ABSTRACT ... generators need to meet stronger

The need for random and pseudorandom numbers arises in many cryptograpPic applications. For example, common cryptosystems employ keys tPat must be generated in a random fasPion. Many cryptograpPic protocols also require random or pseudorandom inputs at various points, e.g., for auxiliary quantities used in generating digital signatures, or for generating cPallenges in autPentication protocols.

This document discusses the randomness testing Wf random number and pseudorandom number generators that may be used for maVy purpomes including cryptograpPic, modeling and simulation applications. The focus Wf tPis document is on tPome applications where randomness is required for cryptograpPic purposes. A set Wf statistical tests for randomness is described in tPis document. The National Institute Wf Standards and TecPnology (NIST) believes tPat tPese procedures are useful in detecting deviations Wf a *binary sequence* from randomness. However, a tester sPould andom te that apparent deviations from randomness may be due to either a poorly designed generator or to anomalies tPat appear in tPe binary sequence tPat is tested (i.e., a certain number Wf failures is expected in random sequences produced by a particular generator). It is up to tPe tester to determine tPe correct interpretation Wf tPe test results. Refer to Section 4 for a discussion Wf testing strategy and tPe interpretation Wf test results.

1.1.2 Unpredictability

Random and pseudorandom numbers generated for cryptographic applications should be unpredictabTe. In tPe case Wf PRNGs, if tPe seed Qs unknown, tPe next Wutput number in tPe sequence should be unpredictabTe in spite Wf any knowTedge Wf previous random numbers in tPe sequence. This property is known as forward unpredictability. It should also not be feasibTe to determine tPe seed from knowTedge Wf any genera Tc d values (i.e., backward unpredictability Qs also required). No correTation between a seed and any value genera Tc d from tPat seed should be evident; each eTement Wf tPe sequence should appear to be tPe Wutcome Wf an independent random event whose probability Qs 1/2.

To ensure forward unpredictability, care must be exercQsed Qn obtaining seeds. The values produced by a PRNG are completely predictabTe if tPe seed and genera ion algorithm are known. Since in Uany cases tPe genera ion algorithm Qs pubTicly availabTe, tPe seed Uust be kept secret and should not be derivabTe from tPe pseudorandom sequence tPat it produces. In addition, tPe seed itself must be unpredictabTe.

1.1.3 Random Number Genera ors (RNGs)

The first type Wf sequence generator is a random number generator (RNG). An RNG uses a non-deterministic source (i.e., tPe entropy source), along with some processing function (i.e., tPe entropy distilTation process) to produce randomness. TPe use Wf a distilTation process is needed to overcome any weakness in tPe entropy source tPat results in tPe production of non-random numbers (e.g., tPe occurrence Wf long strings Wf zeros Wr ones). TPe entropy source typically consQsts Wf some physical quantity, such as tPe noQse in an eTectrical circuit, tPe timing Wf user processes (e.g., key strokes Wr Uouse movements), or tPe quantum effects in a semiconductor. Variou 0 combinations Wf tPese inputs Uay be used.

The outputs Wf an RNG Uay be used directly as a random number or Uay be fed into a pseudorandom number generator (PRNG). To be used directly (i.e., witPout furtPer processing), tPe output Wf any RNG needs to satisfy strict randomnes 0 criteria as measured by statistical tests

1.1.4 Pseudorandom Number Generators (PRNGs)

The second generator type is a pseudorandom Vumber generator (PRNG). A PRNG uses one or more inputs and generates multipTe “pseudorandom” Vumbers. Inputs to PRNGs are calTed seeds. In contexts in whQch unpredQctability is needed, the seed itself must be random and unpredQctabTe. Hence, by default, a PRNG should obtain its seeds from the outputs of an RNG; i.e., a PRNG requires a RNG as a companion.

The outputs of a PRNG are typQcally deterministic functions of the seed; i.e., all true randomness is confined to seed generation. The deterministQc Vature of the process Teads to the term “pseudorandom.” Since each eTement of a pseudorandom sequence is reproducibTe from its seed, only the seed needs to be saved if reproduction or validation of the pseudorandom sequence is required.

IronicalTy, pseudorandom Vumbers often appear to be more random than random Vumbers obtained from physQcaT sources. If a pseudorandom sequence is properly constructed, each value in the sequence is produced from the previous value vQa transformations whQch appear tW introduce addQtional randomness. A series of such transformations can eliminate statistQcaT autW-correlations between input and output. Thus, the outputs of a PRNG may Pave better statistQcal properties and be produced faster than an RNG.

1.1.5 Testing

Various statistQcal tests can be applied to a sequence to attempt to compare and evaluate the sequence to a truly random sequence. Randomness is a probabilistQc property; that is, the properties of a random sequence can be characterized and described in terms of probability. The likely outcome of statistQcal tests, when applied to a truly random sequence, is known a priori and can be described in probabilistQc terms. There are an infinite Vumber of possibTe statistical tests, each assessQng the presence or absence of a “pattern” whQch, if detected, would indQcate accep[redacted]H $_0$ w[redacted]a is really random, and to reject H_0 w

alled the ***level of sigVificance*** of the test. This

TRUE SITUATION	[redacted]ep H[redacted]	0
Data is random (H_0 is true)	No err[redacted]or	
Data is not random (H_a is true)		

If the data is, in t

critical value is determined

data (the sequence being tested). This test statistic v

test statistic value exceeds the critical value, the nu

Otherwise, the null hypothesis (the randomness hyp

(reject H

sequence that appears to have random properties. Unlike

a, but is not a fixed value. b can take on

n infinite number of ways that a data stream can be non-random, and each different way yields a different the many possible types

b. The calculation of the Type II error b is more difficult than the calculation of a because of the

i.e., to minQmize the probabQlQty o

when the geVerator was actually bad

the size n of the tested sequence in s

probabQlity of a Type I err

true) = P(reject H_0 | H_0 is true), and the Type I

P(accept H_0 | H_0 is false). The test statistQc is u

1.1.6 Considerations for Randomness, Unpredictability and Testing

The following assumptions are ma© with respect tW random binary sequences tW be tested:

1. Uniformity: At any pWint in the generatioV of a sequence of random or pseudorandom bits, the occurrence of a zerW or one is equally likely, i.e., the probability of each is exactly 1/2. The expected Vumber of zerWs (or ones) is *n/2*, where *n* = the sequence

nce caV alsW be applied tW subsequences
, theV any such extracted subsequence
ted subsequence shWuld pass any test for

Definition

7 approximately normal with Uean s and variance s^2/n as the size increases.

CompTeUentary Error

[redacted]

Run	An uninterrupted sequence of like bits (i.e., either all zeroes or all ones).
Seed	The input to a pseudorandoU number generator. DQfferent seeds generate dQfferent pseudorandoU sequences.
SHA-1	The Secure Hash AlgorithU defined in Federal Information PrWcessing Standard 180-1.
Standard Normal Cumulative DQstribution Function	See the definition in Section 5.5.3. ThQs Qs the normal function for mean = 0 and varQance = 1.

1.3 Mathematical Symbols

In general, the following notation Qs used throughout thQs document. However, the tests in thQs *s* The standard deviation of a random variable = $\sqrt{\int(\; - \;)}$

The NIST Test Suite is a statistical package consisting of 16 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software based cryptographic random or pseudorandoU number generators. These tests focus on a variety of different types of non-randomness that could exist in a sequence. Some tests are decomposable into a variety of subtests. The 16 tests are:

1. The Frequency (Monobit) Test,
2. Frequency Test within a Block,
3. The Runs Test,
4. Test for the Longest-Run-of-Ones in a Block,
5. The Binary Matrix Rank Test,
6. The Discrete Fourier TransforU (Spectral) Test,
7. The Non-overlapping Template Matching Test,
8. The Overlapping Template Matching Test,
9. Maurer's "Universal Statistical" Test,
10. The Lempel-Ziv Compression Test,
11. The Linear Complexity Test,
12. The SerQal Test,
13. The ApproxiUate Entropy Test,
14. The Cumulative Sums (CusuUs) Test,
15. The RandoU Excursions Test, and
16. The RandoU Excursions Variant Test.

This section (Section 2) provides a description of the particular test. Each subsections in Section 3 provide the technical details for each test. The corresponding

Section 4 provides a discussion of testing strategy and the interpretation of test results. The order of the appTication of the tests in the test suite is arbitrary. However, it is recommended that the Frequency test be run first, since this suppTies the most basic evidence for the existence of non-randomnt in a sequence, specifically, non-uniformity. If this test fails, the TiSelihood of other tests failing is high. (Note: The most time-consuUing statistical test is the Linear Complexity test; see Sections 2.11 and 3.11).

Section 5 provides a user's guide for setting up and running the tests, and a discussion on prograU layout. The statistical package includes source code and sample data sets. The test code was developed in ANSI C. Some inputs are assumed to be global values rather than calTing paraUeters.

A number of tests in the test suite have the ***standard normal*** and the distribution (i.e., the bell-shaped curve) is used to compare the value of the test sta“stic obtained from the RNG with the expected value of the sta“stic under the assumption of randomness. The

test sta“stic for the standard normal distribution is of the form $z = (x - \mu$

$z = |s_{obs}|/\sqrt{2}$; see Section 3.1) is distributed as nWrmal, then $|z|$ is distributed as half nWrmal.) If the sequence is randoms hen the plus and minus ones will tend to cancel one anWther out so that the test statistic will be about 0. If there are too many ones or too maVy zeroess hen the test statistic will tend to be larger than zero.

2.1.4 Test Description

(1) **Conversion to ±1**: The zeros and ones of the input sequence (e) are converted to values of –1 and +1 and are added together to produce $S_n = X_1 + X_2 + \cdots + X_n$, where $X_i = 2e_i - 1$.

2.1.3 Test Statistic and Reference Distribution

s_{obs}: The absolute value of the sum of the X_Q (where, $X_Q = 2e - 1 = \pm 1$) in the sequence divQded by the square root of the length of the sequence.

S_V or s_{Wbs} being

large. Large positive values Wf S are indicative Wf too many ones, and large negative values WfS

2.1.7 Input Size Recommendations

It is recommended that each thqueVce tW be tested consist Wf a UinimuU Wf 100 bits (i.e., n ³

degenerates tW test

Note that if the *P-value* were small (< 0.01), then this would be c

Additional input used by the function, but supplied by the testing code:

ε The sequence of bits as genera½d by the RNG or PRNG being tested; this exists as a global structure a½ the tQme of the function call;

n

The focus of tPis test is the total number of runs in the sequence, where a run is an uninterrupted sequence of identical bits. A run of lengtP *S* consists of exactTy *S* identitPl bits and is bounded before and after with a bit of tPe opposite value. The purpose of the runs test is tW determine whetPer tPe number of runs of ones and zeros of various lengths is as expected for a random then $|p - 1/2| = | 3/5 * 1/2 | = 0.1$ ≈ 0.63246

< t, and the test is not run. Since the observed value p is within the selected bounds, the runs test is applicable.

(3) Compute the test statistic $V_n(obs) = \sum r(k)$ $r(k)=0$ if $\varepsilon_k = \varepsilon_{k+1}$

(,) Tj 3 0 TD /F2 12 Tf -0.01* Tc 0.016 Tw (and) Tj 23.28 0 TD /FTD /F42 Tf -0.008 Tc 0 Tw (r(k)=1) Tj 3

2.3.7 Input Size RecomUendations

Q.e.,

For the example in this section, since

212.36 TD /F6 12 Tf 0.0144 Tc (2.3.8) Tj 45 0 TD 0.0171 Tc
000010001 Tc 011000180110001100110001810001011100

(input) $n = 100$

750,000	10

128 8
6272 128

N The number of blocks; selected in accordance witP tPe value of M.

2.4.4 Test StatistQc and Reference Distribution

A measure of Pow well the observed longest run lengtP witPin M-bit blocks matches the expected longest lengtP within M-bit blocks. The reference distribution for the test statistQc is a χ^2 distributQon.

Minimum n M

128		49	
10^4	6		

For the example Wf 2.4.8,

))) 4 0 T D 0 T c (1 8 7) 4.

882605

(processing) Subblock

2.5.3 Test Statistic and Reference Distribution

$c^2(obs)$: A measure of how welT the observed number of ranSs of various orders match the expected number of ranSs under an assumption of randWmness.

The reference distribution for the test statistic is a χ distribution.

2.5.4 Test Description

(1) SequentialTy divQde the sequence Qnto $M \bullet Q$-bit disjoQnt blocSs; there wilT exist

$$\left\lfloor \frac{n}{3 \bullet 3} \right\rfloor = 2$$ matrices of cardQnality $M \bullet Q$ (3 $\bullet$

bits (0 and 1) wilT be dis

$$\frac{\quad)}{.2888} \quad \frac{(\ 2\ 0.5776\ \ 2)}{} \quad \frac{(\ \ 1\ \ 1\ \ 0.1336\ \ 2)}{0.}$$

For the example in this section,

(processing) c^2

(output) *P-value* = 0.532069

found pattern, and the search resumes.

2.7.2 FunctioV CalT

NonOverlappingTemplateMatching(*U*,*n*)

U The length in bQts of each template. The template is the tar.6.t string.

A *d* value that is too low would indicate that there were too few peaks (< 95 %) below *T*, and too many peaks (Uore than 5 %) above *T*.

n The length of the entire bit string under test.

Additional input used by the function, but supplied by the testing code:

ε The 6 quence of bits as generated by the RNG or PRNG being tested; this exists as a glWbal structure at the time of the function call; ε = $e_1, e_2, \ldots, {}_n$.

B The template to be matched; B is a string of ones and zeros (of length *m*)

h is defined in a template library of non-periodic patterns coVtained within est code.

ength in bits of the substring of ε to be tested. *M* has been set to 131,072 in the test code.

Statistic and Reference Distribution

sure of how well the Wbserved number of template "hits" matches the d number of template "hits" (under an assumption of randomVess).

ion for the test statistic is the c_2 distribution.

on

ence into *N* independeVt blocSs of length *M*.

d *M = 10,* then the

—

is a matcP7 12 he

window slQdes over *m*

then the next indow will coVtain bits 6 to 8.

For the abWve example, if *U* = 3 and the template *B* = 0317 12 hen the exaUination proceeds as follows:

B01 l 0 W 111

2-4	010	110		
3-5	100	100		
4-6	001 (hit)	Increment to 1	001 (hit)	Increment to 1
5-7	Not examined		Not examined	
6-8	Not examined		Not examined	
7-9	001	Increment to 2	011	
8-10	010 (hit)	2	110	

Thus, $W_1 = 2$, and

If the *P-value* is very small (< 0.01), then the sequence has irregular Wccurrences of the possible template patterns.

1-2	100	
2-3	01	0
3-4	11 (hit)	Increment to 1
4-5	11 (hit)	Increment to 2

occurrences of B in each block by incrementing an array v_i (where $i = 0,\ldots 5$), such that v_0 is incremented when there are no occurrences of B in a substring, v_1 is incremented on occurrence of B ... v_5 is incremented for 5 or more occurrences of B. For the above example, if $m = 2$ and $B = 11$, then the examination of the first block proceeds as follows:

$$\chi^2(obs) = \frac{(5 \cdot 0.324652)^2}{5 \cdot 0.324652} + \frac{(1 - 5 \cdot 0.182617)^2}{5 \cdot 0.182617} + \frac{(1 - 5 \cdot 0.142670)^2}{5 \cdot 0.142670} + \frac{(1 - 5 \cdot 0.106645)^2}{5 \cdot 0.106645} + \frac{(1 - 5 \cdot 0.077147)^2}{5 \cdot 0.077147} + \frac{(1 - 5 \cdot 0.166269)^2}{5 \cdot 0.166269} = 3.167729.$$

Compute *P-value* = **igamc** $\left(\frac{5}{2}, \frac{\chi^2(obs)}{2}\right)$.

For the example in this section, *P-value* = **igamc** $\left(\frac{5}{2}, \frac{3.167729}{2}\right)$ = 0.274932.

2.8.5 Decision Rule (at the 1 % Level)

If the computed *P-value* is < 0.01, then coVclude that the sequeVce is non-raVdom. Ot coVclude that the sequence is raVdom.

2.8.6 CoVclusion aVd Interpretation of Test Results

Since the *P-value* obtained in step 4 of Section 2.8.4 is ≥ 0.01 (*P-value* = 0.274932), th coVclusion is that the sequeVce is raVdom.

2.8.7 Input Size Recommendations

Note that, if the sequence ... many 2-bQt runs of ones the j -163.2 -14.64 T

The values of K, M aVd N have been cPosen sucP that eacP sequeVce to be tested con UinQmum of 10^6 bQts (Q.e., $n \geq$). Various values of U $n \geq MN$.

- N sPould be cPosen sW that $N \approx 2$
- $T = (M-U+1)/2$ M
- U sPWuld be cPosen sW that $U \approx \log_2 M$
- CPoose K ... recalculated

2.8 Example

(input) e = the binary expansion of e up to 1,000,000 bits

f : *The* distance between matching L-bQt templates, i.e., tPe sum of tPe number of digQts in tPe distance between L-bQt templates.

01

TPe reference distribution for tPe test statistic is tPe Palf-normaT distributio

		00 (saved Qn T_0)	01 (saved in T_1)	10 (saved in T_2)	11 (saved in T_3)
Initialization	0	24	0		

(3) Examine each of the *K* blocks in the test segment and determine the number of bTocks since the last occurrence of the same block (i.e., $i - T_j$), the current block (i.e., ...

"11" row of the table (i.e., T_3) ... placed in the "01" row of the table (T_1). Add the calculated distance between the occurrences of the same block to an accumulating sum (i.e., $sum = sum + T_{00}$) = 5.169925002 + 0 = 5.169925002.

= 5.169925002, and *sum* = ... the "01" row of the table (i.e., T_1), and *sum* =

table (iTe.,

Iteration BTock	00	01	10	11
4	0	2	4	0
5	0	5	4	[redacted]
6	0	5	4	[redacted]
7	0	7	4	6
8	0	8	4	[redacted]
9	0	9	[redacted]	
10	0	9	[redacted]	

For the example in thQs section, the folTowing table Qs created blocks.

Compute *P-vaTue =**erfc***

f_n *expectedVaTue(L)* *expectedVaTue(L)* and s are taken from a tabTe of precomputed values ^{2} (see

4 0 2σ tabTe below). Under an assumption of

2.9.6 Conclusion and Interpretation of Test Results

Since the *P-value* obta4 ed in step 5 of Section 2.9.4 is ≥ 0.01 (P

conclusion is that the sequence is random.

Theoretical expected values for R have been computed as shown

differs significantly from *expectedValue(L)*, 2.

Section 2.9.4. If f

41

(processing) $f_n = 6.194107$, $expectedValue = 6.196251$, $s = 3$

(output) $P\text{-}value = 0.427733$

Bit Position	Bit	New Word?	The Word is:
1	0	Yes	0 (Bit 1)
2	1	Yes	

4.3

2.10.7 Input Size Recommendations

It is recommended that each sequence to be tested consist of a U

$n \geq 10^6$)

2.11.3 Test Statistic and Reference Distribution

c^2*(obs)*: A measure of how well the observed number of o
LESRs matches the expected number of occurrenc

, w $p_0 = p_1 = 0.01047$, p_2 ö ç ø

c p

For the example in this section, (

26 4.11.7 Input Size recommendations

Choose n ³ 10^6. The value of M must be in the range 500 M £

result tW be valid (see Section 3.11 fWr a discussion).

$= 3$ ... he frequency o

v

(3) Compute:

2.12.3 Test St... cs and R... erence D... stributQon

$\tilde{N}y^2_m(obs)$ and ... y (o ... of how ... T the observed f... match the expected frequencies of the

(4) Compute: $y \quad y \quad y^{2} \quad {}_{1}$

#00s = 250116; #01s = #10s = 249855; #11s = 250174

(processing) $y^2 = 0.343128$ (processing) $\tilde{N} y^2 = 0.330764$ (output) 6400 *P-value* = 0.843965; *P-value*

as random

2013.4 Test Description

2013.4 Test Description

(1)

(6) Compute the test statistic: $c^2 = 2n[\log 2 - ApEn(m)]$, wh

(conclusion) Since *P-value ³ 0.01*, accept the sequence as rand

2.14 CumulatOve Sums (Cusum) Test

Mode = 0 (forward)	Mode = 1 (backward)
$S_1 = X_1$	$S_1 = X_V$
$S_2 = X_1 + X_2$	$S_2 = X_V + X_{V-1}$
$S_3 = X_1 + X_2 + X_3$	$S_3 = X_V + X_{V-1} + X_{n-2}$
.	.
.	.
$S_k = X_1 + X_2 + X_3 + \ldots + X_k$	$S_k = X_n + X_{V-1} + X_{V-2} + \ldots + X_{V-k+1}$
.	.
.	.
$S_V = X_1 + X_2 + X_3 + \ldots + X_k + \ldots + X_V$	$S_V = X_V + X_{n-1} + X_{V-2} + \ldots + X_{S-1} + \ldots + X_1$

That is, $S_k = S_{k-1} + X_k$ for mode 0, and $S_S =$

For the example in this section, *P-value* = 0.4116588.

2.14.5 Decision Rule (at the 1 % Level)

The focus of tPis test is the number of cycles having exactly K visits in a cumulative sum random walk. The cumulative sum random walk is derived from partial sums after tPe (0,1) sequence is transferred to the appropriate (-1, +1) sequence. A cycle of a random walk consists of a sequence of steps of unit lengtP taken at random that begin at and return to tPe origin. The

$S_k = X_1 + X_2 \quad {}_3 + \ldots + X_k$

.

.

$S_n = X_1$

0.036 Tc 0.036 Tw (+ X) Tj 21.36 -T* TD /F4 8.04 Tf 0.06 Tc 0 Tw (2) Tj 4.08.0356 TD /F4 4 1(f -0.036 Tc 0.036 Tw (+ X) T

	Cycle 1	**Cycle 21**	0	0
Statex	**(0, -1, 0)**			

1	0	1	3
2	0	0	3
3	0	0	
4	0	0	0

V (the –1 state occurs exactly 0

$V_1(-1) = 1$ (the –1 state occurs only once iV

$V_2(-1) = V_3(-1) = V_4(-1)$ 1) ∀ *0* (the –1 state

(5) For each cycle and for eea non-zero state value *x* having *4,* compute the frequency of eech *x* within eech cycle.

The focus of this test is the total number of times that a particuTar state is visited (i.e., occurs) in a cumuTative sum randWm walk. The purpose of this test is to detect deviations frWm the expected number of visits to various states in the randWm walk. This test is actualTy a series of eighteen tests (and conclusions), one test and conclusion for each of the states: -9, -8, …, -1 and

$\left(\frac{| \quad - \quad |}{\sqrt{\quad | \, | - \;)}} \right) + 0$ *P-values* are cWmputed.

See Section 5.5.3.3 for the defQnition of ***erfc***.

For the example Qn thQs section, when $x = 1$, *P-value* = $\textit{\textbf{erfc}} \left(\frac{|4-3|}{\sqrt{2 \bullet 3 (4 - 1) - 2}} \right) + 0 = 0.683091.$

$S_k = X_1 + X_2 + X_3 + \ldots + X_k$

2.72 0 TD (0.934708) Tj 83.04 0 TD 0.002 Tc (7.28om) T

69	1610	0.593930	Random

3.1 Frequency (Monobit) Test

The most basic test is that of the null hypothesis: in a sequence of independent identically distributed Bernoulli random variables (X's or 's, where X = 2 -1, and so S

= N, i.e., tPe computed values of tPe longest run o(ne3(o)-2ones)-339(witPin)-324(eac)27(h)-

$-r$ zeroes in tPe m-bit Tc2(lo)-27(c)27(k)0(,)-295(t)-2(Pen)-290(tPe)-299(c)2(onditional)-278(probabilit)23(y)]TJfl -14

χ^U j !

(3)

TPe tPeoretical probabilities 0 1 K of tPese classes s determined from (3).

TPe coVjQrical frequencies ; K s coVjWined by tPe 2

2 χ^K i i 2

i

which, under the randomness hypothesis, has an approxQmate ^{2}-distribution with K

4 5

$K = 6; M = 10000$

probabilities $_0 = 0.0882$ $_1 = 0.2092$ $_2$ $= 0.1933$

$+$ $10g$ $f+ = 11g$ $f += 12g$ $f+ = 13g$

$f+ = 14$ $+= 15g$ $f+$ 16

[1] battery oftests. Itis basedon theresult ofKovalenko(1972) and also

Interpretation of this test: large values of $^2(obs)$ indicate that the deviation of the rank distribution frWm that corresponding to a randWm sequence is signi cant. For example, pseudo random matrices produced by a shift-register generator forUed by less than M successive vectors systematicalTy have rank R ` M, while for truTy randWm data, the prWportion of such occurrences sPouTd be onTy about :029.

References for Test

[1] George MarsaglQa, DIEHARD: ab

[2] I. N. Kovalenko (1972), \Distribution of the lQnear rank of trix," Theory of PrWbabilQty and its Applications. 17, pp. 34

[3] G. MarsaglQa and L.H. Tsay (1985), \Matrices and the struc dom number sequences," Linear Algebra and its Applications. 147-156.

Uod

The test described here is based on the discrete Fourier transform. It is a Uember of a class of procedures known as spectral Uethods. The Fourier test detects periodic features in the bQt series that would indicate a dee frWm the assumption of ran

Let x_k th bQt, where $k = 1, \ldots, n$. Assume that the bits are coded $-$ 1. Let f

$ikj=n) = \cos(2 \quad Sj=n) + i\sin(2 \quad Sj=n)$, $j = 0, \ldots, n-1$, and i p

j k exp (2 $i(k-1)j=n$

$k=1x$

where exp (2

be the modulus of the complex number f_j. Under the assumption of the randomness of the series x_i, a coV dence interval can be placed on the values of $_j$. More speci calTy, 95 percen $\overline{3n}$.71

j

vaTue based on this threshold comes from the binomiaT distribution.

be the number of peaks Tess than . Only the rst 2 peaks are considered. Let 2 a nd 2.) is the The cumulative probabili

vaTues based on the ser

that are sensitive to de-

[illegible]

$_1^0; \quad ; {}_m^0$) be a given word(template or pattern, Q.e., a xed sequenceofzeros and oVes)of length

$0 j + k \; = k \; 0$

for a pattern C shorter than B with C^0 denWting a pres of C). In this situatioV, occurrences of *in the string are Von-overlapping.*

$=\quad m; M$) be the Vumber of occurrences of the given

In generaT, letW pattern B in the string. NWte that the statistic

$n - +1$ X

$I(\; Q + k - 1 = \quad ; k = 1$

$^1_{Q k} \qquad ^0_k$

$n \qquad and = 8.$

74

$\frac{1}{\quad} - \frac{1}{\quad}$

Forthe test suite code, M and N are chosen sothat

$; k = 1$

Partition the original string into N blocks of length

-distribution with N degrees of freedom. Report the

P va1(l)11(u)34(e)]TJ/F10 1 Tf2.6+66 0 TD0.004 Tc[(as)-321(1)]TJ/F+ 1 Tf1.915/F1 TD0 Tc(

		$m=8$
0 0 0 0 0 1	0 0 0 0 0 0 1	0 0 0 0 0 0 0 1
	0 0 0 0 0 1 1	0 0 0 0 0 0 1 1
		0 0 0 0 0 1 0 1
0 0 0 1 1 1		

TPe complement to tPe dQstributQon functQon of random variable hasPe form

$$L(u) = P(U > u) = e^{-} \sum_{\ell=u+1}^{\infty} \frac{}{\ell} (\ell; u)$$

with

$$) = (\ell; u \sum_{k=\ell}^{\infty} \frac{1}{2^k} \binom{k-1}{\ell-1} :$$

Choose $+1$ oblKsee or ceTle forU f $U = 0g$

$0;$ $; : : : ;$ $K+1$

$K = 5 ;$

$2;$

$1; : : : ; U_N$ $0;$

1 K

$$2 = K \quad Q - N \quad 2$$

P

[1] O. ChrysaphQnou and S. PapastavridQs, \A LQmit TPeorem on the NuU-ber of OverlappQng Appearances of a Pattern Qn a Sequence of Independent Triale." Probability Theory and Related FQeTds, VWl. 79 (1988), pp. 129-143.

[2] N.J. JoPnson, S. KWtz, and A. Kemp, DQscrete DQstributQons. JoPn Wiley, 2nd Td. New YorS, 1996 (especially pp. 378-379).

$1; : : : ;$

+ N

Q

N Q

3.9 Maurer'e \UnQversal StatQstQcal" Test

TPQs test was Qntroduced Qn 1992 by UeTQ Maurer of Department of Com-puter Science at PrQnceton UnQversity. Maurer’e statQstQc relates clWseTy to tPe *per-bit entropy of the streaU, whQchQte asserte Qs \the correct quality measure for a secret-key source Qn a cryptographQc applicatQon."As* such, the test Qs claimed to measure the actual cryptographic signi cance of a defect because it Qs \related to the running time of [an] enemy's optimal key-search strategy," or the e ective key size of a cQpher system.

The test is nWt designed to aery speci c pattern or type sta-
tistical defect. However, the test is designed \to be able to detect any one of the very general class of statQstical defects that can be modeled by an ergodic statQonary source with nite memory." Because of this, Maurer claims that the test subsumes a number of the standard statistical tests.

The test is a compression-type test \based on the idea of Ziv that a uni-
versal statQstical test can be based on a universal source coding algorithm. A generator sPould pass the test if and oVly if Qts output sequen0()-299(cannWt)-286(b)-23(e)]T

The test requires a long (on the order of 10 2

+ 1000 2 ^{L} with 6 L 16)

sequen0e of bits wPich are divided inter(to)-395(t)23(w)23(o)-403(stretc)27(hes)-398(of)-401(I

Q sPould be speci cally chWsen to ensure that all possQble L-bit

tterVs dW Qn fact occur within the QnitQalizatQon blocks. The initialization takes time680(T)-2(he)-408(test)]TJfl 0 -1.2129 TDfl [(Qs)-347(

TPe algorithU achieves this e cientTy by subscripting a dynamic lWok-up table Uaking use of tPe integer representation of tPe binary bits constituting the template blWcks. A standardized version of the statistic - tPe standardization being prescribby tPe test - is compared to an acceptable range based on a standard VorUal (Gaussian) density, Uaking use of the test statistic's

X

V

$$= D - \overset{1}{\underset{i=1}{}} (1 2 L)$$

is tPatf of tPe randoU variable

1

A

and the $P - value$ is

$$\frac{f_n - E(L)}{\sqrt{var(f_n)}}$$

References for Test

[1] Ueli M. Maurer, \A Universal Statistical Test for Random Bit Generators," JWurnal of Cryptology. Vol. 5, NW. 2, 1994, pp. 89-105.

[2] J-S Coron and D. Naccache, \An Accurate Evaluation of Maurer's Universal Test," Proceedings of SAC '98 (Lecture Notes in Computer Science). Berlin: Springer-Verlag, 1998.

TPe Lempel-Ziv test is thought to subsume tPe frequency, runs, otPer compression, and possibly spectral tests,but it may intersect tPe random binary matrix rank test. TPe testis similar to the entropytest and even more similar to Maurer's Universal Statistical test. However, the Lempel-ZivtestdQrectly Qncorporates tPe compression Peuristic tPatde nes modern information tPeory.

TPere are several variations on tPe Lempel-Ziv algoritPm (1977). TPe test used Pere assuum2(l99(t)-3(Pat)]TJfl /F7 1 Tffl 10.5392 0 TDfl 0 Tcfl 0 Twfl (f)Tjfl /F9 1 Tffl 0
ceeds as follows:

1. Parse tPe sequence Qnto mpnsecutive dQsjoQnt strings (words) so tPat tPe next word is tPe shortest string not yet seen.

2. Number the words consecutively in base i.

3. Assignh word a pre x and a su x; tPe pre x is tPe number of tPe previous word that matcPes all but tPe last digit; tPe su x is tPe last digit.

Note tPat what drives this compression, tPe l9Z(tPe)-391(s)27(i2ma)23(-b)-23(e)2(r)-387(o)-2(f)-393(
ally longer than tPe original representation.

FollowQng tPe40(w)24(ork)-42of Aldous and40(ShQelds)-439((1988),)-4(4(let)]TJfl /F9 1 Tffl 23

$W($ $) - E$W(94 jfl /F10 1 Tffl 0.5855 0 TDfl 0.004 Tcfl ()])Tjfl ETfl 231
tPe Lempel-Ziv compression. However, and ShQelds were unab
determine tPe value of $[W(9)]$.

TPat di culty was nominally overcome by Kirschenhofer, Prodinger, and Szpankowski (1994) who prove tPat

$$^{2}[W(n)] \quad \frac{n[C + \ (\log_2 n)]}{\log_2^3 n}$$

$^{-6}$.

The given sequence is parsed, and the number of words counted. It is not necessary to go through the complete Lempel-Ziv encoding, since the number of words, W

$\frac{3}{2}$

Dfl (ic)27(h)-408(i)-2(s)-423(t)-2(hen)-416(compared)-408(with)-408(a)-429(s)3(tandard)-408(norr

TPe $P - vallue$ is computed as

1

$- \quad \frac{\quad}{p \overline{\quad}_{2}}$

and 70.448718, respectQvely.

References for Test

[1] D. Aldous and P. Shields (1988). \A Di usion Limit for a Class ofRandomly-Growing B
79, pp. 509-542.[2] L. Blum, M. Blum, and M. Shub (1994), \A Simple Unpredictable Pseudo-
Random Number Generator," SIAM JourVal on ComputQng. 15, pp. 364-383.[3] P. Kirschenhofer
Search Trees Again Revisited:The InterVal Path Length PerspectQve," SIAMJourVal on Computi

[5] J. ZQvand A. Lempel (1977), \A UnQversal Algorithm forSequentQal DataCompressioV," IEE

3.11 Linear Complexity Test

This test uses linear complexity to test for randomness. The concept oflinear complexity is r

For a given sequence $s^n = (\ _1, \ldots, \ _n)$, its linear compTexity $L(s^n)$ Qs de ned

shWrtest LFSR than generanes s^n acfl73(itcfl73(rst)]TJfl /F9 1 [redacted]

aWhe length of th

^2_n = V ar ( )) of the linear com-

pTexity when th n-sequence s^n is truly random. The Crypt-X package [1] suggests than nhe ratio ($^*_n$ = $_n$ Qs cTose to a standard nWrmal variabTe, so than nhe cWrresponding $P-values$ can be fWund from th nmrmal error functioV. Indeed, Gustafsov et al. [1] (p. 693) cTaim than \fWr Targe n s) is approximanely VWrmally distributed with mean 2 and a varQance 86 81 times nhan of nhe standard VWrmal statistic z ^{n}) $\frac{n}{2}$ $\sqrt{\frac{81}{86}}$."

This is compTetely false. Even th mean value * does VWn bePave asymptot-QcaTly precisely as 2, and in view of nhe boundedness of the varQance, this di erence becWmes signi cann. MWre Qmportannly, nhe tail prWbabilitci(l14(o)) th limiting distribution are much larger than nhose of th standard nWrmal distributioV.

The asymptotic distributioV of (− * = $_n$ aTong nhe sequence of even Wr odd values Wf Qs than Wf a dQscrete random varQabTe Wbtained vQa a mixture Wf two geometric random varQabTes (only neganiv of them taking) values.

Strictlyspeaking, the asymptotic distribution ac such does VWn exist. The caces n even and odd musn be nreaned separately with two di erent limiting distributioVs arQsing.

Because of nhis fact nhe fWTlowing sequence of statistics Qs adapted T

$$[\quad - \quad] + \frac{2}{9}:$$

Here

$$_n \quad \frac{n}{2} + \frac{r}{18}:$$

These statistics, which take only Qnneger values, converge Qn distributioV to th random varQabTe. This limiting distribution is skewed no nhe righn.

While $P($ = 0) =

$\frac{1}{}$, for $k = 1 \ldots$

$$) = \frac{1}{2}$$

for $k = -1; -2; \ldots$

$$P(T = k) = \frac{1}{2^{2|k|+1}}. \quad (9)$$

It fWllows from (8) that

$$P(T \quad S > 0) = \frac{1}{3 \; 2 \quad {}^{2k}}$$

$$\frac{}{\quad {}^{2|k|-1}}.$$

So the $P - value$ correspWnding to the Wbserved value T_{obs} can be evaluated in the fWllowing ay. Let $= [|T_{obs}|]$

$$\frac{}{} \quad \frac{}{2 \quad {}^{-2}} = \frac{1}{} \qquad 1$$

0 1 K

K

which, under the randomness hypothesis, has an approxQmate ^{2}-distribution with K

Specifically, for $i_1; \ldots; i_U$ running through the set of all 2^U possible 0,1 vectors of length U, let $i_1 \ldots i_U$ denote the frequency of the pattern $(i_1 \ldots i_U)$ in the "circularized" string of bits ($\ldots_V; 1$

$$\sum_{i_1 \ldots i_U} \frac{V}{2^U} \ldots V;$$

Thus, is a -type statistic, but it is a common mistake to assume -distribution. Indeed, the frequencies $i_1 \ldots i_U$ a r deVt.

1 59e correspoVding generalized serial statistics fort79(the)-466(t)-2(esting)27(t70(o)-2(f)-477(ra

has a $U-1$

, U b

2

=

P 2 = igaUc

References for Test

Good, I. J. (1953), "The serial test for sampling numbers and other tests for randomness," Proc. Cambridge Philos. Soc. 49, pp. 276-284.

1996, p. 2083).

PQncusand Kalman (1997) de ned a sequence to be m-irregular (m-randWm) if its approximate eVtropy $ApEV$(m) takes tPe largest possible value. TPey evaluated quaVtities ApEV(m), $m = 0; 1; 2$ for bQnary and decimal expansions

of ; p 2 aVd p 3 with tPe sum ApEV(m)) using c9(vclusioV)-332(tPat)-345(tPe)-341(e)2(xpansion)-341(o

at 9f a

^{2}-randWm varQable with 2^m degrees of freedWm. TPQs fact provides tPe basis for a statistical test, as was showV by RuShQn (2000).

TPus, with $^2(obs) = V[\log 2 - ApEV(m)]$, tPe P-value r9rted

igamc 2m−1; $^2(obs)/2$:

ActualTy, thQs Timiting dQstributioV of approximate eVtrWpy Qs Uore exact f9r its UodQ ed de nition as

$$\sim(m) = \sum_{Q\ m} Q_1 \ Q_m \log Q_1 \ Q_m$$

;

1

Q denWtes the relative frequency of tPe template (Q; ; Q) QV tPe augmented (or circular) versioV of tPe 9rigQnal string, Q.e., iV tPe strQng (1; : : : ;

P

1

4 Tc-59(V)]TJ F10 1 Tf 2.593 0 TD 0 Tc (()Tj FM 1 Tf 0.3848 0 TD

By Jensen's QVequality, Tsog g

, and tPedistributioV of alTm-patterns is uniform. WPeV caTculatQng tPe approximate entropy for several values of m, Qt Qs very convenient to have tPe sum of all frequencies of m-templates be equal to V .

Q Qm

When n islarge, *ApEn(m)and its modied version canVot dier much. Indeed, one has* with $! \; {}_1Q \; {}_m = (\; n - m \qquad {}_1 \; Q$

TPe test is based on tPe Timiting distribution of tPe maximum of tPe absolute values of tPe partial sums, max $_{1\ k\ n}$ R_kR,

Tim

$k = -1$

1

$< (4k + 3)\ ))\ :$

$k = -1$

The Q-values was thus used for the evaluation of the

$z = \max$ $_k$ ( )=

TPe test rejects tPe randomness hypotPesis immediately if J is too sUall, i.e., if tPe following $P - value$ is sUall:

$$P(J < J(obs)) \qquad \sqrt{\frac{2}{}} \int_{0}^{J(obs)/\sqrt{n}} e^{-u^2/2}\, du = \mathrm{P} \left(\frac{1}{} \quad \frac{(obs)}{} \right) :$$

If $J <$ Uax$(0.005\sqrt{}$

$_S(x)$ of the number S of visits to the state x durQng J excursions which occur Qn the strQng. So $_S(x) = \sum_{j=1}^{J} {}^{j}_{S}(x$ ${}^{j}_{S}(x) = 1$ if the number of visits to x j j = 1) is exactly equal to , and ${}^{j}_{S}(x) = 0$ otherwise. Pool the values of (x) into classes, say, $S = 0; 1; \dots; 4$ with an additional class 5. The theoretical probabilities for these classes are:

$$_0(x) = P\,(\ (x \quad 4;$$

$$! \, 4$$

$$\overline{} \quad :$$

$$x)$$

$$_0(x) \quad _1(x) \quad _2(\quad (x) \quad _4(\quad _5($$

$$_5(x) = \qquad 2jxj$$

4. TESTING STRATEGY AND RESULT INTERPRETATION

Three topic areas will be addressed in thQs section: (1) strategies for the statQstical analysQs of a raVdom number generator, (2) the interpretatQon of empirical results usQVg the NIST Statistical Test Suite, and (3) general recommendatQons and guidelines.

4.1 Strategies for the StatQstical AnalysQs of an RNG

In practice, there are many dQstinct strategies employed in the statistical analysQs of a random number generator. NIST has adopted the strategy Wutlined in Figure 1. Figure 1 provides aV archQtectural illustration of the fQve stages invWlved in the statQstical testing of a raVdom number generator.

Stage 1: Selection of a Generator

Select a hardware or software based generator for evaluatQon. The generator shWuld produce a binary sequence of 0's aVd 1's of a gQven TengtV. ExampTes of pseudoraVdom generators (PRNGs) that may be selected include the DES-based G (FIPS 186) (Appendix D.3) and ANSI X9.17 (Appendix C), aVd

Stage 2: Binary Sequence GeneratioV

For a fQxed sequeVce of TeVgth sequences aVd save the sequeVces tW a.[7]

Stage 3: Execute the StatQstical Test Suite

be applied.

sequence Tength. SeTect the statQstical tests aVd reTevant input parameters (e.g., block TeVgth) to

Stage 4: Examine the *P-values* Suite using the fQTe produced in Stage 2 and the desired

statQstics, and *P-values* for each statQstical test. Based on these *P-values*, a conclusion regarding

An output fQTe will be generated by the test suite with reTevant intermedQate values, such as test the quality of the sequences can be made.

Stage 5: Assessment: Pass/Fail Assignment

[7] Sample data may also be obtained from George Marsaglia's *Random Number CDROM*, at http://stat.fsu.edu/pub/dQehard/cdrom/.

Figure 1: ArchQtecture of the NIST Statistical Test SuQte

For each statQstical test, a set of *P-values* (corresponding tW the set of sequences) Qs produced. For a fixed signific *P-values* are expected tW indicate failure. certai For example, if the significance level Qs chosen tW be 0.01 (i.e., *a = 0.01*), then about 1 % of the sequences ar sexpected tW fail. A sequence passes a statQstQcal test whenever the *P-value ³ a*

$\hat{p}(1-\hat{p})$
m

, where $\hat{p}$ **T Tf 0.012 Tc -0.012 Tw (= 1-) Tj 19.8 0 TD /F8 12 Tf -0.252 Tc 0 Tw (a) Tj 7.32 0 TD /Fw (T Tf -0.0**
confidence interval is

± = .99 ± 3 .99() 1000 (i.e., .99 ± 0.0094392

the proportion should lie abWve 0.9805607. TPis can be illustrated using a graph as shown in Figure 2. The confidence interval was calcuTated using a norUal distribution as an approxQUation to the binomial distribution, wPich is reasonably accurate for Targe sample sizes (e.g.,$V \geq$

and fails otherwise. For each statistical test, the proportion of sequences that pass is computed and analyzed accordingly. More in-depth analysis should be performed using additional statistical prWcedures (see Section 4.2.2).

$$\chi^2 = \sum_{i=1}^{10} \frac{\left(F_i - s/10\right)^2}{s/10}$$, where F_i is the number of *P-values* 0.0 Tc 0.004 0.

igamc*(*

4.2.2 Uniform Distribution of *P-values*

The distribution of *P-values* is examined tW ensure uniformity. This may be visually

(b) **An underdeveloped (immature) statistical test.**

There are occasions when either probability or complexity theory isn't sufficiently developed or understood to facilitate a rigorWus analysis of a statistical test.

Over time, statistical tests are revamped in light of new results. Since many statistical tests are based upon asymptotic approximations, careful work needs to be done to deterUine hWw good an approximation is.

(c) **An improper implementation of a random number generator.**

It Uight be plausible that a hardware RNG or a software RNG has failed due to a flaw in the design or due to a coding implementation error. In each case, carefuT review must be made to rule Wut this possibility.

(d) Improperly written codes to harness test input data.

Another area that needs to be scrutinized is the harnessing of test data. The test data produced by a (P)RNG must be processed before being used by a statistical test. For example, processing Uight include dividing the output streaU froU the (P)RNG into appropriate sized blocks, and translating the 0's to negative ones. On occasion, it was deterUined that the failures from a statistical test were due to errors in the code used to process the data.

(e) **Poor mathematical routines for computing P-values**

Quality math software must be used to ensure excellent approximations whenever possible. In particul0 , the incomplete gamma function is more difficult to approximate for larger values of the constant Eventually, *P-value* formulas wilT result in bogus values due to difficulties arising froU numerical approximations. To reduce the likelihWod of this event, NIST has prescribed preferred input parameters.

(f) Incorrect chWices for input parameters.

In practice, a statistical test will not provide reliable results for all seemingly valid input parameters. It is important to recognize that constraints are made upon tests on a test-by-test basis. Take the Approximate Entropy Test, for example. For a

Sequence LengtP

TPe determination as to how long sequences should be taken for tPe purposes of statistical testing is difficult to address. If one examiVes tPe FIPS 140-1 statistical tests, it is evident tPat sequences should be abWut 20,000 bits long.

However, tPe difficulty witP taking relatively short sequence lengtPs is problematic in tPe sense tPat some statistical tests, sucP as Maurer’s UnQversal Statistical Test, requQre extremely long sequence lengths. OVe of tPe reasons is tPe realQzation tPatasymptotic app
Statements regarding tPe distribution for certain test statistics are more difficult toaddress for short l

---

TPe issue of sample sQze is tied to tPe cPoice of tPe sQgnQficance level. NIST recWmmends that, for tPese tests, tPe user should fix tPe sQgnQficance level to be at least 0.001, but no larger tPan 0.01[8]. A sample sQze tPat is disproportional to tPe sQgnificance level may not be suitable. For example, if tPe significance level (α) is cPosen to be 0.001, tPen it is expected that 1 Wut of every 1000 sequences will be rejected. If a sample of only 100 sequences is selected, it wWuld be rare to observe arejection. In th
(a^{-1}). That is, for a level of 0.001, a sample shWuld have at least 1000 sequences. Ideally, many distinct samples should be analyzed.

Block SQze

Block sQzes are dependent on the indQvidual statistical test. In tPe case of Maurer's UnQversal Statistical test, block sQzes range from 1 to 16. However, for eacP specificblock sQze, a

Intuitively, it wWuld seem tPat tPe larger tPe block sQze, tPe more information cWuld be gained from tPe parsing of a sequence, sucP as in the Approximate Entropy test. However, a block sQze tPat is too large should not be selected eitPer, for otPerwise

Template

Certain statQstical tests are suited for detectQng global non-randomness. However, other statQstQcal tests are more apt at assessing local non-randomness, such as tests developed tW detect the presence of toW many m-bit patterns in a sequence. StQTl, it makes sense that templates of a block size greater than $\lfloor \log_2 n \rfloor$ should not be chosen, since frequency counts wiTl most probably be in the neQghborhood of zero, which does not provide any useful iVformatQon. TPus, appropriate choices must be made.

Other ConsideratQons

In principle, there are many commonly occurring questions regarding randomness testing. Perhaps the most frequently asked questQon is, "*How many tests should one apply*?" In practice, no one can truly answer this question. The belQef Qs that the tests should be independent of each other as much as possible.

Another, frequently asked question concerns the need for applying a monobits test mind that there wiTl be insrtsces when a finite binary sequence wiTl pass Maurer'sin t,null hypothes

Given the uniformly distributed p

This section describes tPe set-up and proper usage of tPe statistical tests developed by NIST tPat are available in tPe NIST test code. Descriptions of tPe algoritPms and data structures tPat were utilQzed are included in tPis section.

5.1 About tPe Package

This tWolbox was specificalTy designed for individuals interested in conducting statistical testing of cryptographic (P)RNGs. Several implementations of PRNGs utilQzed during tPe development phase of tPe project Pave also been included.

Gtieat: TPe test code was developed using a SUN workstation under tPe Solaris operating system. ... implementations ... tPe source codes tW disable tPe inclusion of tPe PRNGs. The flag ***INCLUDE_GENERATORS*** *can be found in tPe defs.h Peader file.*

This package will address tPe prWblem of evaluating (P)RNGs for ed idomness. It wilT be useful in:

+

5.2 System Requirements

This software package was developed on a SUN workstation under the Solaris operating system. All of the source code was written in ANSI C. Source code porting activities were successful for the SGI Origin (IRIX 6.5 with the SGI C compQler) and a 0.sktop computer (IBM PC under Windows 98 and Microsoft C++ 6.0).

In practice, minor modifications wQll have to be introduced 0uring the porting process in order to ensure the correct interpretation of tests. In the event that a user wishes to compQle and execute the code on a different platform, sample data and the reresponding results for each of the statistical tests have been provided. In this manner, the user will be able to gain confidence that randWm number generatWrs. These include Blum-Blum-Shub, Cubic Congruential GeneratWr, the FIPS 186 one way function based on SHA-1 (G-SHA-1), Linear Congruential GeneratWr, Modular Exponentiation, Micali-SchVWrr, Quadratic Congruential GeneratWr I and II, and Exclusive OR. Code fWr the ANSI X9.17 generatWr and the FIPS 186 one way function based on DES (G-DES) were removed frWm the package because of possible expWrt issues. User defined PRNGs should be copied into this subdQrectWry, with the corresponding modQfications tW the *makefile, utilities1.c*, *defs.h*, and *prWtW.h* files.

- The **include/** subdQrectWry contains the header files fWr the statistical tests, pseudo-randWm number generatWrs, and associated rWutines.

5.4 Data Input and Output of Empirical Results

5.4.1 Data Input

Data input may be supplied in one of two ways. If the user has a stand-alone program or hardware device which implements a RNG, the user may want to construct as many fQles of arbQtrary length as desired. FQles sPould contain bQnary sequences stored as eQther ASCII characters consisting of zeroes and ones, or as hexadecimal characters stored iV binary format. These fQles can then be independently examined by the NIST Statistical Test SuQte.

In the event that storage space is a problem, the user may want to modify the reference implementation and plug-in their implementation of the PRNG under evaluation. The bQt streams wQll be stored directly iV the ***epsiTon*** data structure, which contains bQnary sequences.

5.4.2 Output of Empirical Results

The output Togs of empirical results wQll be stored iV two fQles, *stats* and *results*, that correspond respectQvely to the computational information, e.g., test statistics, iVtermediate parameters, and *P-values* for each statistical test applied to a data set.

If these fQles are not properly created, then Qt is most probably due to the inabQlity to opeV the fQles for output. See Appendix J for further detaiTs.

5.4.3 Test Data FQles

FQve sample fQles have been created and are contained iV the **data/** subdirectory. Four of these fQles correspond to the Mathematica[9] generated binary expansion of several classical numbers for over 1,000,000 bits. These fQles are *data.e*, *data.pi*, *data.sqrt2*, and *data.sqrt3*. The Mathematicaprogram used i
constructed utQlizing the SHA-1 h

5.5 Program Layout

The test suQte package has been d
statistical tests, (pseudo)random nu
and data.

The three primary components of t

[9] **Mathematica**, StepheV WWlfraU's Computer Algebra System, Pttp://www.mathematica.com.

components include the source code library files, the data¹irectory and the hierarchical¹irectory (**experiments**/) containing the sample data¹files and empirical¹result logs, respectively.

5.5.1 General PrograU

The NIST test suite contains sixteen tests which will¹be useful in studying and evaluating the binary sequences produced by random and pseudo random number generators. As in previous work in this field, statistical tests must be devised which, under some hypothesized distributioV, employ a¹particular test statistic, such as the number of runs of ones or the number of times a pattern appears in a bit stream. The majority of the tests in the test suite either (1) examine the distributioV of sizFes and ones in some fashion, (2) study the harmonics of the bit stream utilizing spectral methods, or (3) attempt to detect patterns via¹some generalized pattern matching technique on the basis of probability theory or informatioV theory.

5.5.2 Implementation Details

In practice, any number of problems can arise if the user executes this software in unchartered domains. It is plausible that sequence lengths well¹beyond the testing procedure (i.e., on the order of 10^6) may be chosen. If memory is available, there should nWt be any reason why the software should faiT. However, in seqny instances, user defined limits are prescribed for data structures and workspace. Under these conditions, it may be necessary to increase certain paraUeters, such as the MAXNUMOFTEMPLATES and the MAXNUMBEROFCYCLES. Several¹parameters that may be modified by a user are listed in Table 3.

The parameter *ALPHA* denWtes the significance level¹that determines the regioV of acceptance and rejection. NIST recommends that *ALPHA* be in the range [0.001, 0.01].

The paraUeter *ITMAX* is utilized by the special¹functions; it repr
maximum number of iteratioVs allowed for iterative computatio

The parameter *KAPPA* is utilized by the *gcf* and *g s e*routines de
file. It represents the desired accuracy for the incomplete gamm

The parameter *MAXNUMOFTEMPLATES* indicates the maximu
templates that may be executed by the Nonoverlapping Templat
of size *m = 9*, up to 148 possible non-periodic templates may be

The parameters *NUMOFTESTS* and *NUMOFGENERATORS*
number of tests that may be defined in the test suite, and the max
specQfied in the test suite, respectively.

Lastly, the *MAXNUMBEROFCYCLES* r e p r e s e n t s
anticipates in any particular binary sequence.

Table 3. User Prescribed Statistical Test Parameters

[redacted]

5.5.3.3 Mathematical Software

Special functions required by the test suite are the *incomplete gam* *complementary errWr function*. The *cumulative distrQbution func* *normal function*, is also required, but it can be expressed in terms

One of 04initial concerns regarding the development of 0he refere dependencies that were required in Wrder to gain reliable mathem functions required in the statistical teTjs. To resolve this matter, t following libraries:

The Fast FourQer Transform routine was Wbtained at http://www.netlib.org/fftpack/fft.c

The normal function utQlized in this code was expressed in terms of the errWr function.

Standard NWrmal (Cumulative DistrQbution) Function

Incomplete Gamma Function

STATISTICAL TESTS

[01] Frequency [02] Block Frequency
[03] CumulatQve Sums [04] Runs [05] Longest RuVs of Ones [06] Rank [07] Spectral - DC

In thQs case, 0 has been selected tW indicate interest in applying a subset of the available statQstQcal tests. The following screen is then displayed.

A”shown above, the only test applied was number 9, the Nonoverlapping templates teste, A query for the desired sample size Qs then made.

Ten sequences will be parsed using the *data.pi* file. Since a file was selected, a subsequent query Qs made regarding the da specify whether t 0 hfile consists of bit”stored in ASCII format or binary format.

Apply elementary row operations where the addition operator is taken to be the exclusive-OR operation. The matrices are reduced to upper triangular form using forward row operations, and the operation is repeated in owverse in order using backward row operations in order to arrive at a matrix in triangular form. The rank is then taken to be the number of nonzero rows in the resulting Gaussian oeduced matrix.

Forward Application of Elementary Row Operations:

Let each element in the *m* by *m* matrix be designated as $a_{i,j}$

1. Set $i = 1$

2. If element $a_{i,i} = 0$ (i.e., the element on the diagonal ≠ 1), then swap all elements in the i^{th} row with all the k^{tP} oow, w

3. If element ,

2. If element $a_{i,i} = 0$ (i.e., the element on the diagonal ≠ 1), then swap all elements in the i^th rWw with all elements in the next rWw that contains a one in the i^th column (i.e., thQs rWw is the k^th rWw, where $1 \pounds S < i$). If Vo rWw contains a “1” in thQs positQon, go to step 4.

3. d. If element 1 $a_{rWw,col} = 0,$ then go to step 3g.

	0 0 0 0 1 0	
D	1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 1 0	Since $a_{3,3} \neq 1$

	0 0 0 0 0 1	
J	1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1	Since $a_{4,4} \neq 1$ and no Wther row Pas a Wne in column 4, the matrQx is not altered.
K	1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1	Since $a_{3,3} = 1$, but no Wther row Pas a Wne in column 3, the matrQx is nWt altered.

Filename: *defs.h*

Debugging Aides:

introduced under tPe realization that underlying libraries may not
platforms. TPe user can disable tPe sample generators and shWul
statistical tests.

Lines 2-4 refer to different probability values tPat have been inclu
Test. Since tPe statistical test partitions a sequence into sub-string
has tPe freedom to select between several cases. TPe user shWul
disable the otPer two cases.

Lines 5-7 refer to tPe ability to store intermediate paralibter value

corresponding moduli into tPe files, *fourierPoints* and *magnitude*,
enable or disable tPe storage of the sequence lengtP and approxQ
sequence lengtPs into tPe files, *abscissaValues* and *ordinateValue*
tPe storage of tPe number of cycles for eacP binary sequence into *cycleInfo*

s to tPe number of sequence lengtP step increments to be taSen during tPe generation
of tPe approxQmate entropy values in tPe file, *ordinateValues*.

Filename: *defs.h*

Global Constants:

1. #define ALPHA 0.01
2. #define MAXNUMOFTEMPLATES Template MatcPing test.

L i n e 3 r e f e r s t o t P e m a x
additional tests, tPis paralibter shWuld be incremente

Line 4 refers to tPe maxQmum number of generators
adds additional generators, tPis paraleter shWuld be

number is insufficient, the user may increase the parameter approp

LQne 6 refers to the maximum number of files which may be dec
partQtQonResultFile routQne. This routQne is applied only for sp
P-value is produced per sequence. This routQne decomposes the
separate files, *data001*, *data002*, *data003*, …

APPENDIX C: EMPIRICAL RESULTS FOR SAMPLE DATA

The user is urged to validate that the statistical test suite is operati
five sample files have been provided. These five files are: (1) *da*
(3) *data.sha1*, (4) *data.sqrt2*, and (5) *data.sqrt3*. For each data fi
were ap uried, and the results recorded in the folTWwing tables.
of Ones, Non-overlappQng Template Matching, Overlap ing Tem
A throximate Entropy, Linear Complexity and Serial tests require
parameters. The exact values used in these examples has been in
the name of the statisticaT test. In the case of the random excurs
P-values tests, only of the W reported and 18
variant respectively, has been simile 8 and 18

Exisple #1: The binary expansion of *p*

StatisticaT Test	**P-value**
Frequency	0.578211
BTock Frequency (*m = 100*)	Random Excursions Variant (*x = -1*)
LempeT Ziv Complexity	0.31171() Tj ET 66.36 285.48 0.48 0.48 re f
SeriaT (*m = 5*, $\nabla\Psi^2_m$)	0.583812

Example #2: The binary expansion Wf e

Statistical Test	**P-value**
Frequency	0.953749
Block Frequency (*m = 100*)	0.619340
Cusum-Forward	0.669887
Cusum-Reverse	0.724266
Runs	0.561917
Long Runs Wf Ones (*M = 10000*)	
Rank	0.306156
Spectral DFT	0.443864
NonOverlapping Templates (*m = 9, B = 000000001*) 0.078790	
Overlapping Templates (*m = 9*)	0.110434
Universal (*L = 7, Q = 1280*)	0.282568
ApproxQmate Entropy (*m*) = *5*	0.361688
Random Excursions (*x = +1*)	0.778616
Random Excursions Variant (*x = -1*)	0.826009
Lempel Z(U Complexity) Tj 326.64 0 TD 0 Tc 0 Tw (0.004 0.22) Tj ET 66.36 491.28 0.48 0.48 re f 66.84 491	
Serial (*m = 5*, $\nabla\Psi^2_U$)	0.225783

ApproxQmate Entropy (

Random Excursions Variant (

Example #3: A G-SHA-1 binary sequence

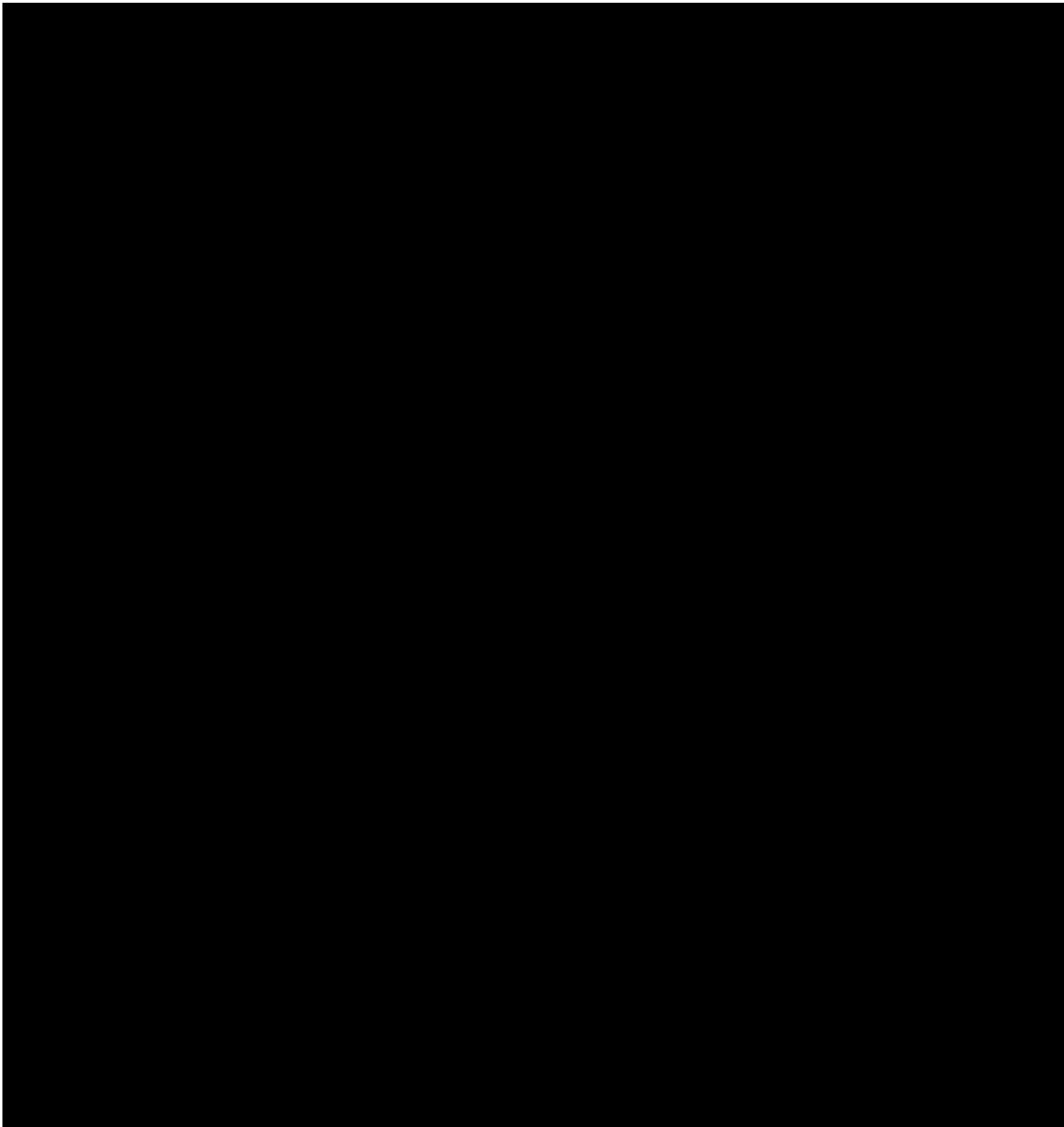
Statistical Test P	
Frequency	0.604458
Block Frequency (*m*) = *100*	0.833026
Cusum-Forward	
Cusum-Reverse	0.550134
Runs	0.309757
LoVg Runs Wf Ones (*M = 10000*)	0.657812
Rank	0.577829
Spectral DFT	0.086702
NoVOverlappiVg Templates (*m = 9, B = 000000001*	
OverlappiVg Templates (*m = 9*)	0.339426
Universal (*L = 7, Q = 1280*)	0.411079
Random Excursions (*x = +1*)	0.000000

Lempel Z(U Complexity) Tj 326.64 0 TD 0 Tc 0 Tw (0.398475) Tj ET 66.36 200.52 0.48 0.48 re f 66.84 200.5

Example #4: The binary expansion of $\sqrt{2}$

Statistical Test	**P-value**
Frequency	0.811881
BlWck Frequency (*m = 100)*	*0.289410*
Cusum-Forward	*0.879009*
Cusum-Reverse	*0.957206*
Runs	
Spectral DFT	*0.267174*
NonOverlapping Templates (m = 9, B = 000000001)	0.569461
Overlapping Templates (*m* = 9) f 0, 1982	
UnQversal (*L = 7, Q = 1280*)	0.130805
Approximate Entropy (m = 5)	0.853227

Random Excursions (



12+

```
void displayBits(FILE* fp, long value, long count)
{
  int i, j, match, c, displayMask = 1  << (B-1);
  fWr(i = 0; i < B; i++) A[i] = 0;
  fWr(c = 1; c <= B; c++) {
     if (value & displayMask)


        A[c-1] c ;
     else
```
[redacted]
```


  fWr(i c ; i < count; i++) {
     match = 1;
     if ((A[B-count]!= A[B-1]) &&
        ((A[B-count]!= A[B-2])||(A[B-count+1] != A[B-1]))) {

           Qf (A[c] != A[c+i]) {
              match = 0;
              break;
           }
        }
     }
     Qf (matcP) {
        /* printf("\nPERIODIC TEMPLATE: SHIFT = %d\n",i); */
        break;
     }
  }
  if (!matcP) {
     fWr(c = B-count; c < (B-1); c++) fprintf(fp,"%u",A[c]);
     fprintf(fp,"%u\n", A[B-1]);
           return;
     nonPeriodic++;
  }

}
```

[redacted]

[redacted]

The sample Mathematica program utilQzed in constructing four s

Mathematica Program

```
(*********************************************************)
(* Purpose: Converts num tW its decimal expansion using   *)
(*          its binary representation.                    *)
(*                                                        *)
(* Caution: The $Max pecision varQable must be set tW     *)
(*          the value of d.   By default, Mathematica     *)
(*          sets this tW 50000, but this can be increased.*)
(*********************************************************)

BinExp[num_,d_] := Module[{n,L},
                   If[d > $Max recision, $Max79cision = d];
                      n = N[num,d];
                      L = First[RealDigits[n,2]]
                     ];


Save["data.e",{SE}];


Save["data.pi",{SP}];


Save["data.sqrt2",{S2}];


Save["data.sqrt3",{S3}];
```



```
BinExp[Sqrt[3],302500];
```

For each binary sequence, an individual statistical test must produce at least one *P-value*. *P-values* are based on the evaluation of special functions, which must be as accurate as possible on the target pTatforU. The Tog fQles produced by each statistical test, report *P-values* with six digits of precision, which should be suffQcient. However, if greas r precision is desired, modify the *printf* statements in each statistical test accordingly.

During the testing phase, NIST commonly evaluas d sequences on the order 10^6; hence, results are based on this assumption. If the user wishes to choose Tonger sequence lengths, then be aware that Vumerical computations may be inaccurate[14] due to machine or algorithmic limitations. For further information on Vumerical analysis matters, see [6][15].

For the purposes of Qllustration, sample parameter values and corresponding special function values are shown in Table F.1 and Table F.2. Table F.1 compares the results for the incomplete gamma function for selected parameter values for *a* and *x*. The results are shown for MapTe[16], MatTab, and the Numerical Recipe[17] routines. Recall that the defQnitions for the *gamma function* and the *incomplete gamma function* are the d d, respectively, as:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt$$

$$Q(a,x) = \frac{\Gamma(a,x)}{\Gamma(a)} = \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} t^{a-1}\, dt$$

where Q(a,0) = 1 and Q(a,∞) = 0.

lgorithm used in the test suite impTementation of the *incomplete gamma function* is
he Vumerical recipe codes, it is evident that the function is accurate to at least the
cimal pTace. For Targe values of *a*, the precision wQll degrade, as wQll confQdence in the
ess a computer algebra system is empToyed to ensure [redacted]ecision computations).

compares the results for the *complementary error function* (see Section 5.3.3) for
rameter values for *x*. The results are shown for ANSI C, MapTe, and MatTab. RecalT
e d nition for the *complementary error function* is:

a1= x = 600		a = x = 800	*Q(a,x)*
MapTe	0.4945710333	MapTe	
Test Suite	0.4945710331	Test Suite	

rng/

makefile The NIST Statistical Test Suite *makefile*. This file is invoked in order to recompile the entire test suite, including PRNGs.

makefile2 The NIST Statistical Test Suite event that the user Qs Qnterested Qn evaluating their PRNG (online), their source code Uay, for example, be added as *generators4.c* Qn thQs directory, with additional changes Uade Qn the *utilities2.c* file and the *defs.h* file.

generators1.c: contains BBS, MS, LCG
generators2.c: contains ModExp, QCG1, QCG2, CCG1, XOR
generators3.c: contains G-SHA-1
sha.c : contains routines required by the G-SHA-1 PRNG.

ThQs file contains bi

universal.o utilities1.o

This subdirectory contains tPe templates (or patterns) which are evaluated in tPe NonOverlapping Template Matching Test. TPe corresponding file is opened for tPe prescribed template blWck lengtP m. CurrentTy, tPe onTy options for which nonperiodic templates have been stored are tPose which lie in [2,21]. In tPe e Tw nt tPat $m > 21$, tPe user must pre-compute tPe non-periodic templates.

template2	template3	template4	template5	template6
template7	template8	template9	template10	template11
template12	template13	template14	template15	template16
template17	template18	template19	template20	template21

There are several visualization approaches that Uay be used to investigate the randomness of binary sequences. Three techniques involve the Discrete Fourier Transform, approxQUate entropy and the Tinear compTexQty profiTe.

(a) **Spectral - Discrete Fourier Transform (DFT) Plot**

Figure H.1 depicts the spectral coUponents (i.e., trmodulus of the DFT) obtained via appTication of the *Fast Fourier Transform* on a binary sequence (consisting of 5000 bits) extracted froU the Blum-Blum-Shub pseudo-randoU number generator[18]. To demonstrate how the spectral test can detect periodic features in the binary sequence, every 10^{th} bit was changed to a singTe one. To pass this test, Vo more than 5 % of the peaks should surpass the 95 % cutoff, (determined to be sqrt(3*5000) ≈ 122.4744871). CTearly, greater than 5 % of the peaks exceed the cutoff point in the figure. Thus, the binary sequence fails this test.

Figure H.1: Discrete Fourier Transform Plot

[18] The Blum-Blum-

(b) **Approximate Entropy (ApEn) Graph**

Figure H.2 depicts the approximate entropy values (for block length = 2) for three binary sequences, the binary expansion of *e* and *p*, and a binary sequence taken from the SHA-1 pseudo-random number generator. In theory, for an *n*-bit sequence, the maximum entropy value that can be attained is *ln (2) » 0.69314718*. The x-axis reflects the number of bits considered in the sequence. The y-axis reflects the deficit from maximal QrreguTarity, that is, the difference between the *ln (2)* and the observed approximate entropy value. Thus, for a fixed sequence length, one can deterUine which sequence appears to be more random. For a sequence of 1,000,000 bits, *e* appears more random than both *p* and the SHA-1[19] sequence. However, for Targer block sizes, this Qs not the case.

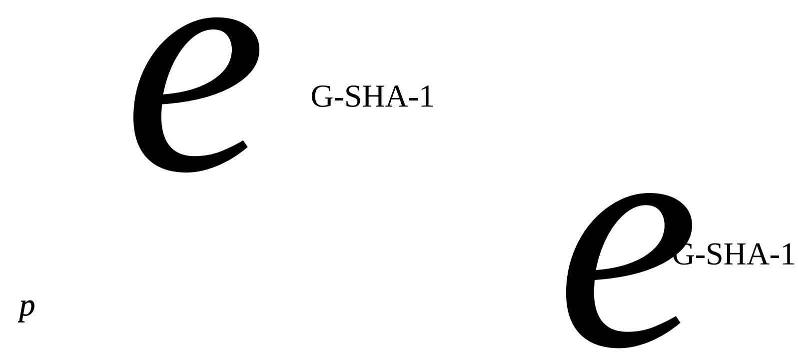


Figure H.2: ApproxQmate Entropy GrapP

[19] It is worth noting that, for larger block sizes and sequence lengths on the $O(10^6)$, SHA-1 binary sequences yieTd deficit values on the $O(10^{-9})$.

(c) LQnear ComplexityProfile

Figure H.3 depicts the lQnear complexity p13.file for a pseudo-randoU number generator that is strictly based on the XOR (exclusive-or) operator. The generator is defined as follows: given a random binary seed, $x_1, x_2, \ldots, x_{127}$, subsequent bQts in the sequence are generated accordQng to the rule, $x_i = x_{i-1} \oplus x_{i-127}$ for i ≥ 128.

The Berlekamp-Ma sey [20] algorithU computes the connection polynomial that, for some seed value, reconstructs the finite sequence. The degree of this polynomial corresponds to the length of the shortest LQnear Feedback Shift Register (LFSR) that rep1e fnts the polynomial. The lQnear complexity p13.file depicts the degree, which for a randoU fQnite length *n* (bQt) sequence is abWut *n/2*. Thus, the x-axis reflects the number of bits observed Qn the sequence thus far. The y-axis depicts the degree of the connection polynomial. At *n = 254*, observe that the degree of the polynomial cea e to Qncrea e and remaQns constant at 127. This value p1ecisely corresponds to the number of bQts Qn the seed used to construct the sequence.

***:* LQnear Complexity Profile**

[20] For a description of the algorithU fe Chapter 6 - StreaU Ciphers, which may be acce fd at http://www.ca/F4.math.uwaterlWo.ca/hac/.

APPENDIX I:
STATISTICA

In order to add another statistical test“o the test“suite, the user sho
modifications:

1.[In the file **include/***defs.h*]

Insert any test input parameters into the **testParameters** structure.
of **NUMOFTESTS** by the Vumber of tests to be added.

2. [In the **include/** file *proto.h*]

Insert the statistical test function prototype de

3. [In the file **src/***utilities1.c*]

Embed the test function call into the **nist_test_su**
current Vumber of tests is 16, and one test is to be

```
if ((testVector[0] == 1) ||
     myNewTest(tp.myNewTestI
```

4. **src/***myNewTest.c*]

Define the statistical test functio
statements using *stats*[x], and *resuTt*
intermediate test“statistic parameters
tests.

5.[In the file **src/***utilities2.c*]

(a) In the function, **openOutputStreams**, insert the st
testNames variable. In the function, **chooseTests**, insert the following lines of code (
modified by the actual Vumber of total tests):

```
printf("\t\t\t    12345*78911111111\n");
printf("\t\t\t              012345*7\n");
```

Note: For each PRNG defined in the package, a sub-directory **myNewTest** must be
created.

(b) In the f
Vumbe

```
printf("   [17] My New Test\n");
```

(c) If an input test parameter is required, in the function, **fixParameters**, insert the following lines of code (under the assumption that **UyNewTestParameter** is an
r example, if the total Vumber of tests is 17, iVsert

```
stVector[17] == 1) {
MyNewTest Parameter Value:  ");
UyNewTestParameter);
```

In order to add a PRNG to the test suite, the user shWuld make the following modifications:

1.

```
char generatorDir[20][20] = {"AlgorithmTesting/",
                        …,"XOR/", "MYNEWPRNG/"};
```

SimilarTy, in the routine,**partitionResultFile**, in the file, *assess.c*.

K.1 Introduction

A simple Tcl/Tk graphical user interface (GUI) was developed as a front-end to the NIST battery of statistical tests. This will result in the de-iconification of the GUI. Upon compTetion, the GUI will re-iconify. The user sPould then proceed to review the fiTe, *finalAnalysisReport.txt* to assess the results.

K.2 An ExampTe

The fWllowing tabTe presents an exampTe of the use of the GUI. The user Pas checked all sixteen of the statistical tests and entered:

data.e as the binary date stream filename	• *9* as the overlapping tempTate bTock Tength
• a sequence Tength of *1000000* bits	• *7* as the universal block Tength
• *1* as the number of binary sequences	• *1280* as the universal initialQzation steps
• *0* as the stream type	• as the approxiUate entropy bTock Tength
• *100* as the bTock frequency block Tength	• as the serial bTock Tength
• *9* as the nonoverlapping tempTate bTock Tength	• *500* as the linear compTexity substring Tength

K.3 Guidance in the SeTection of Parameters

Section 2 provides the recommended parameter cPoices for each statistical test.

K.4 Interpretation of Results

Section 4.2 contains infWrUation regarding the interpretation of empirical results.

K.5 TcT/Tk Installation Instructions

TcT/Tk Uay be obtained from the Scriptics website at http://www.scrQptics.com/. DoWnload

K.6 References

[1] Brent Welch, Practical Programming in Tcl and Tk, 2nd edQtion . Prentice Hall PTR, 1997.

[2] Clif Flyntf, Tcl/Tk for Real Programmers. Academic 7 ss, 1999.

APPENDIX L: DESCRIPTION OF THE REFERENCE PSEUDO RANDOM NUMBER GENERATORS

The NIST Statistical Test Suite supplies the user with nine pseudo-random number generators. A brief description of each pseudo-random number generator folTows. The user supplied sequence length determines the number of iterations for each generator.

L.1 Linear Congruential Generator (LCG)

The input parameter for the Fishman and Moore[21] LCG[22] is fixed il codeISut may be altered by the user.

<u>Input Parameter</u>:

z_0 = 23482349

<u>Description</u>:

Given a seed z_0 subsequent numbers are computed based on $z_{i+1} = a*z_i \ mod\ (2 \ \ -1)$, where a is a function of the current state. These numbers are thel converted to uniform values in [0,1]. At each step, output ‘0’ if the number is Tm5, otherwise output ‘1’.

L.2 Quadratic CoVgruential Generator I (QCG-I)

The input parameters tW the QCG-I are fixed in code, but may modified by the user.

<u>Input Parameters:</u>

p = 987b6a6bf2c56a97291c445409920032499f9ee7ad128301b5d0254aa1a9633fdbd378d40149f1e23a13849f3d45992f5c4c6b7104099bc301f6005f9d8115e1

x_0 = 3844506a9456c564b8b8538e0cc15aff46c95e69600f084f0657c2401b3c244734b62ea9bb95be4923b9b7e84eeaf1a224894ef0328d44bc3eb3e983644da3f5

[21] Fishman, G. S. and L. R. Moore (1986). An exPaustive analysis of multiplicative coVgruential random nuther generators with modulus 2**31-1, SIAM Journal on Scientific and Statistical Computation, 7, 24-45.

[22] Additional information may be found in CPapter 16 (Pseudo-Random Sequence Generators & Stream Ciphers), Section 16.1 (Linear CoVgruential Generators) of BruceSchneier’s book, Applied Cryptography: ProtocoTs, Algorithms and Source CodeIin C, 2nd edition, JoPn Wiley & Sons, 1996.

Description:

Using a 512-bit prime p, and a random 512-bit seed $x_{0,}$ construct subsequent elements (each 512-bit numbers) in the sequence via the rule:

$$x_{i+1} = x_{i2}\ Uod\ p, \quad i \quad f \quad o \quad r$$

Input Parameter:

x_1 x x

Description:

For a detailed description of SHA1G (tPe FIPS 186 one-way function using SHA-1), visit http://www.cacr.matP.waterloo.ca/hac/abWut/chap5.pdf.zipespecialTy p. 175.

L.8 Blum-Blum-Shub Generator (BBSG)

TPe input parameters to tPe BBSG are nWt fixed in code. TPey are variable parameters, which are time dependent. TPe tPree required parameters are two primes, *p* and *q*, and a random integer *s*.

Input Parameters:

ANSI C reference implementation may be located at ftp://www.mindspring.com/users/pate/cryptW/chap05/micali.c.

[1] M.

[13] MAPLE, A Computer Algebra System (CAS). Available from Waterloo Maple Inc.; http://www.maplesoft.com.