Pseudozufallsgeneratoren

Rupert J. Hartung

Vorlesung Kryptographie, SS 2007

Gliederung

Zufall in Algorithmen

Zweck zufälliger Algorithmen:

- Effizienz (in Erwartung),
- Simulation typischer Fälle,
- Zufall als Ergebnis gewollt (z.B. Spiele).
- ...

Kryptographische Algorithmen

Rolle des Zufalls in der Kryptographie:

- Schlüsselauswahl,
- randomisierte Verschlüsselung,
- Verhehlen des Klartextes (’hiding’),
- . . .

Unser Ziel

erzeuge effizient Werte einer ZV

Probleme mit dem Zufall

- Woher bekommt man Zufall?
- Was ist Zufall?
- Gibt es überhaupt Zufall?

Woher bekommt man Zufall?

- Ziel: ZV mit gegebener Verteilung realisieren
- für große Klasse von ZV genügen unabh. gleichverteilte ZV

Was ist Zufall?

Vergleich mit ‚echten' Zufallsbits

Verhalten soll möglichst *unregelmäßig, unvorhersagbar* sein

Dilemma

– rein deterministisch kann kein Zufall erzeugt werden ! –

Anzahl der Ausgänge hängt von der Anzahl der Zufallsbits ab

nur Aussage für viele Ziehungen

Ansatz

Wir richten uns nach folgender Strategie:

- nimm bereits zufällige Eingabe ω an (’Seed’: Samen, Saat);
- dieser Seed wird deterministisch vergrößert;
- das (längere) Ergebnis soll möglichst ’zufällig’ aussehen.

Philosophische Bemerkung

- Unterschied *randomness* und *coincidence*
- zur Seed-Erzeugung benutzen wir „Konizidenzen“

Polynom-Iteration

Sei X eine Zufallsvariable, $f \in \mathbb{Z}_N[x]$.

Betrachte die Folge

$$X_0 := X \mod N, \qquad X_{i+1} = f(X_i) \mod N$$

Wie „zufällig“ ist die Folge $(X_i)_i$?

Linear Congruential Generator

Erster Versuch:

$$f(x) = ax + c$$

Beispiel: $N = 9, \quad a = 4, \quad c = 1$

$$0, \quad 1, \quad 5, \quad 3, \quad 4, \quad 8, \quad 6, \quad 7, \quad 2, \quad 0, \quad 1, \quad 5, \quad \ldots$$
$$0, \quad 1, \quad 1, \quad 1, \quad 0, \quad 0, \quad 0, \quad 1, \quad 0, \quad 1, \quad 1, \quad \ldots$$

Periode

Beobachtung: Bei der Polynom-Iteration wiederholen sich Werte X_i spätestens nach N Proben

$\rightsquigarrow$ nur wenige Iterationen (meist $\mathcal{O}(\log N)$)

Mängel des LCG

Sei $(Z_i)_i$ Folge unabhängiger gleichverteilter ZV in $\mathbb{Z}_N$.

Lemma: Für jedes Paar $(u, v) \in \mathbb{Z}_N^2$ ist
$\mathbb{P}(Z_1 = u,\, Z_2 = v) = N^{-2}$.

Für $(X_i)_i$ gilt das nicht: X_1 legt X_2 fest.

$(X_i)_i$ **schon nach einer Iteration klar von Zufallsfolge unterscheidbar!**

Anforderung: Statistische Tests

- Untersuchung von $\mathbb{P}(Z_1 = u, Z_2 = v)$ ist typischer „Test“
- Forderung: Unterscheidung von (X_i) und (Z_i) ist schwer
- statistischer Test als prob. Turingmaschine
- daher: Wertebereich von (X_i), (Z_i) muss unbeschränkt wachsen (’Ensembles’)

Statistischer Test

Seien $(X_i)_i$ ZV auf $\{0,1\}^{l(i)}$,
sei $(Z_i)_i$ unabhängig gleichverteilt darauf (l ein Polynom).

Definition: Ein *statistischer Test* ist ein probabilistischer Polynomialzeit-Algorithmus mit Ausgabe in $\{0,1\}$. Setze

$$\Delta(\mathcal{D}, X_i) := \mathbb{P}\left[\mathcal{D}(X_i) = 1\right] \quad - \quad \mathbb{P}\left[\mathcal{D}(Z_i) = 1\right]$$

Unterscheiden wird modelliert als verschiedene Ausgaben

Pseudozufall

$$\Delta(\mathcal{D}, X_i) := \mathbb{P}\left[\mathcal{D}(X_i) = 1\right] \quad - \quad \mathbb{P}\left[\mathcal{D}(Z_i) = 1\right]$$

Definition: $(X_i)_i$ heißt *pseudozufällig*, wenn für jeden statistischen Test $\mathcal{D}$ und alle $t > 0$

$$\Delta(\mathcal{D}, X_i) = \mathcal{O}(i^{-t})$$

($\Delta(\mathcal{D}, X_i)$ ist vernachlässigbar in i).
($(X_i)_i$ und $(Z_i)_i$ heißen *ununterscheidbar*.)

Pseudozufallsgenerator

Definition: Ein *Pseudozufallsgenerator* ist ein deterministischer Algorithmus, der auf eine i-Bit-Eingabe hin eine Funktion

$$F_i : \{0,1\}^i \longrightarrow \{0,1\}^{l(i)}$$

berechnet (für ein Polynom l mit $l(i) > i \ \forall\, i$), so dass

$$F_i(Y_i)$$

pseudozufällig ist (für auf $\{0,1\}^i$ gleichverteilte Y_i; modelliert den Seed).

Ensembles aus der Polynom-Iteration

Ensembles durch Polynom-Iteration erzeugen:

- Modul und Koeffizienten in Abhängigkeit von i und Seed wählen,
- Anzahl der erzeugten Iteration wieder als Funktion von i (oft $\mathcal{O}(\log i)$)

Vorhersage

Andere Intuition zum Zufall: nächstes Bit unvorhersagbar

Definition: Ein *Prädiktor* $\mathcal{P}$ an der Stelle j ist ein probabilistischer Polynomialzeit-Algorithmus mit Ausgabe in $\{0,1\}$, der als Eingabe die ersten $j-1$ Bits von X_i erhält.

Idee: $\mathcal{P}$ soll das j-te Bit vorhersagen.

Definition: (X_i) heißt *unvorhersagbar*, wenn für jeden Prädiktor $\mathcal{P}$

$$E(\mathcal{P},(X_i),j) := \mathbb{P}\left[\mathcal{P}(X_{i1},\ldots,X_{i,j-1}) = X_{ij}\right] - \frac{1}{2}$$

vernachlässigbar ist.

Satz von Yao

Satz: (X_i) pseudozufällig $\iff$ unvorhersagbar.

Beweis

Lemma 1: Sei $\mathcal{P}$ ein Prädiktor zu (X_i).
Dann gibt es einen statistischen Test $\mathcal{D}$ mit

- $|\mathcal{D}| = |\mathcal{P}| + \mathcal{O}(1)$, und
- $\Delta(\mathcal{D}, X_i) \geq E(\mathcal{P}, (X_i), j) + w$, und w ist vernachlässigbar.

Korollar: (X_i) pseudozufällig $\Longrightarrow$ unvorhersagbar.

Beweis Lemma 1

Lemma 1: Sei $\mathcal{P}$ ein Prädiktor zu (X_i).
Dann gibt es einen statistischen Test $\mathcal{D}$ mit

- $|\mathcal{D}| = |\mathcal{P}| + \mathcal{O}(1)$, und
- $\Delta(\mathcal{D}, X_i) > E(\mathcal{P}, (X_i), j) + w$, und w ist vernachlässigbar.

Statistischer Test: Gib 1 aus, falls $\mathcal{P}$ recht hat, und 0 sonst.

(Beweis)

Lemma 2: Sei $\mathcal{D}$ ein statistischer Test mit $\Delta(\mathcal{D}, X_i) \geq \delta(i)$. Dann gibt es $0 < j \leq l(i)$ und einen Prädiktor $\mathcal{P}$ mit

- $|\mathcal{P}| = |\mathcal{D}| + \mathcal{O}(l(i))$, und
- $E(\mathcal{P}, (X_i), j) < \delta(i)/l(i)$.

Korollar: (X_i) unvorhersagbar $\Longrightarrow$ pseudozufällig.

Beweis Lemma 2

Definiere gemischte („hybride“) Verteilungen:

$$
\begin{aligned}
H_i^{(0)}: &\quad X_{i1}, &&\dots && X_{i,l(i)-1}, && X_{i,l(i)} \\
H_i^{(1)}: &\quad X_{i1}, &&\dots && X_{i,l(i)-1}, && U_{l(i)} \\
&\dots \\
H_i^{(j)}: &\quad X_{i1}, \quad \dots \quad X_{i,j}, && U_{j+1} && \dots && U_{l(i)} \\
&\dots \\
H_i^{(n)}: &\quad U_1 &&\dots && && U_{l(i)}
\end{aligned}
$$

wo U_j gleichverteilt auf $\{0,1\}$.

(Beweis Lemma 2)

$$p_j := \mathbb{P}\left[\mathcal{D}(H_i^{(j)}) = 1\right]$$

Es gibt j mit

$$p_{j+1} - p_j \geq \frac{\delta}{l(i)}$$

Konstruiere Prädiktor: Wenn $\mathcal{D}(x_{i1}, \ldots, x_{ij}, u_{j+1}, \ldots, u_{l(i)} = 1$, gib u_{j+1} aus

$1 - u_{j+1}$ sonst.

Der Blum-Blum-Shub-Generator

Am Mittwoch wurde gezeigt:

Satz: Unter der QR-Annahme ist der Blum-Blum-Shub-Generator perfekt pseudozufällig.

Problematik

- Pseudozufall gibt es ’nur’ unter kryptographischen Annahmen
- Widerlegung der RSA-Annahme vernichtet auch den RSA-Generator
- allgemeinere hinreichende Bedingung für die Existenz gesucht

Basisannahmen

Grundlegende kryptographische Annahmen postulieren nicht Sicherheitseigenschaften *konkreter* Objekte, sondern die *Existenz* von „sicheren“ Objekten.
Zum Beispiel:

- kollisionsfreie Hash-Funktionen
- Einweg-Permutationen
- Einweg-Funktionen
- . . .

Wiederholung: Einweg-Funktion

Definition: Eine Funktion

$$f : \{0,1\}^* \longrightarrow \{0,1\}^*$$

heißt *Einweg-Funktion*, wenn

- für jeden probabilistischen Polynomialzeit-Algorithmus $\mathcal{A}$

$$\mathbb{P}\left[f\ (\mathcal{A}(f(U_n), 1^n)) = f(U_n)\right]$$

  vernachlässigbar in n ist.
- f in (deterministischer) Polynomial-Zeit berechenbar ist.

Eine *Einweg-Permutation* f ist eine bijektive Einweg-Funktion.
Wir verlangen zusätzlich $f(\{0,1\}^n) \subseteq \{0,1\}^n$ (längenerhaltend).

Rolle der Annahmen

- Existenz von Einweg-Funktionen ist die kryptographische Minimalforderung
- Einweg-Eigenschaft einer konkreten Funktion (z.B. $(p, q) \to pq$) würde sofort die Existenz garantieren
- in vielen Anwendungen wird implizit die Existenz von Pseudozufallsgeneratoren angenommen: *auch eine kryptographische Basisannahme!*

Bedingungen für Pseudozufall

- wir ermitteln logische Beziehungen zwischen diesen Basisannahmen
- **Blum, Micali '84:** $\exists$ Einweg-Permutationen $\Longrightarrow \exists$ Pseudozufallsgeneratoren
- **Håstad & al. '99:** $\exists$ Einweg-Funktionen $\Longrightarrow \exists$ Pseudozufallsgeneratoren
- offensichtlich ist letzteres eine Äquivalenz-Aussage

Ein Bit genügt

Erinnerung: *Pseudozufallsgenerator* von Typ $l(x)$ ist det. Algorithmus, der zu n-Bit-Eingabe

$$F_n : \{0,1\}^n \longrightarrow \{0,1\}^{l(n)}$$

berechnet. ($l(x) \geq x + 1$ Polynom) mit $F_i(U_n)$ pseudozufällig

Bezeichne mit F sowohl Generator als auch Funktion $\{0,1\}^* \longrightarrow \{0,1\}^*$.

Satz: Sei $l(x)$ ein Polynom.
Dann $\exists$ Pseudozufallsgenerator vom Typ $l(x) \iff \exists$ Pseudozufallsgenerator vom Typ $x + 1$

Beweis

Sei F Pseudozufallsgenerator vom Typ $x + 1$.
Definiere

$$G_{l(x)}(u) := (x_1, \ldots, x_{l(n)})$$

durch

$$u_0 := u, \qquad F(u_i) = (u_{i+1}, x_{i+1})$$

(Diagramm)
Betrachte die Verteilung von

$$(u_1, \ldots, u_j, x_{j+1}, \ldots, x_{l(n)})$$

für $j = 1, \ldots, l(n)$. Hier sind die u_i unabhängig gleichverteilt auf $\{0,1\}^n$.

Hardcore-Prädikat

Definition: Sei $F : \{0,1\}^* \longrightarrow \{0,1\}^*$ eine Funktion.
Ein *Hardcore-Prädikat* (auch: *hard core, hard-core bit, hidden bit*) zu F ist eine Funktion

$$B : \{0,1\}^* \longrightarrow \{0,1\}$$

so dass

- für jeden probabilistischen Polynomialzeit-Algorithmus $\mathcal{P}$

$$\mathbb{P}\left[\mathcal{P}(F(U_n), 1^n) = B(U_n)\right] - \frac{1}{2}$$

  vernachlässigbar (in n) ist; und
- B in deterministischer Polynomial-Zeit berechenbar ist.

Existenz von PRG

Satz: Sei F eine Einweg-Permutation und B ein Hardcore-Prädikat zu F.
Dann ist G mit

$$G(x) := (F(x),\ B(x))$$

ein Pseudozufallsgenerator vom Typ $x + 1$.

Beweis:

- Zeige: $G(x)$ ist unvorhersagbar.
- j-te Bit von $F(x)$ unvorhersagbar, da gleichverteilt
- $B(x)$ ist unvorhersagbar nach Voraussetzung
- F, B effizient berechenbar $\Longrightarrow$ G effizient berechenbar

Hardcore-Prädikat aus Einwegfunktion

Satz: Sei f eine Einwegfunktion.
Dann ist

$$(x, r) \longmapsto xr^t$$

ein Hardcore-Prädikat zu $(x, r) \longmapsto (f(x), r)$ (wobei $x, r \in \{0,1\}^n$).

Korollar: Wenn Einweg-Permutationen existieren, dann auch Pseudozufalls-Generatoren.

Beweis-Idee

Beweis:

- Angenommen, Algorithmus $\mathcal{P}$ sage xr^t aus $(f(x), r)$ voraus
- gib $\mathcal{P}$ die Eingaben $(f(x), r)$ sowie $(f(x), r \oplus e_i)$
- Ausgabe verschieden $\iff$ $x_i = 1$

$\mathcal{P}$ könne xr^t aus $(f(x), r)$ für unendlich viele n mit Wahrscheinlichkeit $\geq \frac{1}{2} + \frac{1}{p(n)}$ voraussagen (p Polynom).

Wir müssen Algorithmus $\mathcal{A}$ konstruieren, der in probabilistischer Polynomial-Zeit *für unendlich viele n* das Urbild x von $f(x)$ mit Wahrscheinlichkeit $\geq \frac{1}{2} + \frac{1}{p'(n)}$ bestimmen kann (p' Polynom).

Wir betrachten nur noch n wie oben.

Vorgehensweise

- wähle x
- wähle $s_1, \ldots, s_k \in_U \{0,1\}^n$
- für alle i rate $c_i \in_U \{0,1\}$ (als xs_i^t)
- für alle $J \subseteq \{1, \ldots, k\}$ bilde $r_J := \oplus_J s_j$ und $b_J := \oplus_J c_j$
- für alle J, alle i frage das Orakel $y_J := b_J \oplus \mathcal{P}(1^n, f(x), r_J \oplus e_i)$
- für jedes i entscheidet die Mehrheit der y_J über $\tilde{x}_i$

Terminiert in Polynomial-Zeit $\iff$ k ist logarithmisch in n
Wähle $k := \lceil \log(2np(n)^2 + 1) \rceil$
Berechne Erfolgs-Wahrscheinlichkeit $(x; c_i; \tilde{x}_i)$

Genügend x

Lemma: Sei $s(x_0) := \mathbb{P}[\mathcal{P}(1^n, f(x_0), r) = x_0 r^t]$.
Klar: $s(x)$ ist ZV (wegen x).
Dann gilt:

$$\mathbb{P}_x[s(x) \geq \frac{1}{2} + \frac{1}{2p(n)}] \geq \frac{1}{2p(n)}$$

Beweisidee: Markoffsche Ungleichung

$$\mathbb{P}[S \geq c] < \frac{\mathbb{E}S}{c}$$

Die Mehrheit hat recht

Lemma: Sei n wie vor und x so dass $s(x) \geq \frac{1}{2} + \frac{1}{2p(n)}$. Dann gilt

$$\mathbb{P}\left[\text{für die Mehrheit der } J \text{ gilt } y_{J,i} = x_i\right] > 1 - \frac{1}{2n}$$

Beweis: Anwenden der Tschebytscheff-Ungleichung

$$\mathbb{P}\left[\left|\sum_i (Z_i - \mathbb{E}Z_1)\right| > n\delta\right] < \frac{\mathrm{Var}Z_1}{n\delta^2}$$

auf $Z_i := \mathbb{I}(\tilde{x}_i = x_i)$.

Insgesamt: Erfolg für alle Bits mit Wahrscheinlichkeit $\geq \frac{1}{2}$
Zusammen

$$\frac{1}{2p(n)}\frac{1}{2^l}\frac{1}{2}$$

ist (reziprok) polynomiell

Einweg-Funktionen genügen

Satz (Håstad, Impagliazzo, Levin, Luby ’99): Wenn Einweg-*Funktionen* existieren, dann auch Pseudozufalls-Generatoren.

- wesentlich allgemeiner als Blum-Micali:
  auch wenn es Einweg-Funktionen gibt, müssen Einweg-Permutationen nicht existieren
- der Beweis verwendet „Glättungen“ der Verteilung von $f(U_n)$ mittels wiederholtes Ziehen sowie Hashen

Entropie

Definition: Sei X eine ZV. Die *Entropie* von X ist

$$\mathbb{H}(X) := -\mathbb{E}_Y \log \mathbb{P}(X = Y)$$

wo $X \stackrel{d}{=} Y$.

- anschaulich: $\mathbb{H}(X)$ ist die Anzahl der involvierten Zufallsbits
- $\mathbb{H}(U_n) = n$
- $\mathbb{H}(c) = 0$
- $0 \leq \mathbb{H}(X) \leq n$, Randfälle s.o.
- $\mathbb{H}(f(X)) \leq \mathbb{H}(X)$

Abschwächung „Pseudozufall“

Wir können annehmen: $F : \{0,1\}^n \longrightarrow \{0,1\}^{n+1}$

- Pseudozufalls-Generator: Output ununterscheidbar von der Gleichverteilung
- Pseudoentropie-Generator: Output ununterscheidbar von einer Verteilung mit Entropie $>$ verwendete Zufallsbits
- Entropie-Fälscher: Output ununterscheidbar von einer Verteilung mit Entropie $>$ wirkliche Entropie des Outputs

Abschwächung „Pseudozufall“(II)

Sei $(x, y) \mapsto h_y(x)$ universelle Hash-Funktion
(d.h. $\mathbb{P}[h_Y(x) = a, h_Y(x') = a'] = 2^{-m}$). Es wird gezeigt, dass:

- f Pseudoentropie-Generator $\Longrightarrow (u, y) \longmapsto (h_y(f^k(u)), y)$ Pseudozufalls-Generator
- f Entropie-Fälscher $\Longrightarrow (u, r) \longmapsto (f^k(u), h_r(u), r)$ Pseudoentropie-Generator
- f Einweg-Funktion $\Longrightarrow$
  $(x, y, i, r, u) \longmapsto (h(\mathrm{mult}(x, y)), f^k(x, i, r), u, y)$
  Entropie-Fälscher

Konstruktion zu ineffizient für Anwendungen