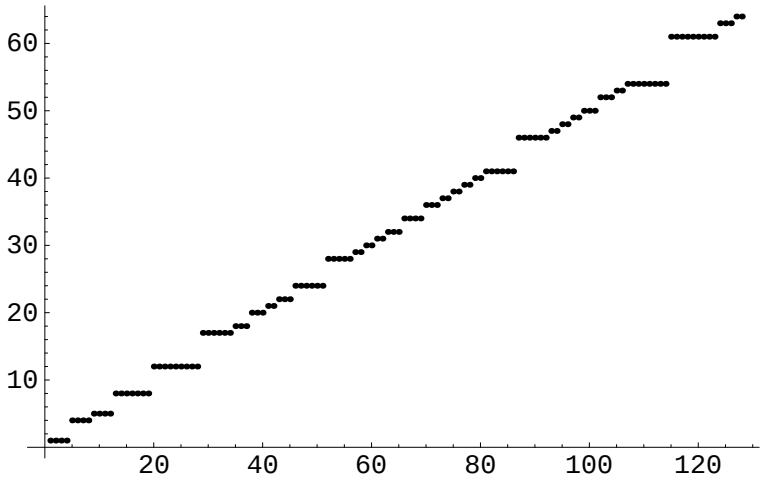


Für die ersten 128 Bits der Folge, die in 4.1 von einem „perfekten“ Zufallsgenerator erzeugt werden wird, ist das Linearitätsprofil:

$$(0, 1, 1, 1, 1, 4, 4, 4, 4, 5, 5, 5, 5, 8, 8, 8, 8, 8, 8, 8, 8, 12, 12, 12, 12,$$

$$12, 12, 12, 12, 12, 17, 17, 17, 17, 17, 17, 18, 18, 18, 20, 20, 20, 21, 21,$$
$$22, 22, 22, 24, 24, 24, 24, 24, 24, 28, 28, 28, 28, 28, 29, 29, 30, 30, 31,$$
$$31, 32, 32, 32, 34, 34, 34, 34, 36, 36, 36, 37, 37, 38, 38, 39, 39, 40, 40,$$
$$41, 41, 41, 41, 41, 41, 46, 46, 46, 46, 46, 46, 47, 47, 48, 48, 49, 49, 50,$$
$$50, 50, 52, 52, 52, 53, 53, 54, 54, 54, 54, 54, 54, 54, 54, 61, 61, 61, 61,$$
$$61, 61, 61, 61, 61, 63, 63, 63, 64, 64),$$

und das graphisch dargestellt sieht so aus:



Man sieht im zweiten Fall die unregelmäßige Schwankung um die Diagonale, wie es sich für eine „gute“ Zufallsfolge gehört. Im ersten Fall ist dieser Effekt auch vorhanden, aber nur, bis die lineare Komplexität der Folge erreicht ist.

3.4 Die Verteilung der linearen Komplexität

Die Verteilung der linearen Komplexität von Bitfolgen fester Länge lässt sich exakt bestimmen. Bei gegebener Folge $u = (u_0, \ldots, u_{N-1}) \in \mathbb{F}_2^N$ sei $\tilde{u} = (u_0, \ldots, u_N) \in \mathbb{F}_2^{N+1}$ eine Verlängerung um 1 Bit. Die Beziehung zwischen $\lambda(\tilde{u})$ und $\lambda(u)$ wird durch die MASSEY-Rekursion beschrieben: Sei

$$\delta = \begin{cases} 0, & \text{wenn die Vorhersage stimmt,} \\ 1 & \text{sonst;} \end{cases}$$

mit „Vorhersage“ ist der Wert gemeint, den das zu u konstruierte Schieberegister als nächsten liefert. Dann gilt

$$\lambda(\tilde{u}) = \begin{cases} \lambda(u), & \text{wenn } \delta = 0, \\ \lambda(u), & \text{wenn } \delta = 1 \text{ und } \lambda(u) > \frac{N}{2}, \\ N + 1 - \lambda(u), & \text{wenn } \delta = 1 \text{ und } \lambda(u) \leq \frac{N}{2}. \end{cases}$$

Im mittleren Fall wird zwar ein neues Schieberegister benötigt, aber dieses hat dieselbe Länge.

Nun wird eine Formel für die Anzahl aller Folgen der Länge N aufgestellt, die eine gegebene lineare Komplexität l haben. Sei dazu

$$\begin{aligned} M_N(l) &:= \{u \in \mathbb{F}_2^N \mid \lambda(u) = l\} \quad \text{für } N \geq 1 \text{ und } l \in \mathbb{N}, \\ \mu_N(l) &:= \#M_N(l). \end{aligned}$$

Folgende Aussagen sind unmittelbar klar:

- $0 \leq \mu_N(l) \leq 2^N$,
- $\mu_N(l) = 0$ für $l > N$,
- $\sum_{l=0}^{N} \mu_N(l) = 2^N$.

Damit lässt sich nun die Rekursion von $\mu_{N+1}(l)$ auf $\mu_N(l)$ explizit machen.

1. Fall: $0 \leq l \leq \frac{N}{2}$. Jedes $u \in \mathbb{F}_2^N$ hat zwei mögliche Fortsetzungen: $u_N = 0$ oder 1. Genau eine davon stimmt mit der Vorhersage überein und führt zu $\tilde{u} \in M_{N+1}(l)$; die andere führt zu $\tilde{u} \in M_{N+1}(N+1-l)$. Da es keine anderen Beiträge zu $M_{N+1}(l)$ geben kann, folgt $\mu_{N+1}(l) = \mu_N(l)$.

2. Fall: $l = \frac{N+1}{2}$ (was natürlich nur für ungerades N auftreten kann). Das richtig vorhergesagte u_N führt zu $\tilde{u} \in M_{N+1}(l)$, das falsch vorhergesagte wegen der MASSEY-Rekursion aber ebenfalls. Daher folgt $\mu_{N+1}(l) = 2 \cdot \mu_N(l)$.

3. Fall: $l \geq \frac{N}{2} + 1$. Beide möglichen Fortsetzungen führen zu einem $\tilde{u} \in M_{N+1}(l)$. Dazu kommt noch je ein Element von der falsch vorhergesagten Fortsetzung aller $u \in M_{N+1-l}(l)$ aus dem ersten Fall. Also ist $\mu_{N+1}(l) = 2 \cdot \mu_N(l) + \mu_{N+1-l}(l)$.

Zusammengefasst:

Hilfssatz 4 *Die Häufigkeitsfunktion $\mu_N(l)$ für Bitfolgen der Länge N mit linearer Komplexität l erfüllt die Rekursion*

$$\mu_{N+1}(l) = \begin{cases} \mu_N(l), & \text{falls } 0 \le l \le \frac{N}{2}, \\ 2 \cdot \mu_N(l), & \text{falls } l = \frac{N+1}{2}, \\ 2 \cdot \mu_N(l) + \mu_{N+1-l}(l), & \text{falls } l \ge \frac{N}{2} + 1. \end{cases}$$

Daraus lässt sich leicht eine explizite Formel gewinnen:

Satz 2 *Die Häufigkeitsfunktion $\mu_N(l)$ für Bitfolgen der Länge N mit linearer Komplexität l ist gegeben durch*

$$\mu_N(l) = \begin{cases} 1, & \text{falls } l = 0, \\ 2^{2l-1}, & \text{falls } 1 \le l \le \frac{N}{2}, \\ 2^{2(N-l)}, & \text{falls } \frac{N+1}{2} \le l \le N, \\ 0, & \text{falls } l > N. \end{cases}$$

Beweis. Im Fall $n = 1$ ist $M_1(0) = \{(0)\}$, $M_1(1) = \{(1)\}$, also $\mu_1(0) = \mu_1(1) = 1$.

Jetzt wird per Induktion von N auf $N+1$ geschlossen. Der Fall $l = 0$ ist dabei trivial, da $M_{N+1}(0) = \{(0, \ldots, 0)\}$, $\mu_{N+1}(0) = 1$. Nun werden wieder drei Fälle unterschieden:

1. Fall: $1 \le l \le \frac{N}{2}$. Hier ist erst recht $1 \le l \le \frac{N+1}{2}$, und $\mu_{N+1}(l) = \mu_N(l) = 2^{2l-1}$.

2. Fall: $l = \frac{N+1}{2}$ (N ungerade). Hier ist $\mu_N(l) = 2^{2(N-l)}$ und der Exponent $2N - 2l = 2N - N - 1 = N - 1 = 2l - 2$, also $\mu_{N+1}(l) = 2 \cdot 2^{2(N-l)} = 2^{2l-2+1} = 2^{2l-1}$.

3. Fall: $l \ge \frac{N}{2} + 1$. Hier ist wieder $\mu_N(l) = 2^{2(N-l)}$. Für $l' = N + 1 - l$ gilt $l' \le N + 1 - \frac{N}{2} - 1 = \frac{N}{2}$, also $\mu_N(l') = 2^{2l'-1}$. Also folgt $\mu_{N+1}(l) = 2\mu_N(l) + \mu_N(l') = 2^{2N-2l+1} + 2^{2N-2l+1} = 2^{2N-2l+2} = 2^{2(N+1-l)}$.

Damit ist der Beweis vollständig. $\diamond$

Die Tabelle 2 bis $N = 10$ und $l = 10$ ergibt ein interessantes Bild.

Beobachtungen:

- Die Zeile l wird ab $N = 2l$ konstant (rot markiert), die Diagonale jeweils ab $N = 2l - 1$ (blau markiert).

	1	2	3	4	5	6	7	8	9	10	$N \rightarrow$
0	1	1	1	1	1	1	1	1	1	1	
1	1	2	2	2	2	2	2	2	2	2	
2		1	4	8	8	8	8	8	8	8	
3			1	4	16	32	32	32	32	32	
4				1	4	16	64	128	128	128	
5					1	4	16	64	256	512	
6						1	4	16	64	256	
7							1	4	16	64	
8								1	4	16	
9									1	4	
10										1	
l											
$\downarrow$											

Tabelle 2: Die Verteilung der linearen Komplexität

- Jede Spalte N enthält von $l = 1$ bis $l = N$ die Zweierpotenzen 2^k, $k = 0, \ldots, N-1$, jeweils genau einmal, erst die ungeraden Zweierpotenzen (rot) in aufsteigender, dann die geraden (blau) in absteigender Reihenfolge.
- Zu jeder Länge N gibt es jeweils genau eine Folge mit der linearen Komplexität 0 und N.

3.5 Der Erwartungswert für die lineare Komplexität

Nachdem nun die Verteilung der linearen Komplexität exakt bestimmt ist, lassen sich auch Erwartungswert und Varianz bei fester Länge exakt bestimmen:

Hauptsatz 1 (RUEPPEL) *Für den Mittelwert*

$$E_N = \frac{1}{2^N} \cdot \sum_{u \in \mathbb{F}_2^N} \lambda(u)$$

und die Varianz V_N der linearen Komplexität aller Bitfolgen der Länge N gilt:

$$\begin{aligned} E_N &= \frac{N}{2} + \frac{2}{9} + \frac{\varepsilon}{18} - \frac{N}{3 \cdot 2^N} - \frac{2}{9 \cdot 2^N} \approx \frac{N}{2}, \\ V_N &= \frac{86}{81} - \frac{14-\varepsilon}{27} \cdot \frac{N}{2^N} - \frac{82-2\varepsilon}{81} \cdot \frac{1}{2^N} - \frac{9N^2+12N+4}{81} \cdot \frac{1}{2^{2N}} \approx \frac{86}{81} \end{aligned}$$

mit $\varepsilon = 0$ für gerades, $\varepsilon = 1$ für ungerades N.

Bemerkenswert ist, dass die Varianz von N praktisch unabhängig ist.

Zum Beweis muss man etwas ausholen; dabei treten Summen auf, die mit einem Trick aus der Analysis geschlossen ausgewertet werden können.

Hilfssatz 5 *Für die Funktion*

$$f\colon \mathbb{R} - \{1\} \longrightarrow \mathbb{R}, \quad f(x) = \frac{x^{r+1} - x}{x-1},$$

gilt:

$$f'(x) = \frac{1}{(x-1)^2} \cdot \left[rx^{r+1} - (r+1)x^r + 1 \right],$$

$$f''(x) = \frac{1}{(x-1)^3} \cdot \left[(r^2-r)x^{r+1} - 2(r^2-1)x^r + (r^2+r)x^{r-1} - 2 \right],$$

$$x^2 f''(x) + x f'(x) = \frac{x}{(x-1)^3} \cdot \left[r^2 x^{r+2} - (2r^2+2r-1)x^{r+1} + (r+1)^2 x^r - x - 1 \right].$$

Beweis. Das folgt durch direkte Rechnung. ◇

Mit Hilfe von f lassen sich folgende Summen berechnen:

Korollar 1 *Für alle $x \in \mathbb{R}$, $x \neq 1$, gilt:*

$$\begin{aligned} \sum_{i=1}^{r} x^i &= \frac{1}{x-1} \cdot \left[x^{r+1} - x \right], \\ \sum_{i=1}^{r} i x^i &= \frac{x}{(x-1)^2} \cdot \left[rx^{r+1} - (r+1)x^r + 1 \right], \\ \sum_{i=1}^{r} i^2 x^i &= \frac{x}{(x-1)^3} \cdot \left[r^2 x^{r+2} - (2r^2+2r-1)x^{r+1} + (r+1)^2 x^r - x - 1 \right]. \end{aligned}$$

Beweis. Aus der bekannten Formel für die geometrische Reihe folgt

$$\sum_{i=1}^{r} x^i = x \cdot \sum_{i=0}^{r-1} x^i = x \cdot \frac{x^r - 1}{x - 1} = f(x),$$

$$\sum_{i=1}^{r} i x^i = x \cdot \sum_{i=1}^{r} i x^{i-1} = x \cdot f'(x),$$

$$\sum_{i=1}^{r} i^2 x^i = \sum_{i=1}^{r} i(i-1)x^i + \sum_{i=1}^{r} i x^i = x^2 \cdot f''(x) + x \cdot f'(x).$$

Die behaupteten Formeln folgen also aus dem Hilfssatz 5. $\diamond$

Korollar 2

$$\begin{aligned} \sum_{i=1}^{r} i\, 2^{2i-1} &= \frac{3r-1}{9} \cdot 2^{2r+1} + \frac{2}{9}, \\ \sum_{i=1}^{r} i^2\, 2^{2i-1} &= \frac{3r^2-2r}{9} \cdot 2^{2r+1} + \frac{5}{27} \cdot 2^{2r+1} - \frac{10}{27}. \end{aligned}$$

Beweis.

$$\sum_{i=1}^{r} i 2^{2i-1} = \frac{1}{2} \cdot \sum_{i=1}^{r} i 4^i = \frac{1}{2} \cdot \frac{4}{9} \cdot \left[r4^{r+1} - (r+1)4^r + 1 \right] = \frac{2}{9} \cdot \left[3r4^r - 4^r + 1 \right],$$

$$\begin{aligned} \sum_{i=1}^{r} i^2 2^{2i-1} = \frac{1}{2} \cdot \sum_{i=1}^{r} i^2 4^i &= \frac{1}{2} \cdot \frac{4}{27} \cdot \left[r^2 4^{r+2} - (2r^2+2r-1)4^{r+1} + (r+1)^2 4^r - 5 \right] \\ &= \frac{2}{27} \cdot \left[(9r^2 - 6r + 5) \cdot 4^r - 5 \right]. \end{aligned}$$

$\diamond$

Der Mittelwert über die lineare Komplexität ist nun

$$E_N = \frac{1}{2^N} \cdot \sum_{u \in \mathbb{F}_2^N} \lambda(u) = \frac{1}{2^N} \cdot \sum_{l=0}^{N} l \cdot \mu_N(l),$$

$$2^N E_N = \underbrace{\sum_{l=1}^{\lfloor \frac{N}{2} \rfloor} l \cdot 2^{2l-1}}_{S_1} + \underbrace{\sum_{l=\lceil \frac{N+1}{2} \rceil}^{N} l \cdot 2^{2(N-l)}}_{S_2}.$$

Sei zunächst N gerade. Dann ist

$$\begin{aligned}
S_1 &= \sum_{l=1}^{\frac{N}{2}} l \cdot 2^{2l-1} = \frac{3N-2}{18} \cdot 2^{N+1} + \frac{2}{9} = \frac{N}{3} \cdot 2^N - \frac{2}{9} \cdot 2^N + \frac{2}{9}, \\
S_2 &= \sum_{l=\frac{N}{2}+1}^{N} l \cdot 4^{N-l} \overset{k=N-l}{=} \sum_{k=0}^{\frac{N}{2}-1} (N-k) \cdot 4^k = N \cdot \sum_{k=0}^{\frac{N}{2}-1} 4^k - \sum_{k=0}^{\frac{N}{2}-1} k \cdot 4^k \\
&= N \cdot \frac{4^{N/2}-1}{3} - \frac{4}{9} \cdot \left[(\frac{N}{2}-1) \cdot 4^{\frac{N}{2}} - \frac{N}{2} \cdot 4^{\frac{N}{2}-1} + 1 \right] \\
&= \frac{N}{3} \cdot 2^N - \frac{N}{3} - \frac{4}{9} \cdot \left[\frac{N}{2} \cdot 2^N - 2^N - \frac{N}{8} \cdot 2^N + 1 \right] \\
&= \left(\frac{N}{6} + \frac{4}{9} \right) \cdot 2^N - \frac{N}{3} - \frac{4}{9}.
\end{aligned}$$

Zusammen ergibt das

$$2^N E_N = \frac{N}{2} \cdot 2^N + \frac{2}{9} \cdot 2^N - \frac{N}{3} - \frac{2}{9},$$

womit die erste Formel von Hauptsatz 1 für gerades N bewiesen ist.

Für ungerades N ist

$$\begin{aligned}
S_1 &= \sum_{l=1}^{\frac{N-1}{2}} l \cdot 2^{2l-1} = \frac{3(N-1)-2}{18} \cdot 2^N + \frac{2}{9} = \frac{3N-5}{18} \cdot 2^N + \frac{2}{9} \\
&= \frac{N}{6} \cdot 2^N - \frac{5}{18} \cdot 2^N + \frac{2}{9}, \\
S_2 &= \sum_{l=\frac{N+1}{2}}^{N} l \cdot 4^{N-l} \overset{k=N-l}{=} \sum_{k=0}^{\frac{N-1}{2}} (N-k) \cdot 4^k = N \cdot \sum_{k=0}^{\frac{N-1}{2}} 4^k - \sum_{k=0}^{\frac{N-1}{2}} k \cdot 4^k \\
&= N \cdot \frac{4^{(N+1)/2}-1}{3} - \frac{4}{9} \cdot \left[\frac{N-1}{2} \cdot 4^{\frac{N+1}{2}} - \frac{N+1}{2} \cdot 4^{\frac{N-1}{2}} + 1 \right] \\
&= \frac{N}{3} \cdot 2^{N+1} - \frac{N}{3} - \frac{4}{9} \cdot \left[\frac{N-1}{2} \cdot 2^{N+1} - \frac{N+1}{2} \cdot 2^{N-1} + 1 \right] \\
&= \frac{2N}{3} \cdot 2^N - \frac{N}{3} - \frac{4N}{9} \cdot 2^N + \frac{4}{9} \cdot 2^N + \frac{N}{9} \cdot 2^N + \frac{1}{9} \cdot 2^N - \frac{4}{9} \\
&= \left(\frac{N}{3} + \frac{5}{9} \right) \cdot 2^N - \frac{N}{3} - \frac{4}{9},
\end{aligned}$$

$$2^N E_N = \frac{N}{2} \cdot 2^N + \frac{5}{18} \cdot 2^N - \frac{N}{3} - \frac{2}{9},$$

womit die erste Formel von Hauptsatz 1 auch für ungerades N bewiesen ist.

Nun zur Berechnung der Varianz V_N. Es ist

$$\begin{aligned} V_N + 2^N E_N^2 &= \frac{1}{2^N} \cdot \sum_{u \in \mathbb{F}_2^N} \lambda(u)^2 = \frac{1}{2^N} \cdot \sum_{l=0}^{N} l^2 \cdot \mu_N(l), \\ &= \underbrace{\sum_{l=1}^{\lfloor \frac{N}{2} \rfloor} l^2 \cdot 2^{2l-1}}_{S_3} + \underbrace{\sum_{l=\lceil \frac{N+1}{2} \rceil}^{N} l^2 \cdot 4^{N-l}}_{S_4}. \end{aligned}$$

Sei wieder zuerst N gerade. Dann ergibt die erste Summe

$$\begin{aligned} S_3 &= \sum_{l=1}^{\frac{N}{2}} l^2 \cdot 2^{2l-1} = \frac{3 \cdot \frac{N^2}{4} - 2 \cdot \frac{N}{2}}{9} \cdot 2^{N+1} + \frac{5}{27} \cdot 2^{N+1} - \frac{10}{27} \\ &= \frac{N^2}{6} \cdot 2^N - \frac{2N}{9} \cdot 2^N + \frac{10}{27} \cdot 2^N - \frac{10}{27}. \end{aligned}$$

Die zweite Summe wird weiter zerlegt:

$$\begin{aligned} S_4 &= \sum_{l=\frac{N}{2}+1}^{N} l^2 \cdot 4^{N-l} \overset{k=N-l}{=} \sum_{k=0}^{\frac{N}{2}-1} (N-k)^2 \cdot 4^k \\ &= \underbrace{N^2 \cdot \sum_{k=0}^{\frac{N}{2}-1} 4^k}_{S_{4a}} \underbrace{-2\,N \cdot \sum_{k=0}^{\frac{N}{2}-1} k \cdot 4^k}_{S_{4b}} + \underbrace{\sum_{k=0}^{\frac{N}{2}-1} k^2 \cdot 4^k}_{S_{4c}} \end{aligned}$$

Diese werden einzeln ausgewertet:

$$\begin{aligned} S_{4a} &= N^2 \cdot \frac{4^{\frac{N}{2}} - 1}{3} = \frac{N^2}{3} \cdot 2^N - \frac{N^2}{3}, \\ S_{4b} &= N \cdot \frac{4}{9} \cdot \left[(\frac{N}{2} - 1) \cdot 4^{\frac{N}{2}} - \frac{N}{2} \cdot 4^{\frac{N}{2}-1} + 1 \right] \\ &= \frac{4N}{9} \cdot \left[\frac{N}{2} \cdot 2^N - 2^N - \frac{N}{8} \cdot 2^N + 1 \right] = \frac{N^2}{6} \cdot 2^N - \frac{4N}{9} \cdot 2^N + \frac{4N}{9}, \\ S_{4c} &= \frac{4}{27} \cdot \left[(\frac{N}{2} - 1)^2 \cdot 4^{\frac{N}{2}+1} - \left(2 \cdot (\frac{N}{2} - 1)^2 + 2 \cdot (\frac{N}{2} - 1) - 1 \right) \cdot 4^{\frac{N}{2}} \right. \\ &\qquad \left. + (\frac{N}{2})^2 \cdot 4^{\frac{N}{2}-1} - 5 \right] \\ &= \frac{4}{27} \cdot \left[2 \cdot (\frac{N^2}{4} - N + 1) \cdot 2^N - N \cdot 2^N + 2 \cdot 2^N + 2^N + \frac{N^2}{16} \cdot 2^N - 5 \right] \\ &= \frac{1}{12} \cdot N^2 \cdot 2^N - \frac{4}{9} \cdot N \cdot 2^N + \frac{20}{27} \cdot 2^N - \frac{20}{27}. \end{aligned}$$

Dazu kommt noch

$$\begin{aligned} 2^N \cdot E_N^2 &= \left[\frac{N}{2} + \frac{2}{9} - \frac{N}{3 \cdot 2^N} - \frac{2}{9 \cdot 2^N}\right]^2 \cdot 2^N \\ &= \frac{N^2}{4} \cdot 2^N + \frac{2N}{9} \cdot 2^N + \frac{4}{81} \cdot 2^N - \frac{N^2}{3} - \frac{10N}{27} - \frac{8}{81} \\ &\quad + \frac{N^2}{9 \cdot 2^N} + \frac{4N}{27 \cdot 2^N} + \frac{4}{81 \cdot 2^N} \,. \end{aligned}$$

Alles zusammen ergibt

$$2^N \cdot V_N = \frac{86}{81} \cdot 2^N - \frac{14N}{27} - \frac{82}{81} - \frac{N^2}{9 \cdot 2^N} - \frac{4N}{27 \cdot 2^N} - \frac{4}{81 \cdot 2^N},$$

womit die zweite Formel von Hauptsatz 1 für gerades N bewiesen ist.

Die entsprechende Berechnung für ungerades N ist:

$$\begin{aligned} S_3 &= \sum_{l=1}^{\frac{N-1}{2}} l^2 \cdot 2^{2l-1} = \frac{N^2}{12} \cdot 2^N - \frac{5N}{18} \cdot 2^N + \frac{41}{108} \cdot 2^N - \frac{10}{27} \,, \\ S_{4a} &= N^2 \cdot \sum_{k=0}^{\frac{N-1}{2}} 4^k = \frac{2N^2}{3} \cdot 2^N - \frac{N^2}{3} \,, \\ S_{4b} &= N \cdot \sum_{k=0}^{\frac{N-1}{2}} k \cdot 4^k = \frac{N^2}{3} \cdot 2^N - \frac{5N}{9} \cdot 2^N + \frac{4N}{9} \,, \\ S_{4c} &= \sum_{k=0}^{\frac{N-1}{2}} k^2 \cdot 4^k = \frac{N^2}{6} \cdot 2^N - \frac{5N}{9} \cdot 2^N + \frac{41}{54} \cdot 2^N - \frac{20}{27} \,, \\ 2^N \cdot E_N^2 &= \left[\frac{N}{2} + \frac{5}{18} - \frac{N}{3 \cdot 2^N} - \frac{2}{9 \cdot 2^N}\right]^2 \cdot 2^N \\ &= \frac{N^2}{4} \cdot 2^N + \frac{5N}{18} \cdot 2^N + \frac{25}{324} \cdot 2^N - \frac{N^2}{3} - \frac{11N}{27} - \frac{10}{81} \\ &\quad + \frac{N^2}{9 \cdot 2^N} + \frac{4N}{27 \cdot 2^N} + \frac{4}{81 \cdot 2^N} \,. \end{aligned}$$

Alles zusammen ergibt

$$\begin{aligned} 2^N \cdot V_N &= S_3 + S_{4a} - 2 \cdot S_{4b} + S_{4c} - 2^N \cdot E_N^2 \\ &= \frac{86}{81} \cdot 2^N - \frac{13N}{27} - \frac{80}{81} - \frac{9N^2 + 12N + 4}{81 \cdot 2^N} \,, \end{aligned}$$

womit Hauptsatz 1 vollständig bewiesen ist.

3.6 Lineare Komplexität und TURING-Komplexität

Eine **universelle** TURING-**Maschine** kann jede andere TURING-Maschine durch ein geeignetes Programm simulieren. Sei **M** eine solche, und sei $u \in \mathbb{F}_2^n$ eine Bitfolge der Länge n. Dann ist die TURING-KOLMOGOROV-CHAITIN-**Komplexität** $\chi(u)$ gleich der Länge des kürzesten Programms von **M**, das u als Output produziert. Ein Programm der ungefähren Länge n gibt es immer: Es nimmt einfach u als Input-Folge und gibt diese wieder aus.

Anmerkung. Die Funktion $\chi : \mathbb{F}_2^* \longrightarrow \mathbb{N}$ ist selbst nicht berechenbar; d. h., es gibt keine TURING-Maschine, die χ berechnet. Daher ist die TURING-KOLMOGOROV-CHAITIN-Komplexität als Komplexitätsmaß kaum praktisch verwendbar. Sie ist allerdings in den letzten Jahren durch die Arbeiten von VITANYI und anderen in präzisierter Form wieder in Mode gekommen; siehe dazu etwa:

- MING LI, PAUL VITANYI: *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, New York 1993, 1997.

Eine wichtige Aussage der Theorie ist:

$$\frac{1}{2^n} \cdot \#\{u \in \mathbb{F}_2^n \mid \chi(u) > n \cdot (1-\varepsilon)\} > 1 - \frac{1}{2^{n\varepsilon - 1}},$$

d. h., fast alle Folgen haben eine TKC-Komplexität nahe am maximalen Wert, also keine wesentlich kürzere Beschreibung als durch vollständiges Hinschreiben. Eine gängige Interpretation dieses Sachverhaltes ist: „Fast alle Folgen sind zufällig." Dies entspricht der intuitiven Vorstellung von Zufall sehr gut. Eine Folge mit einer kurzen Beschreibung wie „eine Million Mal abwechselnd 0 und 1" wird nämlich niemand als auch nur im geringsten zufällig ansehen.

Das Komplexitätsmaß „lineare Komplexität" λ, das auf dem sehr speziellen Maschinenmodell des linearen Schieberegisters beruht, hat dagegen auf den ersten Blick gravierende Mängel. Die Folge „999999 Mal die 0, dann eine 1" hat eine sehr geringe TKC-Komplexität – und eine sehr geringe intuitive Zufälligkeit –, aber die lineare Komplexität 1 Million. Der Vorteil der linearen Komplexität ist, wie gesehen, ihre leichte explizite Bestimmbarkeit, und sie beschreibt „im allgemeinen" die Zufälligkeit ener Bitfolge doch recht gut. Diese Aussage lässt sich überraschend präzise fassen (ohne Beweis):

Satz 3 (BETH/DAI, EUROCRYPT 89)

$$\begin{aligned}
\frac{1}{2^n} \cdot \#\{u \in \mathbb{F}_2^n \mid (1-\varepsilon)\lambda(u) \le \chi(u)\} &\ge 1 - \frac{8}{3 \cdot 2^{\frac{n\varepsilon}{2-\varepsilon}}}, \\
\frac{1}{2^n} \cdot \#\{u \in \mathbb{F}_2^n \mid (1-\varepsilon)\chi(u) \le \lambda(u)\} &\ge 1 - \frac{1}{3} \cdot \frac{1}{2^{n\varepsilon - (1-\varepsilon)(1+\log n)+1}} - \frac{1}{3} \cdot \frac{1}{2^n}.
\end{aligned}$$

Interpretation: „Für fast alle Bitfolgen stimmen die lineare Komplexität und die TKC-Komplexität mit vernachlässigbarer Abweichung überein.“

Damit ist klar, dass die lineare Komplexität trotz ihrer Einfachheit ein gutes Komplexitätsmaß ist, und dass Bitfolgen hoher linearer Komplexität im allgemeinen auch mit anderen Ansätzen nicht kürzer erklärbar sind. Sie sind also kryptographisch brauchbar. (Ein anderer Ansatz wäre etwa, die lineare Komplexität auf andere endliche Körper oder Restklassenringe zu verallgemeinern – das würde also dem Kryptoanalytiker kaum nützen.) Jedes effiziente Vorhersageverfahren im Sinne der Kryptoanalyse von Bitstromchiffren wäre auch eine Kurzbeschreibung im Sinne der TKC-Komplexität, so dass man als Fazit festhalten kann: *Bitfolgen hoher linearer Komplexität sind im allgemeinen nicht vorhersagbar.*

Natürlich gibt es auch Ansätze, nichtlineare Schieberegister zur Beurteilung der Komplexität heranzuziehen, siehe etwa:

- Agnes Hui Chan, Richard A. Games: On the quadratic span of periodic sequences. Crypto 89, 82–89.
- Cees J. A. Jansen, Dick E. Boekee: The shortest feedback shift register that can generate a given sequence. Crypto 89, 90–96.

3.7 Nichtlinearität für Schieberegister – Ansätze

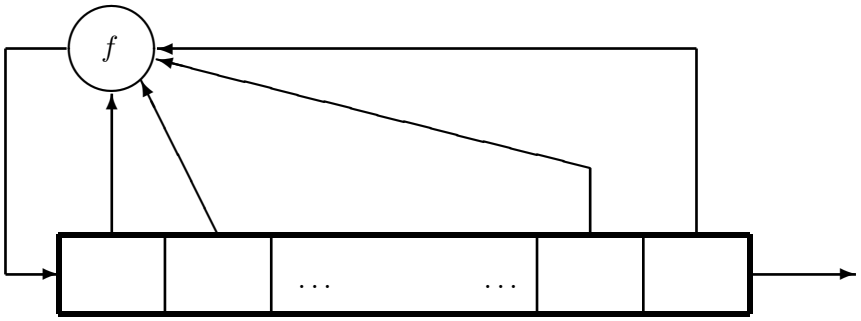
Lineare Schieberegister sind beliebt – vor allem bei Elektro-Ingenieuren und beim Militär – denn sie sind

- sehr einfach zu realisieren,
- extrem effizient, vor allem in Hardware,
- als Zufallsgeneratoren für statistische Zwecke sehr gut geeignet,
- problemlos parallel zu betreiben,
- aber leider kryptologisch völlig unsicher.

Um die positiven Eigenschaften zu nutzen und die kryptologische Schwäche zu vermeiden, gibt es verschiedene Ansätze.

Ansatz 1: Nichtlineare Rückkopplung

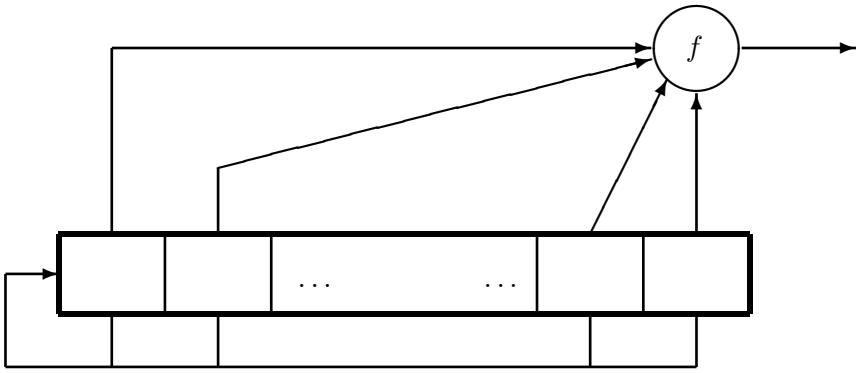
Die nichtlineare Rückkopplung (nonlinear feedback) folgt dem Schema:



Sie wurde schon in Abschnitt 2.6 behandelt und dort als kryptographisch nicht hinreichend sicher eingestuft.

Ansatz 2: Nichtlinearer Ausgabefilter

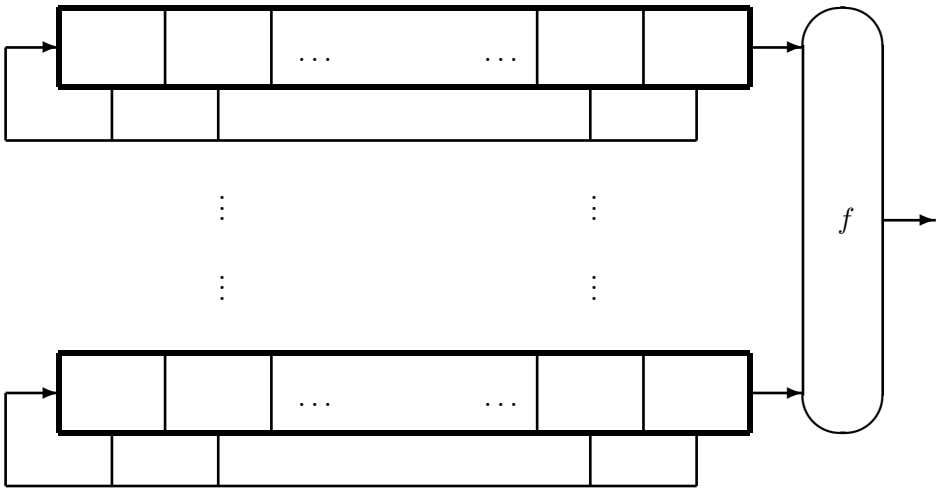
Der nichtlineare Ausgabefilter (nonlinear feed forward) folgt dem Schema:



Das Schieberegister selbst ist linear. Der nichtlineare Ausgabefilter ist ein Spezialfall des nächsten Ansatzes. (**Übungsaufgabe:** Wie?)

Ansatz 3: Nichtlinearer Kombinierer

Hier wird eine „Batterie“ aus n linearen Schieberegistern – die durchaus unterschiedliche Länge haben können und sollen – parallel betrieben. Ihre Outputfolgen werden in eine BOOLEsche Funktion $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ gefüttert:



Die ausführliche Diskussion dieses Verfahrens folgt in Abschnitt 3.8 ff. Natürlich kann man auch allgemeiner eine BOOLEsche Abbildung $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^q$ verwenden, die in jedem Takt q Bits ausgibt.

Ansatz 4: Auswahlsteuerung/Dezimierung/Taktung

Weitere Möglichkeiten bestehen in verschiedenen Ansätzen zur Steuerung einer Batterie von n parallel betriebenen linearen Schieberegistern

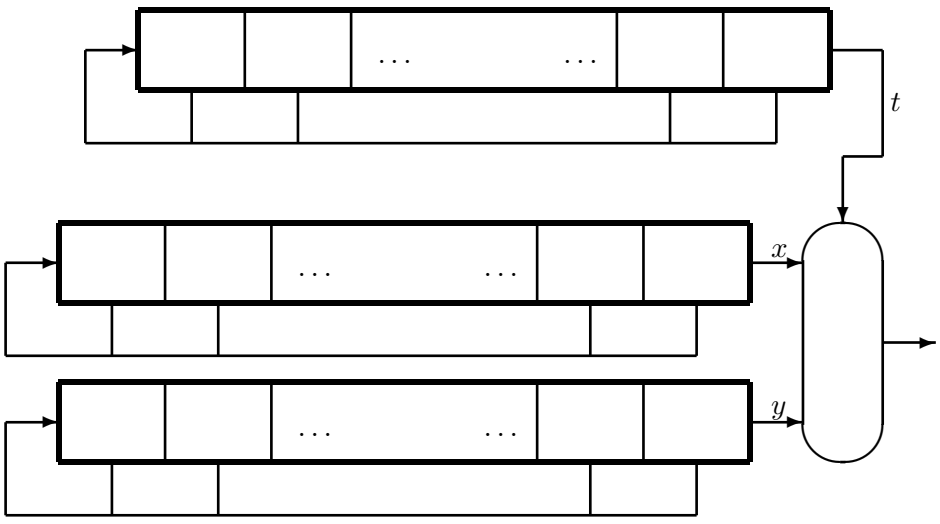
durch ein weiteres lineares Schieberegister:

- Bei der **Auswahlsteuerung** wird je nach Zustand des „Hilfsregisters“ das aktuelle Output-Bit von genau einem der „Batterie-Register“ als Output des Zufallsgenerators ausgewählt. Allgemeiner kann man auch eine Auswahl „r aus n“ treffen.
- Bei der **Dezimierung** nimmt man im allgemeinen $n = 1$ an und gibt das Output-Bit des einen Batterie-Registers nur dann aus, wenn das Hilfsregister einen bestimmten Zustand hat. Diese Art der Dezimierung kann man natürlich analog auf jede Bitfolge anwenden.
- Bei der **Taktung** gibt der Zustand des Hilfsregisters an, welche der Batterie-Register im aktuellen Taktzyklus weitergeschoben werden (und um wieviele Positionen) und welche in ihrem momentanen Zustand bleiben. Das ist vergleichbar mit der Steuerlogik von Rotor-Maschinen.

Diese Ansätze lassen sich oft bequem auch als nichtlineare Kombinierer schreiben, so dass Ansatz 3 als *der* Ansatz zur Rettung der linearen Schieberegister angesehen werden kann.

Beispiel: Der Geffe-Generator

Das einfachste Beispiel für die Auswahlsteuerung ist der Geffe-Generator, der durch das folgende Schema beschrieben wird:



Die Ausgabe ist x, wenn $t = 0$, und y, wenn $t = 1$. Das kann man so als Formel ausdrücken:

$$\begin{aligned} u &= \begin{cases} x, & \text{wenn } t = 0, \\ y, & \text{wenn } t = 1 \end{cases} \\ &= (1-t)x + ty = x + tx + ty. \end{aligned}$$

Also lässt sich der GEFFE-Generator auch durch einen nichtlinearen Kombinierer mit einer BOOLEschen Funktion $f\colon \mathbb{F}_2^3 \longrightarrow \mathbb{F}_2$ vom Grad 2 beschreiben.

3.8 Nichtlineare Kombinierer

Ein nichtlinearer Kombinierer wird beschrieben durch eine BOOLEsche Funktion $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ sowie eine Batterie aus n linearen Schieberegistern der Längen $l_1, \ldots, l_n$.

Bemerkungen

1. Sind die Schieberegister und f bekannt, so besteht ein Schlüssel der zugehörigen Bitstrom-Chiffre aus dem n-Tupel der Startvektoren, also hat der Schlüsselraum die Größe $2^{l_1} \cdots 2^{l_n}$.

2. Die lineare Komplexität der erzeugten Bitfolge ist „im allgemeinen“ $f(l_1, \ldots, l_n)$, wobei f in algebraischer Normalform geschrieben und als Polynom $f \in \mathbb{Z}[X]$ ausgewertet wird. Insbesondere ist ein möglichst hoher Grad von f erstrebenswert.

Beispiele

1. Der GEFFE-Generator ließ sich als nichtlinearer Kombinierer mit $f(x, t, y) = x + tx + ty$ deuten. Er hat die sehr große Periode $(2^{l_1} - 1)(2^{l_2} - 1)(2^{l_3} - 1)$, einen Schlüsselraum der Größe $2^{l_1+l_2+l_3}$, sowie die recht beachtliche lineare Komplexität $l_1 + l_1 l_2 + l_2 l_3$.

2. Nimmt man bei n beliebigen „Batterie-Registern“ als Kombinier-Funktion $f = T_1 \cdots T_n \in \mathbb{F}_2[T]$, also einfach die Multiplikation in $\mathbb{F}_2$, so hat die erzeugte Folge die lineare Komplexität $\leq l_1 \cdots l_n = f(l_1, \ldots, l_n)$. Hinreichend für die Gleichheit ist:

   (a) Alle charakteristischen Polynome der Batterie-Register sind irreduzibel,

   (b) die Längen $l_1, \ldots, l_n$ sind paarweise teilerfremd.

3.9 Korrelationsattacken – die Achillesferse der Kombinierer

Sie $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ die Kombinierfunktion eines nichtlinearen Kombinierers. Die Anzahl

$$K_f := \#\{x = (x_1, \ldots, x_n) \in \mathbb{F}_2^n \mid f(x) = x_1\}$$

misst, wie oft der Funktionswert mit dem ersten Argument übereinstimmt. Ist sie $> 2^{n-1}$, so ist die Wahrscheinlichkeit für diese Übereinstimmung

$$p = \frac{1}{2^n} \cdot K_f > \frac{1}{2},$$

also überdurchschnittlich. Die kombinierte Outputfolge „korreliert“ also stärker mit dem Output des ersten linearen Schieberegisters, als zufällig zu erwarten wäre.

Diesen Effekt kann sich der Kryptoanalytiker bei einem Angriff mit bekanntem Klartext zunutze machen: Die (ersten) Schlüsselbits $b_0, \ldots, b_{r-1}$ seien bekannt. Mit einer Exhaustion über die 2^{l_1} Startvektoren des ersten Registers erzeugt man jedesmal die Folge $u_0, \ldots, u_{r-1}$ und zählt die Koinzidenzen. Zu erwarten ist

$$\frac{1}{2^r} \cdot \#\{i \mid u_i = b_i\} \approx \begin{cases} p & \text{beim richtigen Startvektor,} \\ \frac{1}{2} & \text{sonst.} \end{cases}$$

Falls r groß genug ist, kann man also den echten Startvektor des ersten Registers mit einem Aufwand $\sim 2^{l_1}$ bestimmen. Macht man dann mit den anderen Registern genauso weiter, gelingt die Identifikation des gesamten Schlüssels mit einem Aufwand $\sim 2^{l_1} + \cdots + 2^{l_n}$. Das ist zwar exponentiell, aber wesentlich geringer als der Aufwand $\sim 2^{l_1} \cdots 2^{l_n}$ für die naive vollständige Schlüsselsuche.

In der Sprache von Kapitel II haben wir hier die lineare Relation (T_1, T) für f ausgenutzt. Klar ist, dass man analog jede lineare Relation ausnutzen kann, um die Komplexität der vollständigen Schlüsselsuche zu reduzieren.

Beispiel: Geffe-Generator. Hier werden die Korrelationen durch die folgende Tabelle beschrieben:

x	0	0	0	0	1	1	1	1
t	0	0	1	1	0	0	1	1
y	0	1	0	1	0	1	0	1
$f(x,t,y)$	0	0	0	1	1	1	0	1

Als Wahrscheinlichkeit für die Übereinstimmung erhält man also

$$p = \begin{cases} \frac{3}{4} & \text{für das Register 1,} \\ \frac{1}{2} & \text{für das Register 2 (Steuerung),} \\ \frac{3}{4} & \text{für das Register 3.} \end{cases}$$

Daher lassen sich bei einer Korrelationsattacke die Startwerte für die Register 1 und 3 – die Batterieregister – leicht schon aus kurzen Outputfolgen bestimmen; den Startwert für Register 2, das Steuerungsregister, findet man dann auch leicht durch Exhaustion seiner Startvektoren.

Aus der bisherigen Diskussion lassen sich als Design-Kriterien für nichtlineare Kombinierer herleiten:

- Die einzelnen Batterieregister müssen möglichst lang sein; genaueres siehe unten.
- Die Kombinierfunktion f muss balanciert sein
- ... und soll eine möglichst hohe Nichtlinearität, d. h., ein möglichst geringes lineares Potenzial haben.

Zum letzten Punkt: Wegen der nötigen Balanciertheit sind krumme Funktionen trotz ihrer „Korrelationsimmunität“ nicht geeignet. Die nächst beste Eigenschaft ist „resilient“ = unter den balancierten maximal nichtlinear.

Wie lang sollen die Batterieregister sein? Es gibt verschiedene Ansätze zu „schnellen“ Korrelationsattacken, z. B. mit Hilfe der Walsh-Transformation, besonders gegen dünn besetzte Rückkopplungspolynome. Diese reduzieren zwar nicht die Komplexitätsklasse des Angriffs, aber der Aufwand wird um einen beträchtlichen Proportionalitätsfaktor verringert. Auf diese Weise werden Register angreifbar, die bis zu 100 Koeffizienten 1 im Rückkopplungspolynom haben. Folgerung:

- Die einzelnen linearen Schieberegister sollten mindestens 200 Bits lang sein und eine „dicht besetzte“ Rückkopplung besitzen.

Ein eleganter Ausweg, der die Korrelationsattacke zusammenbrechen lässt, wurde von Rueppel vorgeschlagen: eine zeitabhängige Kombinierfunktion, also eine Familie $(f_t)_{t\in\mathbb{N}}$ zu verwenden.

Als Fazit kann man festhalten: Mit linearen Schieberegistern und nichtlinearen Kombinierern lassen sich sich ziemlich effiziente Pseudozufallsgeneratoren aufbauen. Für deren kryptologische Sicherheit gibt es zwar keine umfassende befriedigende Theorie, aber durchaus eine Absicherung, die – ähnlich wie bei Bitblock-Chiffren – auf der Theorie der Nichtlinearität Boolescher Funktionen beruht.