Überblick über Zufallszahlengeneratoren

Seminararbeit für die Lehrveranstaltung
KV Web Security (353.031) im Sommersemester 2013

Severin Schürz (0955499)

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einführung

Die Generierung von Zufallszahlen stellt eine wichtige Grundlage für die moderne Kryptographie dar. Kryptographie wiederum nimmt einen hohen Stellenwert in den heutigen Kommunikationstechnologien ein, um einen sicheren Austausch von Daten zu ermöglichen. Die dabei verwendeten Zufallszahlen müssen hohen Anforderungen genügen, um keine Angriffsmöglichkeiten auf die gesicherten Systeme zu ermöglichen. Die Generierung qualitativ hochwertiger Zufallszahlen stellt daher ein wichtiges Thema in der Wissenschaft und Praxis dar (Chen et al. 2011, S. 1678).

Konkrete Anwendungsgebiete von Zufallszahlengeneratoren in der Kryptographie sind die Erzeugung geheimer Schlüssel, aber sie liefern auch zufälligen Input für die Erstellung von digitalen Signaturen oder sogenannten „Challenges“ in Authentifizierungsprotokollen (Rukhin et al. 2010 2010, S. 1:1), wo sie das Wiederabspielen aufgezeichneter Sitzungen verhindern. Neben kryptographischer Anwendungen spielen Zufallszahlen auch in Simulationsexperimenten oder zur Auswahl von Stichproben in der Statistik eine wichtige Rolle (Abu & Muslim 2008, S. 381). In solchen Anwendungsbereichen bestehen allerdings wesentlich andere Anforderungen an die verwendeten Zufallszahlen als in der Kryptographie. Beispielsweise sollen bei Simulationen bei mehreren Durchläufen eines Experiments immer wieder die gleichen Zufallszahlen generiert werden, während die Zufallszahlen für kryptographische Anwendungen nicht vorhersagbar sein sollen (Abu & Muslim 2008, S. 381). Außerdem werden bei Simulationen oft spezielle, von der Gleichverteilung abweichende, statistische Verteilungen (z.B. Normalverteilung) der erzeugten Zufallszahlen gefordert. Aufgrund der unterschiedlichen Anforderungen wird in dieser Arbeit nur auf die Generierung von Zufallszahlen für kryptographische Anwendungen eingegangen.

Diese Arbeit ist wie folgt strukturiert: In Abschnitt 2 wird auf Grundlagen, wie Begriffsdefinitionen und die Einteilung von Zufallszahlengeneratoren eingegangen. Abschnitt 3 präsentiert konkrete Möglichkeiten zur Implementierung echter Zufallszahlengeneratoren. In Abschnitt 4 wird untersucht, welche Arten von Angriffen auf Zufallszahlengeneratoren möglich sind und wie Zufallszahlengeneratoren vor diesen geschützt werden können. Danach werden in Abschnitt 5 Tests vorgestellt, mit denen die Qualität erzeugter Zufallszahlen überprüft werden kann. Abschnitt 6 schließlich fasst die wesentlichsten Erkenntnisse zusammen und zieht ein Resümee.

2. Grundlagen

Eine Zufallszahl ist laut Haahr (1999) eine Zahl, die aus einer Menge von Zahlen ausgewählt wurde, wobei die Auswahl jeder in dieser Menge vorhandenen Zahl gleich wahrscheinlich ist. Die Anforderung der genau gleichen Wahrscheinlichkeit besteht zumindest dann, wenn, wie in der Kryptographie üblich, gleichverteilte Zufallszahlen produziert werden sollen. Bei einer Folge von Zufallszahlen (wie auch von Zufallszahlengeneratoren erzeugt) muss die Auswahl jeder einzelnen Zahl statistisch unabhängig von der Auswahl der restlichen Zahlen in der Folge sein (Haahr 1999).

Anforderungen an Zufallszahlen für den Einsatz in der Kryptographie sind laut (Rukhin et al. 2010):

- **Zufälligkeit:**
  Bei der Erzeugung einer Folge von Zufallszahlen, dargestellt durch eine Folge von zufällig erzeugten Bits, muss die Wahrscheinlichkeit, dass ein erzeugtes Bit zu einer Eins (oder eben einer Null) wird, genau bei ½ liegen. Außerdem müssen die einzelnen Erzeugungen der Bits

voneinander unabhängig sein, d.h. eine Erzeugung eines Bits darf keinen Einfluss auf die weiteren zufällig erzeugten Bits haben (Rukhin et al. 2010, S. 1:1). Das heißt vor allem, dass keine sich wiederholenden Muster auftreten dürfen.

- **Unvorhersagbarkeit:**
  Bei Kenntnis einer oder mehrerer erzeugter Zufallszahlen können die zukünftig erzeugten Zufallszahlen nicht vorhergesagt werden (Rukhin et al. 2010, S. 1:1).

Zur Erzeugung von Zufallszahlen werden Zufallszahlengeneratoren eingesetzt, die von den meisten Autoren (z.B. Rukhin et al. 2010, Abu 2008, Chan 2011) wie folgt eingeteilt werden:

- **Pseudozufallszahlengeneratoren** (PNRG, engl. für *Pseudo Random Number Generator*):
  Pseudozufallszahlengeneratoren verwenden einen deterministischen Algorithmus, um aus einem vorgegebenen Startwert (*Seed*) eine Folge von Pseudozufallszahlen zu berechnen (Rukhin et al. 2010, S. 1:2). Durch die deterministische Arbeitsweise liefern solche Generatoren bei gleichem Input (Seed) immer den gleichen Output. Unvorhersagbarkeit ist also nicht gegeben. Pseudozufallszahlengeneratoren eignen sich besonders gut für Simulationsexperimente, da für die Wiederholung des Experiments nur der Seed gespeichert werden muss.
- **Echte Zufallszahlengeneratoren** (TRNG, engl. für *True Random Number Generator*):
  Echte Zufallszahlengeneratoren, auch nichtdeterministische Zufallszahlengeneratoren genannt, verwenden nichtdeterministische Quellen, um echte Zufallszahlen zu erzeugen (Rukhin et al. 2010, S. 1:2). Solche Quellen sind zumeist physikalische Vorgänge, wie das Frequenzrauschen in der Atmosphäre, thermisches Rauschen in Widerständen oder radioaktiver Zerfall (Chan et al. 2011, S. 161; Kwok & Lam 2006, S. 1) Es kann aber auch vom Benutzer (z.B. Mausbewegungen) oder von gewöhnlicher PC-Hardware (z.B. Festplatten-Zugriffszeiten) stammende „Zufälligkeit“ eingesetzt werden, dabei ist die erreichbare Geschwindigkeit der Zufallszahlenerzeugung meist aber eher begrenzt (Chan et al. 2011, S. 161). Einige Beispiele für die Implementierung von echten Zufallszahlengeneratoren werden in Abschnitt 3 vorgestellt.

Entsprechend dieser Einteilung von Zufallszahlengeneratoren wird auch bei Zufallszahlen als deren Output zwischen Pseudozufallszahlengeneratoren und echten Zufallszahlen unterschieden.

Einige Quellen (z.B. Suciu et al. (2011) oder Colesa et al. (2008)) sehen Zufallszahlengeneratoren, die zwar nicht auf echtem Zufall beruhen, deren Ausgabe jedoch aufgrund der Komplexität des dahinter liegenden Generierungsprozesses nicht vorhersagbar ist, als eine eigene Klasse von Zufallszahlengeneratoren. Diese werden dann als nicht-vorhersagbare Zufallszahlengeneratoren (URNG, engl. für *Unpredictable Random Number Generator*) bezeichnet. Darunter fallen Zufallszahlengeneratoren, die auf Benutzereingaben (z.B. Mausbewegungen) aufbauen, oder solche, die die Unvorhersagbarkeit von Zuständen einzelner Hardwarebauteile zu einem bestimmten Zeitpunkt ausnützen. Entsprechend werden von diesen Quellen dann nur solche Zufallszahlengeneratoren, die auf physikalischen Vorgängen aufbauen, als echte Zufallszahlengeneratoren klassifiziert.

Außerdem wird die Kombination von einem echten Zufallszahlengenerator mit einem Pseudozufallszahlengeneratoren, der den zufälligen Output des ersteren weiter verarbeitet, teilweise als hybrider Zufallszahlengenerator bezeichnet (Thamrin et al. 2009). Die Grenze zu einem echtem Zufallszahlengenerator mit einer Aufbereitungsfunktion (siehe Abschnitt 3) ist dabei jedoch fließend.

In dieser Arbeit wird nur auf echte Zufallszahlengeneratoren näher eingegangen, da Zufallszahlen für kryptographische Anwendungen Unvorhersagbarkeit aufweisen müssen. In der Kryptographie können Pseudozufallszahlengeneratoren jedoch zur „Nachbearbeitung" der Ergebnisse von echten Zufallszahlengeneratoren in Form von hybriden Zufallszahlengeneratoren Anwendung finden.

3. Implementierungsmöglichkeiten

In diesem Abschnitt werden verschiedene Implementierungsmöglichkeiten echter Zufallszahlengeneratoren behandelt. Laut Rukhin et al. (2010, S. 1:2) benutzt jeder echte Zufallszahlengenerator eine „Entropie-Quelle", also eine nicht-deterministisch arbeitendes System, sowie meist eine Aufbereitungsfunktion für den Output der Entropie-Quelle. Es können dabei auch mehrere Entropie-Quellen kombiniert werden. Laut Sunar et al. (2007, S. 109) ist die Güte echter Zufallszahlengeneratoren abhängig von der Qualität dreier Komponenten:

- **Entropie-Quelle:**
  Die Entropie-Quelle ist die kritischste Komponente echter Zufallszahlengeneratoren, da sie die grundsätzlich zur Verfügung stehende „Zufälligkeit" (Entropie) bestimmt. Falls die Entropiequelle einen Bias (statistische Verzerrung) aufweist, muss dieser durch den Extraktionsmechanismus oder durch eine Aufbereitungsfunktion kompensiert werden.
- **Extraktionsmechanismus:**
  Der Auslesemechanismus zum Messen der zufällig schwankenden Größe aus der Entropie-Quelle soll möglichst viel Entropie „sammeln", die Quelle jedoch dabei möglichst nicht beeinflussen.
- **Aufbereitungsfunktion:**
  Eine Aufbereitungsfunktion soll Unzulänglichkeiten der Entropie-Quelle oder des Extraktionsmechanismus kompensieren, oder aber auch Toleranz gegen Veränderungen in der Umwelt oder gegen Angriffe ermöglichen. Laut Sunar et al. (2007, S. 109) kommen einige vorgeschlagene Implementierungen von echten Zufallszahlengeneratoren auch ohne Aufbereitungsfunktion aus.

Im Folgenden werden einige Implementierungsmöglichkeiten bzw. konkrete in der Literatur vorgestellte Implementierungen von echten Zufallszahlengeneratoren präsentiert. In den Unterabschnitten 3.1 bis 3.3 werden Zufallszahlengeneratoren beschrieben, die auf physikalischen Phänomenen beruhen, in den Abschnitten 3.4 und 3.5 solche, die Zufallszahlen aus direkten oder indirekten Benutzereingaben erzeugen. Unter 3.6 wird noch kurz auf Zufallszahlengeneratoren, die mit gewöhnlicher PC-Hardware und gleichzeitig ohne notwendige Benutzerinteraktion arbeiten, eingegangen.

3.1. Ringoszillatorbasierte Zufallszahlengeneratoren

Viele in der einschlägigen Literatur (Sunar et al. 2007; Yoo et al. 2010; Maiti et al. 2009; Tkacik 2003) vorgestellte Zufallszahlengeneratoren beruhen auf Ringoszillatoren. Ein Ringoszillator ist eine elektronische Baugruppe, bei der eine ungerade Anzahl von Invertern in Form eines Rings hintereinander geschaltet wird (Sunar et al. 2007, S. 110). Der Ausgang des letzten Inverters ist also mit dem Eingang des ersten Inverters verbunden. Durch die ungerade Anzahl von „Invertierungen" des Signals entsteht bei einem „Null" repräsentierenden Signal am Eingang des ersten Inverters ein

„Eins“ repräsentierendes Signal am Ausgang des letzten Inverters. Durch die Rückkopplung oszilliert das Signal am Ausgang jedes Inverters zwischen „Null“ und „Eins“ in Form einer Rechteckschwingung. Eine wichtige Eigenschaft der Frequenz dieser Schwingung ist, dass sie kleinen zufälligen Schwankungen, genannt *Jitter*, um die *ideale Frequenz* der Schaltung unterworfen ist und daher als Entropiequelle für echte Zufallszahlengeneratoren verwendet werden kann (Sunar et al. 2007, S. 110-111). Die ideale Frequenz der Schwingung hängt von der Anzahl der Inverter und deren Verzögerungszeit ab.

Im Folgenden wird die von Sunar et al. (2007) beschriebene Implementierung eines ringoszillatorbasierten Zufallszahlengenerator näher beschrieben. Bei diesem Vorschlag wird das schwingende Signal mehrerer Ringoszillatoren durch eine XOR-Funktion verknüpft. Ein digitales Ausgabesignal mit einer zufälligen Folge von Nullen und Einsen wird durch periodisches „Sampling“ der Ausgabe der XOR-Funktion erreicht. Dieser Aufbau ist in Abbildung 1 schematisch dargestellt. Die zufälligen Schwankungen eines von *einem* Ringoszillator erzeugten Signals treten nur in zeitlich kleinen Bereichen auf. Dies wird in Abbildung 2 gezeigt, wo die mehrfachen senkrechten Linien den möglichen Bereich des Signalübergangs symbolisieren. Der Rest des Signals ist deterministisch, daher ist (zumindest bei kleiner Zahl von kombinierten Ringoszillatoren) auch ein Teil des von der XOR-Funktion aus mehreren Ringoszillatoren erzeugten Signals deterministisch. Durch Lösung eines Optimierungsproblems können eine Anzahl von Oszillatoren sowie deren ideale Frequenz bestimmt werden, bei der der Output der XOR-Funktion zum größten Teil (ca. 90 %) nichtdeterministisch ist.

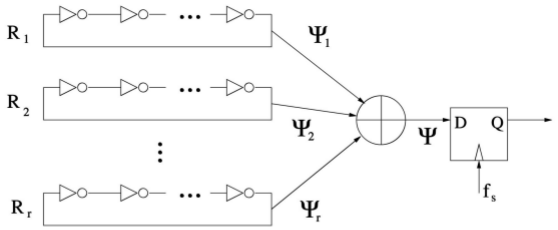

Abbildung 1: Schematisches Schaltbild für die Verknüpfung mehrere Ringoszillatoren inklusive Sampling (Sunar et al. 2007, S. 111)

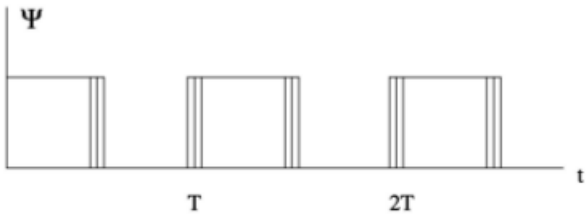

Abbildung 2: Signalverlauf eines Ringoszillators mit der Periode T (Sunar et al. 2007, S. 111)

Um den Effekt der verbleibenden deterministischen Teile auf die erzeugten Zufallszahlen (Vorhersagbarkeit) zu eliminieren, wird eine Aufbereitungsfunktion auf die durch das Sampling gewonnenen Bits angewendet. Dabei handelt es sich um eine sogenannte *resilierende Funktion* (engl. *resilient function*, „Abfederungsfunktion“), die jeweils eine Folge von „gesampelten“ Bits zusammennimmt und daraus eine kürzere und dafür zufälligere Folge von Bits erzeugt. Für die genaue mathematische Definition dieser Funktion wird auf Sunar et al. (2007) verwiesen.

3.2. Auf dem thermischen Rauschen basierende Zufallszahlengeneratoren

Eine weitere, in vielen echten Zufallszahlengeneratoren (u.a. Wang et al. 2005; Kaenel & Takayanagi 2007) verwendete Entropie-Quelle ist das *thermische Rauschen* oder *Wärmerauschen*. Dieses entsteht durch die thermische (brownsche) Bewegung von Elektronen in einem elektrischen Widerstand, wodurch kleine, zufällig schwankende Spannungen zwischen den Enden des Leiters entstehen. Für eine genauere Beschreibung des thermischen Rauschens wird auf das Standardwerk von Davenport und Root (1987, S. 185) verwiesen. Wesentlich für die Anwendung in Zufallszahlengeneratoren ist die

Eigenschaft der Zufälligkeit der Spannungsschwankungen, die bei entsprechender Aufbereitung zur Erzeugung von Zufallszahlen genutzt werden können.

Laut Wang et al. (2005, S. 3970) zeigen die an einem Widerstand gemessenen, durch das thermische Rauschen entstandenen Spannungen unter anderem aufgrund von Temperaturveränderungen während des Betriebs oder deterministischen Einflüssen von außen (z.B. elektromagnetische Schwingungen) auch bei gutem Design der Schaltung gewisse Regularitäten. Eine Beeinflussbarkeit ergibt sich laut Kaenel und Takayanagi (2007, S. 269) vor allem aufgrund der kleinen Spannungen von unter 1 mV. Um dieses Problem zu eliminieren, sind entsprechende Gegenmaßnahmen erforderlich. Die von Kaenel und Takayanagi (2007) vorgeschlagene Schaltung beispielsweise kombiniert einen auf dem thermischen Rauschen basierenden Zufallszahlengenerator (*Thermal noise RNG*) mit einem auf *chaotischen Oszillatoren* basierenden Zufallszahlengenerator (*Chaotic RNG*), deren Outputs durch eine XOR-Funktion verknüpft werden (Aufbau siehe Abbildung 3). Da der chaotische Zufallszahlengenerator einen wesentlich höheren Signalpegel (und damit ein wesentlich günstigeres Signal-Rausch-Verhältnis) liefert, sinkt die Beeinflussbarkeit des Zufallszahlengenerators.

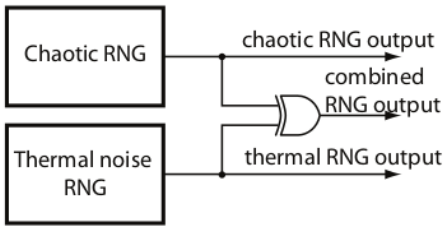

Abbildung 3: Design-Konzept des von Kaenel und Takayanagi vorgestellten Zufallszahlengenerators (Kaenel & Takayanagi 2007, S. 269)

Als weitere Implementierungsvariante eines auf dem thermischen Rauschen beruhenden Zufallszahlengenerators wird hier noch kurz auf den Vorschlag von Wang et al. (2005) eingegangen. Dieses Design verwendet die SHA-2(512)-Hashfunktion (512 Bit langer Hashwert), um die erzeugten Zufallszahlen „zufälliger“ zu machen. Es werden Blöcke von aus der Entropie-Quelle gewonnenen Bits in die hardwaremäßig implementierte Hashfunktion eingespeist, um 512 Bit lange Zufallszahlen zu erzeugen. Um die Entropie der Ergebnisse weiter zu verbessern, werden die Bits aus der Entropie-Quelle mit der letzten erzeugten Zufallszahl kombiniert, bevor sie der SHA-2-Funktion übergeben werden. Dabei würde auch bei einem genau konstanten Spannungswert der Entropie-Quelle eine immer neue und zufällig wirkende Zahl generiert, der Generator würde allerdings vollkommen deterministisch arbeiten.

3.3. Sonstige auf physikalischen Phänomenen basierende Zufallszahlengeneratoren

Neben den in den Abschnitten 3.1 und 3.2 beschriebenen Phänomenen finden sich in der Literatur (Chan et al. 2011, S. 161; Kwok & Lam 2006, S. 1; Wang et al. 2005, S. 3970) noch Hinweise auf weitere physikalische Effekte, die für die Realisierung einer Entropie-Quelle genutzt werden können. Diese sind unter anderem

- das Frequenzrauschen in der Atmosphäre (wie es bei einem schlecht eingestellten Radio zu hören ist, für eine konkrete Implementierung siehe Brederlow et al. (2007)),
- radioaktiver Zerfall,

- aus Halbleiterbauelementen gewonnenes Rauschen (für eine konkrete Implementierung siehe Killmann und Schindler (2008)) und
- zufällige quantenmechanische Vorgänge (für eine konkrete Implementierung siehe Jennewein et al. (1999)).

3.4. Auf direkten Benutzereingaben basierende Zufallszahlengeneratoren

Zufallszahlengeneratoren können als Entropie-Quelle direkte Benutzereingaben wie z.B. Mausbewegungen oder Tastatureingaben (betätigte Tasten, Zeitpunkt und Länge der Betätigung) nutzen (Chan et al. 2011, S. 161). Da es sich dabei um vom Menschen willkürlich ausgeführte Aktionen handelt, ist eine entsprechende Aufbereitung entscheidend.

Als konkretes Beispiel für einen Zufallszahlengenerator, der auf direkten Benutzereingaben aufbaut, wird hier der Vorschlag von Marton et al. (2011) vorgestellt. Dieser wird von den Autoren als *Unpredictable Random Number Generator (URNG)* (Definition siehe Abschnitt 2) klassifiziert. Der Aufbau könnte auch als hybrider Zufallszahlengenerator bezeichnet werden, da die zufälligen, auf Mausbewegungen des Benutzers gewonnenen Daten durch Pseudozufallszahlengeneratoren weiter aufbereitet werden.

Konkret wurde der Zufallszahlengenerator von Marton et al. (2011) so umgesetzt, dass ein am Bildschirm angezeigtes Fenster rasterförmig unterteilt wurde, wobei jeder Zelle des Rasters ein Algorithmus zur Erzeugung von Pseudozufallszahlen zugeordnet wird (siehe Abbildung 4). Die verwendeten Algorithmen können variiert werden, ebenso können einzelne Algorithmen mehrmals mit unterschiedlicher Parametrisierung im Raster „platziert“ werden. Für eine Beschreibung der einzelnen Pseudozufallszahlenalgorithmen wird auf Marton et al. (2011, S. 1-3) verwiesen.

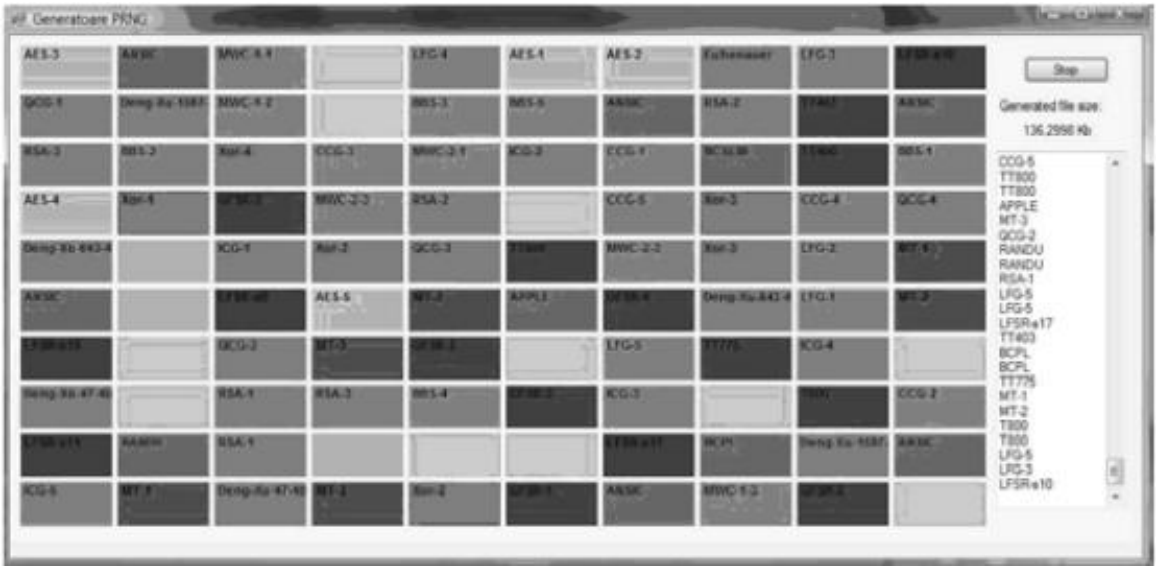

Abbildung 4: Oberfläche des von Marton et al. vorgestellten Zufallszahlengenerators (Marton et al. 2011, S. 3)

Wird die Generierung der Zufallszahlen gestartet, so arbeiten die Pseudozufallszahlengeneratoren je nach Konfiguration sequentiell oder parallel. Auch für die Aktivierung und Deaktivierung der Ausführung der einzelnen Algorithmen sind verschiedene Strategien anwendbar (Start beim Eindringen der Maus in die Rasterzelle und Stopp beim Verlassen; Start beim Eindringen und Stopp beim nächsten Eindringen). Ist ein Algorithmus aktiv, so produziert er Zufallszahlen mit den

Koordinaten der Mausposition im (gesamten) Fenster als Startwert (Seed). Alle Pseudozufallszahlengeneratoren produzieren durch entsprechende Konfiguration Pseudozufallszahlen zwischen 0 und 1, sodass die Ergebnisse einfach kombiniert werden können. Die durch die jeweils aktiven Algorithmen erzeugten Zufallszahlen werden in einem gemeinsamen Speicherort abgelegt und stellen das Ergebnis des Zufallszahlengenerators dar.

Bei der Analyse der Ergebnisse wurde von Marton et al. (2011) festgestellt, dass die verwendete Kombination der Algorithmen zur Pseudozufallszahlerzeugung für die erreichte Qualität entscheidend ist. Bei Verwendung von nur wenigen Algorithmen mit geringer „Zufälligkeit“ (z.B. *Eichenauer-Generator* und *Blum Blum Shub*, Beschreibung siehe Marton et al. (2011, S. 1-3)) zeigten sich unbefriedigende statistische Eigenschaften der Ergebnisse. Unter anderem waren lange Folgen von Nullen erkennbar. Durch Kombination verschiedenster Algorithmen mit komplementären Charakteristiken und verschiedenen Längen von möglicherweise existierenden periodischen Mustern konnten jedoch Zufallszahlen mit guten statistischen Eigenschaften und vor allem Unvorhersagbarkeit erzeugt werden.

3.5. Auf Daten von mobilen Geräten basierende Zufallszahlengeneratoren

Hier wird beispielhaft ein von Suciu et al. (2011) vorgestelltes Konzept präsentiert, das ebenfalls von Benutzern kommende Daten zur Erzeugung von Zufallszahlen nutzt. Dabei werden vier Datenquellen verwendet, wobei sich jede auf eine Hardware-Komponente von Smartphones bezieht. Die vier Datenquellen sind

- der Beschleunigungssensor,
- der Magnetfeldsensor („Kompass“),
- der Lagesensor und
- der GPS-Empfänger.

Suciu et al. (2011) weisen jedoch darauf hin, dass das Konzept sehr einfach auf weitere vor allem in neueren Smartphones enthaltene Datenquellen (z.B. Thermometer) erweitert werden kann.

Der Zufallszahlengenerator besteht aus einem Aufzeichnungsmodul, das Daten aus den beschriebenen Quellen vor der Erzeugung von Zufallszahlen aufzeichnet und einem Kombinationsmodul, das aus den aufgezeichneten Daten schließlich Zufallszahlen generiert.
Im Aufzeichnungsmodul wird jedes erfasste Datum (z.B. ein Beschleunigungswert) je nach Sensor als 32- oder 64-Bit-Zahl gespeichert. Handelt es sich um eine 32-Bit-Zahl, so muss diese vor der Kombination auf eine 64-Bit-Zahl erweitert werden, wobei dazu ein Algorithmus verwendet wird, der die Entropie der Bits innerhalb der Zahl möglichst erhält.
Im Kombinationsmodul wird ein spezieller Kombinationsalgorithmus angewandt, der zuerst die zu verwendenden Datenquellen selektiert (v.a. GPS steht nicht immer zur Verfügung), dann die Daten zwischen den Datenquellen untereinander kombiniert (Datum für Datum, z.B. mit einer XOR-Verknüpfung) und schließlich je nach benötigter Länge der Zufallszahl eine Menge von Bits in jedem Datum selektiert. Bei letzterem Schritt werden bei einer benötigten Länge von *n* die *n* niederwertigsten Bits selektiert, da sich diese am schnellsten ändern, beispielsweise bereits bei kleinen Standortänderungen des Benutzers.

Ein wichtiger Vorteil dieses auf dem Benutzerverhalten basierenden Zufallszahlengenerators ist, dass der Benutzer zur Generierung einer Zufallszahl nicht selbst zusätzliche Aktivitäten zur Erzeugung der

„Zufälligkeit“ (z.B. Bewegung der Maus) ausführen muss. Gleichzeitig ist auch keine spezielle Hardware mit integrierter Entropie-Quelle erforderlich. Suciu et al. (2011) konnten experimentell durch Anwendung der NIST-Testsuite (siehe Abschnitt 5) zwar eine ausreichende Qualität der erzeugten Zufallszahlen nachweisen, jedoch sehen die Autoren noch Potential für Verbesserungen bei der Qualität der Daten aus einzelnen Sensoren sowie beim Selektionsmechanismus zur Auswahl geeigneter Datenquellen.

3.6. Auf gewöhnlicher PC-Hardware basierende Zufallszahlengeneratoren

Die Ansätze zur Generierung von Zufallszahlen, die auf zufälligen physikalischen Phänomenen beruhen, können mit relativ hohen Datenraten gute Zufallszahlen erzeugen, benötigen jedoch spezielle Hardware. Die Ansätze, die auf Nutzereingaben basieren, benötigen eine Person, die sie betätigt und generieren Zufallszahlen potentiell nur sehr langsam. Es wurden daher auch Zufallszahlengeneratoren entwickelt, die mit gewöhnlicher Hardware ohne erforderliche Benutzerinteraktion Zufallszahlen generieren, wobei die „Zufälligkeit“ hier oft nicht auf wirklich zufälligen physikalischen Phänomenen beruht, sondern auf der Nicht-Nachvollziehbarkeit komplexer deterministischer Vorgänge.

Ein Beispiel wäre das von Colesa et al. (2008) vorgestellte Verfahren, das den praktisch nicht-vorhersagbaren Ausgang sogenannter „Race Conditions“ bei der Ausführung paralleler Prozesse ausnützt. Die Autoren weisen darauf hin, dass neben der Komplexität des Schedulings auch tatsächlicher „physikalischer Zufall“ unter anderem durch Interrupts von nicht-perfekten (zufällig in ihrer Frequenz schwankenden) Timern zur Unvorhersagbarkeit des Outputs führt. Es kann also von einem echten Zufallszahlengenerator gesprochen werden, von den Autoren wird der Vorschlag als nicht-vorhersagbarer Zufallszahlengenerator (URNG; siehe Abschnitt 2) klassifiziert.

Als weiteres Beispiel für auf gewöhnlicher PC-Hardware basierende Zufallszahlengeneratoren sei hier auf die Nutzung von nicht-vorhersagbaren Festplatten-Zugriffszeiten (Chan et al. 2011, S. 161) verwiesen.

4. Angriffsmöglichkeiten und Schutz

Da viele kryptographische Anwendungen (z.B. Schlüsselgenerierung) auf der Nichtvorhersagbarkeit zufälliger Daten beruhen, können sich Angriffe auf Zufallszahlengeneratoren richten, um Zugriff auf ein gesichertes System zu erhalten. Entsprechend des Fokus der Arbeit wird hier auf Angriffe auf echte Zufallszahlengeneratoren eingegangen, und nicht die Vorhersage der „Zufallszahlen“ bei Pseudozufallszahlengeneratoren. In der Literatur (Sunar et al. 2007, Bayon et al. 2012) wird bei Angriffen auf echte Zufallszahlengeneratoren zwischen

- Fehlereinschleusungen (engl. *fault-introduction*, aktive Attacken) und
- Bit-Analyse und -Vorhersage (engl. *bit-deduction*, passive Attacken)

unterschieden.

Im Folgenden wird auf Attacken durch Fehlereinschleusungen näher eingegangen. Laut Sunar et al. (2007, S. 116) ist das Ziel bei Fehlereinschleusungen, eine Verzerrung der statistischen Eigenschaften des Outputs, also ein Bias, und damit eine (zumindest einfachere) Vorhersagbarkeit der Ergebnisse zu erreichen. Dazu können verschiedene Arten von Attacken durchgeführt werden, wobei von Sunar et al.

(2007, S. 116) eine Einteilung in noninvasive und invasive Attacken vorgenommen wird.
Als *noninvasive Attacke* wird jede *externe* Einflussnahme auf einen Zufallszahlengenerator bezeichnet. Dazu gehören beispielsweise das Bestrahlen mit elektromagnetischen Wellen oder das Zuführen von Wärme zur Erhöhung der Temperatur (Sunar et al. 2007, S. 116). Laut dieser Definition wird also bereits für noninvasive Attacken ein physischer Zugang zum Zufallszahlengenerator benötigt.
Invasive Attacken hingegen nehmen dauernde Veränderungen am Zufallszahlengenerator (bzw. am Chip auf dem dieser implementiert ist) vor. Beispielsweise könnte eine von einem Transistor geschaltete elektrische Verbindung permanent geöffnet oder kurzgeschlossen werden, um die Arbeitsweise der Schaltung zu verändern (Sunar et al. 2007, S. 116-117).

Bei der Analyse der Angriffssicherheit von echten Zufallszahlengeneratoren müssen immer alle Komponenten von diesen betrachtet werden. So wird von Sunar et al. (2007) die Angriffssicherheit des in Abschnitt 3.1 näher beschriebenen Zufallszahlengenerators in Bezug auf die Ringoszillatoren, die XOR-Verknüpfung, den Sampler, sowie die „Abfederungsfunktion“ beschrieben.

Aufgrund der großen Verbreitung und der näheren Beschreibung in Abschnitt 3.1 wird hier beispielhaft genauer auf ringoszillatorbasierte Zufallszahlengeneratoren eingegangen. Markettos und Moore (2009) beschreiben als wichtigste Angriffsmöglichkeit auf ringoszillatorbasierte Zufallszahlengeneratoren die Einschleusung bestimmter Frequenzen in die Stromversorgung, also eine noninvasive Attacke. Dadurch kann die (sonst zufällig schwankende) Frequenz der eingesetzten Oszillatoren stabilisiert werden, wobei für die physikalische Erklärung für diese Beeinflussungsmöglichkeit auf Markettos und Moore (2009) verwiesen wird. Die Autoren konnten dabei einen Angriff auf ein 2004 von einer britischen Bank eingeführtes Chipkarten-Zahlungssytem durchführen. Bei diesem System befindet sich ein Zufallszahlengenerator im Lesegerät, der „Challenges“ für den Authentifizierungsmechanismus erzeugt. Durch Einspeisung bestimmte Frequenzen in die Stromversorgung des Zufallszahlengenerators kann die Qualität der erzeugten Zufallszahlen signifikant verschlechtert werden, der Output weist einen Bias auf und „besteht“ die NIST- bzw. DIEHARD-Tests nicht mehr (zu den Tests siehe Abschnitt 5). Es werden wesentlich weniger verschiedene Challenges generiert, wodurch bei einer Brute-Force-Attacke deutlich weniger Versuche notwendig sind, um wiederholt die gleiche Challenge zu erhalten und damit eine Replay-Attacke mit einer aufgezeichneten Sitzung durchführen zu können. Genaue Angaben über die Anzahl der notwendigen Versuche wurden nicht publiziert, jedoch wurde in einem ähnlichen Experiment der Autoren die Zahl der von einem ringoszillatorbasierten Zufallszahlengenerator ausgegebenen Zahlen von 2^{64} auf 3300 reduziert, was ungefähr einer Teilung durch $5 * 10^{15}$ entspricht.

Ein Schutz vor solchen Attacken ist laut Markettos und Moore (2009, S. 13) durch Glättung der Eingangsspannung (Filterung „gefährlicher“ Frequenzen) oder Verbesserung des Designs der Ringoszillatoren in Bezug auf deren Anzahl und Frequenz (Verhinderung eines Bias auch bei Störfrequenzen) möglich. Als einfachere Maßnahme kann aber auch unter Umständen die Zahl der Zugriffsversuche auf das System durch geeignete Mechanismen begrenzt werden, um Brute-Force-Attacken auf beeinflusste Zufallszahlengeneratoren zu verhindern.

Bayon et al. (2012) beschreiben die Möglichkeit kontaktloser Angriffe auf ringoszillatorbasierte Zufallszahlengeneratoren durch elektromagnetische Wellen. Auch hier konnte ein Bias im Output von auf bis zu 50 Ringoszillatoren basierenden (und bisher als sicher geltenden) Zufallszahlengeneratoren erreicht werden. Schutz vor solchen Angriffen ist durch eine entsprechende elektromagnetische Abschirmung möglich.

Auch auf Zufallszahlengeneratoren, die auf dem thermischen Rauschen beruhen, können Angriffe durch Beeinflussung der Stromversorgung oder elektromagnetische Störungen erfolgen (Wang et al. 2005). Die von Wang et al. (2005) zur Aufbereitung der Zufallszahlen eingesetzte SHA-2-Funktion (siehe auch Abschnitt 3.2) eliminiert zwar die Korrelation zwischen äußeren Einflüssen und dem Output, die Eigenschaft der niedrigen Anzahl der verschiedenen Outputs bei einem erfolgreich attackierten Zufallszahlengenerator bleibt aber bestehen. Weitere Schutzmaßnahmen (z.B. Abschirmung) können also durch eine Aufbereitung der Daten aus der Entropie-Quelle nicht vollständig ersetzt werden. Ein wichtiger nachgelagerter Prüfschritt zur Verhinderung von Angriffen sind jedoch Selbsttests, für die auf das Ende von Abschnitt 5 verwiesen wird.

Die zuvor beschriebenen Angriffsmöglichkeiten sind zwar non-invasiv, benötigen aber trotzdem eine physische Zugangsmöglichkeit des Angreifers zum Zufallszahlengenerator. Im Folgenden soll noch kurz auf die Analyse von Output-Bits und deren Vorhersage eingegangen werden. Dichtl (2003) beschreibt in seiner Arbeit, wie aufgrund der Analyse einer Menge bekannter Outputs eines ringoszillatorbasierten Zufallszahlengenerators die Anzahl der eingesetzten Oszillatoren, deren ideale Frequenz, sowie die Sampling-Frequenz relativ zuverlässig abgeschätzt werden kann, wodurch Aussagen über zukünftige Outputs möglich werden. Auch wenn heutige echte Zufallszahlengeneratoren einen wesentlich komplexeren Aufbau (z.B. Kombination mehrerer Entropie-Quellen) besitzen, konnte nichts desto trotz in einer aktuelleren Arbeit von Baudet et al. (2011) gezeigt werden, dass auch bei vielen neueren Zufallszahlengeneratoren durch Analyse einer ausreichenden Menge von Output-Bits zumindest optimierte Brute-Force-Attacken möglich sind. Dies gilt vor allem, wenn „Over-Sampling“ vorliegt, also die Ausgabe der Ringoszillatoren zu häufig zur Erzeugung einer Zufallszahl ausgelesen wird. Die Erfolgsaussichten solcher Analyse-Attacken hängen wesentlich von den statistischen Eigenschaften der erzeugten Zufallszahlen ab. Daher ist es wichtig, die Qualität von Zufallszahlen im Hinblick auf ihre Verteilung messen zu können, wozu es standardisierte Tests gibt. Auf solche wird im nächsten Abschnitt näher eingegangen.

Abschließend soll in diesem Kapitel noch auf Seitenkanalattacken auf echte Zufallszahlengeneratoren eingegangen werden. Dabei handelt es sich weder um Fehlereinschleusungen, noch um reine Bit-Analyse und –Vorhersage, Seitenkanalattacken werden jedoch auch als passive Attacken klassifiziert. Bei Seitenkanalattacken werden laut Bayrak et al. (2012, S. 20:1) Informationen wie der Stromverbrauch, die elektromagnetische Abstrahlung oder akustische Geräusche von kryptographischen Geräten genutzt, um geheime Informationen auszuforschen. Bei solchen geheimen Informationen kann es sich beispielsweise um den aktuellen Zustand eines Zufallszahlengenerators handeln, der Schlüsse auf die zukünftig erzeugten Zahlen zulässt. Auf die technische Funktionsweise solcher Attacken wird hier nicht näher eingegangen. Schutz vor Seitenkanalattacken kann durch entsprechende Abschirmungsmaßnahmen sowie Vermeidung einer Korrelation zwischen von außen messbaren Größen und den generierten Zufallszahlen geboten werden. Zur weiteren Absicherung existieren aber auch komplexere Maßnahmen, wie das von Bayrak et al. (2012) vorgestellte zufällige Vertauschen von voneinander unabhängigen Befehlen bei in Zufallszahlengeneratoren eingesetzten Algorithmen, wodurch Zustandsinformationen weniger leicht zur Vorhersage des Outputs genutzt werden können.

5. Tests für Zufallszahlengeneratoren

In Abschnitt 2 wurde auf die Zufälligkeit der Ergebnisse als eine wichtige Anforderung an Zufallszahlengeneratoren hingewiesen. Das Vorhandensein von Zufälligkeit kann durch auf den Ergebnissen durchgeführte statistische Tests überprüft werden, wobei (nur) untersucht werden kann, ob die vorliegende Verteilung der Ergebnisse zufällig erscheint in dem Sinne, dass die Verteilung statistische Eigenschaften aufweist, die auch eine echt zufällige Zahlenfolge (als theoretisches Konstrukt in der Statistik) aufweisen würde (Rukhin et al. 2010, S. 1:3). Daher können diese Tests auch zur Überprüfung von Pseudozufallsgeneratoren eingesetzt werden.

Wie bei jedem statistischen Test gibt es auch bei einem Test auf Zufälligkeit eine Nullhypothese (H_0) und eine Alternativhypothese (H_a). Die Nullhypothese besagt, dass Zufälligkeit vorliegt, die Alternativhypothese besagt, dass keine Zufälligkeit vorliegt. Neben der Formulierung der Hypothesen ist vor der Testdurchführung außerdem ein Signifikanzniveau (z.B. 1 %) festzulegen, das die Wahrscheinlichkeit des Verwerfens der Nullhypothese angibt, wenn diese in Wirklichkeit erfüllt ist. Dadurch kann die Wahrscheinlichkeit, dass ein Test fehlschlägt, weil die für den Test verwendeten Ergebnisse zufälligerweise bestimmte Eigenschaften (z.B. Muster) aufweisen, also die Wahrscheinlichkeit für einen „Fehlalarm“ (α-Fehler) kontrolliert werden (Rukhin et al. 2010, S. 1:3-1:4).

In dieser Arbeit soll exemplarisch etwas näher auf eine Sammlung von solchen Tests auf Zufälligkeit, eine sogenannte Testsuite, eingegangen werden. Konkret handelt es sich dabei um die *NIST Test Suite* des amerikanischen *National Institute of Standards and Technology (NIST)*, die in der aktuellen Version in Rukhin et al. (2010) publiziert wurde. Diese Testsuite wird in der Literatur häufig zitiert und viele Zufallszahlengeneratoren wurden durch die enthaltenen Tests überprüft.

Die NIST Test Suite besteht aus 15 einzelnen Tests, die auf die Erkennung verschiedener Arten von „Nicht-Zufälligkeit“ abzielen. Dabei überprüft der Frequency (Monobit) Test beispielsweise, ob der Anteil der Einsen in einem Bitstrom nahe bei ½ liegt, wie dies bei einer echt zufälligen Bitfolge zu erwarten wäre. Je nach Länge der überprüften Bitfolge und festgelegtem Signifikanzniveau ist eine größere oder kleinere Abweichung von ½ zulässig.
Ein weiterer Test ist der Runs Test, der generierte Zufallszahlen auf das Nicht-Vorhandensein von langen, ununterbrochenen Folgen („Runs“) von Nullen oder Einsen überprüft.
Für eine Auflistung aller 15 Tests inklusive Beschreibung der Durchführung wird auf Rukhin et al. 2010 (S. 2:1-2:40) verwiesen. Aufgrund der unterschiedlichen Zielsetzungen der einzelnen Tests ist es sinnvoll, immer mehrere bzw. alle von ihnen zur Überprüfung eines Zufallszahlengenerators anzuwenden. Zur effizienten Durchführung der Tests wird vom NIST ein Software-Tool mit dem Namen *sts* (Statistical Test Suite) kostenlos auf der NIST-Website (NIST 2012) zum Download angeboten.

Eine ähnliche zu der vom NIST publizierten Testsuite wird im vom deutschen Bundesamt für Sicherheit in der Informationstechnik (BSI) herausgegebenen Standard AIS 31 (Killmann & Schindler 2008, S. 44-56) beschrieben. Viele Zufallszahlengeneratoren werden auch nach der etwas älteren DIEHARD-Testsuite, publiziert von Marsaglia (1995), geprüft.

Eine sehr hohe statistische Qualität der Ergebnisse eines Zufallszahlengenerators, nachgewiesen durch die Anwendung von Testsuiten, impliziert nicht automatisch die Sicherheit eines Zufallszahlengenerators im Einsatz, da die Tests bei Durchführung im „Standardbetrieb“ nichts über Möglichkeiten zu Fehlereinschleusungen bzw. Seitenkanalattacken (siehe Abschnitt 4) aussagen. Statistische Tests können allerdings zeigen, ob ein Versuch zur Einflussnahme auf einen

Zufallszahlengenerator Erfolg gezeigt hat oder sich die Qualität des Outputs durch Umwelteinflüsse verschlechtert hat. Daher sind beispielsweise im Standard FIPS 140-2 des NIST Selbsttests für zertifizierte Zufallszahlengeneratoren vorgesehen, die nach deren Aktivierung („Power-Up Tests“) bzw. kontinuierlich durchgeführt werden müssen (NIST 2001, S. 30, 33-36). Bei Erkennung eines Fehlers dürfen die erzeugten Zufallszahlen nicht mehr verwendet werden. Auch im deutschen AIS-Standard sind Beispiele für Selbsttests angegeben (Killmann & Schindler 2008, S. 111-116). Die beschriebenen Selbsttest-Mechanismen sind allerdings meist weniger komplex als die Testsuiten und beschränken sich teilweise (wie z.B. der in FIPS 140-2 beschriebene „Power-Up Test“) auf die Erkennung von Ähnlichkeiten in zwei aufeinander folgenden Bitfolgen.

6. Konklusion / Zusammenfassung

Zu Zufallszahlengeneratoren gibt es eine Vielzahl von wissenschaftlichen Publikationen, in denen hauptsächlich konkrete Designs von echten Zufallszahlengeneratoren vorgestellt werden. Viele davon sind innerhalb der letzten Dekade erschienen. Darin spiegelt sich die zunehmende Bedeutung der Kryptographie für die Nutzung des Internets für sicherheitsrelevante Anwendungen, wie z.B. Online-Banking oder elektronische Geschäftsabschlüsse, wieder.

Die meisten echten Zufallszahlengeneratoren nutzen physikalische Phänomene, hauptsächlich Ringoszillatoren oder thermisches Rauschen, als Entropie-Quelle. Es kommen aber auch Benutzereingaben als Input zur Anwendung, wenngleich auch in einem eingeschränkteren Anwendungsfeld.

In den entsprechenden Publikationen zu echten Zufallszahlengeneratoren werden zumeist auch Testresultate (NIST, DIEHARD) präsentiert, außerdem wird die allgemeine Angriffssicherheit analysiert. Üblicherweise werden die vorgestellten echten Zufallszahlengeneratoren als schwer angreifbar bezeichnet, dennoch gelingen immer wieder Attacken (z.B. der von Markettos und Moore (2009) beschriebene Angriff auf ein Kartenzahlungssystem durch Beeinflussung eines ringoszillatorbasierten Zufallszahlengenerators). Daher werden auch immer wieder Verbesserungsvorschläge zu einzelnen Designs (z.B. Yoo et al. (2010)) bzw. völlig neue Aufbauten publiziert.

Von behördlicher Seite (z.B. amerikanisches NIST oder deutsches BSI) werden umfangreiche Standards vorgegeben, aber auch Evaluierungsmethoden (Tests) zur Verfügung gestellt. Dadurch steht ein „Framework“ zur Umsetzung von sicheren Zufallszahlengeneratoren zur Verfügung. Aufgrund der technischen Weiterentwicklung auch auf der „Angreiferseite“ wird jedoch eine ständige Verbesserung der Technologien zur Erzeugung von Zufallszahlen für kryptographische Anwendungen erforderlich sein. Zusammenfassend lässt sich also sagen, dass es *den* (auch in Zukunft) absolut sicheren Zufallszahlengenerator nicht gibt, und diesen wird es wohl auch in absehbarer Zeit nicht geben.

Literatur

Abu, N. A., & Muslim, Z. (2008). Random room noise for cryptographic key. In Digital Ecosystems and Technologies, 2008. DEST 2008. 2nd IEEE International Conference on (S. 381–387). doi:10.1109/DEST.2008.4635143

Baudet, M., Lubicz, D., Micolod, J., & Tassiaux, A. (2011). On the Security of Oscillator-Based Random Number Generators. Journal of Cryptology, 24(2), 398–425. doi:10.1007/s00145-010-9089-3

Bayon, P., Bossuet, L., Aubert, A., Fischer, V., Poucheret, F., Robisson, B., & Maurine, P. (2012). Contactless Electromagnetic Active Attack on Ring Oscillator Based True Random Number Generator. In W. Schindler & S. Huss (Hrsg.), Constructive Side-Channel Analysis and Secure Design (Bd. 7275, S. 151–166). Springer Berlin Heidelberg. doi:10.1007/978-3-642-29912-4_12

Bayrak, A. G., Velickovic, N., Ienne, P., & Burleson, W. (2012). An architecture-independent instruction shuffler to protect against side-channel attacks. ACM Trans. Archit. Code Optim., 8(4), 20:1–20:19. doi:10.1145/2086696.2086699

Brederlow, R., Prakash, R., Paulus, C., & Thewes, R. (2006). A low-power true random number generator using random telegraph noise of single oxide-traps. In Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International (S. 1666–1675). doi:10.1109/ISSCC.2006.1696222

Chan, J. J. M., Sharma, B., Lv, J., Thomas, G., Thulasiram, R., & Thulasiraman, P. (2011). True Random Number Generator Using GPUs and Histogram Equalization Techniques. In High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on (S. 161–170). doi:10.1109/HPCC.2011.30

Chen, I.-T., Tsai, J.-M., & Tzeng, J. (2011). Audio random number generator and its application. In Machine Learning and Cybernetics (ICMLC), 2011 International Conference on (Bd. 4, S. 1678–1683). doi:10.1109/ICMLC.2011.6017002

Colesa, A., Tudoran, R., & Banescu, S. (2008). Software Random Number Generation Based on Race Conditions (S. 439–444). IEEE. doi:10.1109/SYNASC.2008.36

Davenport, W. B., & Root, W. L. (1987). An introduction to the theory of random signals and noise. New York: IEEE Press.

Dichtl, M. (2003). How to Predict the Output of a Hardware Random Number Generator. In C. Walter, Ç. Koç, & C. Paar (Hrsg.), Cryptographic Hardware and Embedded Systems - CHES 2003 (Bd. 2779, S. 181–188). Springer Berlin Heidelberg. Abgerufen von http://dx.doi.org/10.1007/978-3-540-45238-6_15

Haahr, M. (1999). Introduction to Randomness and Random Numbers. Abgerufen 8. Mai 2013, von http://www.random.org/randomness/

Jennewein, T., Achleitner, U., Weihs, G., Weinfurter, H., & Zeilinger, A. (2000). A fast and compact quantum random number generator. Review of Scientific Instruments, 71(4), 1675–1680. doi:10.1063/1.1150518

Kaenel, V., & Takayanagi, T. (2007). Dual True Random Number Generators for Cryptographic Applications Embedded on a 200 Million Device Dual CPU SoC. In Custom Integrated Circuits Conference, 2007. CICC ’07. IEEE (S. 269–272). doi:10.1109/CICC.2007.4405730

Killmann, W., & Schindler, W. (2008). A Design for a Physical RNG with Robust Entropy Estimators. In Proceeding sof the 10th international workshop on Cryptographic Hardware and Embedded Systems (S. 146–163). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/978-3-540-85053-3_10

Kwok, S. H. M., & Lam, E. Y. (2006). FPGA-based High-speed True Random Number Generator for Cryptographic Applications. In TENCON 2006. 2006 IEEE Region 10 Conference (S. 1–4). doi:10.1109/TENCON.2006.344013

Maiti, A., Nagesh, R., Reddy, A., & Schaumont, P. (2009). Physical unclonable function and true random number generator: a compact and scalable implementation. In Proceedings of the 19th ACM Great Lakes symposium on VLSI (S. 425–428). New York, NY, USA: ACM. doi:10.1145/1531542.1531639

Markettos, A. T., & Moore, S. W. (2009). The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators. In Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems (S. 317–331). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/978-3-642-04138-9_23

Marsaglia, G. (1995). The Marsaglia Random Number CDROM including the Diehard Battery of Tests. Abgerufen 17. Mai 2013, von http://stat.fsu.edu/pub/diehard/

Marton, K., Suciu, A., & Petricean, D. (2011). A parallel unpredictable random number generator (S. 1–5). IEEE. doi:10.1109/RoEduNet.2011.5993701

National Institute of Standards and Technology (NIST). (2001). FIPS PUB 140-2 - Security Requirements for Cryptographic Modules (Includes Change Notices as of December 3, 2002).

National Institute of Standards and Technology (NIST). (2012). NIST.gov - Computer Security Division - Computer Security Resource Center - Download Documentation and Software. Abgerufen 21. Mai 2013, von http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html

Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., … Vo, S. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications.

Suciu, A., Lebu, D., & Marton, K. (2011). Unpredictable random number generator based on mobile sensors. In Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on (S. 445–448). doi:10.1109/ICCP.2011.6047913

Sunar, B., Martin, W. J., & Stinson, D. R. (2007). A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks. Computers, IEEE Transactions on, 56(1), 109–119. doi:10.1109/TC.2007.250627

Thamrin, N. M., Witjaksono, G., Nuruddin, A., & Abdullah, M. S. (2009). An Enhanced Hardware-based Hybrid Random Number Generator for Cryptosystem. In Information Management and Engineering, 2009. ICIME ’09. International Conference on (S. 152–156). doi:10.1109/ICIME.2009.115

Tkacik, T. (2003). A Hardware Random Number Generator. In B. Kaliski, çetinK. Koç, & C. Paar (Hrsg.), Cryptographic Hardware and Embedded Systems - CHES 2002 (Bd. 2523, S. 450–453). Springer Berlin Heidelberg. doi:10.1007/3-540-36400-5_32

Wang, Y.-H., Zhang, H.-G., Shen, Z.-D., & Li, K.-S. (2005). Thermal noise random number generator based on SHA-2 (512). In Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on (Bd. 7, S. 3970–3974 Vol. 7). doi:10.1109/ICMLC.2005.1527631

Yoo, S.-K., Karakoyunlu, D., Birand, B., & Sunar, B. (2010). Improving the Robustness of Ring Oscillator TRNGs. ACM Trans. Reconfigurable Technol. Syst., 3(2), 9:1–9:30. doi:10.1145/1754386.1754390