Granger-inspired Predictive Structure Test for RBG/RNG Evaluation

Joshua H. Sylvester*, Micah A. Thornton†, Mitchell A. Thornton*, and Eric C. Larson*
*Darwin Deason Institute for Cybersecurity, Southern Methodist University, Dallas, TX, USA
†Department of Mathematics, Texas Woman's University, Denton, TX, USA
{jsylvester, mitch, eclarson}@smu.edu, mthornton11@twu.edu

***Abstract*—New standards for post-quantum cryptography rely on new methods to evaluate the quality of random bit generators. Existing approaches largely rely on correlation-based tests, which are limited in their ability to capture dependence. In particular, they do not directly assess independence or reveal potential cause-and-effect relationships within the sequence. To address this gap, we introduce a methodology for evaluating random generators that draws on the concept of causality as a measure of predictability. We introduce the Granger-inspired Predictive Structure Test, which evaluates subsequences of a bitstream by comparing a logistic regression model against a null model, enabling the detection of predictive dependencies that correlation-based methods overlook. We show that, under assumptions aligning our method with a Granger causality test, the method detects long-term predictive dependencies and identifies non-randomness at a significantly higher rate than widely used statistical tests.**

***Index Terms*—Cryptography, Bitstreams, Causality, Random Number Generation**

I. Introduction

In December 2024, the U.S. National Institute of Standards and Technology (NIST) released FIPS 203, 204, and 205, establishing standards for "Post-Quantum Cryptography" (PQC), with a fourth standard expected soon [1]–[3]. PQC is designed to guard against cryptographic system attacks from quantum computers (QC) that are known to be effective against "Public Key Infrastructure" (PKI). While no "Cryptographically Relevant Quantum Computer" (CRQC) yet exists, experts anticipate emergence within five to ten years [4]. This timeline necessitates a worldwide migration from PKI to PQC.

A critical component of PQC is the use of random bit and number generators (RBG/RNG) with significantly higher quality and performance than those supporting PKI [5]. An additional requirement is that, to facilitate the migration from PKI to PQC, new RBG/RNGs must guard against *both* conventional and QC-based attacks, particularly the class of cyber-attacks known as random number generator attacks (RNGA) [6]. Although the idea of a high-quality RBG/RNG is intuitive, formally establishing quality metrics remains a long-standing challenge, as it is impossible to prove that a sequence of values is or is not truly random [7]. Existing tests typically assess whether output distributions are uniform or whether sequences exhibit statistical anomalies [8]–[12]. The NIST Statistical Test Suite (NIST-STS) represents one of the most widely adopted frameworks for randomness evaluation, providing a collection of statistical tests designed to detect various forms of non-randomness [5]. However, these tests focus largely on probability mass function properties, leaving other critical aspects, such as independence, less directly examined.

In the context of an RNG stream, independence requires that past bits provide no predictive information about future bits. Violations of these properties can indicate exploitable structure in RBG/RNG output. Current correlation-based tests detect certain associations between values, but correlation alone does not establish causation. A more general vulnerability regarding independence arises if past subsequences *cause* future subsequences, enabling prediction beyond chance. Such causal dependencies are especially concerning in PQC contexts, where even subtle predictability undermines security [6]. Due to the lack of tests that focus on independence, we are motivated to devise new tests based on causal inference. While detecting generalized cause-and-effect is as elusive as detecting true randomness, causal inference methods have been devised under various assumptions, including in economics [13], medical trials [14], [15] and artificial intelligence [16].

Of particular interest to this work is Granger causality (GC). Originally proposed by Clive Granger, GC tests whether a time series x is useful in forecasting another time series y [13]. Put simply, when forecasting a variable y, the framework tests whether a linear model fit to *both* past values of x and y exhibits better performance than a linear model fit only to past values of x. In the RBG/RNG context, a stream can be viewed as a concatenation of stochastic time series observations, making the Granger causal model a natural conceptual foundation for developing a new causal inference test to assess RBG/RNG supporting the PQC standards.

Motivated by the GC framework, we ask whether knowledge of past subsequences of a bitstream significantly improves prediction of future subsequences. We term this approach the Granger-inspired *Predictive Structure Test* (PST). Our central research question is whether the PST can detect a statistically significant improvement in predictive power over a null model. Such an improvement would reveal structure and identify deviations from randomness that traditional RNG/RBG tests may fail to capture.

II. Methodology

The PST fits two logistic regression models and compares them via a log-likelihood ratio (LLR) test. The first model is a *null* (constant-only) model, which serves as the baseline for

comparison, and the second is a *full* model fit to the actual data that incorporates both immediately preceding bits and longer-lagged bits.

A. Bitstream Tabulation Procedure

Because a bitstream's format is not inherently suitable for logistic regression, it must first be transformed into a tabular dataset. We accomplish this by performing a sliding-window procedure across the bitstream, generating one table row for each window position. The procedure uses three windows: (1) **Target**: the bit to predict, (2) **Recent**: bits immediately preceding the target bit with potential short-term influence, (3) **Offset**: earlier bits with potential long-term influence.

This procedure is parameterized by the window-size (w_{pst}), which is the length (in terms of bits) for both the recent and offset windows, and the offset (o_{pst}), which is the number of windows separating the end of the offset window from the start of target window. In practice, $o_{\text{pst}} \geq 1$ to avoid duplicating the recent window.

At each step, bits from the recent and offset windows are concatenated and added as a row of predictors to a table. The resulting dataset contains $2 \times w_{\text{pst}}$ predictor bits and a single target bit. The windows are then shifted forward by one bit, repeating the process until the end of the bitstream is reached.

B. Testing Procedure

The null model is a constant-only regression model fit to the target bits, y_i. Theoretically, the log-likelihood of the null model can be computed directly using the maximum likelihood estimate of the success probability, $\hat{p} = \frac{1}{n}\sum_{i=1}^{n} y_i$, yielding:

$$\ell_{\text{Null}} = \sum_{i=1}^{n} \left[y_i \log(\hat{p}) + (1 - y_i) \log(1 - \hat{p}) \right]. \tag{1}$$

Shown in eq. (2), the full model is a logistic regression that uses all predictive bits (from both the recent and offset windows) as well as a learned constant to predict the corresponding target bit. Here, σ denotes the sigmoid function, and $X_{\text{Tar.}}$, $X_{\text{Rec.}}$, and $X_{\text{Off.}}$ represent the target bit, recent bits, and offset bits, respectively.

$$P(X_{\text{Tar.}} = 1 \mid X_{\text{Rec.}}, X_{\text{Off.}}) = \sigma\Big(\alpha + \textstyle\sum_{j=1}^{w} \beta_j X_{\text{Rec.},j} + \sum_{j=1}^{w} \gamma_j X_{\text{Off.},j}\Big) \tag{2}$$

The log-likelihoods of the null and full models are then compared using a one-tailed LLR test:

$$H_0 : \ell_{\text{Null}} \geq \ell_{\text{Full}}, \qquad H_A : \ell_{\text{Null}} < \ell_{\text{Full}} \tag{3}$$

$$\lambda = -2 \times (\ell_{\text{Null}} - \ell_{\text{Full}}) \tag{4}$$

$$\lambda \sim \chi^2(2w_{\text{pst}}) \tag{5}$$

where ℓ_{Null} and ℓ_{Full} are the log-likelihoods of the null and full models, respectively, and $2w_{\text{pst}}$ is the number of additional parameters in the full model relative to the null model.

Our approach draws inspiration from Granger causality, which tests whether incorporating lagged values of one time series improves prediction of another beyond what can be achieved using only the target series' own history. In traditional Granger causality, this involves comparing a restricted autoregressive model against an unrestricted model that includes additional predictive variables. Our PST adapts this framework to the bitstream context: rather than comparing against an autoregressive baseline, we compare against the fundamental assumption of randomness itself—represented by our null model. This comparison allows us to detect any predictive structure in the bitstream, whether from recent or offset windows, that deviates from true randomness.

III. Dataset

To establish a dataset, we retrieved a 10-million-bit sample from the NIST Randomness Beacon [17]. This unmodified bitstream serves as our high-quality random baseline, which we then modify by introducing controlled dependencies to assess whether our method detects predictability when present. We define two parameters for this overwrite procedure: the overwrite window size w_{ow} and overwrite offset o_{ow}. These parameters govern the spatial arrangement of the procedure: w_{ow} specifies the number of bits used to compute the overwrite probability at each step, and o_{ow} determines the separation between the context window and the future bit to be overwritten.

To control the strength and pattern of the induced dependencies, we randomly initialize a coefficient vector of length w_{ow}. For each overwrite location, the following procedure is applied exactly w_{ow} times:

1) Extract the current w_{ow} bits from the bitstream (context window) and compute their dot product with the coefficient vector, adding a constant bias term of 1.0.
2) Pass this value through a sigmoid function to obtain an overwrite probability.
3) Compare this probability against a pre-generated random threshold in $[0, 1]$.
4) If the probability exceeds the threshold, overwrite the target bit—located $w_{\text{ow}} \times o_{\text{ow}}$ positions ahead of the end of the context window—with a 1; otherwise, overwrite it with a 0. E.g., given $w_{ow} = 8$ and $o_{ow} = 6$ when the context window includes bit positions [0-8), the target bit to be overwritten is at bit position 56.
5) Shift the context window forward by one bit and repeat until w_{ow} consecutive overwrites have been performed.

Overwrite locations are non-overlapping with randomized spacing to prevent alignment effects and periodic patterns.

For the experimental datasets, we vary the total number of overwrite locations from 0 to 3000 in increments of 50. For each setting, we generate 30 independently modified bitstreams by re-sampling the coefficient vector for each run. This design allows us to systematically evaluate the sensitivity of our method to increasing amounts of injected structure, while maintaining statistical robustness through repeated trials. We note that the number of overwrites is small compared to the total length of the bitstream— 3000 overwrites when $w_{ow} = 8$ only represents 0.24% of the bitstream.

TABLE I: Recall rates for detecting non-randomness in bitstreams with varying numbers of predictive overwrites.

Method	**Recall versus number of overwritten sequences** (For each cell, n = 30)									
	300	600	900	1200	1500	1800	2100	2400	2700	3000
Predictive Structure Test (ours)	0.00	0.00	**0.30**	**0.57**	**0.63**	**0.80**	**0.97**	**1.00**	**1.00**	**1.00**
Approximate Entropy	0.00	0.00	0.00	0.00	0.00	0.13	0.13	0.17	0.23	0.27
Block Frequency	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.13
Cumulative Sums Forward	0.00	0.00	0.10	0.13	0.37	0.43	0.47	0.50	0.53	0.63
Cumulative Sums Reverse	0.00	0.00	0.10	0.13	0.13	0.13	0.13	0.33	0.43	0.53
Discrete Fourier Transform	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Frequency	0.00	**0.03**	0.13	0.13	0.13	0.17	0.17	0.40	0.50	0.60
Linear Complexity	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Longest Run of Ones	0.00	0.00	0.00	0.00	0.07	0.17	0.23	0.23	0.27	0.27
Non-overlapping Template	0.00	0.00	0.17	0.30	0.33	0.43	0.43	0.43	0.43	0.43
Overlapping Template	0.00	0.00	0.17	0.23	0.30	0.33	0.33	0.37	0.37	0.37
Random Excursions	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00
Random Excursions Variant	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00
Rank	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.07	0.07
Runs	0.00	0.00	0.00	0.17	0.29	0.37	0.37	0.42	0.38	0.29
Serial	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Universal Statistical	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

IV. Results

A. Experimental Parameters

All experiments discussed in Sections IV-B and IV-C use a window size of $w_{\text{pst}} = w_{\text{ow}} = 8$ and an offset of $o_{\text{pst}} = o_{\text{ow}} = 6$. These values were selected to maintain consistency between the test parameters and the dataset generation process, ensuring that the spatial arrangement of the predictive windows in the test aligns with the injected structure in the modified datasets. We recognize that this configuration represents a best-case scenario where the test parameters are matched to the structure present. In practical applications, identifying suitable parameters would require a grid search or other parameter selection strategy. We provide a brief discussion of sensitivity in Section IV-D. In the present study, our goal is to compare the PST to the NIST-STS tests under conditions where predictability is known to be present and should be detectable. To apply the NIST-STS we utilize an implementation provided by the"STandard Entropy Evaluation Report" (STEER) framework [1] We use the default parameters specified by STEER for each of the NIST-STS tests[2].

B. Predictive Dependency Detection

As previously discussed, we conduct experiments varying the number of times a predictive dependency sequence is introduced into the bitstream and for each experiment we conduct 30 variations using different coefficients [3]. We evaluate each test's detection capability using $\alpha = 0.01$, which is standard for the NIST-STS [5]. Table I presents recall rates for each test across all experiments. PST demonstrates substantially higher recall rates compared to NIST-STS tests as the number of predictive dependencies increase. At 2400 overwritten sequences and greater, the PST achieves 100% recall compared to the next best performer, Cumulative Sums Forward (Cusums), which achieved only 63% recall at 3000 overwritten sequences. Additionally, the NIST-STS tests all show plateauing or inconsistent performance, while the PST demonstrates consistent improvement as the amount of introduced dependencies increases, reaching optimal performance.

C. Detailed Comparison with Best Competitor

Figure 1 compares the PST and Cusums by plotting the median p-values and bands for the interquartile ranges. Three key advantages emerge for our method: faster convergence to significant detection, substantially lower variance in performance across configurations, and more consistent p-values below the significance threshold at higher insertion counts.

The Cusums test shows considerable variability, with wide IQR bands indicating inconsistent performance across different coefficient configurations. In contrast, the PST exhibits narrow confidence intervals and steady performance improvement, demonstrating greater reliability for detecting the specific type of long-range predictive dependencies that traditional correlation-based methods may miss. This consistency is particularly valuable for practical randomness testing where reliable detection across diverse scenarios is essential.

[1]https://github.com/SMU-DDI/steer-framework

[2]For tests which report multiple p-values (e.g., the random excursions tests) we take the minimum p-value to give NIST-STS the best chance of detection. For the non-overlapping template matching test we use STEER's 100+ default templates and apply Fisher's combined probability to obtain a single p-value [18].

[3]Note on computation: Testing performed on an M5 Macbook Pro with 16 GB of RAM. Each PST run took $\sim$ 40 sec. and $\sim$ 6GB of RAM.

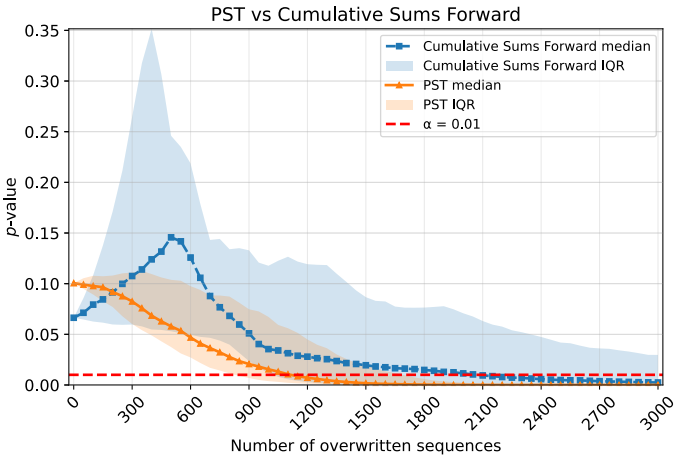

Fig. 1: Median p-values and IQRs of PST vs. Cumulative Sums Forward as 8-bit predictive dependencies increase.

Ratio	Parameters				Recall	
(w_{ow}/w_{pst})	w_{pst}	o_{pst}	w_{ow}	o_{ow}	PST	Cusums
100%	8	6	8	6	1.00	0.63
57.1%	14	3	8	6	0.80	0.47
28.6%	28	1	8	6	0.40	0.47
100%	12	6	12	6	1.00	0.70
100%	16	6	16	6	1.00	0.73
100%	20	6	20	6	1.00	0.83

TABLE II: Recall for bitstreams (n=30) with 3000 predictive sequences, using different PST parameterizations

D. Sensitivity Analysis

We provide a preliminary investigation into the sensitivity of the PST to parameterization, leaving more comprehensive analyses for future work. The PST performs best when w_{pst} and o_{pst} align with w_{ow} and o_{ow}, an ideal setting that would require a grid search to identify. To evaluate less-than-ideal cases, we test larger PST windows ($w_{pst} > w_{ow}$), as shown in the top of Table II. Here the ratio w_{ow}/w_{pst} reflects the share of predictive bits within the PST's offset window. The PST retains high recall even when the overwrite parameters are mismatched with the PST parameters. At the smallest ratio, PST recall (0.40) is only two examples lower than Cusums (0.467), indicating robustness to parameter choice. We further vary w_{ow} to examine whether the Cusums test can match PST as predictability increases. While PST achieves perfect recall, Cusums never does, indicating that predictive influence is more effectively captured by an ideally parameterized PST.

V. Conclusions

We introduced the Granger-inspired Predictive Structure Test (PST) as a method for evaluating the quality of random bit generators and random number generators (RBG/RNG). Our experiments demonstrated that when dependencies are inserted into a bitstream, the PST detects these dependencies far more frequently than methods from the NIST Statistical Test Suite. For example, on bitstreams with the largest amount of predictability, the best-performing NIST test achieved only 63% recall, whereas the PST achieved a perfect 100% recall. Although the PST can capture predictive dependencies that other tests may miss, it is sensitive to parameter choices, necessitating a grid search for optimal performance. Even so, in contexts such as post-quantum cryptography, where the quality of random bitstreams is critical, parameter searches allow the test to be fine-tuned for optimum recall.

References

[1] National Institute of Standards and Technology (US), "Module-lattice-based key-encapsulation mechanism standard," National Institute of Standards and Technology (U.S.), Washington, D.C., Tech. Rep. NIST FIPS 203, Aug. 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf

[2] ——, "Module-lattice-based digital signature standard," National Institute of Standards and Technology (U.S.), Washington, D.C., Tech. Rep. NIST FIPS 204, Aug. 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf

[3] N. I. o. S. a. Technology, "Stateless Hash-Based Digital Signature Standard," U.S. Department of Commerce, Tech. Rep. Federal Information Processing Standard (FIPS) 205, Aug. 2024. [Online]. Available: https://csrc.nist.gov/pubs/fips/205/final

[4] D. M. Mosca, "Quantum Threat TImeline Report 2024," 2024.

[5] NIST Computer Security Resource Center, "Random Bit Generation RBG," https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software, 2024, STS version 2.1.2, Accessed: 2025-03-09.

[6] M. De Bernardi, M. Khouzani, and P. Malacaria, "Pseudo-random number generation using generative adversarial networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 191–200.

[7] G. Marsaglia, "Random Number Generators," *Journal of Modern Applied Statistical Methods*, vol. 2, no. 1, pp. 2–13, May 2003. [Online]. Available: https://jmasm.com/index.php/jmasm/article/view/57

[8] R. G. Brown, "Die$Harder$: A Gnu Public License Random Number Tester," https://webhome.phy.duke.edu/~rgb/General/dieharder.php, 2005, accessed: 2025-03-09.

[9] W. Killmann and W. Schindler, "A Proposal for: Functionality Classes for Random Number Generators," *ser. BDI, Bonn*, 2011.

[10] W. Schindler, *Evaluation Criteria for Physical Random Number Generators*. New York, NY: Springer, 2009, pp. 25–54.

[11] J. Walker, "A Pseudorandom Number Sequence Test Program," https://www.fourmilab.ch/random/, 2008, accessed: 2025-03-09.

[12] G. Marsaglia and W. W. Tsang, "Some Difficult-to-Pass Tests of Randomness," *Journal of Statistical Software*, vol. 7, pp. 1–9, 2002.

[13] C. W. J. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica: Journal of the Econometric Society*, pp. 424–438, 1969.

[14] P. W. Holland, "Statistics and Causal Inference," *Journal of the American Statistical Association*, vol. 81, no. 396, pp. 945–960, 1986, publisher: [American Statistical Association, Taylor & Francis, Ltd.]. [Online]. Available: https://www.jstor.org/stable/2289064

[15] D. B. Rubin, "Bayesian Inference for Causal Effects: The Role of Randomization," *The Annals of Statistics*, vol. 6, no. 1, pp. 34–58, Jan. 1978, publisher: Institute of Mathematical Statistics. [Online]. Available: https://projecteuclid.org/journals/annals-of-statistics/volume-6/issue-1/Bayesian-Inference-for-Causal-Effects-The-Role-of-Randomization/10.1214/aos/1176344064.full

[16] J. Pearl, *Causality*, 2nd ed. Cambridge: Cambridge University Press, 2009. [Online]. Available: https://www.cambridge.org/core/books/causality/B0046844FAE10CBF274D4ACBDAEB5F5B

[17] NIST Computer Security Division – Information Technology Laboratory, "NIST Randomness Beacon (Version 2.0 Beta) - Interoperable Randomness Beacons | CSRC | CSRC," Jun. 2019. [Online]. Available: https://csrc.nist.gov/Projects/interoperable-randomness-beacons/beacon-20

[18] R. A. Fisher, *Statistical methods for research workers*. Edinburgh, Oliver and Boyd, 1925, vol. 1. [Online]. Available: http://archive.org/details/statisticalmethoe7fish