Sicherheit von Zufallszahlengeneratoren

Worrmann, Marcus
HTW Berlin
10313 Berlin
s0583039@htw-berlin.de

***Zusammenfassung*—Im folgenden Artikel werden Sicherheitsaspekte bei der Generierung und im Umgang mit Zufallszahlengeneratoren analysiert. Thematisch liegt der Fokus auf praktischen Anwendungsbeispielen, der Absicherung gegen Fremdzugriffe und der Gewährleistung valider Zufallszahlen. Zudem wird die Bedeutung von Zufallszahlengeneratoren für kryptografische Systeme hervorgehoben, da sie die Grundlage für sichere Schlüssel darstellen. Abschließend werden Angriffsszenarien und Sicherheitsmechanismen vorgestellt, welche Verwendung finden um Schwachstellen zu erkennen und zu beheben.**

I. Einleitung

Zufallszahlengeneratoren spielen eine zentrale Rolle in der modernen Kryptografie und Datensicherheit. Von der sicheren Authentifizierung beim Online-Banking bis hin zu kryptografischen Schlüsseln für Blockchain-Technologien—die Qualität der Zufallszahlen ist entscheidend für die Sicherheit von Systemen. Sichere Schlüssel, die auf Zufallszahlen basieren, sind unverzichtbar, um unvorhersehbare und manipulationssichere Prozesse zu gewährleisten. Der vorliegende Beitrag beleuchtet die verschiedenen Typen von Zufallszahlengeneratoren sowie deren spezifische Eigenschaften. Dabei wird besonderes Augenmerk auf echte Zufallszahlengeneratoren (TRNGs) gelegt, die aufgrund ihrer physikalischen Basis eine hohe Entropie und Sicherheitsresistenz aufweisen.

II. Zufallszahlgeneratoren

A. Definition von Zufallszahlgeneratoren

Haahr [1] definierte 1999 eine Zufallszahl als eine Zahl, die aus einer Menge von Zahlen ausgewählt wurde und bei der jede einzelne Auswahl an Zahlen aus dieser Menge gleich wahrscheinlich ist. Zusätzlich muss jede Auswahl einer Zahl statistisch unabhängig von der Wahl der restlichen Zahlen einer Kombination erfolgen. Laut Rukhin [2] sind folgende Anforderungen an Zufallszahlen definiert:

- **Zufälligkeit:**
  Das Erzeugen einer Folge von Zufallszahlen, welche durch eine Folge von zufälligen Bits generiert wird, muss die Wahrscheinlichkeit, das jedes erzeugte Bit eine *1* oder eine *0* erzeugt wird, identisch sein. Zudem darf das Erzeugen eines Bits keinen Einfluss auf den weiteren Verlauf der Zahlenkombination auswirken.
- **Unvorhersehbarkeit:**
  Ein Vorhersagen zukünftig auftretender Zahlen soll unter Kenntnisstand einer oder mehrer Zufallszahlen nicht möglich sein.

Unter Berücksichtigung dieser beiden Aspekte wird ein Vorhersagen und Generieren von sich wiederholenden Mustern vermieden.

B. PRNG - Pseudo Random Number Generator

Pseudozufallszahlengeneratoren ermöglichen es, unter Angabe eines vorgegebenen Startwertes (Seed) unter Verwendung eines deterministischen Algorithmus eine Folge von Pseudozufallszahlen zu generieren. Aufgrund der deterministischen Verarbeitung ist bei identischem Input(Seed) ein immer gleichender Output zu erwarten.

C. TRNG - True Random Number Generator

Zum Erzeugen echter Zufallszahlen werden nicht deterministische Quellen genutzt. Neben physikalischen Quellen wie dem Frequenzrauschen der Atmosphäre, dem thermischen Rauschen in Widerständen oder radioaktivem Zerfall können ebenso vom Benutzer erzeugte Eingaben (Mausbewegung, akustische Signale) oder von der PC-Hardware generierte Werte (CPU Rechenzeiten, Festplatten-Zugriffszeiten) als Zufallsfaktoren genutzt werden.

D. URNG - Unpredictable Random Number Generator

Die Klasse der nicht-vorhersagbaren Zufallszahlgeneratoren beinhaltet all diejenigen Zufallszahlgeneratoren, welche auf Benutzereingaben oder die Unvorhersehbarkeit von Zuständen basieren. Ebenfalls fallen all die erzeugten Zufallszahlen unter diese Kategorie, welche eine Kombination eines *PRNG* und eines *TRNG* darstellen. Diese Zufallszahlen werden auch als hybride Zufallszahlen bezeichnet. Da bei Pseudozufallszahlengeneratoren die Unvorhersehbarkeit bezogen auf die Generierung aufgehoben ist, wird im weiteren Verlauf nicht weiter auf diese Form der Generierung eingegangen.

III. Implementierungsmöglichkeiten

Im späteren Verlauf wird auf spezifische Angriffs- und Sicherheitsmechanismen am Beispiel ausgewählter *TRNGs* eingeganen. Um das Verständnis für die Empfindlichkeit dieser Systeme ist es wichtig, einen Einblick in die grundlegende Funktionsweise dieser Generatoren zu erhalten.
Nachfolgend anhand von drei diskreten Beispielen.

A. Ringoszillatorbasierte Zufallszahlgeneratoren

Viele der heute bekannten Zufallszahlgeneratoren beruhen auf Ringoszillatoren. Diese setzen sich als Baugruppe einer ungeraden Anzahl hintereinander geschaltenen Inverter in Form eines Rings zusammen. Dadurch bildet das letzte

Glied der Schaltung den erneuten Eingang zum ersten Inverter. Aufgrund der ungeraden Anzahl von Elementen oszilliert das Signal am Ausgang zwischen einer logischen *1* und *0* im Rechteckschwingungsverlauf.
Die Idealfrequenz und -schwingung hängt hierbei von der Anzahl und Bauart der Inverter ab, wodurch es je nach Baugruppe zu kleinen zufälligen Schwankungen (Jitter) kommt.
Verbindet man nun mehrere Ringoszillatoren mit einer *XOR-Funktion*, so entstehen auf Grundlage der Schwankungen eines jeden Signals kurzzeitig nicht deterministische Pegel.
Um die Vorhersagbarkeit des deterministischen Anteils zu minimieren, werden die gewonnen Bits (Sample) anschließend noch mittels einer Abfederungsfunktion gebündelt und aus diesem Zusammenschluss eine kürzere neue Bitfolge generiert.
Sunar [3] verweist in seiner Arbeit auf die genaue mathematische Funktion des Algorithmus.
Nachfolgende Figure 1 stellt schematisch einen Output dieses Systems dar.

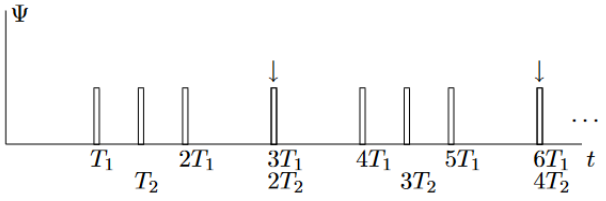

Abbildung 1. Output-Sample eines *XOR* mit zwei oszillierenden Outputs der Periode T_1 und T_2. [3]

B. RNGs auf Grundlage thermischen Rauschens

Als weitere Entropie-Quellen zum Generieren echter Zufallszahlen ist das *thermische Rauschen* oder auch *Wärmerauschen*, welches auf dem physikalischen Prinzip der brownschen Bewegung von Elektronen innerhalb eines elektrischen Widerstands beruht.
Diese marginalen, zufällig auftretenden Schwankungen zwischen den Enden eines Leiters wird zum Erzeugen von Zufallszahlen genutzt. Da auch diese Form der Zufallszahlgenerierung durch äußere Einflüsse manipulierbar ist, wird beispielsweise bei einer Schaltung, welche von Kaenel & Takayanagi [4] entworfen wurde zusätzlich mit einem chaotischen Oszillator verknüpft.
Dieser verstärkt die zum Teil geringen Abweichungen der Signalpegel des *thermischen Rauschens*, sodass die Beeinflussbarkeit stark abnimmt.

C. Hardware von Mobilgeräten

Suciu beschreibt in seinem Buch [5], das Datenquellen einzelner Hardwarekomponenten von Smartphones ebenfalls zum Generieren genutzt werden können. Mögliche, durch Zufälligkeit betroffene Komponenten wären zum Beispiel:

- der Lagesensor
- der GPS-Empfänger
- der Magnetfeldkondensator

Hierbei werden die gesammelten Daten in zwei Schritten verarbeitet und zu einer Zufallszahl generiert.
Zunächst wird ein Aufzeichnungsmodul mit Daten einer zufälligen Quelle beschrieben und diese Daten anschließend durch ein Kombinationsmodul verarbeitet.
Im Kombinationsmodul wird ein spezieller Kombinationsalgorithmus verwendet. Dieser bestimmt zunächst die verwendete Quelle der Datenaufzeichnung, anschließend werden die Daten untereinander kombiniert und abschließend je nach benötigter Länge der Zufallszahl eine *n* Menge an Bits aus der erzeugten Bitkette selektiert. Häufig handelt es sich dabei um die niederwertigsten Bits, da sich diese bereits bei kleinsten Abweichungen rekombinieren.

IV. Angriffsmöglichkeiten

Die Generierung einer Zahlenfolge oder Schlüssels beruht auf der nicht deterministischen Erzeugung durch den Algorithmus. Angriffe richten sich aus diesem Grund bei *TRNG* nicht auf die Ermittlung des eigentlichen Schlüssels sondern vielmehr auf die Erzeugung selbst. Hierbei wird unter passiven und aktiven Angriffen sowie folgenden Angriffsmöglichkeiten unterschieden:

- Fehlereinschleusung (fault-introduction , activ attack)
- Bit-Analyse und -Vorhersage (bit-detection, passiv attack)
- Seitenkanalattacken (side-channel attacks, passiv attack)

A. Fehlereinschleusung

Ziel der Fehlereinschleusung ist eine Verzerrung der statistischen Eigenschaften des Schlüssels/Outputs, ein sogenanntes Bias. Dadurch soll eine vereinfachte Vorhersagbarkeit des Ergebnisses erreicht werden. Sunar [3] nimmt eine Einteilung in invasive und noninvasive Attacken vor.

- **noninvasive Attacken:**
  Wird auf ein System extern Einfluss ausgeübt, spricht man von noninvasiven Attacken. Arten der Einflussnahme können das Bestrahlen mit elektromagnetischen Wellen oder eine Wärmezufuhr sein. Aus der Manipulation des Systems geht hervor, dass ein physischer Zugang für den Angriff vorausgesetzt sein muss.
- **invasive Attacken:**
  Invasive Attacken setzen keinen realen Zugang zum System voraus. Attacken richten sich viel mehr auf das Verändern des Systems selbst. So können systemisch geschaltete elektronische Verbindungen permanent geöffnet oder kurzgeschlossen werden.

Eine qualitative Einschätzung bezüglich der Sicherheit eines Schlüssels bedingt immer ein Berücksichtigen der Stabilität gegenüber beiden Attacken.
Sowohl der Input, welcher dem Angriff zugrunde liegt, als auch der infolgedessen generierte Output, können hierbei von tragender Rolle sein.
So beschreiben *Markettos & Moore* in Ihrem Buch [6] wie die Sicherheit eines ringoszillatorbasierten Zufallszahlgenerators durch das Einschleusen bestimmter Frequenzen in das Stromnetz zu einer essenziellen Verschlechterung der Qualität der erzeugten Zufallszahlen führte. Hierbei wurde die physikalische Eigenschaft der *CMOS* ausgenutzt und eine Glättung der Eingangsspannungen simuliert.

Diese Form des noninvasiven Eingreifens bildet ebenso wie die Qualität des Outputs selbst einen hohen Anspruch auf die perspektivische und realistische Sicherheit von Zufallszahlen, benötigt jedoch einen physischen Zugang zum System selbst.

B. Bit-Analyse und -Vorhersage

Eine weitere Variante abseits der im Vorfeld beschriebenen Fehlereinschleusung ist das Exponieren und Prognostizieren des Output selbst.
Eine Möglichkeit hierbei beschreibt *Dichtl* in seiner Arbeit [7]. Darin betont er, dass auf Grundlage des Wissens über die Anzahl der genutzten Oszillatoren sowie deren Ideal- und Samplingfrequenzen eine statistische Kalkulierbarkeit des Ergebnisses möglich ist.
So stellt die Verletzbarkeit von sicheren Zufallszahlen durch ein *Over-Sampling*, also dem häufigen Auslesen eines Systems zum Ermitteln bestimmter Auffälligkeiten, Wiederholungen oder Muster eine ebenso valide Sicherungsmaßnahme wie der Sicherung durch Fremdzugriff bei der Erzeugung des Schlüssels selbst dar.

C. Seitenkanalattacken

Im Fokus von Seitenkanalattacken liegt das Auslesen der Funktions- und Arbeitsweise des zugrunde liegenden Systems beziehungsweise seiner Hardware. Laut Bayrak [8] werden hierbei Daten des Stromverbrauchs, elektromagnetische Abstrahlung oder akustische Geräusche kryptografischer Geräte genutzt um geheime Informationen zu erhalten.
Für eine erfolgreiche Entschlüsselung der Zufallszahl sind verschiedene Faktoren nötig sowie ein direkter Zugang zur Hardware. Zunächst wird z.B. der Stromverbrauch während des Verschlüsselns eines Klartexts gemessen. Anschließend wird mithilfe von statistischen Methoden der Stromverbrauch untersucht. Grundlagen wie der Algorithmus welcher auf dem zu encodierenden Gerät operiert sind zu heutiger Zeit oftmals einlesbar und zugänglich.
Ziel der Entschlüsselung ist es, eine Deterministik zu erreichen, wodurch zukünftige Zufallszahlen vorhersagbar sind.

V. Prüfkonzepte

Ob der generierte Schlüssel oder die erzeugte Zufallszahl eine tatsächliche Zufälligkeit aufweist, kann mithilfe diverser statistischer Tests geprüft werden. Dabei wird ein Ergebnis dahingehend überprüft, ob die Verteilung der Zahlenfolge eine echt zufällige statistische Verteilung aufweist.
Statistische Tests basieren auf der Untersuchung, ob eine Nullhypothese (H_0) oder eine Alternativhypothese (H_a) vorliegt. Liegt eine Nullhypothese vor, so besteht statistische Zufälligkeit. Liegt eine Alternativhypothese vor, so gibt es keine statistische Zufälligkeit.
Zusätzlich zur Formulierung von Hypothesen ist es notwendig, ein Signifikanzniveau festzulegen. Dieses dient der Validierung der Nullhypothese. So wird vermieden, dass ungünstig gewählte Testszenarien und -eigenschaften, z.B. solche mit zufällig häufenden Mustern keinen Fehlalarm (α-*Fehler*) bewirken.

A. NIST Test Suite

Das *National Institute of Standards and Technology* hat eine *NIST Test Suite* veröffentlicht. Diese beinhaltet 15 diverse Tests, welche auf das Erkennen verschiedener Arten einer fehlenden Zufälligkeit abzielt. [2]
So gibt es Testszenarien für:

- **Monobit:**
  Überprüfung, ob die Verteilung an *1*'en innerhalb eines Bitstroms nahezu $\frac{1}{2}$ beträgt. Abhängig vom Signifikanzniveau kann eine größere oder kleine Abweichung ebenfalls ein valides Ergebnis erzielen.
- **Runs-Test:**
  Nachweis, ob es innerhalb einer Zufallszahl lange ununterbrochene Folgen (*Runs*) von *0*'en und *1*'en gibt.
- **DTF-Test:**
  Bei der Analyse mittels einer diskreten Fourier-Transformation wird auf Peaks innerhalb einer Sequenz bzw. Spektrums geprüft. Diese konkludieren periodisch auftretende Muster innerhalb der Zahlenfolge, was eine Zufälligkeit ausschließen kann. Ebenso wie bei der *Monobit-Analyse* ist hier eine korrekte Definition des Signifikanzniveaus von ausschlaggebender Rolle.

B. DIEHARD

Marsaglia definierte bereits 1995 diverse Testfälle, um die Qualität einer Zufallszahl zu überprüfen. [9]
Diese wurden zum Teil von der *NIST* weiter entwickelt, stellen zum Teil jedoch eine Ergänzung dar. Einer dieser fast 20 Jahre alten Testfälle, ist das Zählen von *1*'en.
In diesem Prüffall werden *1*'en entweder zufällige oder in aufsteigender Reihenfolge in einem Cluster gezählt und zu einem Wort übersetzt. Anschließend werden alle gebildeten Worte auf Ähnlichkeit geprüft. Die Qualität der Zufallszahl definiert sich hierbei über die Abwesenheit von Worthäufungen.

VI. Sicherheitsmechanismen

Da trotz der im Vorfeld genannten Möglichkeiten auf Prüfung von Zufälligkeit und Nachweisbarkeit von statistischer Qualität die Ergebnisse von Zufallszahlgeneratoren im Standardbetrieb kein Garant für absolute Sicherheit oder die Möglichkeit von Fehlereinschleusungen bzw. Seitenkanalattacken sind, bedarf es auf Entwicklungsseite einiger Vorkehrungen, um die *RNGs* zur Laufzeit zu schützen. Diverse Möglichkeiten der Absicherung bestehen sowohl auf Konstruktionsebene als auch innerhalb des Algorithmus der Zufallszahlgenerierung.

A. Konstruktionsebene

1) Chaos-Oszillatoren: Eine Möglichkeit, die Anfälligkeit gegenüber manipulativer Eingriffe aufgrund einer prozessbedingten Schaltung vorzubeugen ist, einen Chaos-Oszillator innerhalb des Generators zu implementieren.
Der im Kapitel III-B vorgestellte *Chaotic RNG* sorgt bei Systemen, welche mit geringen Spannungen von zum Teil 1mV arbeiten, diese Niederspannungsbereiche aufgrund von Verstärkern weniger kalkulierbar zu gestalten.

2) Abschirmung: Viele der non-invasiven Angriffe richten sich auf das Manipulieren des Stromflusses oder Einspeisens elektromagnetischer Impulse , um kalkulierbare oder wiederholende Zustände auf dem Zielgerät zu erzielen. Durch eine ausreichende Abschirmung gegenüber externen Zugriffen sowie eine Glättung von Spannungsversorungen innerhalb des Systems beugen eine Verzerrung durch externe Quellen vor.

B. Algorithmus

1) Random Shuffling: Während der Laufzeit des Algorithmus greift unabhängig der Eingangsparametrisierung eine *ADD-Instruktion* auf den Vorgang ein und verknüpft zwei eingeladene Variablen. Der dadurch erzeugte Wert wird wiederum als prozessinterne Variable widergespiegelt.
Da dieser Vorgang rein zufällig und gelöst der hardwareseitigen Datenbeschaffung geschieht, wird eine Verzerrung der gesammelten Daten vorgenommen.

VII. Fazit

Seit des Einzugs der Kryptografie in nahezu alle Bereiche des Internets und mit steigender Bedeutung von Datensicherheit im Alltag gewinnt die Qualität und Reliabilität von sicheren Zufallszahlen immer mehr an Bedeutung.
TRNGs bilden im Bereich der Zuverlässlichkeit und Sicherheit die tragende Rolle. So wird durch die Nutzung physikalischer Phänomene eine schwer zu manipulierende Entropie-Quelle implementiert.
Auch unter dem Aspekt Algorithmen und Zufallszahlen durch die Nutzung bewährter Werkzeuge wie dem *NIST Test Suite* oder denen von *DIEHARD* zu unterziehen, trägt zu einer erhöhten Angriffssicherheit bei.
So sind Zufallszahlengeneratoren zwar immer wieder Ziel von Angriffen, bieten aufgrund des großen Interesses, der breiten Nutzungsmöglichkeit und der Fortschritte in der Forschung eine bis dato verlässliche Quelle zum Erzeugen sicherer Zufallszahlen und Schlüssel.

Literatur

[1] M. Haahr, "Introduction to randomness and random numbers," *Random. org, June*, 1999.

[2] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert *et al.*, *A statistical test suite for random and pseudorandom number generators for cryptographic applications*. US Department of Commerce, Technology Administration, National Institute of . . . , 2001, vol. 22.

[3] B. Sunar, W. J. Martin, and D. R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *IEEE Transactions on computers*, vol. 56, no. 1, pp. 109–119, 2006.

[4] Y.-H. Wang, H.-G. Zhang, Z.-D. Shen, and K.-S. Li, "Thermal noise random number generator based on sha-2 (512)," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 7. IEEE, 2005, pp. 3970–3974.

[5] K. Marton, A. Suciu, and D. Petricean, "A parallel unpredictable random number generator," in *2011 RoEduNet International Conference 10th Edition: Networking in Education and Research*. IEEE, 2011, pp. 1–5.

[6] A. T. Markettos and S. W. Moore, "The frequency injection attack on ring-oscillator-based true random number generators," pp. 317–331, 2009.

[7] M. Dichtl, "How to predict the output of a hardware random number generator," pp. 181–188, 2003.

[8] A. G. Bayrak, N. Velickovic, P. Ienne, and W. Burleson, "An architecture-independent instruction shuffler to protect against side-channel attacks," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 8, no. 4, pp. 1–19, 2012.

[9] G. Marsaglia, "The marsaglia random number cdrom including the diehard battery of tests of randomness," *http://www. stat. fsu. edu/pub/diehard/*, 2008.