2.8 Analyse bei gestutztem Output

Schwieriger wird die Kryptoanalyse, wenn der Zufallsgenerator nicht alle Bits der erzeugten Zahlen ausgibt. Das kann auf Absicht beruhen, aber auch als Nebeneffekt dadurch entstehen, dass die Zahlen ins reelle Intervall $[0, 1]$ oder sonstwie transformiert und dabei gerundet werden; besonders wenn der Modul m keine Zweierpotenz ist, muss man den Verlust einiger gering signifikanter Bits in Erwägung ziehen. Die Differenzenfolge ist dann natürlich auch nur ungefähr bekannt, die größten gemeinsamen Teiler sind nicht mehr zu ermitteln, und die Algorithmen von PLUMSTEAD-BOYAR und BOYAR/KRAWCZYK brechen zusammen.

Sind die Parameter des Zufallsgenerators bekannt, kann man es mit systematischem Probieren versuchen. Für die folgende Überlegung (die im übrigen nicht streng durchgeführt wird) muss der Zufallsgenerator nicht einmal notwendig linear sein. Nehmen wir an, es werden n-Bit-Zahlen erzeugt, aber jeweils nur q Bits ausgegeben und $n - q$ Bits zurückgehalten. Die ausgegebenen Bits stammen jeweils von festen, bekannten Positionen. Dann gibt es zu jedem ausgegebenen q-Bit-Fragment 2^{n-q} mögliche vollständige Werte; anders ausgedrückt, enthält eine zufällig gewählte n-Bit-Zahl mit der Wahrscheinlichkeit $1/2^q$ die vorgegebenen Bits an den richtigen Stellen.

Die weitere Überlegung wird hier nur exemplarisch für den Fall durchgeführt, dass die q ausgegebenen Bits die Leitbits sind. Man zerlegt also den Startwert x in $x = x_0 2^{n-q} + x_1$ mit $0 \le x_1 < 2^{n-q}$. Der Wert x_0, die ersten q Bits, ist bekannt. Der Angreifer startet eine Exhaustion über die 2^{n-q} verschiedenen möglichen Werte für x_1. Zu jeder Wahl von x_1 bildet er $x = x_0 2^{n-q} + x_1$ und $y = s(x)$ mit der erzeugenden Funktion s des betrachteten Zufallsgenerators. Diesen Wert y vergleicht er mit den ihm bekannten führenden q Bits des wahren Wertes. Ist der Zufallsgenerator statistisch gut, so ist die Wahrscheinlichkeit eines Treffers hierbei $1/2^q$. Das bedeutet, dass von den 2^{n-q} Werten für x_0 noch ungefähr 2^{n-2q} übrig bleiben. Falls $q \ge \frac{n}{2}$, können wir also genau einen Treffer erwarten. Ansonsten wird weitergemacht. Nach k Schritten ist die Trefferzahl ungefähr 2^{n-kq}. Die zu erwartende nötige Schrittzahl ist also $\ge k$ nur, wenn $kq \le n$, also $q \ge \frac{n}{k}$. Bei $q = \frac{1}{4}$ (zum Beispiel $n = 32$, $q = 8$, d. h., Ausgabe von 8 Bit einer 32-Bit-Zahl) reichen also vier q-Bit-Fragmente (wobei man im Beispiel allerdings schon 2^{24} Zahlen durchprobieren muss). Dieses Probierverfahren ist bei kleinem Modul m durchführbar; der Aufwand wächst aber exponentiell mit m (wenn der Anteil $r = \frac{q}{n}$ der ausgegebenen Bits gegen 1 beschränkt bleibt).

Für lineare Kongruenzgeneratoren haben FRIEZE/KENNAN/LAGARIAS, HÅSTAD/SHAMIR und J. STERN ein besseres (probabilistisches) Verfahren entwickelt, dessen erster Schritt im folgenden Satz resümiert wird (ohne Beweis).

Satz 7 (FRIEZE, KANNAN, LAGARIAS) *Sei* $0 \le r \le 1$ *und* p_n *die Wahr-*

scheinlichkeit, dass das Verfahren nicht den Modul m eines linearen Kongruenzgenerators bestimmt. Dann gilt für beliebiges $\varepsilon > 0$

$$p_n = \mathrm{O}(n^{\frac{5r-3}{2}+\varepsilon}).$$

Insbesondere $p_n \to 0$ mit $n \to \infty$, wenn $r > \frac{2}{5}$. *Es gelingt also mit großer Wahrscheinlichkeit, m zu finden, wenn mehr als $2/5$ der Leitbits ausgegeben werden.*

Im zweiten Schritt geht es darum, den Multiplikator a unter der Annahme zu bestimmen, dass der Modul m schon bekannt ist. Im dritten Schritt sind noch die vollen Zahlen x_i oder die Differenzen y_i zu bestimmen. Auch dies gelingt außer für eine vernachlässigbare Menge von Multiplikatoren, die mit wachsendem m immer kleiner gewählt werden kann, und für die „guten“ Multiplikatoren benötigt man nur noch etwas mehr als ein Drittel der Leitbits von x_0, x_1, x_2 und x_3, um die volle Bitinformation herzuleiten. Ähnliche, etwas schwächere Ergebnisse hat J. Stern auch für den Fall gefunden, dass statt der Leitbits „innere Bits“ der erzeugten Zahlen ausgegeben werden.

Die Kryptoanalyse der linearen Kongruenzgeneratoren hat also grundsätzliche Schwächen aufgedeckt, und zwar unabhängig davon, ob ein solcher Generator statistisch gute oder schlechte Eigenschaften hat.

Trotzdem sind die linearen Kongruenzgeneratoren für statistische Anwendungen durchaus brauchbar, denn es scheint extrem unwahrscheinlich, dass ein Anwendungsprogramm „aus Versehen“ die nötigen Schritte enthält, um einen linearen Kongruenzgenerator zu knacken und so seinen Determinismus aufzudecken. Für die kryptographische Anwendung sind die linearen Kongruenzgeneratoren auch bei gestutztem Output aber ein für allemal disqualifiziert. Offen ist allerdings, ob die Einwände auch zutreffen, wenn man nur „ganz wenige“ Bits ausgibt (etwa nur ein Viertel oder gar nur $\log\log(m)$ Bits).

Literaturverweise

- J. Stern: Secret linear congruential generators are not cryptographically secure. FOCS 28 (1987), 421–426.
- Frieze/Håstad/Kannan/Lagarias/Shamir: Reconstructing truncated integer variables satisfying linear congruences. SIAM J. Comput. 17 (1988), 262–280.
- J. Boyar: Inferring sequences produced by a linear congruential generator missing low-oerder bits. J. Cryptology 1 (1989), 177–184.