An On-Chain Gaussian Pseudo-Random Number Generator

Simon Tian*

2021/7/31

Abstract

An on-chain Gaussian pseudo-random number generator is proposed in this article. It relies on the count of 1's in the binary representation of a hashed value produced by the `keccak256` hashing algorithm. By Lyapunov Central Limit Theorem, this count after proper transformations, has a Gaussian distribution. The algorithm has $O(1)$ complexity and is easy to implement. It can open up many possibilities for blockchains.

Introduction

Suppose a reliable and verifiable source of randomness is available on-chain. It can be obtained via an off-chain Oracle such as Chainlink or sophisticated on-chain algorithms. This source of randomness is required as the input.

The second assumption is the `keccak256` algorithm provides uniformly distributed values in the output space $[0, 2^{256} - 1]$. In other words, in the binary representation of a hashed value, every digit has equal chance of being `0` or `1`. That is equivalent with having a Bernoulli distribution with success probability `0.5`, i.e., $X_i \sim Bernoulli(0.5)$, where X_i is the outcome of the i-th digit. Numerical studies show that although the observed probabilities based on a sample of N hashed values have deviations from `0.5`, statistical significance gets smaller as N gets larger. In other words, for $X_i \sim Bernoulli(p_i)$ and the point estimator $\hat{p}_i = \bar{X}_{iN} = \dfrac{1}{N}\sum_{j=1}^{N} X_{ij}$, according to Weak Law of Large Numbers,

$$lim_{N\to\infty} P(|\bar{X}_{iN} - 0.5| < \epsilon) = 1,$$

for every $\epsilon > 0$. A thorough numerical study on this assumption can be found in a separate article.

The third assumption is the outcomes of digits must be independent of each other, i.e., knowing the outcome of one digit does not provide any additional knowledge about any other digit. This assumption is also validated in the same study above.

Methodology

Based on the three assumptions above, a Gaussian random number generator can be obtained taking the sum of outcomes of these digits. Lyapunov Central Limit Theorem provides the necessary condition and theoretical basis for this algorithm. We start by writing the probability of having x successes, or 1's, out of a total of n digits as

$$P(X = x) = \sum_{A\in F_x} \prod_{i\in A} p_i \prod_{i\in A^c} (1 - p_i),$$

*simon@dtopia.me

where F_x is the set of all subsets of x integers that can be selected from $\{0, 1, 2, 3, ...\}$. For example, if $n = 3$, then $F_2 = \{\{0, 1\}, \{0, 2\}, \{1, 2\}\}$. A^c is the complement of A, i.e., $A^c = \{0, 1, ..., n\} \setminus A$.

Notice the outcomes of digits are assumed to be independent and the probabilities may not be all equal, but each has the mean p_i and variance $p_i(1 - p_i)$, i.e., $E(X_i) = p_i \in (0, 1)$ and $var(X_i) = p_i(1 - p_i) \in (0, 1)$.

Lyapunov Central Limit Theorem

This theorem states even if the random variables are not necessarily identically distributed, although they have to be independent, the central limit theorem is still valid under Lyapunov condition.

The Lyapunov condition: Suppose $\{X_1, \ldots, X_n\}$ is a sequence of independent random variables, each with finite expected value μ_i and variance σ_i^2. Define $s_n^2 = \sum_{i=1}^n \sigma_i^2$. If for some $\delta > 0$, *Lyapnov*'s condition

$$\lim_{n\to\infty} \frac{1}{s_n^{2+\delta}} \sum_{i=1}^n \mathbb{E}\left[|X_i - \mu_i|^{2+\delta}\right] = 0$$

is satisfied, then a sum of $\frac{X_i - \mu_i}{s_n}$ converges in distribution to a standard Gaussian distribution, as $n \to \infty$:

$$\frac{1}{s_n} \sum_{i=1}^n (X_i - \mu_i) \ \xrightarrow{d}\ \mathcal{N}(0, 1)$$

In this article, the binary outcome of each digit $X_i \sim Bernoulli(p_i)$ is the random variable of concern. As previously discussed, X_i has mean p_i and variance $p_i(1 - p_i)$ and independence among outcomes of digits can be reasonably assumed. And the quantity s_n^2 can be written as $\sum_{i=1}^n p_i(1 - p_i)$.

Next is to check if the Lyapnov's condition is satisfied for some $\delta > 0$, and in particular, $\delta = 1$ is checked. The denominator in the *Lyapnov*'s condition can then be written as $\left[\sum_{i=1}^n p_i(1 - p_i)\right]^{3/2}$, and the second term can be written as $\sum_{i=1}^n \left[p_i(1 - p_i)^3 + p_i^3(1 - p_i)\right]$.

For the denominator, there must exist a value $p_m \in (0, 1)$ such that $p_m(1 - p_m)$ is the smallest among all p_i's, i.e., $p_i(1 - p_i) \geq p_m(1 - p_m)$ for all $i = 0, 1, ..., n$. The denominator then must be greater than or equal to $[np_m(1 - p_m)]^{3/2}$.

For the second term, it can be easily proved that the term $p_i(1 - p_i)^3 + p_i^3(1 - p_i)$ achieves the maximum when $p_i = 1/2$, hence, $p_i(1 - p_i)^3 + p_i^3(1 - p_i) <= 1/8$, for $i = 1, 2, ..., n$. And the whole term must be bounded by $n/8$.

The numerator has an upper bound at $n/8$ and the denominator has a lower bound at $Cn^{3/2}$, where $C = p_m^{3/2}(1 - p_m)^{3/2}$, then the Lyapunov's condition must be smaller than $\frac{C}{n^{1/2}}$, which goes to 0, as $n \to \infty$.

In conclusion, the *Lyapnov*'s condition holds when $\delta = 1$ and the sum of all digits in the binary representation of a hashed value by `keccak256` algorithm converges to a Gaussian distribution after some transformation, i.e.,

$$\frac{1}{s_n} \sum_{i=1}^n (X_i - p_i) \xrightarrow{d} \mathcal{N}(0, 1),$$

where $s_n = \sqrt{\sum_{i=1}^n p_i(1 - p_i)}$.

In practice, we have $\mu_i \approx 0.5$ and $n = 256$ which is large enough to see the asymptomatic effect.

Therefore, the variable $Y = \dfrac{S - 128}{8}$ is approximately a standard Gaussian random variable, where $S \equiv \sum_{i=1}^{256} X_i$. The reason this is only an approximation is S can only be discrete integers from 0 to 256. whereas, a Gaussian distribution is a continuous distribution on the range $(-\infty, \infty)$. However, the integer values have good approximation to a Gaussian distribution. See the graph below for a detailed comparison between the CDFs of a standard Gaussian distribution and this algorithm. Besides, numerically, given $P(S < -16) + P(S > 16) < 2e - 57$, the probability that is more extreme than the limits is so small the approximation can be considered good enough.

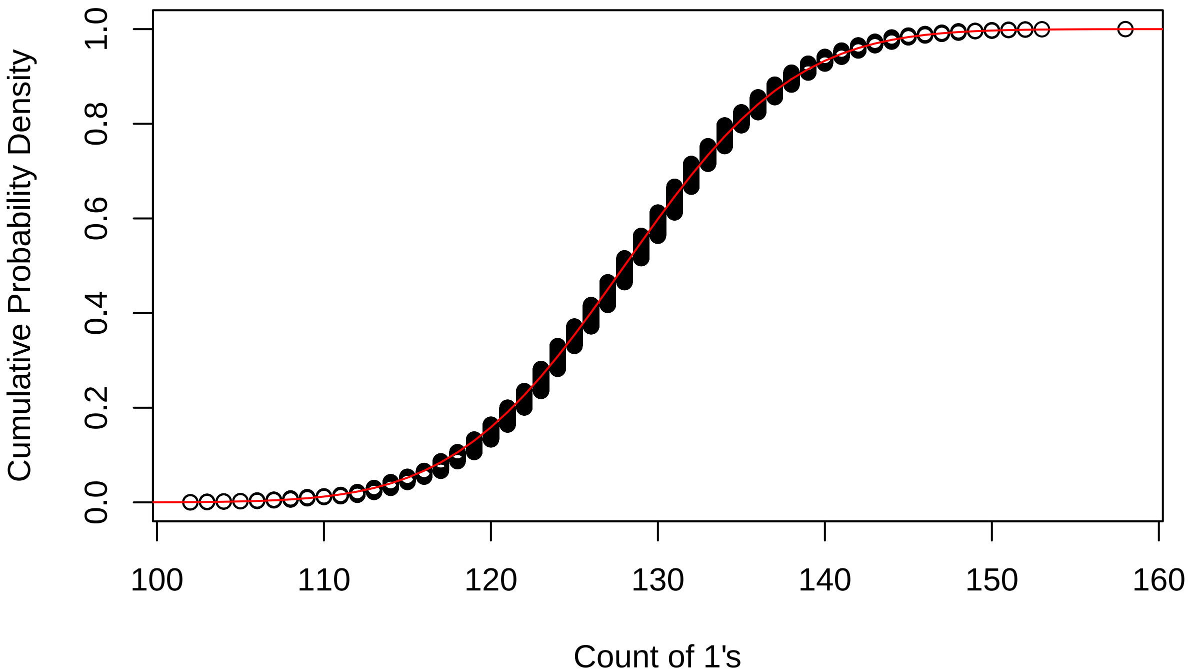

Figure 1: Cumulative Probability Density plots. The black dots are the quantiles of 3,000 random counts of 1's produced by this algorithm, and the red solid line is the theoretical Gaussian CDF curve.

If this algorithm is to be implemented in `Solidity` that does not support floating number operations, and only integers are available, $Z \equiv 1000Y = 125S - 16000$ can be used as a scaled Gaussian random number with precision up to three decimal points. If a higher precision of one more decimal point is desired, four `keccak256` hashed values can be concatenated as a `1024`-digit long array, and the sum of `1`'s in this array can be scaled to have precision up to four decimal points.

Conclusion

This article introduces a novel on-chain Gaussian random number generator, by counting the number of 1's in the binary representation of a hashed value produced by the `keccak256` hashing algorithm. Lyapunov Central Limit Theorem provides the necessary theoretical condition and validation of this algorithm. It can open up many possibilities for blockchains.