# Statistical Testing of Random Number Generators

Juan Soto
National Institute of Standards & Technology
100 Bureau Drive, Stop 8930
Gaithersburg, MD 20899-8930
soto@nist.gov
(301) 975-4641

Abstract: *Random Number Generators[1] (RNGs) are an important building block for algorithms and protocols in cryptography. They are paramount in the construction of encryption keys and other cryptographic algorithm parameters. In practice, statistical testing is employed to gather evidence that a generator indeed produces numbers that appear to be random. Few resources are readily available to researchers in academia and industry who wish to analyze their newly developed RNG. To address this problem, NIST has developed new metrics that may be employed to investigate the randomness of cryptographic RNGs. In this paper, issues such as statistical test suites, evaluation frameworks, and the interpretation of results are addressed.*

## 1.0 Introduction

In computer security, suitable metrics are needed to investigate the degree of randomness for binary sequences produced by cryptographic random number generators (RNGs). Today, researchers are developing new hardware and software based RNGs. However, few standards address statistical analysis techniques that should be employed in practice. This paper will: (1) list statistical test suite sources, (2) illustrate several evaluation approaches, (3) briefly describe the National Institute of Standards and Technology (NIST) Statistical Test Suite and its application in the systematic evaluation of cryptographic RNGs, (4) establish guidelines for the interpretation of test results, and finally (5) express a few closing remarks.

## 2.0 Statistical Test Suites

For those interested in analyzing their cryptographic RNG, several options are available. **Table 1** highlights batteries of statistical tests that are available or will be available in the near future.

**Table 1. Batteries of Statistical Tests**

| Source/Affiliation | Statistical Tests |
|---|---|
| 1. Donald Knuth/*Stanford University* | The Art Of Computer Programming Vol. 2 Seminumerical Algorithms |

[1] Throughout this paper, the term, random number generators, refers to both hardware based RNGs and software based RNGs, i.e., pseudo random number generators (PRNGs).

| | |
|---|---|
| 2. George Marsaglia/*Florida State University* | DIEHARD |
| 3. Helen Gustafson, et. al./ *Queensland University of Technology* | Crypt-XS |
| 4. Alfred Menezes, et. al./*CRC Press, Inc.* | Handbook of Applied Cryptography |
| 5. Andrew Rukhin, et. al./*NIST ITL* | NIST Statistical Test Suite |

In Donald Knuth's book, **The Art of Computer Programming, Seminumerical Algorithms, Volume 2,** he describes several empirical tests which include the: **frequency**, **serial**, **gap**, **poker**, **coupon collector's**, **permutation**, **run**, **maximum-of-t**, **collision**, **birthday spacings**, and **serial correlation**. For further information, visit http://www-cs-faculty.stanford.edu/~knuth/taocp.html.

The **DIEHARD** suite of statistical tests developed by George Marsaglia consists of fifteen tests, namely the: **birthday spacings**, **overlapping permutations**, **ranks of 31x31 and 32x32 matrices**, **ranks of 6x8 matrices**, **monkey tests on 20-bit Words**, **monkey tests OPSO, OQSO, DNA**, **count the 1's in a stream of bytes**, **count the 1's in specific bytes**, **parking lot**, **minimum distance**, **random spheres**, **squeeze**, **overlapping sums**, **runs**, and **craps**. Additional information may be found at http://stat.fsu.edu/~geo\diehard.html

The **Crypt-XS** suite of statistical tests was developed by researchers at the Information Security Research Centre at Queensland University of Technology in Australia. Crypt-XS tests include the **frequency**, **binary derivative**, **change point**, **runs**, **sequence complexity** and **linear complexity**. For additional information visit http://www.isrc.qut.edu.au/cryptx/index.html.

The **NIST Statistical Test Suite** is the result of collaborations between the Computer Security Division and the Statistical Engineering Division at NIST. Statistical tests in the package include the: **frequency**, **block frequency**, **cumulative sums**, **runs**, **long runs**, *Marsaglia's* **rank**, **spectral** (based on the Discrete Fourier Transform), **nonoverlapping template matchings**, **overlapping template matchings**, *Maurer's* **universal statistical**, **approximate entropy** (based on the work of Pincus, Singer and Kalman), **random excursions** (due to *Baron* and *Rukhin*), **Lempel-Ziv complexity**, **linear complexity**, and **serial**. Additional information may be found at http://www.itl.nist.gov/div893/staff/soto/jshome.html.

## 3.0 Evaluation Approaches

Different approaches have been taken by designers of statistical tests. Given a binary sequence *s*, we want to establish whether or not *s* passed or failed a statistical test. In this paper we will compare three different viewpoints.

### 3.1 Case A: Threshold Values

One approach is to compute a test statistic for a binary sequence *s* and compare it to a threshold value. The decision rule in this case states that a binary sequence fails this test "whenever the value of *c(s)* falls below the threshold value." For example, the *sequence complexity test* described in the Crypt-XS package is based on Lempel-Ziv compression. Given *s*, we compute its *sequence complexity*, *c(s)*. In order to determine whether the sequence passed this test, we need to compare *c(s)* with the threshold value, $n/(\log_2 n)$.

### 3.2 Case B: Fixed Ranges

A second approach involves computing a test statistic for *s* as before. However, in this case, the decision rule states that "*s* fails a test if the test statistic falls outside of a range." For example, if the *frequency test* is applied to a binary sequence *s* consisting of 800 bits, and we define our test statistic to be the *number of ones* in *s*, we expect roughly 400 zeroes and 400 ones. If the significance level is fixed at 5%, then *s* fails the test if the *number of ones* falls outside the range $400 \pm 1.96/2 * \sqrt{800} = [373,427]$.

### 3.3 Case C: Probability Values

A third approach involves computing a test statistic for *s* and its corresponding probability value (P-value). Typically, test statistics are constructed so that large values of a statistic suggest a non-random sequence. The P-value is the probability of obtaining a test statistic as large or larger than the one observed if the sequence is random. Hence, small values (conventionally, P-values < 0.05 or P-values < 0.01) are interpreted as evidence that a sequence is unlikely to be random. The decision rule in this case states that "for a fixed significance value $\alpha$, *s* fails the statistical test if its P-value < $\alpha$." Typically, $\alpha$ is taken to be a value in the interval [0.001,0.01]. For example, if *s* is a sequence of 1,000,000 bits, and we apply a *runs test*, then our test statistic *V*, the *total number of runs*, should be roughly 500,000. Suppose $V = 499996$; then its P-value = $\text{erfc}\left(\left|\frac{V - 2np(1-p)}{2\sqrt{2np}(1-p)}\right|\right) = 0.994876$, where *n* is the sequence length and *p* is the total number of ones divided by *n*. Clearly, *s* passes the test since the P-value is very close to one.

*Note: This list is not exhaustive and was chosen to illustrate contrasting techniques.*

The limitations of each of these cases are as follows:

**Case A:** The use of threshold values may not be a sufficiently stringent measure. A sequence complexity measure, which exceeds a threshold value, *may be* non-random. Empirical evidence[2] utilizing the SHA-1 generator suggests that for sequence lengths of

[2] Research work by Leung and Tavares [4] indicates that for 64 bit blocks, the expected sequence complexity value is approximately 13, which agrees with our empirical results. An approximate expected

1,000,000, the mean is close to 50,778 which is far greater than the threshold value, $ceil\left(10^6/\log_2 10^6\right)$ = 50,172. A 1,000,000, bit sequence counterexample was observed using the file, *canada.bit*[3]. The sequence consisting of 50.3726% zeroes and 49.6274% ones, clearly fails a monobits test[4], however, its sequence complexity measure of 50,553 would exceed 50,172, and hence, the sequence passes the Crypt-XS sequence complexity test.

**Case B:** The use of fixed ranges implies that significance levels and acceptable ranges are pre-computed. If significance levels are modified in the future, the range values must be recomputed.

**Case C:** The use of P-values is non-trivial in some cases, but has the added advantage that they do not require the specification of the significance level, $\alpha$. Once a P-value has been computed, the P-value can be compared to an arbitrary $\alpha$. Typically, P-values are computed utilizing special functions such as the:

**Standard Normal (Cumulative Probability Distribution) Function**

$$\Phi(z) = \frac{1}{\sqrt{2p}} \int_{-\infty}^{z} e^{-u^2/2} du$$

**Complementary Error Function**

$$erfc(z) = \frac{2}{\sqrt{p}} \int_{z}^{\infty} e^{-u^2} du$$

**Incomplete Gamma Function**

$$Q(a,x) \equiv 1 - P(a,x) \equiv \frac{\Gamma(a,x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_{x}^{\infty} e^{-t} t^{a-1} dt$$

where Q(a,0) = 1 and Q(a,$\infty$) = 0.

Of course, it must be emphasized that *Q(a,x)* is not easily computed, especially for large *a*, and one must resort to numerical methods to achieve accurate results. Among these three case scenarios, NIST chose case C for its statistical test suite due to its flexibility.

## 4.0 The NIST Statistical Test Suite

Let us proceed to describe the NIST test suite in more detail. We begin by highlighting our evaluation framework and then list the defects that each test was designed to detect.

---

value for sequence complexity was established by Mund [8]; however, our experiments suggest that for larger sequence lengths, $O(10^6)$, this may not be a good approximation.

[3] This file may be found on Marsaglia's **Random Number CDROM**, http://stat.fsu.edu/pub/diehard/cdrom/

[4] A monobits test examines the distribution of zeroes and ones in a binary sequence.

### 4.1 The NIST Framework

The NIST framework, like many tests, is based on hypothesis testing. A hypothesis test is a procedure for determining if an assertion about a characteristic of a population is reasonable. In this case, the test involves determining whether or not a specific sequence of zeroes and ones is random. **Table 2** illustrates the step by step process that is followed in the evaluation of a single binary sequence. Additional information on hypothesis testing terminology may be found in the appendix.

**Table 2. Evaluation Procedure For A Single Binary Sequence**

| Step By Step Process | Comments |
|---|---|
| 1. State your null hypothesis. | Assume that the binary sequence is random. |
| 2. Compute a sequence test statistic. | Testing is carried out at the bit level. |
| 3. Compute the P-value. | P-value $\in$ [0, 1]. |
| 4. Compare the P-value to $\alpha$. | Fix $\alpha$, where $\alpha \in$ (0.001, 0.01]. *Success* is declared whenever P-value $\geq \alpha$; otherwise, *failure* is declared. |

### 4.2 The NIST Statistical Tests

Though much attention could be given in fully describing each of the statistical tests, we will focus strictly on the types of defects that this battery of statistical tests was designed to detect. **Table 3** describes the general characteristics of each of the statistical tests.

**Table 3. Characteristics of the NIST Statistical Tests**

| Statistical Test | Defect Detected |
|---|---|
| 1. Frequency | Too many zeroes or ones. |
| 2. Cumulative Sums | Too many zeroes or ones at the beginning of the sequence. |
| 3. Longest Runs Of Ones | Deviation of the distribution of long runs of ones. |
| 4. Runs | Large (small) total number of runs indicates that the oscillation[5] in the bit stream is too fast (too slow). |
| 5. Rank | Deviation of the rank distribution from a corresponding random sequence, due to periodicity[6]. |
| 6. Spectral | Periodic features in the bit stream. |
| 7. Non-overlapping Template Matchings | Too many occurrences of non-periodic templates. |
| 8. Overlapping Template Matchings | Too many occurrences of m-bit runs of ones. |
| 9. Universal Statistical | Compressibility[7] (regularity). |

[5] *Oscillation* refers to abrupt changes between runs of zeroes or runs of ones.
[6] *Periodicity* refers to sub-sequences that repeat.

| | |
|---|---|
| 10. Random Excursions | Deviation from the distribution of the number of visits of a random walk[8] to a certain state. |
| 11. Random Excursion Variant | Deviation from the distribution of the total number of visits (*across many random walks*) to a certain state. |
| 12. Approximate Entropy | Non-uniform distribution of m-length words. Small values of ApEn(m) imply strong regularity. |
| 13. Serial | Non-uniform distribution of m-length words. Similar to Approximate Entropy. |
| 14. Lempel-Ziv Complexity | More compressed than a truly random sequence. |
| 15. Linear Complexity | Deviation from the distribution of the linear complexity[9] for finite length (sub)strings. |

## 5.0 Interpretation of Empirical Evidence

The principal problem lies in the many approaches that can be taken in order to determine the effectiveness of a statistical test. We begin by describing a set of numerical experiments that was conducted utilizing the NIST developed statistical tests.

### 5.1 Numerical Experiments

*Three pseudo-random number generators* (G-SHA-1, Blum-Blum-Shub, Cubic Congruential Generator) were selected[10]; along with *five statistical tests*, numerically coded as (1 = frequency, 2 = cumulative sum, 3 = runs, 4 = spectral, 5 = approximate entropy); a *sequence length* = 1,000,000; a *sample size*[11] = 300; and a *significance level*, $\alpha = 0.01$. For each generator, five statistical tests were applied to each binary sequence. In all, this resulted in (300)(5)(3) = 4500 P-values.

### 5.2 Goodness of Fit Distributional Tests & Graphical Analysis of Empirical Results

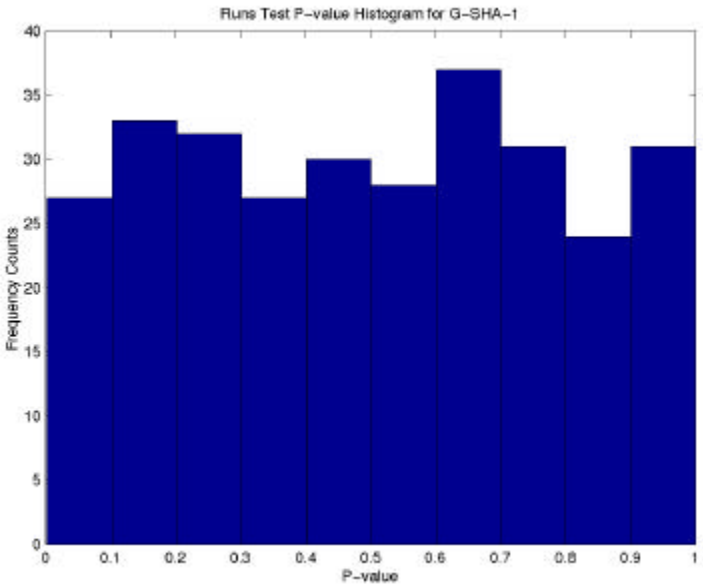
Once the statistical tests had been applied, we wished to determine how well the empirical results matched their theoretical counterparts. This could be accomplished by assessing the goodness of fit of the distribution of P-values to a uniform distribution. This could be done in one of several ways. *One approach*, involves taking the mean and variance of the P-values and comparing it to the mean (0.5) and variance (1/12) for a uniform distribution. A *second approach* involves computing a chi-square statistic with nine degrees of freedom based on the frequency counts of P-values among bins determined by discretizing[12] the unit interval by ten.

---

[7] *Compressibility* refers to the existence of a sub-sequence that represents the entire sequence.
[8] A 1-D *random walk* is a sequence of steps, each of whose characteristics is determined by chance.
[9] *Linear complexity* is the length of the shortest linear feedback shift register that generates the sequence.
[10] Selection was based on the inclusion of a cryptographically secure PRNG (Blum-Blum-Shub), an excellent PRNG (SHA-1) and a poor PRNG (Cubic Congruential).
[11] That is, 300 individual sequences were constructed, each consisting of 1,000,000 bits.
[12] Discretizing the unit interval, i.e., subdividing [0,1] into ten equally spaced subintervals.

**Figure 1** illustrates a histogram of P-values obtained from the runs test applied to three hundred sequences generated utilizing G-SHA-1.

***Figure 1: Illustration of the Chi-Square Test to Evaluate Goodness of Fit***




**Figures 2-4** depict a graphical approach to facilitate the interpretation of empirical results. The x-axis consists of individual statistical tests, whereas the y-axis represents the proportion of sequences that passed the corresponding statistical test at the chosen significance level, $\alpha = 0.01$. Ideally, we expect to reject 1 out of every 100 binary sequences, or equivalently, 99% of the sequences should pass (depicted as a solid line in Figures 2-4). However, realistically this won't necessarily be satisfied. The minimum acceptable proportion of binary sequences expected to pass a statistical test was determined utilizing a *confidence interval* defined as $\hat{p} \pm k\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$, where $\hat{p}$ = 1-$\alpha$, is the average P-value; k = 3 is the number of standards deviations, and *n* is the sample size. The dashed line depicted in Figures 2-4 represents that lower bound (97.27%).

As we can see in **Figure 2** and **Figure 3**, none of the proportions falls below this threshold. However, in **Figure 4**, we see that all of our proportions fall below the threshold. This is indicative of a serious problem with this pseudo-random number generator. Further testing on different samples should be conducted to gather more evidence that this is a poor PRNG.

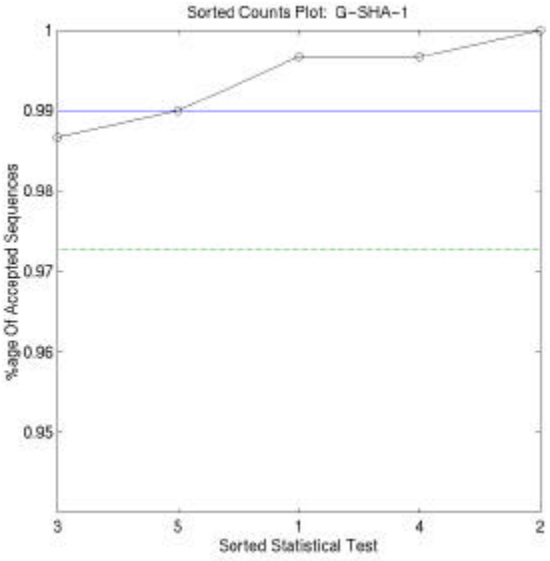



**Figure 2:**
This graph depicts the empirical results for the G-SHA-1 PRNG.

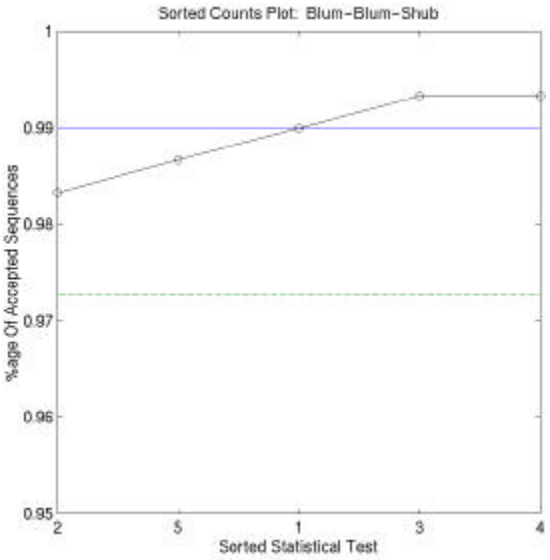



**Figure 3:**
This graph depicts the empirical results for the Blum-Blum-Shub PRNG.

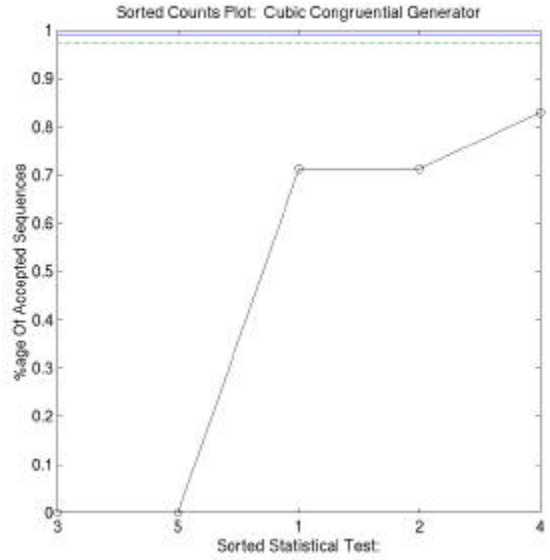



**Figure 4:**
This graph depicts the empirical results for the Cubic Congruential PRNG.

## 6.0 Open Problems

Two problems that must be addressed include (a) the independence of the statistical tests, and (b) the coverage or span of the statistical tests.

The independence of statistical tests seeks to determine whether or not there is any redundancy in applying more tests than are indeed necessary. The coverage or span of the statistical tests seeks to address the problem of how many distinct types of *non-randomness* can be investigated, and to assess whether or not we have a sufficient number of statistical tests to detect any deviation from randomness? To address this problem, research is underway which involves the application of *principal components analysis*[13]. The results seem promising and suggest that NIST has a test suite that contains a nearly independent set of tests.

## Acknowledgments

The author wishes to thank Andrew Rukhin, Mark Vangel, Elaine Barker, Miles Smid, Donna Dodson, and Lawrence Bassham, all members of the *Information Technology Laboratory* at the *National Institute of Standards & Technology* who provided guidance and useful comments on the scope of this paper.

## Summary and Conclusion

Random number generators are an important link in the computer security chain. They are very important in the construction of encryption keys and other cryptographic algorithm parameters.

In this paper, we have introduced new metrics, which may be employed to investigate the randomness of cryptographic RNGs and thus gain additional confidence that random number generators are acceptable from a statistical point of view.

We have described sources of statistical tests, discussed the different evaluation techniques, illustrated numerical experiments conducted utilizing NIST statistical tests, and addressed the problem of analysis of empirical results.

New statistical tests need to continuously be developed to gather evidence that RNGs are of high quality. The NIST Statistical Test Suite is applicable to both software and hardware based RNGs. In addition, the usage of statistical testing can be employed to gain assurance in the proper implementation of cryptographic algorithms in software.

[13] In this context, principal components analysis refers to a methodology to assess correlations among the statistical tests.

## References


[1] Robert W. Baldwin, *"Preliminary Analysis of the BSAFE 3.x Pseudorandom Number Generators,"* **RSA Laboratories Technical Bulletin**, Number 8, September 3, 1998.

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

[3] Helen Gustafson et. al., *"A computer package for measuring strength of encryption algorithms,"* **Journal of Computers & Security**, vol. 13, no. 8, 1994, pp. 687-697.

[4] Donald Knuth, **The Art of Computer Programming, Seminumerical Algorithms**, Volume 2, 3rd edition, Addison Wesley, Reading, Massachusetts, 1998.

[5] A. K. Leung and S. E. Tavares, *"Sequence Complexity as a Test for Cryptographic Systems,"* **Advances in Cryptology - CRYPTO '84 Proceedings**, Lecture Notes in Computer Science Vol. 196, G. Goos & J. Hartmanis editors, Springer-Verlag, 1984.

[6] Ueli Maurer, *"A Universal Statistical Test for Random Bit Generators,"* **Journal of Cryptology**, Vol. 5, No. 2, 1992, pp. 89-105.

[7] Alfred Menezes, et. al., **Handbook of Applied Cryptography**, CRC Press, 1997.

[8] George Marsaglia**, DIEHARD** Statistical Tests: http://stat.fsu.edu/~geo/diehard.html.

[9] Sibylle Mund, *"Ziv-Lempel Complexity for Periodic Sequences and its Cryptographic Application,"* **Advances in Cryptology – EUROCRYPT '91,** Lecture Notes in Computer Science, Springer-Verlag, 1991.

## Appendix. Hypothesis Testing Terminology

| Statistical Term | Definition |
|---|---|
| *test statistic* | A statistic upon which a test of a hypothesis is based. |
| *null hypothesis* | The stated hypothesis. In this case, the null hypothesis is that a binary sequence is random. |
| *alternative hypothesis* | An alternate hypothesis. In our case any non-random characteristic. |
| *significance level* | Usually denoted as, $\alpha$, it is the least upper bound of the probability of an error of type I for all distributions consistent with the null hypothesis. In our case $\alpha \in [0.001, 0.01]$. |
| *type I error* | The likelihood that a test rejects a binary sequence, that was in fact, produced by an acceptable random number generator. |
| *confidence interval* | An interval which is believed, with a pre-assigned degree of confidence, to include the particular value of some parameter being estimated. |

## BIOGRAPHY

In 1991, Juan earned his BS in Computational Mathematics from the *University of Puerto Rico at Humacao*. In 1993, he went on to earn his MS in Applied Mathematics from the *State University of New York at Stony Brook*. In the 1993-1994 academic year, Juan was a mathematics instructor at *Catonsville Community College* in Catonsville, MD. In 1996, he earned an MS in computer science from the *University of Delaware*. Prior to joining the Computer Security Division at NIST he was employed as a software engineer at *Lockheed-Martin Management & Data Systems* in Valley Forge, PA, conducting research and development in computer vision and image understanding. At NIST, Juan has been collaborating with a technical working group on developing statistical tests to evaluate the randomness of binary sequences. His current research interests are in cryptography, applied mathematics and computer algebra.