Cryptography 101: From Theory to Practice

Chapter 3 – Random Generators

Rolf Oppliger

February 7, 2022

Terms of Use

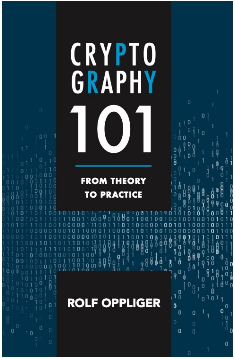
- This work is published with a CC BY-ND 4.0 license (🅭🅯⊜)
  - CC = Creative Commons (🅭)
  - BY = Attribution (🅯)
  - ND = No Derivatives (⊜)

whoami

rolf-oppliger.ch
rolf-oppliger.com

- Swiss National Cyber Security Centre NCSC (scientific employee)
- eSECURITY Technologies Rolf Oppliger (founder and owner)
- University of Zurich (adjunct professor)
- Artech House (author and series editor for information security and privacy)

Reference Book



© Artech House, 2021
ISBN 978-1-63081-846-3

https://books.esecurity.ch/crypto101.html

Challenge Me

Part I

UNKEYED CRYPTOSYSTEMS

Outline

3. Random Generators

Random numbers should not be generated with a method chosen at random.

– Donald E. Knuth

3. Random Generators

3. Random Generators

3.1 Introduction

- The term **randomness** refers to nondeterminism
- If one says that an event is random, then one means that one cannot determine its outcome, or — equivalently — that its outcome is nondeterministic
- Whether randomness really exists is also a philosophical question
- Present knowledge in quantum physics suggests that randomness does exist

3. Random Generators

3.1 Introduction

- Assuming the existence of randomness, one may ask whether it is possible to measure it in one way or another
- In theory, the **Kolmogorov complexity** measures the minimal length of a program for a Turing machine that is able to generate a given sequence of values
- The Kolmogorov complexity is noncomputable
- For a bit sequence generated with a linear feedback shift register (LFSR), one can use the **linear complexity** (and the Berlekamp-Massey algorithm) to measure its randomness
- The linear complexity refers to the size of the shortest LFSR that can generate the sequence

3. Random Generators

3.1 Introduction

- Aside from the questions whether randomness exists and how to measure it, one may wonder how to generate random values
- This is where random generators and random bit generators according to Definition 2.1 come into play
- If one can generate random bits, then one can also generate random numbers (of any size) — and vice versa
- To construct an n-bit random number a, one sets $b_n = 1$, uses the random bit generator to generate $n-1$ random bits $b_{n-1}, \ldots, b_1$, and declares

$$a = \sum_{i=1}^{n} b_i 2^{i-1}$$

3. Random Generators

3.2 Realizations and Implementations

- RFC 4086 (BCP 106) recommends the use of special hardware to generate truly random bits
- But there are situations in which special hardware is not available, and software must be used instead
- Consequently, there is room for both hardware-based and software-based random generators
- In either case, deskewing techniques may be used (to deal with statistical defects)

3. Random Generators

3.2 Realizations and Implementations – Hardware-Based Random Generators

- **Hardware-based** random generators exploit the randomness that may occur in physical processes and phenomena
  - The elapsed time between emission of particles during radioactive decay
  - The thermal noise from a semiconductor diode or resistor
  - The frequency instability of a free-running oscillator
  - The amount a metal-insulator-semiconductor capacitor is charged during a fixed period of time
  - The air turbulence within a sealed disk drive
  - The sound from a microphone or video input from a camera
  - ...

3. Random Generators

3.2 Realizations and Implementations – Software-Based Random Generators

- **Software-based** random generators combine several sources of random-looking values (with a strong mixing function)
  - System clock
  - Elapsed time between keystrokes or mouse movements
  - Content of input/output buffers
  - Input provided by the user
  - Values of operating system variables, such as system load or network statistics
  - ...

3. Random Generators

3.2 Realizations and Implementations – Deskewing Techniques

- Any source of random bits may be defective in the sense that the output bits are biased or correlated
- There are several **deskewing techniques**
- For example, John von Neumann's technique can be used if the output bits are biased
  - A 10 pair is transformed to 1
  - A 01 pair is transformed to 0
  - 00 and 11 pairs are discarded
- A simpler technique is to pass the bit sequence through a cryptographic hash function or block cipher (e.g. with a random key)

3. Random Generators

3.3 Statistical Randomness Testing

- While it is impossible to give a mathematical proof that a generator is a random (bit) generator, statistical randomness testing may help detecting certain kinds of defects or weaknesses
- This is accomplished by taking a sample output sequence and subjecting it to statistical randomness tests
- Each test determines whether the sequence possesses a certain attribute that a truly random sequence would be likely to exhibit

3. Random Generators

3.3 Statistical Randomness Testing

- Test suites
  - Maurice George Kendall and Bernard Babington-Smith (1938)
  - Diehard tests (George Marsaglia, 1995)
  - TestU01 (Pierre L'Ecuyer and Richard Simard, 2007)
  - NIST (2010)
  - ...
- The NIST test suite comprises Maurer's universal statistical test
- The basic idea of this test is that it should not be possible to significantly compress (without loss of information) the output sequence of a random bit generator

3. Random Generators

3.4 Final Remarks

- Random generators are at the core of most systems that employ cryptographic techniques
- In practice, it is often required that random bit generators conform to a security level specified in FIPS PUB 140-2
- Hence, there is room for conformance testing as well as evaluation and certification
- From an application viewpoint, it is important to generate truly random bits (using a random generator) and use them as a seed for a PRG
- PRGs are further addressed in Chapter 7

Questions and Answers

Thank you for your attention