Zusammenfassung „Zufall im Rechner" (Jochen Rondorf)

theoretischer Teil

Es gibt zwei Arten von Zufall: **„echten" und computergenierten Zufall.** Beispiele für echten Zufall, deren Daten aus nicht-deterministischen Quellen stammen, sind Münzwerfen („Kopf" oder „Zahl"), Ziehung der Lottozahlen mit einer Maschine oder Würfeln. Im Bereich der Physik können auch Messungen im Zusammenhang mit der Erzeugung und des Registrierens von Zählimpulsen, z.B. beim radioaktiven Zerfall, echten Zufall erzeugen, im Computerbereich können Tastaturanschläge und Mausbewegungen mit in die Berechnungen einfließen. Manchmal ist es zu aufwendig, reale Zufallszahlen zu erzeugen. Dann verwendet man sogenannte Pseudozufallsgeneratoren. Dies ist ein Algorithmus, der aus einer kurzen Folge von Zufallszahlen, eine lange Folge berechnet, die zufällig „aussieht", die also nicht in Polynomzeit von einer wirklichen zufälligen Folge unterschieden werden kann. Computererzeugte Zufallszahlen sind periodisch. Das heißt, nach einer gewissen Zeit folgen die Zahlen in einer schon zuvor produzierten Reihenfolge. Hieraus folgt, dass Zufallszahlen theoretisch vorhersehbar sind. Zufallszahlen brauchen wir heute in verschiedenen Bereichen. Anwendung finden Pseudozufallsgeneratoren bei statistischen Simulationen, numerischen Analysen, unvoreingenommene Entscheidungsfindungen (auch für optimale Strategien bei Computerspielen) und in der Computerprogrammierung. Der letztgenannte Punkt ist sehr umfangreich. Hier denke man insbesondere an kryptografische Algorithmen, Tests von Effektivität und Effizienz von Programmen und Computerspiele mit Würfeln, Rouletterädern oder gemischten Spielkarten. Für kryptografischen Verfahren (Schlüsselgenierung bei Public-Key-Verschlüsselung, Erzeugung von Onetime-Pads) bedient man sich heutzutage aus sicherheitstechnischen Gründen echtem Zufall.

Es gibt fünf **Eigenschaften von „guten" Zufallszahlen**. Diese sollen nämlich gleichverteilt, unabhängig, mit großer Periode errechnet, reproduzierbar und einfach und schnell (effizient) berechenbar sein. Für sicherheitsrelevante Programme ist es unumgänglich, unvorhersehbare Zufallszahlen erzeugen zu können. Diese sind mit einem Computer aber nur schwer zu generieren. Das liegt daran, weil Rechner so gebaut sind, dass sie nur eine Menge genau definierter Kommandos ausführen können, die bei nochmaligem Aufruf nach genau demselben Schema abgearbeitet werden und dieselben Ergebnisse produzieren. Dadurch kann jeder fixe Algorithmus dazu benutzt werden, genau dieselben Ergebnisse auf einem anderen Computer nachzustellen.

Die Gleichverteilung von Zufallszahlen kann man über die **Entropie** beschreiben. Um Entropie mathematisch zu beschreiben, betrachten wir einen Zufallsgenerator, der n verschiedene Zahlen ausgibt, als Informationsquelle mit endlichem Alphabet $A = (a_1, a_2, ..., a_n)$. Die Wahrscheinlichkeit des Generators, die Zahl a_i auszugeben, nennen wir p_i. Dann ist die negierte Summe über $p_i * \log_2(p_i)$ die Entropie dieser Informationsquelle. ($\log_2(x)$ für $0<x<1$ ist negativ, somit muss die Summe negiert werden, um ein positives Ergebnis

zu erhalten.) Sie gibt ein Maß für den "Informationsgehalt" der Quelle. Die Quelle hat keinen Informationsgehalt, weil wir nicht wissen, welche Information als nächstes kommt. Damit gibt die Entropie wieder, wie "überraschend" ein Zeichen kommen kann, und beim Zufall wollen wir, dass die "Überraschung" maximal ist.

Mathematisch wird der **x^2-Test** (Chi-Quadrat-Test) bewerkstelligt, indem die Summe der Quadrate der einzelnen Häufigkeiten Y_s durch die zu erwartende Häufigkeit $n*p_s$ dividiert wird. (Das n kann vor das Summenzeichen gezogen werden.) Danach wird die Größe der Folge subtrahiert: $V = \frac{1}{n} \sum_{1 \le s \le k} \left(\frac{Y_s^2}{p_s} \right) - n$

Der **Spektraltest** ist ein sehr mächtiger und daher auch sehr komplexer und schwer verständlicher Zufallszahlenreihentest. Die Bedeutung des Spektraltests ist sehr groß und er wird daher oft eingesetzt. Der von Knuth beschriebene Spektraltest richtet sich ausschließlich auf Zufallszahlenreihen, die mittels des LCG generiert wurden. Er untersucht, wie stark eine Zufallszahl von ihrem Vorgänger abhängt. Ein LCG mit etwa der Formel $X_{n+1} = (137\ X_n + 187) \bmod 256$ ergibt ein Diagramm, dass parallele Linien etwa im Winkel 30° aufweist, die teilweise noch im gleichen Abstand zueinander stehen. Sollte sich die Existenz solcher Muster auch bei größeren Moduli beweisen lassen, besteht der Generator den Spektraltest nicht.

Mittels des **Run-Tests** überprüft man, wie viele Zahlen hintereinander ohne Unterbrechung aufwärts ansteigen („run-up") bzw. abwärts abfallen („run-down"). Mit dieser Methode kann ich trotz etwa eines bestandenen Entropietests mit maximaler Entropie eine schlechte Zufallsreihenfolge nachweisen. Zur Interpretation des Run-Tests ist es nötig, in Tabellen nachzuschauen, die ausschließlich für Run-Tests angefertigt wurden, wie gut (relativ wahrscheinlich) oder schlecht (relativ unwahrscheinlich) eine Zufallsfolge ist.

Eine bekannte Art und Weise, Zufallszahlen zu generieren, sind die **linearen Kongruenzgeneratoren** (LCGs) nach Lehmer. Sie sind nach der Formel

$X_n = (a * X_{n-1} + b) \bmod m$ mit ggt(b, m) = 1, 0 < a < m, 0 < b < m

X_0 = Startwert, a = Multiplikator, b = Inkrement, m = Modulus

aufgebaut. Der Modulus m sollte sehr hoch gewählt werden, da die Periode der Zufallszahlenreihe höchstens m betragen kann. Dies liegt daran, dass die Zufallszahl nur von der vorherigen rechnerisch abhängt. Eine Zahl, die das zweite Mal in der Reihe vorkommt, ist damit automatisch Beginn der Periode. Für $m > 10^8$ spricht Knuth von zufriedenstellenden Perioden. Damit die Zahlen des Generators unabhängig sind, darf nicht x_n, sondern nur ein Teil davon ausgegebenen werden, etwa eine geeignete Anzahl der niederwertigsten Bits, nur jedes zweite Bit, oder zwei Bits paarweise XOR-verknüpft. Durch diese Verkleinerung des Ausgabebereichs, vergrößert sich die „Unabhängigkeit". Wer ohne Langzahlarithmetik einen LCG nachbauen möchte, bedient sich am besten der Parameter a = 1103515245, b = 12345, $m = 2^{32}$ = 4294967296. Hierbei ist der Modulus mit

2^{32} so gewählt, dass seine Berechnung durch Shiften sehr schnell durchgeführt werden kann, allerdings auf Kosten der Periodenlänge, die sich auf (m/4) verkürzt.

Auch **Schieberegister mit Rückkopplung** werden zur Erzeugung von Pseudozufallszahlen verwendet. In der Literatur sind vielseitige Varianten von rückgekoppelten Schieberegisterfolgen zu finden. Die Arbeitsweise des dargestellten Schieberegisters kann folgendermaßen beschrieben werden: Immer wenn ein Pseudozufallsbit benötigt wird, selektiert man das niederwertigste Bit des Schieberegisters b1, das zum Ausgabebit wird. Das so gewonnene Ausgabebit wird mit einem Maskenwort (das gleich viele Bits wie das Schieberegister hat) UND-verknüpft. Das Ergebnis dieser UND-Verknüpfung wird mit den einzelnen Bits des Schieberegisters XOR verknüpft. Das Ergebnis dieser XOR-Verknüpfung wird wieder in die Schieberegisterbits eingespeichert. Nach Ausführung dieser Rückkopplungsfunktion werden alle Bits des Schieberegisters um eine Stelle nach rechts geschoben. Das höchstwertige Bit erhält den Wert des Ausgabebits.

praktischer Teil

Für eine **TCP/IP-Verbindung** von einem Client zu einem Host, generiert der Host eine sogenannte **Initial Sequence Number** (ISN). Diese wird benötigt, um jedes Packet einer Verbindung nachverfolgen zu können und um sicherzustellen, dass die Verbindung richtig fortgeführt werden kann. Sowohl Client als auch Host benutzen diese Sequenznummern innerhalb von TCP/IP-Verbindungen. Seit 1985 sind Spekulationen entfacht worden, dass ein Angreifer durch Erraten der nächsten ISN eine unidirektionale Verbindung zu einem Host aufbauen könne. Mittels IP-Spoofing der Quell-IP-Adresse wäre es dem Hacker möglich, mit dem Host zu kommunizieren, obwohl er keine Empfangsbestätigungspakete („acknowledgement pakets"), die die ISN beinhalten, empfangen kann, da diese an die gespoofte IP geschickt werden. Es wurde festgelegt, dass jedem Stream eine eindeutige, zufällige Sequenznummer zugeordnet wird, um die Integrität einer TCP/IP-Verbindung zu gewährleisten. Das TCP-Sequenznummernfeld kann einen 32-Bit-Wert beinhalten. Die RFC-Spezifikationen schreiben einen Wert von 31 Bit vor. Ein Angreifer, der beabsichtigt, eine Verbindung von einer vorgetäuschten IP-Adresse aufzubauen oder eine existierende Verbindung zu kompromittieren, indem er manipulierte Pakete in den TCP-Stream einschleust, muss die ISN wissen.

Der Zufallszahlengenerator der Unix-Devices **/dev/random und /dev/urandom** sammelt Umgebungsrauschen von Gerätetreibern und anderen Quellen in einem Entropie-Pool. Hieraus werden die Zufallszahlen generiert. Bei Lesevorgängen auf die Gerätedatei /dev/random werden nur so viele Zufallsbytes ausgegeben, wie im Entropie-Pool zur Verfügung stehen. Damit genügt dieses Device hohen Ansprüchen bezüglich qualitativ gutem Zufall, etwa für Onetime-Pads oder Schlüsselerstellung für kryptografische Verfahren. Sollten sich keine Bytes mehr im Entropie-Pool befinden, wird /dev/random solange geblockt, bis wieder genug Entropie gesammelt wurde. Im Gegensatz hierzu liefert die Gerätedatei /dev/urandom so viele Bytes aus, wie angefordert wurden.