Using Sky based Entropy for a Publicly Verifiable TRNG

Table of Contents

Abstract

This document intends to submit the final paper of Ibrahim Khalil as part of his Independent Study Module 'Rapid Prototyping and Experimentation - II' under Dr. Debayan Gupta. The report reviews existing software-based randomness techniques, and proposes a design of a publicly verifiable True Random Number Generator (TRNG) based on extracting entropy from sky images. The paper discusses the limitations of this technique and conducts a comparative analysis with existing Pseudo Random Number Generators (PRNGs). Finally, it discusses potential developments in the model and its real-world applications, highlighting the uniqueness of sky-based entropy as a publicly verifiable source of randomness.

Introduction

The generation of random numbers is a basic need in cryptographic security, scientific simulations, and decentralized systems. The use of Pseudo Random Number Generators (PRNGs) is common, though they work on deterministic algorithms, which render them unsuitable for true randomness. The True Random Number Generators (TRNGs) mainly depend on entropy sources based on hardware; however, it is still an open problem on how one can ensure public verifiability. A novel TRNG that borrows from sky images for entropy has been proposed in this paper. By observing sky regions, cloud-movement and light-intensity variations, the system produces verifiable and unpredictable random numbers. The paper begins with a literature review regarding previous methods of achieving randomness, together with a stepwise design discussion of the proposed TRNG, including its validation via statistical randomness tests, and a comparative analysis with traditional PRNGs. Finally, the study concludes with some constraints and applications that bring sky-based entropy to contribute toward the creation of publicly verifiable randomness.

Literature Review

Randomness is an essential aspect in cryptosystems, statistical simulations, and other various computation processes. The first are based in hardware, while the latter regard software ways of generating random numbers through a set of deterministic algorithms. These software techniques essentially fall into two categories: one is based on the degree of statistical properties of the output sequence and the second on the unpredictability that can be observed with respect to any attempts to overturn the randomness. While the former are called Pseudo Random Number Generators (PRNGs), the latter are called Cryptographically Secure Pseudo Random Number Generators (CSPRNGs). PRNGs utilize mathematical functions along with certain seed values, producing sequences of numbers appearing to be random but actually being completely deterministic. The widely known PRNGs consist of Linear Congruential Generator (LCG), Mersenne Twister (MT19937), and XORShift; the oldest amongst them is LCG which operates with recurrence:

$$X_{n+1} = (aX_n + c) \bmod m;$$

where a, c, and m are constant values. Despite its adherence in generation of random numbers, it is claimed to generate poor randomness due to its cycle length being short and the correlative nature of the numbers generated.

Mersenne Twister (MT19937) exhibits quite a long period ($2^{19937}-1$) and is statistically more random. It's treated as an acceptable PRNG in simulation and gaming applications. However, it's predictability under certain circumstances makes it unsuitable for cryptographic purposes. CSPRNGs such as Fortuna or Yarrow or based on cryptographic hash functions (SHA-256, HMAC) or block ciphers (AES-CTR, ChaCha20) have been implemented to amend security considerations. These generators employ techniques to increase the unpredictability and against state compromise attacks. For example, the Fortuna PRNG continuously collects entropy from cryptographically secure sources, thereby ensuring the secure generation of random numbers. Conversely, even the finest of newer approaches are inherently limited, for they cannot incorporate entropy as pocket shocks and applause in software-based randomness generation; in other words, in software-based PRNGs and such, true unpredicted values cannot arise without some kind of outside entropy source. The very natural setting reinforces the point for the need of Random Number Generators, RTNG producing randomness from non-deterministic, physical processes.

<u>Randomness Extractors</u>

Randomness extractors are one of the most important components needed to convert raw entropy from noisy or biased sources into a uniform output of good quality. These extractors are crucial for ensuring that raw entropy sources with correlations, patterns, or biases are addressed for both software and hardware random number generation. A randomness extractor specifies the function Ext that maps weakly random source X and a short truly random seed Y into an almost uniform output R:

$R = \text{Extract}\ (X,Y)$

One of the most well-known extractors is the von Neumann extractor, which corrects biases in binary sequences. Given an input sequence of bits, it processes adjacent pairs: ‘01’ is mapped to ‘0’, ‘10’ to ‘1’, and ‘00’ or ‘11’ are discarded. Although simple, this method significantly reduces the data rate and may require additional extraction techniques for efficiency. More advanced extractors, such as Trevisan’s Extractor and Leftover Hash Lemma, leverage computational hardness assumptions and universal hash functions to distill high-quality randomness from weak sources. The Leftover Hash Lemma states that if an entropy source has sufficient min-entropy, a universal hash function can extract nearly uniform random bits. For real-world applications, randomness extractors are often used in conjunction with TRNGs to ensure output quality. For instance, cloud movements, atmospheric noise, or radioactive decay may introduce patterns that require post-processing through extractors. The proposed sky-based TRNG will incorporate such mechanisms to enhance randomness quality and achieve verifiability.

Randomness Beacon

A randomness beacon is a publicly available source of verifiable random numbers that is accessible in real time. A beacon generates and publishes random values at regular intervals to ensure accountability and to prevent manipulation. One of the examples includes NIST Randomness Beacon, that produces random values which are made public every 60 seconds with time and digital signature. Having a quantum source makes randomness unpredictable and non-retroactive. Each output is kept and retrievable for historical verification of previously produced values. Randomness beacons have a critical role in secure elections, cryptographic key generation, and decentralized protocols. In blockchain-based smart contracts, publicly verifiable randomness is crucial to ensure fair lottery draws, unbiased leader selection, and transparent governance mechanisms. The proposed sky-based entropy TRNG adopts a beacon model in which it provides a publicly accessible, verifiable randomness source. Such an approach appeals to natural entropy from atmospheric phenomena, thus more accessible and independent of specialized equipment, unlike quantum-based or hardware-centric beacons.

There is a consensus that hardware TRNGs are the most reliable true random number generators and source their unpredictable random number generation using such sources as thermal noise, radioactive decay, and the emission of light. However, these TRNGs interface issues of accessibility, cost, and possible security vulnerabilities (for example, chip-based RNGs may have backdoors). The paper will exclude discussions on hardware-based TRNGs because it concentrates on software-based solutions that are not tied to any form of hardware. The focus, therefore, is on alternative entropy sources, which can be captured and processed using software techniques. This sky-based entropy model takes an innovatively different approach: it avoids the constraints put forth by hardware, all the while holding onto public verifiability. Allowing the public to access images of the sky or feed from camera systems provides an unpredictable and thus hard-to-predict or manipulate source of entropy: verifiable randomness is thus observed.

The literature survey gives the overview of the area of research in software-type randomness generation, randomness extractors, and randomness beacons: the groundwork for the proposed TRNG. Although PRNGs are efficient, they are always deterministic, and getting a source of true random numbers requires harnessing entropy from external sources. Randomness extractors play a central role in extracting the entropy from various natural sources and ensuring uniform distribution. The randomness beacon concept thereby puts forth a stronger case for public verifiability in randomness generation. The proposed TRNG promises to provide a new, unique, convenient, and transparent source of randomness without requiring special hardware solutions by drawing entropy from the sky. The next section details how the methodology was designed and implemented.

Proposed TRNG Using Sky-Based Entropy

True Random Number Generators (TRNGs) must rely on some inherently unpredictable physical source to generate randomness. We have proposed using sky-based entropy, where the random variation of the appearance of the sky through time is inherently constant but unpredictable, thus creating a strong publicly verifiable source of randomness. In contrast to software-based Pseudo Random Number Generators (PRNGs), which depend on deterministic algorithms, or even traditional hardware TRNGs that require special and unverifiable entropy sources, our method is just about open and open. The subtle unmanageable fluctuations that happen on sky images are due to atmospheric changes, light scattering, and environmental factors, and they provide a natural pool of entropy. Even though variations in sky images are normally low, in terms of color gradients, contrast, and brightness-microscopic pixel-level changes disallowed any one entity to reproduce the exact image. This ensures that even a slight difference in the image would result in a sea of hashed output that can be described by no one.

The architectural design of this TRNG revolves around the capturing of sky images in real-time and statistic processing. This is done using a hash that extracts entropy from minute pixel variations. The randomness extractor functions by ensuring that as long as an image is not pixel-for-pixel identical to one taken before it, the output produced is uniformly distributed. This property forms the core of randomness—a fragment of difference in an image should yield a very different hash output; thus, no party can predict future outputs having knowledge of previous images. This is reinforced with live-streaming of sky images, with five different captures published every second. By ensuring that we have a continuous stream of images, we offer proof of continuity—one cannot just inject fake images without ruining the expected game flow. An attacker that attempts to insert an image or manipulate the source would not be able to keep the continuance of conditions for a sky. The fact that there is an undamaged video stream then gives a direct proof of authenticity alone, providing a timestamp at which each image is said to be recorded. To further strengthen public verifiability, we introduce a secondary entropy source—a cryptographic signature of real-time stock market data at the time of image acquisition. We concatenate the image hash with a string composed of the prices of 50 highly volatile stocks at the instant of the image capture.

The logic behind this is in the fact that stock prices are volatile and the stock data that become available represent the current unpredictable state of the financial market at any given moment. This serves to anchor the time, just like it would in the classic “kidnapper's newspaper” scenario, where a child was being held hostage, and the ransom note was deemed current by virtue of a newspaper photograph bearing the day's headline news. In this case, the stock price string serves to prove that the output of the randomness we generated corresponded to a point in time that no longer exists, which would be practically impossible to reconstruct or predict: trying to determine or reconstruct the exact combination of prices at some other time. Since both sky images and stock prices change in an independent, unpredictable way, the total entropy is sufficiently large to promise a high-entropy, publicly verifiable TRNG output. Putting together just these two sources of randomness-the unpredictability of the brightness of pixels on one hand and a live trading move with stock prices that cannot be reproduced-creates a system for a source of randomness that is theoretically uniformly balanced as well as practically impossible to interfere with. The live feed helps with verification, while concatenating stock prices works as a time lock that stops the entry of precomputation values. Hence the randomness output is transparent, trustless, publicly accessible, and provides unique value to public randomness beacons and cryptographic applications.

In order to prove that our sky-derived true random number generator (TRNG) is indeed truly random, we must validate it using known randomness tests. Understanding these tests helps in determining how random our numbers are and whether or not any hidden patterns exist. Given the need for more background from our system, we will simply specify which tests to employ and what they mean in layman's terms. As soon as we collect enough random numbers, we will analyze their performance against these tests. The statistical test suite developed by the United States National Institute of Standards and Technology is one of the most accepted tests for random number generation. This set of tests has numerous applications in cryptography and data security, where they are employed to verify that a sequence of numbers is actually random. A very basic one in this suite is the Frequency Test, which checks to see whether we get almost identical counts of 1s and 0s in our binary outcome. If one appears to occur more than the other, it might not be random. The Runs Test checks how 1s and 0s appear in clusters. In the case of complete random sequences, it should naturally produce short clusters of 1s and 0s. If the clusters are too long or too short, it may, however, indicate a certain hidden pattern. The Block Frequency Test performs this task slightly differently, looking for the randomness of connector groups, and not only the bolt-on whole sequence. Meanwhile, the Approximate Entropy Test: Does any pattern repeat too often, thus being easy to predict? Lastly, the Random Excursion Test monitors how the sequence acts when viewed as a type of random walk and verifies its expected development over time.

Along with NIST, we will administer the Diehard Tests, another widely acknowledged suite of tests established by the mathematician George Marsaglia. These tests concentrate on deeper statistical properties of randomness. One is the Birthday Spacing Test, which examines how far apart repeated values occur in a series of numbers. If the numbers cluster, that is, they do not spread out, it is a clue that they might not be truly random. Another is the Overlapping Permutations Test, which looks to see whether some small patterns appeared more frequently than they ought to. If they do, randomness falls under suspicion. And finally, the Rank of Matrices Test builds binary matrices from our numbers and looks at how their ranks are distributed. If the ranks follow an expected pattern, it is more likely that the numbers are random. For now, these tests are the roadmap for how we plan on verifying our TRNG once we have collected a sufficiently large amount of data. When our method fails any of these tests, we will have to inspect the causes closely and overhaul them again. The actual results will be drawn only once sufficient randomness has been collected from our sky-based system.

The traditional random number generators are called a “Pseudo Random Number Generator” (PRNG); they use mathematical formulas to create seemingly random sets of numbers in a sequence. They are not really random since their values depend upon fixed formulas and an initial seed value. If one can know the seed, all subsequent numbers can be predicted. Such a predictability makes PRNG unsuitable for any security-sensitive applications, such as cryptography, where genuine unpredictability is required. Our TRNG, however, is completely different in that it uses images of the sky upon which real-world randomness is based. The factors within a sky cannot be manipulated or predicted, lending it a degree of unpredictability; this is manifested in minute changes in cloud patterns or lights on our sky. Although the overall appearance of the sky does not change erratically from moment to moment, there are pixel-level changes that can give rise to large changes on how an image is processed. The fact that we hash these images means that even the smallest change will yield entirely different finishes, without any chance of predicting beforehand what the random numbers will be. Our technique is driven by something that cannot be easily influenced by anybody; hence, it makes for a stronger source of randomness, unlike PRNGs, which always follow a rigid mathematical structure.

One of the major advantages of using the sky as a source of randomness is its ability to make the randomness publicly verifiable. While this is generally not possible with traditional TRNGs because they are based on hardware components, making the user depend on the good faith of manufacturers to generate true randomness from the beginning to prevent manipulation, in our case, it is a non-issue. We use live images streamed that can be accessed by anybody. Since our images are taken and published at five images per second, proving them to be real and taken at the appropriate time is usually easy. If someone were to attempt to fabricate an image or tamper with the randomness source, the illusion would collapse since they would break the natural continuity of the sky appearance. This affects our method's high transparency and resistance to manipulation. Another major competitive advantage is that our method forwards two independent sources of randomness: sky images and real-time stock prices. The reason we add stock prices is to incorporate a verifiable timestamp. Stock prices keep changing rapidly and cannot be preempted, and hence, they help us throw in additional evidence that our random numbers are being generated exactly at the time they claim.

We take a set of 50 highly volatile stock prices, concatenate them into a string, and hash this string along with the sky image. This means that even if someone somehow managed to manipulate the image, he or she would also have to predicted or change stock prices in real-time -a nearly impossible task. In this example, one could draw comparisons with the classic example of a hostage holding a newspaper to prove that the photo had been taken on the specified day. Sky images fused with stock prices guarantee that the testified method would not only be arbitrary in random style, but also will have been proven by time and effaced from any tarde.

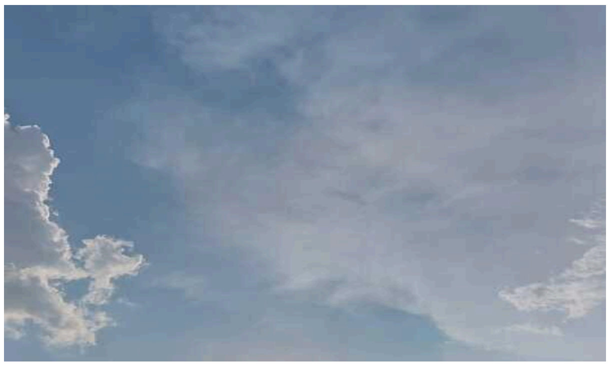
That said, there are challenges and limitations inherent to this approach. The randomness of the sky, for instance, strongly relates to atmospheric conditions. Should the sky be clear from any white clouds or be covered by uniform clouds, it will often result in low variation between the consecutive images, causing entropy extraction to let us down. In the beginning, the camera we have used has major implications for how well we do it because the noise or distortion introduced by the camera could greatly affect the randomness in unexpected ways. Last of all, capturing high-resolution images every second means far more computation than applying simple mathematical PRNGs can create. While it can guarantee us greater randomness, it also means that our method is somewhat slower and more resource-hungry. On the other hand, to achieve practical TRNG, we need to generate random numbers efficiently along with strong security guarantees-along with throughput, which is the number of random bits that we get in one second. While we grab five images a second, alongside processing of those with stock prices, we need to make sure that this is on time for real field applications.

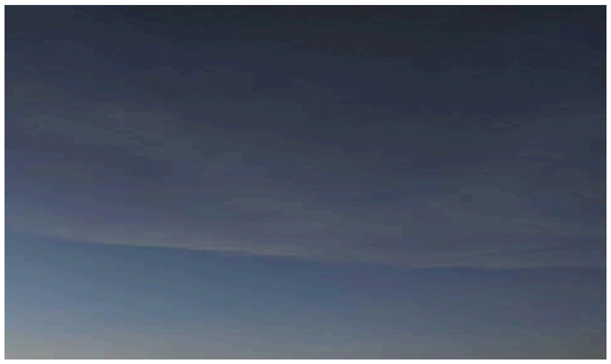
Also important is the latency, or the time interval between capturing an image with the camera and producing and communicating a random output to the user or other applications. In case this time interval becomes intolerably long, then the randomness will not be usable for real-time applications. The next consideration is long-term reliability; we want to ensure that our TRNG will produce systematic randomness of high quality even after months and years in operation.

Environmental effects, such as seasonal effects, atmospheric pollution, or wear and tear on the camera, might have small impacts on the entropy source. Having systems in place to monitor these types of changes over time will likely ensure that our system is stable. Notwithstanding this, our approach gives a genuinely unique and verifiable way of generating true random numbers. Combining the adaptations provided in the sky with independently verifiable stock prices gives a randomness source that is difficult to manipulate, open to all for public knowledge, and resistant to external influence. If our randomness passes all statistical tests, then it could serve as a highly reliable alternative to standard PRNGs and hardware-based TRNGs.

Example Random Number Generation

In order to demonstrate the random number generation process in flow, I have selected 4 images of varying brightness, hue and focus (all of Sonipat skies). I will select one image and proceed to generate a random number from its entropy. Consider pictures 1 through 4 (in order left to right, top to bottom) attached below.





Assume the camera is mounted in this fixed position so as to livestream the sky, publishing 5 images every second. From our line of argumentation, we follow that such a setup is extremely difficult to manipulate, and continuity of stream will enable public verification of the image used at a particular timestamp: that the image is clicked after X time of beginning the livestream, and is not tampered with. Thereafter, the image is hashed using a public hash function to make the output (random number) unpredictable and sensitive to minute changes in the image (this feature is necessary as the image between time X and time X' might not have extreme variations depending on the period of random number generation). The public nature of it makes it possible for anyone to verify that the random number is generated using the same image (published to a web repository).

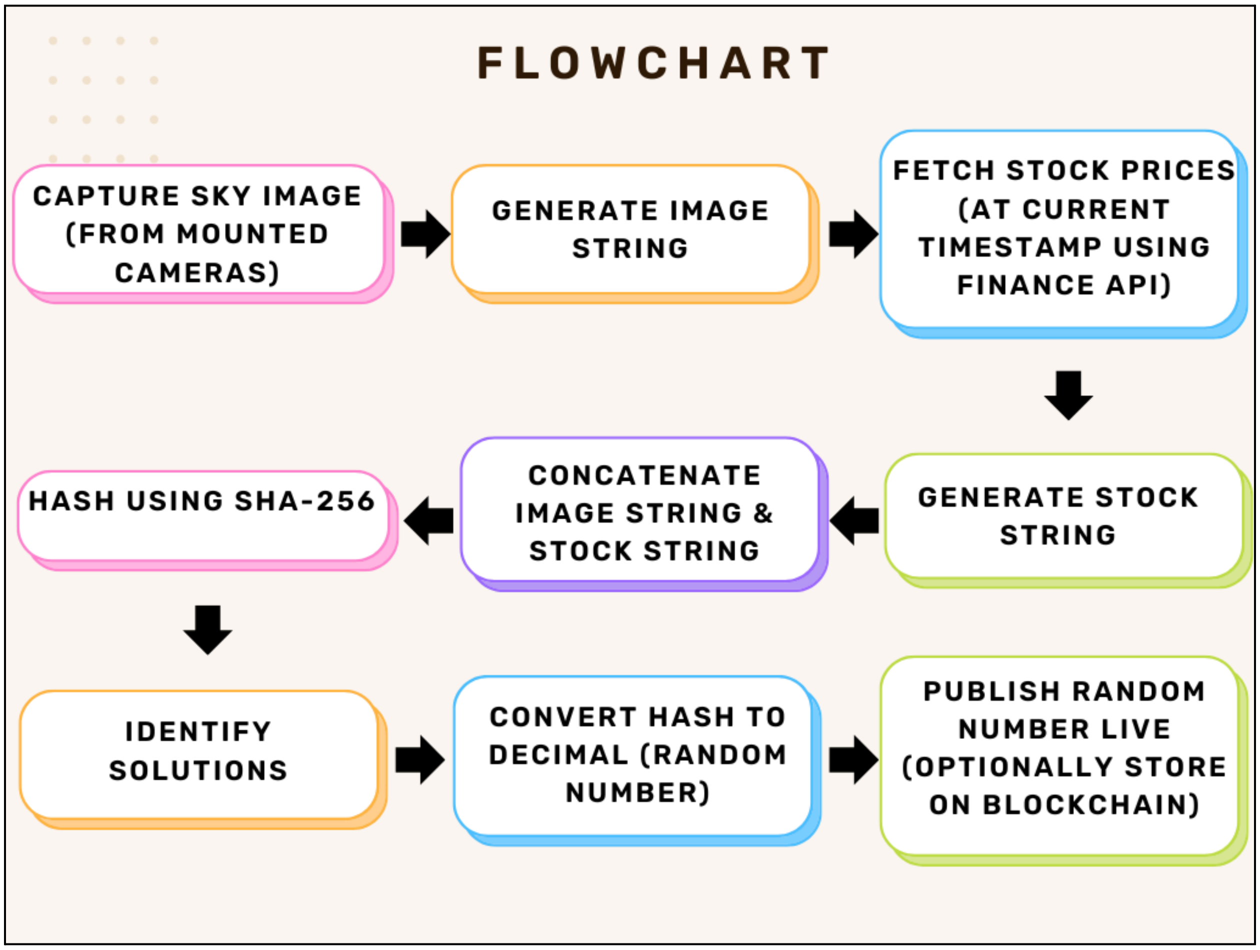
Before hashing the image, the live stock prices (opening, closing, minimum and maximum prices) of 50 volatile stocks at that timestamp is fetched from a publicly available source. The image alongside the concatenated string of data of the stock portfolio (hereafter called stock string) is hashed to produce the random number. This exercise adds entropy as not just the image but also the stock string becomes a source of entropy for the hash output i.e the random number, but the more important function it plays is of public verifiability of the timestamp of generation of the random number. Since the stock string is generated using a high number of volatile stocks, it is nearly impossible to get an accurate prediction of 50 stocks with each of its components at the given timestamp (the minimum, maximum, opening and closing prices). The hash produced hereafter hence is signed at the timestamp, that at least the hash is produced at this time not before, and is publicly verifiable if one knows the timestamp and the image used. Thus, combining continuity that says the image is taken at timestamp T, and that it is signed with the stock string (that the hash is generated at least at timestamp T) produces a publicly verifiable random number.

To show a working example, let's take **picture 1** taken at timestamp 11:58 pm, 8th March IST. We first convert the image to bytestring, fetch stock prices of a portfolio of stocks, and then hash (SHA-256) the image string with the byte string to produce the random number (has converted to decimal).

```
Current Timestamp: 2025-03-08 11:58:12

Image Byte String (First 100 Bytes):
b"\x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x02\x14\x00\x00\x01\xa8
\x08\x02\x00\x00\x00\x91\xf8\xeb!\x00\x01\x00\x00IDATx\x9c\xec\xfdY\x
97\xec:\x92&\x8a}f\x00\xe9C\x0c{\x9f}\x86\xcc\xca\xac\xaa\xcc\xaeR\x9
7Zz\xeb\x17-=\xe9G\xe8\x97\xe8\x87\xe8'\xf5\xbb\xd6\xd2\xbdZ\xbd\xd4]
w\xdd[\xa5\x9b5u"...
Fetched data for AAPL (2025-03-07): {'open': 235.105, 'high': 241.37,
'low': 234.76, 'close': 239.07}

Stock String:
{"AAPL": {"close": 239.07, "high": 241.37, "low": 234.76, "open":
235.105}}

Combined Image + Stock String (First 100 Bytes):
b"\x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x02\x14\x00\x00\x01\xa8
\x08\x02\x00\x00\x00\x91\xf8\xeb!\x00\x01\x00\x00IDATx\x9c\xec\xfdY\x
97\xec:\x92&\x8a}f\x00\xe9C\x0c{\x9f}\x86\xcc\xca\xac\xaa\xcc\xaeR\x9
7Zz\xeb\x17-=\xe9G\xe8\x97\xe8\x87\xe8'\xf5\xbb\xd6\xd2\xbdZ\xbd\xd4]
w\xdd[\xa5\x9b5u"...

SHA-256 Hash:
35b051938da48123eadb919f72692f6dcd5e1b2079ab2c0301219ac4b33ebb10

Hash Converted to Number:
242841090777970706651853696235911000556584998263547394041252621363863
63767568

Generated Random Number:
242841090777970706651853696235911000556584998263547394041252621363863
63767568
```

The range of this number is: {0, … 2^256 - 1}. We can convert this to our custom range by applying the mod x function {0, … x-1}. Even a single bit change in the image string or the stock string will change the random number entirely (property of the hash function), thus making it extremely unpredictable and distributed, as the subsequent images will have differences.

The following flowchart illustrates the process of random number generation:


FLOWCHART
CAPTURE SKY IMAGE (FROM MOUNTED CAMERAS)
GENERATE IMAGE STRING
FETCH STOCK PRICES (AT CURRENT TIMESTAMP USING FINANCE API)
GENERATE STOCK STRING
CONCATENATE IMAGE STRING & STOCK STRING
HASH USING SHA-256
IDENTIFY SOLUTIONS
CONVERT HASH TO DECIMAL (RANDOM NUMBER)
PUBLISH RANDOM NUMBER LIVE (OPTIONALLY STORE ON BLOCKCHAIN)

Limitations and Future Work

While using sky images as a source of randomness has many advantages, it also comes with certain limitations. One major constraint is the dependence on weather conditions. On clear days, when the sky appears uniform without much variation, the entropy extracted from images may be lower than expected. Similarly, during overcast conditions, if the cloud cover remains unchanged for long periods, the amount of randomness obtained from consecutive images could be reduced. Since our method relies on small pixel-level variations, having a less dynamic sky could affect the quality of randomness. To counter this, additional processing techniques may be needed, such as focusing on specific regions of the image where variation is highest or adjusting contrast dynamically to emphasize small changes. Another challenge is hardware reliability. The quality of randomness depends on the camera used to capture the sky. If the camera introduces noise, artifacts, or compression errors, these could distort the entropy extraction process. Additionally, long-term reliability is a concern. Over time, camera sensors degrade, and environmental factors such as dust, humidity, and exposure to sunlight may affect image clarity. Regular maintenance and recalibration may be necessary to ensure that randomness quality remains high. A further limitation is computational efficiency. Unlike pseudo-random number generators (PRNGs) that can generate large amounts of randomness with simple mathematical operations, our method requires continuous image capture, processing, and hashing.

It is more demanding in resources, especially if used with a high-resolution image or if many updates are needed. Pipeline optimization-a way to ensure fast processing while keeping the relevant entropy conserved from image-to-image-would help sustain a healthy balance between speed and attaining a stiff randomness property. Notwithstanding the various challenges, one can improve the techniques in many aspects. One possible enhancement is to combine several sources of entropy. Software-wise, we already combine stock prices with sky images, both providing strong time verification options, but there are potentially more sources from the real world to get good input into the underlying system. For instance, using sensor data from weather stations (wind speed, air pressure, temperature changes) might add a measure of randomness to bolster the uniform improvement of entropy sources. To that end, an alternative approach entails making a switch to adaptive entropy extraction techniques, where the system would select some most-variable, highly selected sky regions rather than provide data from the full image itself. This would assure good randomness quality even under less dynamic conditions of the sky.

Sky images have a verifiable public TRNG developed with the potential for quite a few different real-world applications; one of the most obvious such areas is cryptography. Many cryptographic systems have come to rely on strong randomness to generate secure encryption keys; thus, if the generated randomness is predictable, this would derive security implications. As our approach cannot be manipulated and provides public verifiability, it can serve as a trusty source of entropy for cryptographic protocols, especially competitive ones, where trust in centralized randomness providers is invaluable. Other areas of possible application include simulations and scientific computing. These require the sustained injection of a considerable amount of good quality random numbers into scientific models, from physics to finance and AI. In the context of these, real-life entropy in place of pure PRNGs would allow better realism in simulations-where unpredictably is crucial. Also, this system could be applied in decentralized applications and blockchain systems. Most of the blockchain-based protocols rely on the random generation of things, including fair lottery draws, secure voting, and consensus mechanisms.

Unfortunately, conventional randomness sources do not trust in central authority. A public TRNG based on sky images promises sufficient transparency with regard to its randomness, extending no possibility of centralized trust. It is demonstrated that sky-based entropy probably serves as a dependable verifiable ‘randomness’ source. Continuously photographing the sky and using those images to hash volatile stock prices results in a randomness extraction methodology both shove-proof and open to the public with regard to transparency. While the computations are challenging, given the unpredictability of the weather, they themselves provide a robust alternative to classical randomness generation techniques. The salient feature of this work can be summed up in its verifiability. Unlike hardware-based TRNGs, our approach is scrutinized at every stage and could be classified as a loud box. Anyone can easily verify the images and stock prices used to generate the random bits, making it a much more trustable randomness source for applications that rely on transparent randomness. Various avenues merit further exploration in future investigations. The most relevant one lies in improving the process of entropy extraction so that we may possess an ongoing system producing bits of such randomness quality, even in situations of low-variance in the sky's atmosphere. Another meaningful advance would include scaling our approach to accommodate large-scale applications requiring random bits at higher speeds. A similar approach, incorporating several natural entropy sources such as sensor data, atmospheric noise, or even astronomical observations into one hybrid model, would add resilience to the developed system. In the final analysis, though sky-based randomness is still a nascent idea, it can provide a perspective shift on how we think about generating and verifying random numbers in a decentralized secure public peer-reviewed manner.

To deploy this sky-based TRNG system at an operational level, a company or organization would need certain infrastructural elements to support reliability, efficiency, and verifiability. To begin with, a separate network of HDR and low-light cameras of good quality should be placed at different fixed points for continuous capture of sky images. Such cameras must be mounted on firm pedestals, safeguarded against damage from the elements, and hooked up to a secure cloud server for instant image processing. The system would need a strong data pipeline that could retrieve live stock prices from a reliable financial API, merge them with the captured images, and hash the combined data with a secure cryptographic algorithm such as SHA-256. In order to ensure transparency and verifiability, a public ledger or blockchain may be utilized to store image hashes and stock information at every timestamp, enabling external observers to independently verify the randomness source. Moreover, an optimized software framework with GPU acceleration or parallel computing methods would be required to facilitate the speedy processing of images and stock information, ensuring low latency while maintaining entropy. Lastly, security features and regulatory compliance need to be taken into account, such as encryption of the stored images, secure API connections, and regular audits to ensure against tampering. Deployed at scale, such a system could be applied to industries needing robust verifiable randomness, such as cryptography, gaming, and secure online platforms.