# Random Number Generation Based on Cryptocurrency Prices and Linear Feedback Shift Register

Ayumu Chiba and Shuichi Ichikawa
*Department of Electrical and Electronic Information Engineering*
*Toyohashi University of Technology*
Toyohashi 441-8580, Japan
ichikawa@tut.jp

***Abstract*—This paper presents the design and evaluation of an unpredictable random number generator (URNG) utilizing cryptocurrency prices. A URNG operates using a deterministic algorithm, while utilizing external entropy sources to generate random numbers that are practically unpredictable. The proposed URNG employs a linear feedback shift register (LFSR), whose sampling period are fluctuated by the cryptocurrency price $P_t$ or its logarithmic return $R_t$. Using Bitcoin (BTC) price data, we simulated the proposed URNG and evaluated its randomness with the Diehard test. Our findings suggest that a 40-bit or longer LFSR, with sampling period fluctuations determined by either the least significant bit of $P_t$ or the comparison of $R_t$ to its average value, achieves satisfactory randomness quality. Future work includes evaluating the URNG with other cryptocurrencies and exploring new methods for entropy extraction.**

***Index Terms*—URNG (Unpredictable Random Number Generator), LFSR (Linear Feedback Shift Register), EMH (efficient-market hypothesis), bitcoin**

## I. Introduction

Random numbers are essential in many applications, including security and simulations. Random numbers are typically classified into two categories: True Random Number (TRN) and Pseudo-Random Number (PRN).

TRN is generated from physical phenomena, such as thermal noise or metastability, and is inherently unpredictable. Since TRN relies on physical phenomena, dedicated hardware called TRNG (True Random Number Generator) is essential to generate TRN [1].

PRN, on the other hand, is generated from a deterministic algorithm and an initial value, which means that PRNG (Pseudo-Random Number Generator) can be implemented in either software or hardware. The primary drawback of PRN is that its future values can be predicted by deducing its internal states and generation algorithm.

Suciu et al. [2] introduced the concept of Unpredictable Random Number (URN), which exhibits characteristics between those of PRN and TRN. URN is generated using a deterministic algorithm but incorporates external entropy sources to ensure practical unpredictability. Suciu et al. generated URNs using the performance counters of a microprocessor and the background process activities.

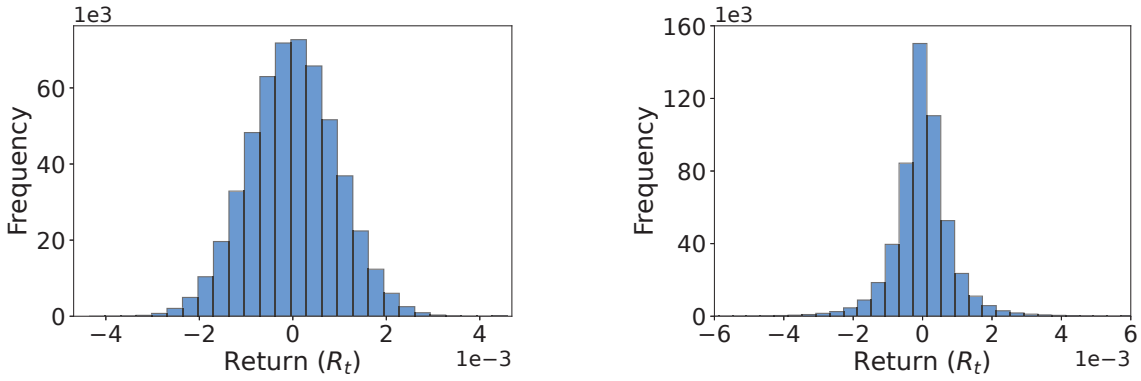
Masaoka et al. [3] proposed a URNG (Unpredictable Random Number Generator) that sequentially samples a built-in LFSR (Linear Feedback Shift Register) in a microprocessor. In this LFSR-based URNG, the least-significant 32 bits of a 128-bit LFSR were sampled as URN. The entropy source for this URNG is the fluctuation of sampling intervals, typically caused by interrupts. The generated sequence of URNs passed the Diehard test [5], although the design requirements of their URNG were not thoroughly discussed. Kamogari and Ichikawa [4] examined the randomness quality of LFSR-based URNG through extensive simulations, elucidating the relationships between the design parameters and the randomness quality.

By utilizing the external entropy source, a URNG can be implemented solely in software. Chiba and Ichikawa [6] proposed using meteorological data as an entropy source for URNG. Meteorological observation data are publicly available on the Internet, with new data uploaded continuously. These data, generated from physical phenomena, are usable as entropy sources. Chiba and Ichikawa [6] generated URNs by using wind direction data to vary the sampling period of an LFSR-based URNG. The derived URNs passed both the Diehard test [5] and the NIST test [7].

The present study examines using cryptocurrency price as an entropy source for an LFSR-based URNG. Cryptocurrencies represent digital assets primarily used as a medium in exchange, employing cryptography to secure transactions and control the creation of new units [8]. In essence, cryptocurrencies are digital money, exchangeable with other currencies in the market, with exchange rates that fluctuate continuously like those in the foreign exchange market.

While stock prices could also be considered, their data are often commercially provided as a paid service with various restrictions. In contrast, real-time cryptocurrency data are freely available and open to the public, making them suitable for research purposes. Additionally, the data rate of cryptocurrency data[1] is much higher than the 4 bits per hour derived

[1]In case of Bitcoin price, 5 decimal digits are derived for each minute.




(a) Simulation

(b) Sample data

Fig. 1: Histogram of $R_t$ (BTC)

from wind direction data [6], suggesting that the generation rate could be significantly higher.

The rest of this paper is organized as follows. Section II introduces the background and related studies of this work. Then, Section III outlines the design of a URNG that utilizes LFSR and cryptocurrency prices, and Section IV presents the evaluation results. We conclude this work in Section V.

## II. Background

### A. Efficient Market Hypothesis

Efficient Market Hypothesis (EMH) is a hypothesis in financial economics that posits: (1) the stock price fully and fairly reflects all available information about that stock, and (2) the stock price should exhibit the appearance of a memoryless random walk [10].

Many studies investigated market efficiency by analyzing the behavior of stock or currency prices. The following are some examples of such studies closely related to the current research.

Doyle and Chen [11] applied a randomness test for RNGs to stock market movements to test market efficiency. They reported that most markets exhibit idiosyncratic recurrent patterns, contradicting the EMH.

Noakes and Rajaratnam [12] used the overlapping serial test to assess market efficiency on the Johannesburg Stock Exchange. They found a high degree of non-randomness in small cap stocks and observed that many stocks exhibited inefficiency during the crisis period.

Noda [13] measured the degree of market efficiency in Japanese stock markets using a time-varying model approach. Tran and Leirvik [14] improved Noda's method by introducing a new measure that facilitates the comparison of market efficiency across assets, time periods, regions, and data frequencies. Tran and Leirvik first investigated Japanese stock markets [14], and later reported the efficiency of cryptocurrency markets [15]. They found that the cryptocurrency markets were mostly inefficient before 2017 but became more efficient over the period 2017–2019 [15].

As demonstrated by the aforementioned studies, cryptocurrency markets are not fully efficient; however, they are partially efficient and have been becoming more efficient in recent years. Therefore, they are regarded as a promising source of entropy for a URNG.

This study investigates the URNG which utilizes the behavior of cryptocurrency data.

### B. Random Walk Model

In a market where EMH holds, the market price is expected to follow a random walk model. Under this situation, the price can be modeled by the following equation, known as the Geometric Brownian Motion (GBM) model.

$$dP_t = \mu P_t dt + \sigma P_t dW_t \quad (1)$$

$$\mu = \bar{R}_t + \sigma^2/2 \quad (2)$$

In Equations (1) and (2), $P_t$ and $R_t$ represent the price and the logarithmic return at time $t$, respectively. $W_t$ is a Wiener process where $W_t \sim N(0,t)$. The parameter $\mu$ represents growth rate, while $\sigma$ represents volatility, or the standard deviation over the sample period. $\bar{R}_t$ in Eq.(2) is the average of $R_t$ over the sample period.

The stochastic differential equation in Eq.(1) is given by Eq. (3).

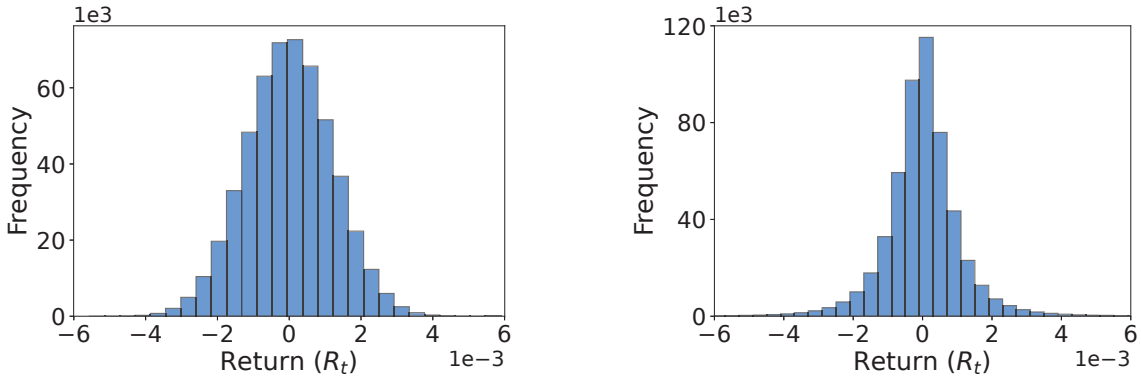
$$P_t = P_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right) \quad (3)$$

Taking the logarithm of both sides, we derive the next equation. Since $P_t$ follows the normal distribution, $\ln P_t$ follows a log-normal distribution.

$$\ln P_t = \ln P_0 + \left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t \quad (4)$$

Similarly, $R_t$ is given by Eq.(5), which follows the normal distribution $R_t \sim N\left(\left(\mu - \sigma^2/2\right)\Delta t,\ \sigma^2 \Delta t\right)$.

$$R_t = \ln\left(\frac{P_{t+1}}{P_t}\right) = \left(\mu - \frac{\sigma^2}{2}\right)\Delta t + \sigma \Delta W_t \quad (5)$$




(a) Simulation

(b) Sample data

Fig. 2: Histogram of $R_t$ (ETH)

TABLE I: Simulation parameters

| | $\mu$ | $\sigma$ |
|---|---|---|
| BTC | -1.646155173e-6 | 0.938358125e-3 |
| ETH | -1.189055401e-6 | 1.201967413e-3 |

### C. Simulation vs. Market Data

In this subsection, the output of the GBM model is compared with actual market data.

Firstly, $P_t$ is simulated every minute for one year, making $\Delta t = 1$. Cryptocurrency prices are retrieved using python library Historic Crypto 0.1.6 [16]. The sampling period was set from 2021-11-01T00:00 to 2022-10-31T23:59, and the closing prices at each minute were used. The values of $\mu$ and $\sigma$ are shown in Table I.

The simulation results and sample data of Bitcoin (BTC) and Ethereum (ETH) are shown in Figures 1 and 2, respectively. Although the sample data do not follow a perfect normal distribution, the distribution is almost symmetrical with respect to the average $\bar{R}_t$ in both cases. By setting an appropriate threshold and comparing the sample to the threshold, it is expected that a certain amount of entropy can be extracted from cryptocurrency prices.

It should be noted that stablecoins are not suitable for entropy extraction due to their minimal price movement. Figure 3 presents the distribution of $R_t$ of USDC, a type of stablecoin. This distribution is far from normal, and is not reliable as an entropy source.

Due to page limits, only the price of BTC is examined in the following discussions.

## III. Design of URNG

As stated in the previous section, the market price should exhibit the appearance of a memory-less random walk if the market is efficient [10]. This naturally suggests that market price data can be used as an entropy source for an LFSR-based URNG. This section describes the design of an LFSR-based URNG using market price data.

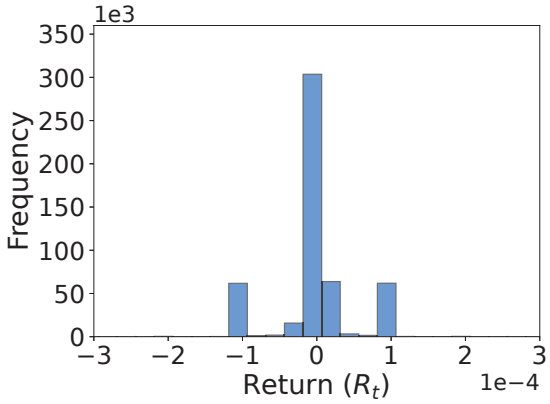



Fig. 3: Histogram of $R_t$ (USDT)

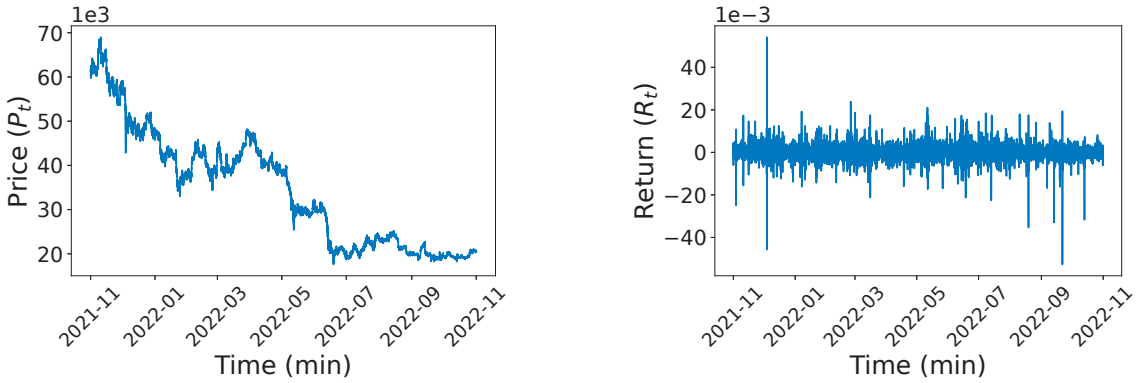
### A. Principles of LFSR-based URNG

An LFSR is a type of PRNG, which is defined as a shift register whose input is given by a linear function of its previous state (i.e., feedback polynomial). The period of an $n$-bit LFSR becomes maximal ($2^n - 1$), if the feedback polynomial is selected to be primitive. LFSRs are utilized in various applications, including communication systems and built-in-self-test of LSIs. LFSR is also acknowledged as a fast and low-cost PRNG suitable for hardware implementation.

The operation principle of an LFSR-based URNG is straightforward. After the $(n-1)$-th sampling of the LFSR, the feedback polynomial is applied $S(n)$ times before the $n$-th sampling. The sampling interval $S(n)$ is given by the following equation, where $\beta$ is a constant (Base Period) and $\alpha(n)$ is the fluctuation factor.

$$S(n) \quad = \quad \alpha(n) + \beta \tag{6}$$

In this study, $\alpha(n)$ is derived from $P_t$ or $R_t$, as stated below. In the following discussion, the array $data$ contains the time series of $P_t$ or $R_t$, and $N$ represents the size of the array. The array $data$ is used cyclically when $n > N$.




(a) $P_t$ ($t_{max} = 525491$) (b) $R_t$ ($t_{max} = 525490$)

Fig. 4: Time-series data at 1 minute time interval from 2021-11-01 to 2022-10-31 (BTC)

Figure 4 presents the time-series data of (a) $P_t$ and (b) $R_t$. As can be easily seen, the characteristics of $P_t$ and $R_t$ are much different. Thus, different methods are required for $P_t$ and $R_t$ to harvest entropy for URNG.

### B. Use of market price

Though $P_t$ in the GBM model follows a normal distribution, the actual $P_t$ behaves differently. As seen in Fig. 4(a), $P_t$ is almost consecutive and drifts following a trend. In this case, it is difficult to extract entropy from the value itself or the upper digits of $P_t$. Therefore, we use the least significant bit (LSB) of $P_t$ as $\alpha(n)$ in the following discussion (Eq.(7)).

$$\alpha(n) \quad = \quad \mathrm{int}\left(data[n \bmod N] \times 10^2\right) \bmod 2 \qquad (7)$$

Since each $P_t$ has two decimal digits under the decimal point, $10^2$ is multiplied before taking its modulo 2.

While it might be possible to extract two ore more bits from each $P_t$, it is left for future studies. In this work, we extract a single bit from each $P_t$ as a baseline of the evaluation.

### C. Use of logarithmic return

As seen in Fig. 4(b), $R_t$ fluctuates around its average value without drift. Though its distribution is not perfectly normal (Fig. 1(b)), it is almost symmetrical. Thus, we utilize this symmetry to extract a single bit from each $R_t$ value.

The first idea is to compare $R_t$ with zero (Eq. (8)), while the second idea uses the average value of $R_t$ instead of zero (Eq. (9)).

$$\alpha(n) \quad = \quad \begin{cases} 1 & (data[n \bmod N] > 0) \\ 0 & (data[n \bmod N] < 0) \end{cases} \qquad (8)$$

$$\alpha(n) \quad = \quad \begin{cases} 1 & \left(data[n \bmod N] > \bar{R_t}\right) \\ 0 & \left(data[n \bmod N] < \bar{R_t}\right) \end{cases} \qquad (9)$$

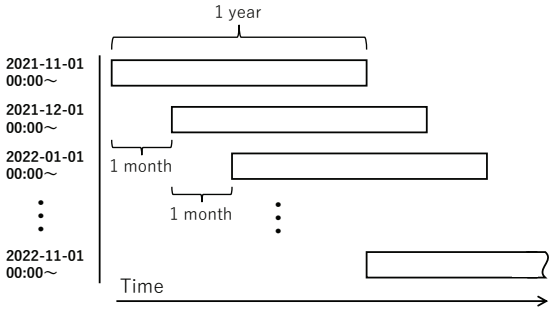



Fig. 5: Rolling window sample period

The above two ideas split the domain into two zones (quantiles), while we can further split the domain into four quantiles (Eq. (10) and Eq. (11)).

$$\alpha(n) \quad = \quad \begin{cases} 1 & (Q_1 \le data[n \bmod N] \le Q_3) \\ 0 & (otherwise) \end{cases} \qquad (10)$$

$$\alpha(n) \quad = \quad \begin{cases} 0 & (data[n \bmod N] < Q_1) \\ 1 & (Q_1 \le data[n \bmod N] < Q_2) \\ 0 & (Q_2 \le data[n \bmod N] < Q_3) \\ 1 & (Q_3 \le data[n \bmod N]) \end{cases} \qquad (11)$$

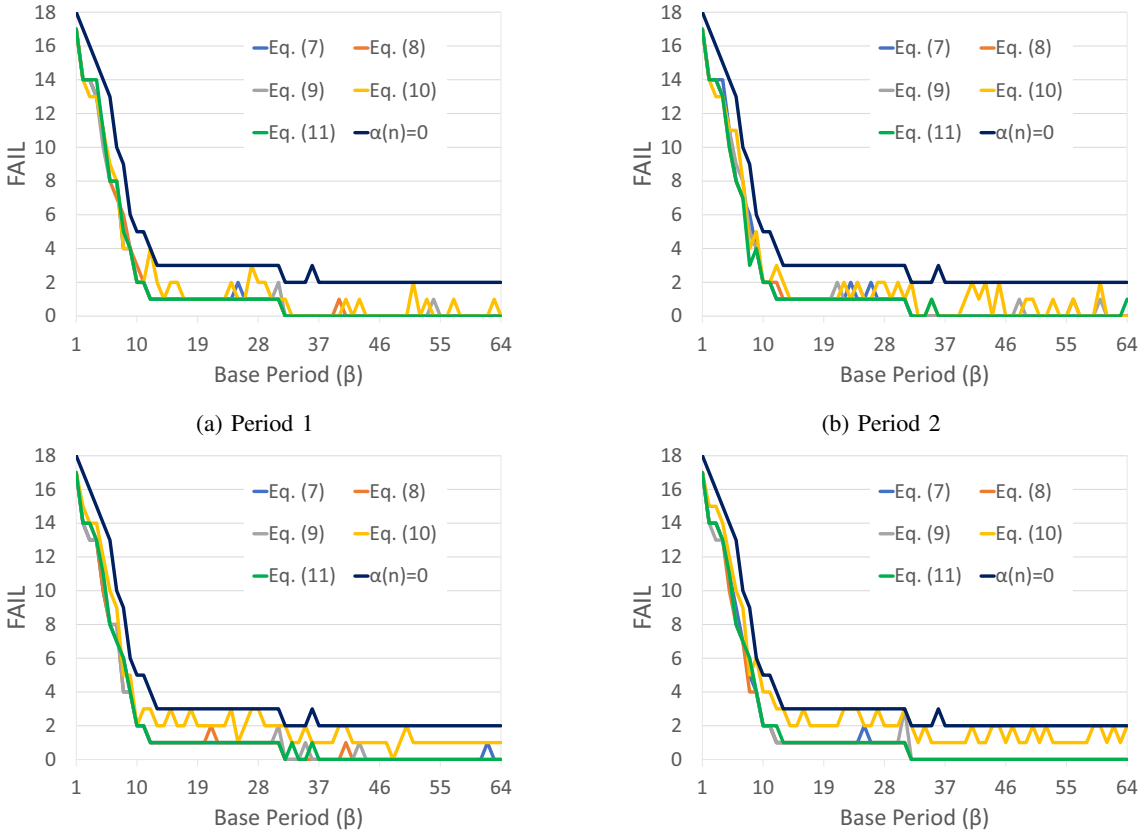
In the above equations, $Q_1$, $Q_2$, and $Q_3$ represent the first quartile (25th percentile), the second quartile (50th percentile), and the third quartile (75th percentile), respectively. In Equations (10) and (11), the expected value of $\alpha(n)$ becomes 0.5.

## IV. Evaluation

### A. Diehard test

As in previous studies [4] [6], this study adopts the Diehard test to evaluate the randomness quality.

The Diehard test [5] consists of 18 individual tests, each of which outputs 1–100 p-values (313 p-values in total). If the sequence under test is random, the p-values should show a uniform distribution between 0 and 1. Although the overall interpretation of these results is not standardized, this study follows the evaluation criteria adopted in the previous




(a) Period 1

(b) Period 2

(c) Period 3

(d) Period 4

Fig. 6: DIEHARD test results on BTC with a 32-bit LFSR

TABLE II: Sample period (BTC)

| | Period |
|---|---|
| Period 1 (max $\Delta P$) | 2021-11-01T00:00〜2022-10-31T23:59 |
| Period 2 (max $\sigma$) | 2021-12-01T00:00〜2022-11-30T23:59 |
| Period 3 (min $\Delta P$) | 2022-07-01T00:00〜2023-06-30T23:59 |
| Period 4 (min $\sigma$) | 2022-10-01T00:00〜2023-09-30T23:59 |

studies [4] [6]. According to these criteria, the results of the 18 tests are categorized into three groups: PASS, WEAK, and FAIL.[2] In the following discussion, we focus on the number of FAILs out of the 18 tests.

### B. Evaluation periods

Another issue resides in the difference among the sampling periods; i.e., the market is calm in some periods, while stormy in other periods. Therefore, we checked the price range $\Delta P$ and the volatility $\sigma$ of the 1-year period with various start dates (Fig. 5), and picked up four periods with max $\Delta P$, max $\sigma$, min $\Delta P$, and min $\sigma$. These periods (Period 1 to 4), listed in Table II, are used for the following evaluations.

### C. 32-bit LFSR

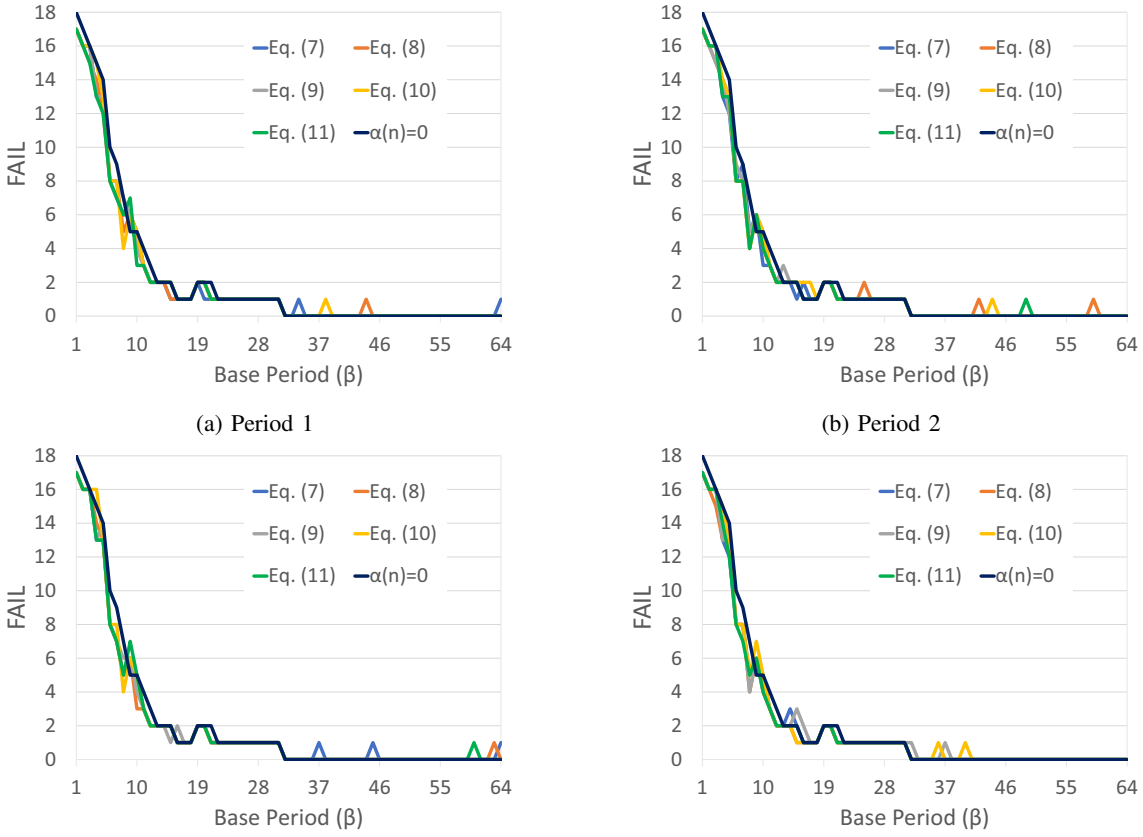
Figure 6 summarizes the Diehard test results of the 32-bit LFSR, whose tap sequence is [32, 30, 17, 12, 3, 1] [17]. Four graphs correspond to four periods in Section IV-B and Table II. The X-axis of each graph corresponds to the Base Period ($\beta$) in Eq. (6), while the Y-axis shows the number of FAILs in the Diehard test. Five lines correspond to the results where $\alpha(n)$ is given by Eq.(7)–Eq.(11). The last line ($\alpha(n) = 0$) represents the case without fluctuation in the sampling period $S(n)$.

In any periods, the results of small $\beta$ yield many FAILs, and the number of FAILs decreases as $\beta$ increases. As an LFSR is a simple PRNG, it cannot pass the randomness test as it is (i.e., $\beta = 1$). However, by applying the feedback polynomial repeatedly, the randomness quality rises as $\beta$ increases.

In case of $\alpha(n) = 0$, the generated sequence cannot pass the Diehard test even with large $\beta$. The sequence of $\alpha(n) = 0$ is a fixed-stride sampling of the output of a 32-bit LFSR; even with its longest period $2^{32} - 1$, a 32-bit LFSR is impossible to pass the Binary Rank 31x31 and 32x32 tests of the Diehard test. In contrast, the randomness quality was improved by adding the fluctuation with Eq.(7)–Eq.(11). For $\beta > 32$, it passes the Diehard test with no FAIL in some cases, though it still yields some FAILs in other cases.

It is notable that Eq. (10) yields more FAILs than Equations (8), (9), and (11), where the results of these three equations are almost comparable.

[2]For more details, refer to the previous studies [4] [6].




(a) Period 1

(b) Period 2

(c) Period 3

(d) Period 4

Fig. 7: DIEHARD test results on BTC with a 40-bit LFSR

It is inferable from the previous study [4] that longer LFSRs will result in higher randomness quality. Thus, we examine a 40-bit LFSR in the next section.

### D. 40-bit LFSR

Figure 7 presents the Diehard test results of the 40-bit LFSR (tap sequence [40, 29, 21, 10] [18]), where the least significant 32 bits are used as a URN in each sample.[3] Compared to Fig. 6, it is evident that the number of FAILs decreased. Notably, the URNG with a 40-bit LFSR can pass the Diehard test even without fluctuation ($\alpha(n) = 0$), although it is a PRNG rather than a URNG. The qualities of Eq. (7) with $P_t$ and Eq. (8)–(11) with $R_t$ are almost comparable and acceptable.

However, there are still some FAILs left in $\beta \geq 32$. Most of these FAILs were caused by OPERM5 test, which is known to be buggy [19] and often reports peculiar FAILs [4]. In this study, FAILs of OPERM5 are disregarded, following Brown's advice [19]. Except for OPERM5, there were only five FAILs summarized in Table III) for ($\beta \geq 32$). These FAILs appear in various periods with different equations, and no specific tendency is observed. Moreover, it is expected that these FAILs will diminish by using a longer LFSR [4].

[3]Upper bits are concealed as in Masaoka et al. [3]

TABLE III: FAILed tests in Figure 7 ($\beta \geq 32$) except for OPERM5 test.

| Period | Eq. | $\beta$ | FAILed test |
|---|---|---|---|
| Period1 | (7) | 64 | Squeeze |
| Period2 | (8) | 59 | Overlapping Sums |
| Period3 | (7) | 37, 45, 64 | Overlapping Sums |
| Period4 | (9) | 32 | Overlapping Sums |

### E. Summary of Evaluation

Eq. (7) with $P_t$ yields almost acceptable randomness quality; particularly, no FAILs were observed with 32-bit LFSR for $\beta > 32$. Since its implementation is simple and straightforward, it is recommended for use with $P_t$.

Eq. (10) and (11) are more complex than Eq. (8) and (9), while the randomness quality is slightly lower or equal. In short, Eq. (10) and (11) have no evident advantages over Eq. (8) and (9). In this sense, there are no special reasons to select Eq. (10) and (11).

Eq. (8) and (9) present similar qualities, but it might be caused by the fact that $\bar{R}_t \sim 0$ in our evaluation conditions. Considering the generality, the use of Eq. (9) might be more reasonable.

## V. Conclusion

This study examined a URNG based on cryptocurrency prices, and presented the evaluation results of its randomness quality. The design of this URNG is based on an LFSR-based URNG [3] [4] [6], where the fluctuation of sampling periods is derived from the cryptocurrency price $P_t$ and its logarithmic return $R_t$. The evaluation was performed for four sampling periods based on volatility $\sigma$ and price range $\Delta P$.

From our evaluation results on BTC, we suggest to use a 40-bits or longer LFSR and the fluctuation of sampling periods to be determined by (1) the least significant bit of $P_t$ or (2) the comparison of $R_t$ with its average value.

Our future work includes (1) the evaluation using the NIST test, (2) the investigation of other cryptocurrencies, and (3) the examination of new methods to harvest entropy in cryptocurrency price data.

## Acknowledgment

This work was partially supported by JSPS KAKENHI Grant Number 20K11733 and 24K14878.

## References

[1] M. Tehranipoor, N. Pundir, N. Vashistha, F. Farahmandi, "True Random Number Generators," in *Hardware Security Primitives*, Springer International Publishing, 2023, pp. 119–144.

[2] A. Suciu, S. Banescu and K. Marton, "Unpredictable random number generator based on hardware performance counters," Digital Information Processing and Communications, pp. 123–137, Springer Berlin Heidelberg, 2011.

[3] H. Masaoka, S. Ichikawa, N. Fujieda, "Random Number Generation from Internal LFSR and Fluctuation of Sampling Interval," IEEJ Transactions on Industry Applications, vol. 141, no. 2, pp. 86–92, 2021. (in Japanese)

[4] H. Kamogari, S. Ichikawa, "Evaluation of a random number generator based on an internal linear feedback shift register," IEEJ Transactions on Industry Applications, vol. 143, no. 2, pp. 87–93, 2023. (in Japanese)

[5] G. Marsaglia, "Diehard battery of tests of randomness (archived)", https://web.archive.org/web/20160125103112/ http://stat.fsu.edu/pub/diehard/

[6] A. Chiba, S. Ichikawa, "Evaluation of Random Number Generator Utilizing Weather Data and LFSR," IEEJ Transactions on Industry Applications, vol. 143, no. 2, pp. 80–86, 2023. (in Japanese)

[7] L.E. Bassham III et al., "SP 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications," National Institute of Standards and Technology, Gaithersburg, MD, 2010.

[8] M. Milutinović, "Cryptocurrency," Ekonomika, vol. 64, no. 1, pp. 105-122, 2018.

[9] E. F. Fama, "Efficient Capital Markets: A Review of Theory and Empirical Work," Journal of Finance, vol. 25, no. 2, pp. 383–417, 1970.

[10] E. F. Fama, "The behavior of stock-market prices," Journal of Business, vol. 38, no. 1, pp. 34–105, 1965.

[11] J. R. Doyle, C.H. Chen, "Patterns in stock market movements tested as random number generators," European Journal of Operational Research, vol. 227, no. 1, pp. 122–132, 2013.

[12] M. A. Noakes, K. Rajaratnam, "Testing market efficiency on the Johannesburg Stock Exchange using the overlapping serial test," Annals of Operations Research, vol. 243, no. 1, pp. 273–300, 2016.

[13] A. Noda, "A test of the adaptive market hypothesis using a time-varying AR model in Japan," Finance Research Letters, vol. 17, pp. 66–71, 2016.

[14] V. L. Tran and T. Leirvik, "A simple but powerful measure of market efficiency," Finance Research Letters, vol. 29, pp. 141–151, 2019.

[15] V. L. Tran and T. Leirvik, "Efficiency in the markets of cryptocurrencies," Finance Research Letters, vol. 35, no. 101382, 2020.

[16] Python Package Index (PyPI), "Historic-Crypto 0.1.6", https://pypi.org/project/Historic-Crypto/

[17] M. Živković, "A table of primitive binary polynomials," Mathematics of Computation, vol. 62, no. 205, pp. 385–386, Jan. 1994.

[18] J. Rajski, J. Tyszer, "Primitive Polynomials Over GF(2) of Degree up to 660 with Uniformly Distributed Coefficients," Journal of Electronic Testing, vol. 19, pp. 645–657, 2003.

[19] R. G. Brown, "Robert G. Brown's General Tools Page," https://webhome.phy.duke.edu/%7ergb/General/dieharder.php (accessed 26 Jun. 2023)