Zufallszahlen
in der Kryptografie

Martin

Chaos Computer Club Cologne e.V.
https://koeln.ccc.de

19. Oktober 2015

Warum brauchen wir Zufall?

- Die Verfahren, die wir kennen brauchen Zufallszahlen als Input
- Um zu zeigen, dass sie sicher sind brauchen wir “echten” Zufall
- Was ist (echter) Zufall?

Warum brauchen wir Zufall?

- Die Verfahren, die wir kennen brauchen Zufallszahlen als Input
- Um zu zeigen, dass sie sicher sind brauchen wir “echten” Zufall
- Was ist (echter) Zufall?

Aufgabe

Aufgabe

Denke an eine Zahl zwischen 1 und 10

War das zufällig?

Aufgabe

Ist 100110 zufällig?
Ist 111110 zufällig?

Aufgabe

Aufgabe

Denke an eine Zahl zwischen 1 und 10

War das zufällig?

Aufgabe

Ist 100110 zufällig?
Ist 111110 zufällig?

Beweisbarkeit

Aufgabe

Gibt es ein Programm das “echt”zufällige Zeichenfolgen ausgibt?
Wer kann das beweisen?

Zufall lässt sich nicht beweisen. Nicht-Zufall bedeutet aber, dass Muster existieren
Programme sind deterministisch.

Beweisbarkeit

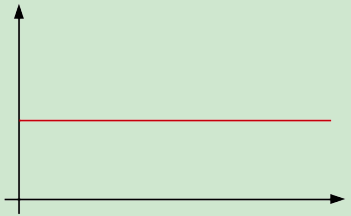
Aufgabe

Gibt es ein Programm das “echt”zufällige Zeichenfolgen ausgibt?
Wer kann das beweisen?

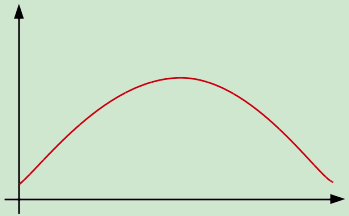
Zufall lässt sich nicht beweisen. Nicht-Zufall bedeutet aber, dass Muster existieren
Programme sind deterministisch.

Kryptografisch sicherer Zufall

Experiment: Gleichverteilung



Wie hoch ist die Wahrscheinlichkeit mit einem Würfel eine 4 zu würfeln?

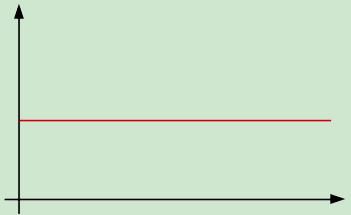


Wie hoch ist die Wahrscheinlichkeit mit acht Würfel eine 12 zu würfeln?

Zufall reicht nicht aus - wir wollen eine Gleichverteilung

Kryptografisch sicherer Zufall

Experiment: Gleichverteilung



Wie hoch ist die Wahrscheinlichkeit mit einem Würfel eine 4 zu würfeln?

Wie hoch ist die Wahrscheinlichkeit mit acht Würfel eine 12 zu würfeln?

Zufall reicht nicht aus - wir wollen eine Gleichverteilung

kryptografisch sicherer Zufall

Experiment: Verstecken

- Ich gebe dir ein Kartendeck
- Suche eine Karte aus.
- Du willst deine Karte geheim halten, wo ist das beste Versteck?

Im Kartenstapel
Die Karte ist die geheime Nachricht, der Kartenstapel die Menge aller Nachrichten...

Art of the Problem, https://www.youtube.com/watch?v=FfZurPKYM2w

kryptografisch sicherer Zufall

Experiment: Verstecken

- Ich gebe dir ein Kartendeck
- Suche eine Karte aus.
- Du willst deine Karte geheim halten, wo ist das beste Versteck?

Im Kartenstapel
Die Karte ist die geheime Nachricht, der Kartenstapel die Menge aller Nachrichten...

Art of the Problem, https://www.youtube.com/watch?v=FfZurPKYM2w

Entropie

- Claude Elwood Shannon 1950
- Wie viel Information steckt in einer Folge von Elementen?
- Wie viele Fragen muss ich stellen um das nächste Element bestimmen zu können?
- Kann ich von den vorherigen Elementen auf das kommende schließen?
- Je größer die Entropie desto schwieriger die Vorhersage....

vgl. Art of the Problem, https://www.youtube.com/watch?v=R4OIXb9aTvQ

Beispiel Entropie

Ich soll dir 8 Bits geben. Die letzten beiden Bits verwende ich für Fehlerkorrektur. Der Informationsgehalt sind 6 Bits.

RNG - Herkunft von “echtem” Zufall

- Zeitintervall bei radioaktivem Zerfall
- elektromagnetisches Rauschen
- astabile Kippstufe (Temperatur, Stromstärke, ...)

teuer, langsam, Gaiga-Müller Zähler zuhause vergessen...

Nächst beste Alternative

Ereignisse die Zufällig erscheinen

- Intervall zwischen Tastenanschlägen, Maus
- Intervall zwischen Interrupts
- Intervall von Netzwerkereignissen
- Intervall von Ereignissen anderer Devices (Speicher, ...)

Problem: Davon gibt es nicht so viel...

PRNG als Alternative

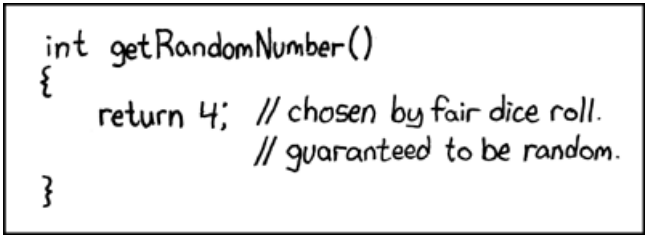


Abbildung: xkcd, https://xkcd.com/221

Jetzt kann ich nicht mehr beweisen, dass mein Verfahren sicher ist.
Aber ich kann den Zufallsraum so wählen, dass der Angreifer ewig braucht. =>
“sicher”

CSPRNG

Wir brauchen CSPRNG (Cryptografically Secure Pseudo Random Number Generator)

- langer Zyklus
- interner Status geheim
- keine Relation zwischen Input und Output (wie schafft man das mit PRNG?)
- Gleichverteilung
- hohe Entropie (20 Zufallsbits sollen 20 Bit Zufall enthalten, nicht nur 2)
- Ausgabe nicht vorhersagbar
- Ununterscheidbarkeit von “echtem” Zufall

title

Inhalt...

Seed

Lösung: (CS)PRNG mit “echtem” Zufall anreichern

1. Wähle einen (deterministischen) Algorithmus
2. füttere etwas “Seed” hinein
3. wiederhole 2

Die Ausgabe hängt vom Seed ab => Erheblich eingeschränkter Raum im Vergleich zu “echtem” Zufall
Aber: Zyklus ist ewig lang (wirklich ewig)

Aufgabe/Beispiel

- Blockchiffre im Counter Modus
- middle squares

Zufalltests

- Random Walk
- Kompressions-Versuch (Erinnerung: Entropie...)
- Versuche ein Polynom über die Zufallsereignisse zu bilden...

/dev/(u)random

- erfüllen (beide!) die Anforderungen für CSPRNG
  - Ausnahme: unmittelbar nach Systemstart ist der Entropie-Pool leer...

Zusammenfassung

- Wir brauchen keinen TRNG - ein guter PRNG kann den Rechenaufwand auch ins unermessliche treiben...
- Anforderungen an CSPSRG: Ununterscheidbarkeit zu RNG - hohe Entropie/Gleichverteilung
- Auch für PRNG brauchen einen kleinen (!) Seed guten Zufall
- Auch wenn es lange dauert: PRNG ist zyklisch in Abhängigkeit vom Seed.

Aufgabe

Aufgabe: Random Walk

Implementiere einen Random Walk mit einem guten CSPRNG (/dev/urandom) und einem schlechten PRNG.

Random Walk: Eine Schildkröte zieht Zahlen, jede Zahl steht für einen Schritt in eine Himmelsrichtung.

```
import turtle
import random
```

Lösung in Python vorhanden, random_walk.py