A Numerical Study On The Keccak256 Hashing Algorithm

Simon Tian[*], PhD

2021/7/17

Abstract

keccak256 algorithm is the default hashing algorithm to many blockchains. It is accepted to be able to provide high quality pseudo randomness. However, formal detailed quantitative analyses of such randomness are not publicly available. This makes the trust on this algorithm up in the air. This article aims to fill in this missing gap by carrying out three sets of experiments to address key concerns around measures such as entropy, correlation and autocorrelation. The quality of randomness of keccak256 is effectively justified.

Introduction

keccak256 algorithm is one hashing algorithm in the SHA-3 family and was selected on October 2012 as the winner of the NIST hash function competition and standardized in 3GPP TS 35.231, FIPS 202 and SP 800-185. It is based on the sponge function and works by mapping an infinite input space to a finite fixed-sized output space. More technical details can be found on the official website https://keccak.team.

Gholipour(2011) carried out statistical tests and claimed the keccak family can pass random test of the NIST Statistical Test Suite and ENT random test, hence concluded the keccak family of hash functions had excellent pseudo randomness. However, the lack of details, statistical reasoning and clarity make the conclusion not quite convincing.

In the next section, three sets of numerical experiments are given out and four quantities are introduced for the evaluation of the quality of randomness of keccak256.

Methodology

Numerical Experiments

Experiment 1 This is the base experiment. Two groups of integer values are generated and then fed into keccak256 as input values. In Group 1, values are the integer sequence from 1 to 10,000 and in Group 2, from $1 + 10^{128}$ to $10,000 + 10^{128}$. To have two groups is to explore two distant regions in the input space.

Experiment 2 A prefix is added to the integer values in the last experiment to make it more practical in a way that it is generally the concatenation of a string prefix and a numerical id being hashed. The two prefixes chosen are `Transaction Number:` and `Tx Id:`. The resulting strings would then be `Transaction Number: 1`, `Transaction Number: 2`, etc., or `Tx Id: 1`, `Tx Id: 2`, etc.

[*] simon@dtopia.me

Experiment 3 `ASCII` letters, punctuation, and digits are randomly chosen to form a string that can be of arbitrary length from `2` to `7` in Group 1 or `32` in Group 2. For example, three such strings can be `SJ2R1Q`, `K?6`, or `*a|Ll2&]x^Xz#PkWiWueAp4#Bwx`. This experiment is to explore a wide and arbitrary region of the input space.

Numerical Measures

Uniformity of hashed values This quantity can be measured by individual probabilities of observing `1` for each bit of a hashed value. Point and interval estimation methods are used in this study. Ideally, the point estimation of true probabilities should be close to 0.5, and the point-wise confidence intervals should cover 0.5 more frequently than required by a significance level. To control the overall Type I error rate, Bonferroni Correction is applied, if the overall confidence level is set at 95%, then each individual test should have the confidence level at $1 - \alpha/256 = .9998$. Note, this is only to study the marginal properties of the outcome of each bit.

Independence among outcomes of digits This property is to check if the knowledge of outcome of one bit provides additional knowledge to the outcome of any other bit. Zhang(2003) examined properties of eight binary vector dissimilarity measures and `Rogers-Tanmoto`, `Correlation` and `Sokal-Michener` measures outperform the others by the discriminative power. They will be adopted to in this test. And ideally outcomes of bits have high mutual dissimilarity measurements.

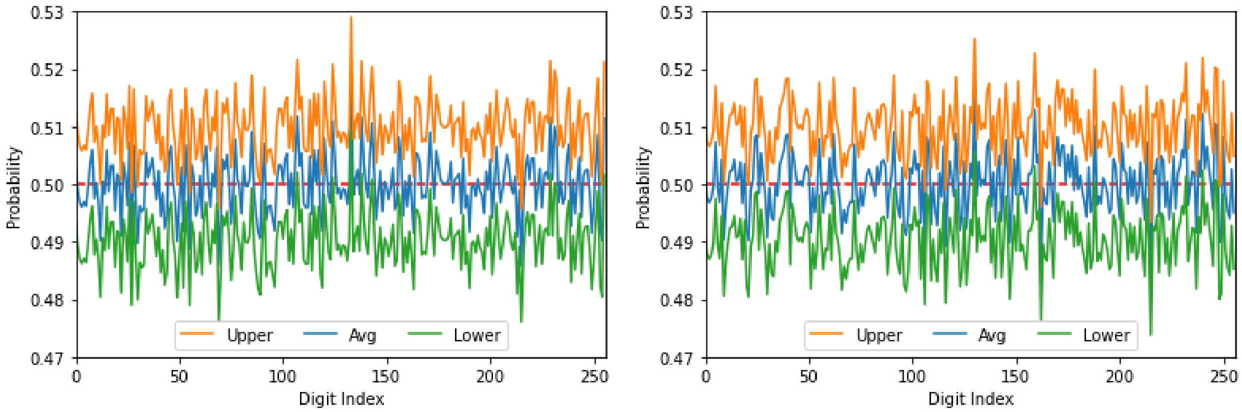
Independence between input values and hashed values This property is to address the concern that input information can be correlated with certain bit in the output hashed value. There is no universally applicable measure on this, however, for the first two experiments in this study, Pearson's correlation can be used to make the assessment. Ideally, the correlation is close to 0.

Autocorrelation for any digit across sequential inputs This measure aims to address the concern that input values, especially sequential values, can produce correlated hashed values. The concept of autocorrelation is borrowed from time series analysis. Suppose the given input integers or strings contain a sequence of consecutive values x indexed by $i = 1, 2, \ldots$, and let the input to the `keccak256` hash function is denoted as y_i, then the hashed values $h_i = h(y_i)$ can be studied with the sequence x_i for auto-correlations, i.e., if outcomes for certain bits are correlated w.r.t. input values. Ideally, the auto-correlations are close to 0 for all lags other than 0.

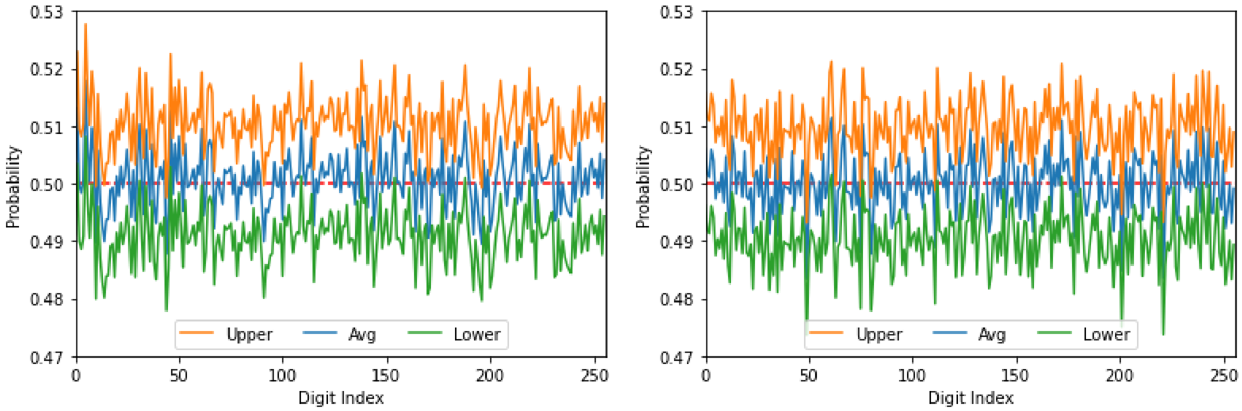
Results

Results are given under each experiment below. Note that the numbers are reproducible. The code for reproducing the results can be found in the GitHub repository for this testing.

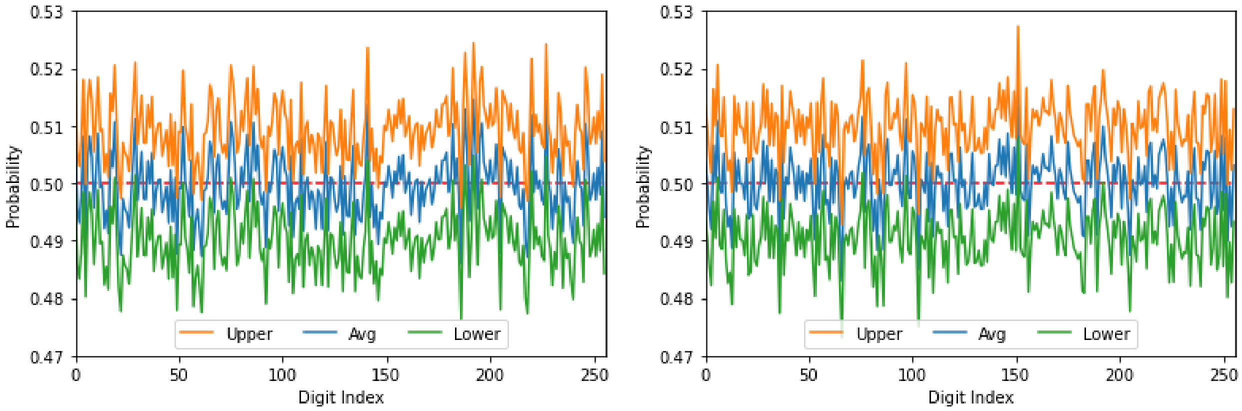
Experiment 1



Experiment 2



Experiment 3



Conclusions

1. The `keccak256` hashing algorithm can generate good uniformity across a wide input space.

2. The outcomes of bits have good mutual independence.
3. The outcomes of bits do not present significant autocorrelation.

References

Gholipour, A. and Mirzakuchaki, S., “A Pseudorandom Number Generator with KECCAK Hash Function”, *International Journal of Computer and Electrical Engineering*, Vol. 3, No. 6, December 2011

Zhang, Bin and Srihari, S., “Properties of Binary Vector Dissimilarity Measures”, 2003