Contents

About De|G PRESS — VII

Preface — XVII

1 Introduction — 1
   1.1 Classes of Random Number Generators — 3
   1.2 Naming RNGs — 5
   1.3 Disambiguating RNG Types — 6
   1.4 Nonuniform RNGs — 7
   1.5 Noncryptographically Secure PRNG Algorithms — 8
   1.6 Cryptographically Secure PRNG Algorithms — 9
      1.6.1 Example of CSPRNG: The SP800-90A CTR DRBG — 11
      1.6.2 Attacking CSPRNGs — 12
   1.7 Controlled Defect RNGs — 14
   1.8 Noise Source Circuits — 15
   1.9 TRNG Extractor/Conditioning Algorithms — 16
   1.10 The Structure of Secure RNG Implementations — 17
      1.10.1 Point A, the Raw Data — 17
      1.10.2 Points D and E, the Health Status Feedback — 18
      1.10.3 Point B, the Seed Quality Data — 18
      1.10.4 Point C, the PRNG Output — 18
   1.11 Pool Extractor Structures — 19
   1.12 What Key to Use? — 20
   1.13 Multiple Input Extractors — 21

2 Entropy Sources — 23
   2.1 Ring Oscillator Entropy Sources — 23
      2.1.1 Ring Oscillator Entropy Source Problems — 25
   2.2 Metastable Phase Collapse Ring Oscillator Entropy Sources — 26
   2.3 Fast Digital TRNG Based on Metastable Ring Oscillator — 30
   2.4 Feedback Stabilized Metastable Latch Entropy Source — 31
   2.5 Infinite Noise Multiplier Entropy Source — 35
   2.6 Diode Breakdown Noise Entropy Source — 39

3 Entropy Extraction — 41
   3.1 The Simplest Extractor, the XOR Gate — 42
   3.2 A Simple Way of Improving the Distribution of Random Numbers that
      Have Known Missing Values Using XOR — 46
      3.2.1 Is This Efficient? — 51
3.2.2 Why This Might Matter: Two Real-World Examples with Very Different Results — 51

3.3 Debiasing Algorithms — 57
3.4 Von Neumann Debiaser Algorithm — 57
3.5 Yuval Peres Debiaser Algorithm — 63
3.6 Blum’s Method Debiaser Algorithm — 66
3.7 Cryptographic Entropy Extractors — 72
3.8 Pinkas Proof, or Why We Cannot Have Nice Things — 72
3.9 Seeded Entropy Extractors — 73
  3.9.1 CBC-MAC Entropy Extractor — 74
  3.9.2 CMAC Entropy Extractor — 77
  3.9.3 HMAC Entropy Extractor — 79
3.10 Multiple Input Entropy Extractors — 81
  3.10.1 Barak, Impagliazzo, Wigderson 3 Input Extractor — 82
  3.10.2 2-EXT 2-Input Extractor — 88

4 Cryptographically Secure Pseudorandom Number Generators — 95
4.1 SP800-90A — 95
  4.1.1 SP800-90A Hash DRBG — 96
  4.1.2 SP800-90A HMAC DRBG — 100
  4.1.3 SP800-90A CTR DRBG — 103
  4.1.4 Observations On the CTR DRBG — 108
4.2 ANSI X9-82 — 109
4.3 Salsa20 — 109
  4.3.1 Salsa20 Hash — 112
4.4 Cha Cha — 115
4.5 Blum Blum Shub — 117

5 Nondeterministic Random Number Generators — 119
5.1 The XOR Construction NRBG — 120
5.2 The Oversampling Construction NRBG — 120
5.3 Standards-Based NRBG Summary — 121

6 Statistically Uniform Noncryptographic PRNGs — 123
6.1 Linear Congruential Generator — 123
6.2 Multiply with Carry Uniform PRNG — 126
6.3 Xorshift Uniform PRNG — 129
6.4 Permuted Congruential Generator — 132
  6.4.1 PCG Naming — 133
  6.4.2 64-32-PCG-XSH-RR — 134
  6.4.3 64-32-PCG-XSH-RS — 135
  6.4.4 128-64-PCG-XSL-RR — 135
7 Gaussian or Normally Distributed PRNGs — 137
  7.1 Box–Muller Transform Normal Random Variate Algorithm — 137
  7.2 The Ziggurat Normal Random Variate Algorithm — 138
  7.3 The ZIGNOR Normal Random Variate Algorithm — 142

8 Testing Random Numbers — 145
  8.1 Known Answer Tests — 145
  8.2 Distinguishability Tests — 147
  8.3 PRNG Test Suites — 147
    8.3.1 Dieharder — 147
    8.3.2 NIST SP800-22 — 148
    8.3.3 SEMB GM/T 0005-2012 — 148
  8.4 Entropy Measurements — 148
  8.5 Shannon Entropy Estimation — 148
  8.6 Min Entropy Estimation — 149
  8.7 Model Equivalence Testing — 151
  8.8 Statistical Prerequisite Testing — 151
    8.8.1 Mean — 151
    8.8.2 Standard Deviation — 152
    8.8.3 The $\chi^2$ Test of Randomness — 154
    8.8.4 Serial Correlation Coefficient — 159
    8.8.5 Lag-$N$ Correlation — 162
    8.8.6 A Note on Bit Ordering — 165
    8.8.7 Efficient Computation of a Correlogram using FFTs — 166
  8.9 The Problem Distinguishing Entropy and Pseudorandomness — 169
  8.10 Statistical Tests of Uniformity — 170
  8.11 Results That are “Too Good” — 170
  8.12 Summary — 171

9 Online Random Number Testing — 173
  9.1 Tagging or Blocking? — 173
  9.2 How Much Data to Test? — 175
  9.3 What Should an Online Test Test for? — 175
  9.4 The FIPS 140-2 Continuous RNG Test — 176
  9.5 The SP800-90B Repetition Count Test — 179
  9.6 The SP800-90B Adaptive Proportion Test — 180
  9.7 Pattern Counting Health Test — 186
  9.8 Online Mean and Serial Correlation Test — 188
  9.9 Online Source Independence Test — 195
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>First, Do not Assume the Contrapositive</td>
<td>199</td>
</tr>
<tr>
<td>10.2</td>
<td>SP800-22rev1a Monobit Test</td>
<td>200</td>
</tr>
<tr>
<td>10.2.1</td>
<td>Application</td>
<td>200</td>
</tr>
<tr>
<td>10.2.2</td>
<td>Procedure</td>
<td>200</td>
</tr>
<tr>
<td>10.2.3</td>
<td>Monobit Test Python Implementation</td>
<td>200</td>
</tr>
<tr>
<td>10.3</td>
<td>SP800-22rev1a Frequency Test Within a Block</td>
<td>201</td>
</tr>
<tr>
<td>10.3.1</td>
<td>Application</td>
<td>201</td>
</tr>
<tr>
<td>10.3.2</td>
<td>Procedure</td>
<td>201</td>
</tr>
<tr>
<td>10.3.3</td>
<td>Frequency Test Within a Block Python Implementation</td>
<td>202</td>
</tr>
<tr>
<td>10.4</td>
<td>SP800-22rev1a Discrete Fourier Transform (DFT) Test</td>
<td>203</td>
</tr>
<tr>
<td>10.4.1</td>
<td>Application</td>
<td>203</td>
</tr>
<tr>
<td>10.4.2</td>
<td>Notes on the DFT Algorithm</td>
<td>204</td>
</tr>
<tr>
<td>10.4.3</td>
<td>Procedure</td>
<td>204</td>
</tr>
<tr>
<td>10.4.4</td>
<td>DFT Test Example Code</td>
<td>207</td>
</tr>
<tr>
<td>10.5</td>
<td>SP800-22rev1a Nonoverlapping Template Matching Test</td>
<td>208</td>
</tr>
<tr>
<td>10.5.1</td>
<td>Application</td>
<td>208</td>
</tr>
<tr>
<td>10.5.2</td>
<td>Procedure</td>
<td>209</td>
</tr>
<tr>
<td>10.5.3</td>
<td>Nonoverlapping Template Matching Test Example Code</td>
<td>210</td>
</tr>
<tr>
<td>10.6</td>
<td>Overlapping Template Matching Test</td>
<td>211</td>
</tr>
<tr>
<td>10.6.1</td>
<td>Application</td>
<td>213</td>
</tr>
<tr>
<td>10.6.2</td>
<td>Procedure</td>
<td>213</td>
</tr>
<tr>
<td>10.6.3</td>
<td>Overlapping Template Matching Test Example Code</td>
<td>215</td>
</tr>
<tr>
<td>10.7</td>
<td>SP800-22rev1a Longest Runs of Ones in a Block Test</td>
<td>217</td>
</tr>
<tr>
<td>10.7.1</td>
<td>Application</td>
<td>217</td>
</tr>
<tr>
<td>10.7.2</td>
<td>Procedure</td>
<td>217</td>
</tr>
<tr>
<td>10.7.3</td>
<td>Longest Runs of Ones in a Block Test Example Code</td>
<td>218</td>
</tr>
<tr>
<td>10.8</td>
<td>SP800-22rev1a Binary Matrix Rank Test</td>
<td>221</td>
</tr>
<tr>
<td>10.8.1</td>
<td>Application</td>
<td>221</td>
</tr>
<tr>
<td>10.8.2</td>
<td>Procedure</td>
<td>221</td>
</tr>
<tr>
<td>10.8.3</td>
<td>SP800-22rev1a Binary Matrix Rank Test Example Code</td>
<td>224</td>
</tr>
<tr>
<td>10.9</td>
<td>SP800-22rev1a Random Excursion Test</td>
<td>226</td>
</tr>
<tr>
<td>10.9.1</td>
<td>Application</td>
<td>226</td>
</tr>
<tr>
<td>10.9.2</td>
<td>Procedure</td>
<td>226</td>
</tr>
<tr>
<td>10.9.3</td>
<td>Random Excursion Test Example Code</td>
<td>228</td>
</tr>
<tr>
<td>10.10</td>
<td>SP800-22rev1a Random Excursion Variant Test</td>
<td>230</td>
</tr>
<tr>
<td>10.10.1</td>
<td>Application</td>
<td>231</td>
</tr>
<tr>
<td>10.10.2</td>
<td>Procedure</td>
<td>231</td>
</tr>
<tr>
<td>10.10.3</td>
<td>Random Excursion Variant Test Example Code</td>
<td>231</td>
</tr>
</tbody>
</table>
10.11 SP800-22rev1a Maurer’s Universal Statistical Test —— 233
  10.11.1 Application —— 233
  10.11.2 Procedure —— 233
  10.11.3 Maurer’s Universal Statistical Test Example Code —— 236

10.12 SP800-22rev1a Linear Complexity Test —— 237
  10.12.1 Application —— 238
  10.12.2 Procedure —— 238
  10.12.3 Linear Complexity Test Example Code —— 239

10.13 SP800-22rev1a Serial Test —— 241
  10.13.1 Application —— 243
  10.13.2 Procedure —— 243
  10.13.3 Serial Test Example Code —— 244

10.14 SP800-22rev1a Cumulative Sums Test —— 245
  10.14.1 Application —— 246
  10.14.2 Procedure —— 246
  10.14.3 Cumulative Sums Test Example Code —— 246

10.15 SP800-22rev1a Approximate Entropy Test —— 248
  10.15.1 Application —— 248
  10.15.2 Procedure —— 248
  10.15.3 Approximate Entropy Test Example Code —— 249

11 Software Tools —— 251
  11.1 hex2bin —— 251
  11.2 bin2hex —— 252
  11.3 cleanhex —— 253
  11.4 djenrandom —— 254
    11.4.1 Pure Model —— 257
    11.4.2 SUMS – Step Update Metastable Source Model —— 258
    11.4.3 Biased Model —— 259
    11.4.4 Correlated Model —— 259
    11.4.5 LCG Model —— 260
    11.4.6 PCG Model —— 261
    11.4.7 XorShift Model —— 261
    11.4.8 Normal Model —— 262
    11.4.9 File Model —— 263
  11.5 quickrdrand —— 265
    11.5.1 Quickrdrand Output Formats —— 266
    11.5.2 Quickrdrand data output size —— 267
  11.6 ent —— 267
    11.6.1 Ent Output Formatting —— 267
    11.6.2 Ent Symbol Size Options —— 268
    11.6.3 Ent Occurrence Counts —— 268
11.6.4 Ent Statistical Metrics — 269
11.6.5 Uses of ent — 270
11.7 djent — 272
11.7.1 Parseable Filenames — 273
11.7.2 Measuring Lag-N Correlation Coefficient — 274
11.7.3 Changing Symbol Size — 274
11.8 Dieharder — 275
11.8.1 Running Dieharder Against Internal Generators — 277
11.8.2 Running Dieharder Against Stdin — 279
11.8.3 Running Dieharder Against File Input — 280
11.9 NIST STS 2.1.2 — 281
11.10 NIST SP800-90B Entropy Assessment Suite — 284

12 RdRand and RdSeed Instructions in x86 CPUs — 289
12.1 Intel DRNG — 289
12.1.1 RdRand Instruction — 291
12.1.2 RdSeed — 292
12.1.3 AMD Support for RdRand and RdSeed — 293

13 Accessing RNGs from Software — 295
13.1 MacOS and BSD getentropy() — 295
13.2 Linux getrand() Syscall — 296
13.3 /dev/random and /dev/urandom — 297
13.4 RdRand and RdSeed — 299
13.5 The Python Random Library — 309
13.5.1 Python Cryptographically Secure Random Numbers — 312
13.6 The Windows CryptGenRand() API — 314

14 Floating-Point Random Numbers — 317
14.1 The Floating-Point Number Distribution — 319
14.2 The Fixed-Exponent Method — 321
14.3 Exponent Distribution Compensation — 323
14.4 Diving by the Largest Random Integer — 325

15 Making a Uniform Random Number Between Nonpower of Two Bounds — 329
15.1 Rejection Method for Nonpower of 2 Bounds — 331
15.2 Large Number, Small Modulus Method for Nonpower of 2 Bounds — 332
15.3 Floating-Point Method for Nonpower of 2 Bounds — 334

16 Generating Random Prime Numbers — 337
16.1 Safe and Strong Primes — 343
17 Additive Distributions — 345
  17.1 Dice — 345
  17.2 Unfair Dice — 346
  17.3 How the Normal Distribution Forms — 346

18 Probability Distributions — 349
  18.1 Names of Probability Distributions — 349
  18.2 Properties of Distributions — 351
  18.3 Fair Dice and Uniform Random Numbers — 352
  18.4 Normal/Gaussian Distributions — 354
    18.4.1 Normal PDFs and CDFs in Programming Languages — 357
    18.4.2 Gnuplot Normal Distribution Functions — 358
    18.4.3 Python Normal Distribution Functions — 359
    18.4.4 R Normal Distribution Functions — 361
    18.4.5 C Normal Distribution Functions — 362
    18.4.6 Excel Normal Distribution Functions — 366
  18.5 Random Walks and Degrees of Freedom — 366
    18.5.1 Expected Final Positions for Random Walks — 371
  18.6 Poisson Point Processes — 372
  18.7 The Binomial Distribution — 374
    18.7.1 The Binomial PMF — 374
    18.7.2 The Binomial Discrete CDF — 378
    18.7.3 The Binomial Quantile Function — 379

19 Quantifying Entropy — 383
  19.1 Rényi Entropy — 383
    19.1.1 Derivation of Shannon Entropy from Rényi Entropy — 384
    19.1.2 Derivation of Min Entropy from Rényi Entropy — 386
  19.2 Min-Entropy of Biased Binary Sources — 387
  19.3 Min-Entropy of Serially Correlated Binary Sources — 387
  19.4 Distance From Uniform — 390

20 Random Methods to Generate π — 393
  20.1 Random Method for Computing π — 393
  20.2 Another Random Method for Computing π — 395

Appendix A Adaptive Proportion Test Cutoff Tables — 397

Appendix B High-Precision Incomplete Beta Function Implementation — 403

Appendix C Incomplete Gamma Function Implementation — 409
Appendix D  Software Tool Sources —— 415

Appendix E  Listing Reference —— 417

Bibliography —— 421

Index —— 423