EntroBeam

Random bit and probability distribution with secure entropy source apply in a blockchain. Ver 1.0

Leon Wesker, entrobeam@gmail.com

Abstract. Obtaining the entropy of random bit generators (RGBs, random number generators, RNGs) in deterministic systems is always a fundamental issue in computer science. Since Blockchain is a deterministic and transparent system, it is challenging to generate secure entropy as it is; Even so, blockchains can provide a solution that creates a source of entropy that constructs an unpredictable output in a way suitable for a blockchain. When users send their arbitrarily specified entropy seeds to the smart contract, the contract collects them up to a specific limit. The gas fee value of each entropy seed transaction is affected by the user-defined value, total transactions per block, hash rate difficulty, and network state. Calculates the median and min-max average of all gas fees from the collected entropy seeds, draws lots one of them as the noise entropy seed through the modular operation of those values, and mixes them with one of the unmixed, pure entropy seed to generate secure entropy. Subtract the mixed seed from the set of collected entropy and add a new entropy seed. And the seed and gas fee of the transaction that performed these calculations is excluded from the process not to affect the secure entropy generation. Transactions that send entropy seeds become secure entropy sequentially in the transaction order, but it is very complicated to predict which modulo will mix with noise entropy seeds. Furthermore, using methods like the Fisher-Yates shuffle results in a more reliable algorithm. All these processes can be called entropy mining.

1. Introduction

Random bits are used to generate an encryption key and a digital signature key to be used for ciphers. In addition, it is imperative to use unbiased random numbers in probability distributions, especially when the number of draws is limited, such as in hypergeometric distributions. Combining and using random bits may differ depending on the application algorithm; importantly, entropy is the root source of random bits. Entropy must be unpredictable, but the nature of the blockchain, where everything works transparently, makes it challenging to achieve the goal, and even more so, entropy cannot rule out the possibility of manipulation. However, by applying the concept that the blockchain forms a more secure network as the participation of nodes and miners increases, it is possible to generate secure entropy even in smart contracts. An arbitrary root source is required to generate entropy. A user who transmits a root source (entropy seed) to a contract by consuming a gas fee as a resource can be likened to an entropy miner. The seeds sent by the entropy miner are collected up to a specific limit to form a kind of entropy chain. Collecting these seeds to create an entropy chain is to draw lots from one of them and mix it with the other entropy seeds. The order in which the entropy seed becomes the secure entropy is sequential in which the entropy seed arrived at the contract. The order in which the entropy seed becomes the secure entropy is treated fairly in the order in which the entropy seed arrives at the contract.

2. The eye of randomness

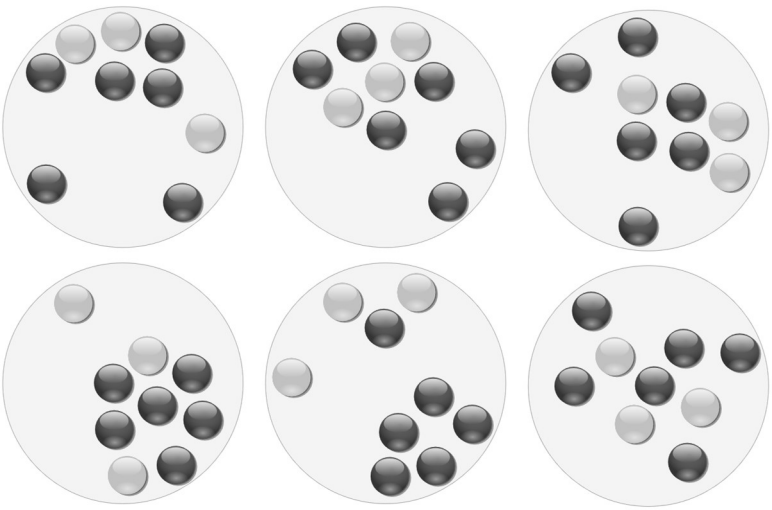
If the variable N describes the number of all marbles in the urn and K describes the number of white marbles, then $N-K$ corresponds to the number of black marbles. In this example, X is the random variable whose outcome is k, the number of white marbles actually drawn in the experiment. Assume that there are 3 white and 6 black marbles in the urn. Standing next to the urn, you close your eyes and draw 2 marbles without replacement. What is the probability that exactly 1 of the 2 are white? Note that although we are looking at success/failure, the data are not accurately modeled by the binomial distribution, because the probability of success on each trial is not the same, as the size of the remaining population changes as we remove each marble.

This is the hypergeometric distribution. The formula for the probability of drawing exactly k from the hypergeometric distribution is as follows.

$P(X=k)=f(k;N,K,n)=\frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$, $\binom{a}{b}=\frac{a!}{b!(a-b)!}$

$$P(X=k)=f(1;9,3,2)=\frac{\binom{3}{1}\binom{9-3}{2-1}}{\binom{9}{2}}=\frac{3\times 6}{36}=0.5=50\%$$

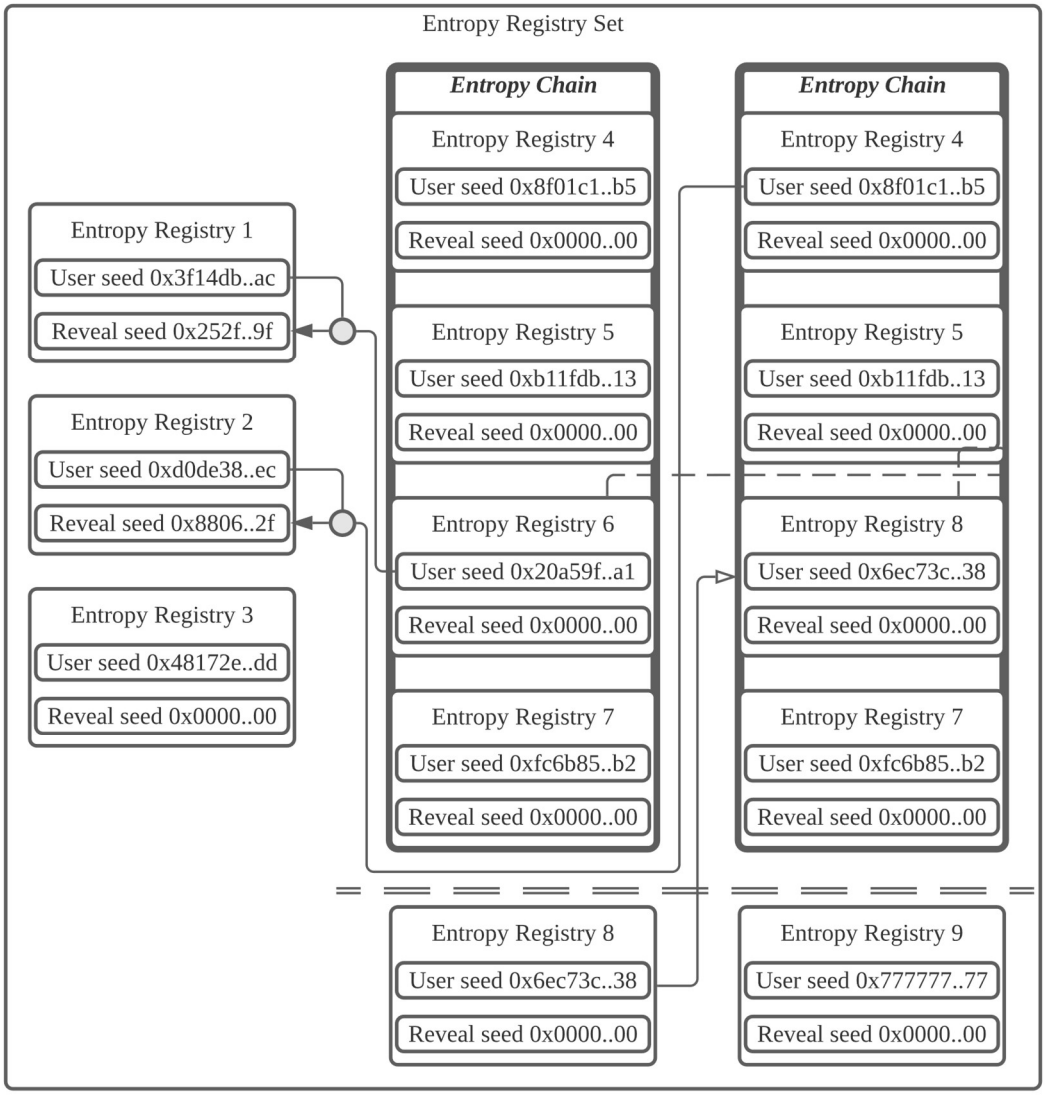
It obviously has a 50% chance of success, but the issue of reliability is another issue entirely.



As you can see from the figure above, the hypergeometric probability itself is reasonable, but the distribution of the marbles is not reliable. You may want to choose the distribution yourself, but that exactly loses randomness; furthermore, it goes against closing your eyes. This means that processes must disclose entropy sources and random number generators, which should not be transparently disclosed. It's a pretty bizarre circumstance, but if compare randomness to a storm, randomness also includes the calmest part, like the eye of a storm. We are trying to solve this process, which can also be called the eye of randomness, with the Entropy registry and Entropy chain.

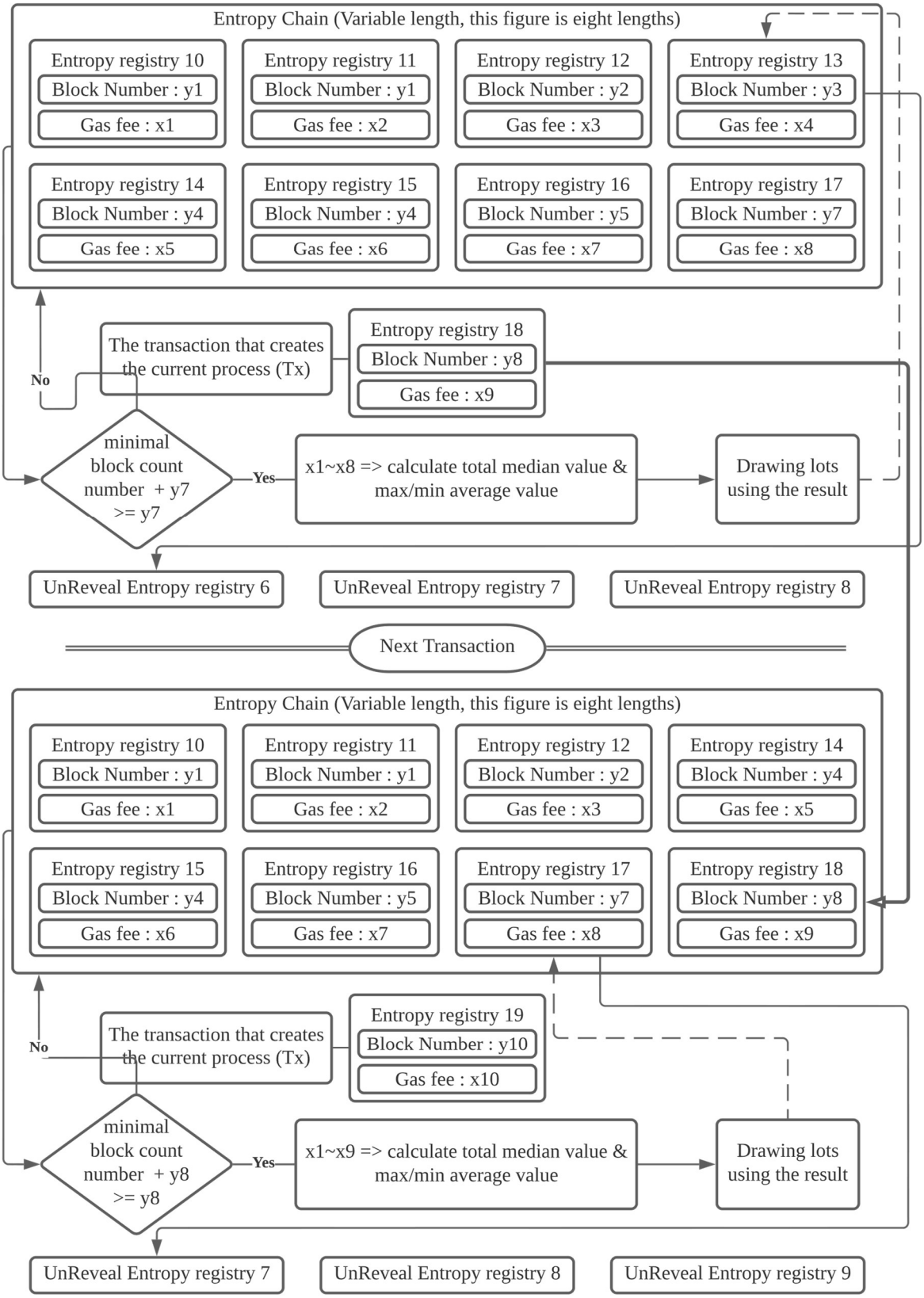
3. Entropy registry

The entropy registry is a database that stores entropy seeds sent by users and the secure entropy generated by the contract. Indexing is possible with the user's entropy seed, and if it is not yet the turn of the secure entropy (Reveal or Settle entropy), the secure entropy returns "0" 256 bits. Data registered in the entropy registry continuously accumulates without being deleted or changed, and duplicate data is not allowed.

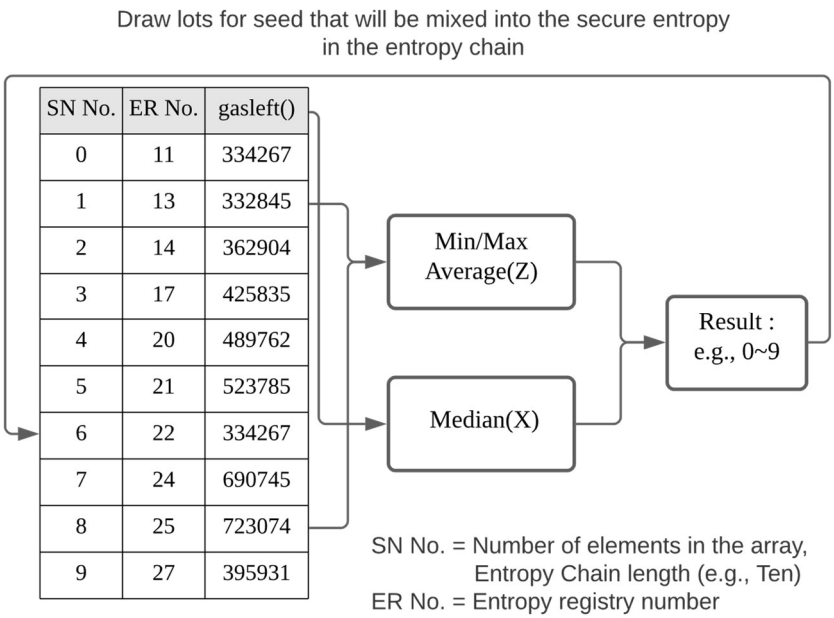


When a user sends a transaction to the entropy seed, each transaction is given a sequence number. Then, Then, 'reveal seed' (the secure entropy of the contract) is allocated forward order according to the sequence number. End users can find Reveal Seeds by indexing User Seeds in the contract's Entropy Registry. And user can also see Reveal Seed bits in the event log of inbound transactions. A process is performed whenever a transaction occurs, but the latest transaction only stores the seed and does not participate in seed generation.

4. Entropy Chain



The entropy chain is another set belonging to the entropy registry set. It is a set used to determine which entropy to deliver to a user entropy that has not yet been assigned a secure entropy (reveal or settle seed). The length of the entropy chain is not fixed. The initial value connects ten transactions, but the length increases as user participation increases. In general, the point to be problematic is that if the length of the entropy chain is p, the probability of predicting which secure entropy will be generated may appear to be $1/p$. But it is far from that. Number of user transactions is n, When multiple transactions occur, and reveal entropy values accumulate, users get the number of cases p^n, which exponentially reduces the probability of predicting the secure entropy. Seeds drawn by lots are removed from the entropy chain. The seed used to generate the secure entropy is removed from the chain, and a 'new seed' is added to the chain. The 'new seed' is the most recent transaction, stored in the entropy registry, and performs all processes such as operating the entropy chain, but is not registered in the entropy chain to prevent it from participating in the creation of secure entropy. (In the figure, Entropy registry [18] is added as 'new seed' in the next transaction, this process was handled by the Entropy registry [19] transaction.)



$$Min\ Max\ Average(Z) = \frac{Z_1 + Z_2}{2}$$

Z_1= Minimum gas fee in the entropy chain data set.

Z_2= Maximum gas fee in the entropy chain data set.

$$Median(X) = \begin{matrix} \text{If } n \text{ is even number} \\ \text{If } n \text{ is odd number} \end{matrix} \begin{cases} X\left[\frac{n}{2}\right] \\ \frac{\left(X\left[\frac{n-1}{2}\right]\right) + X\left[\frac{n+1}{2}\right]}{2} \end{cases}$$

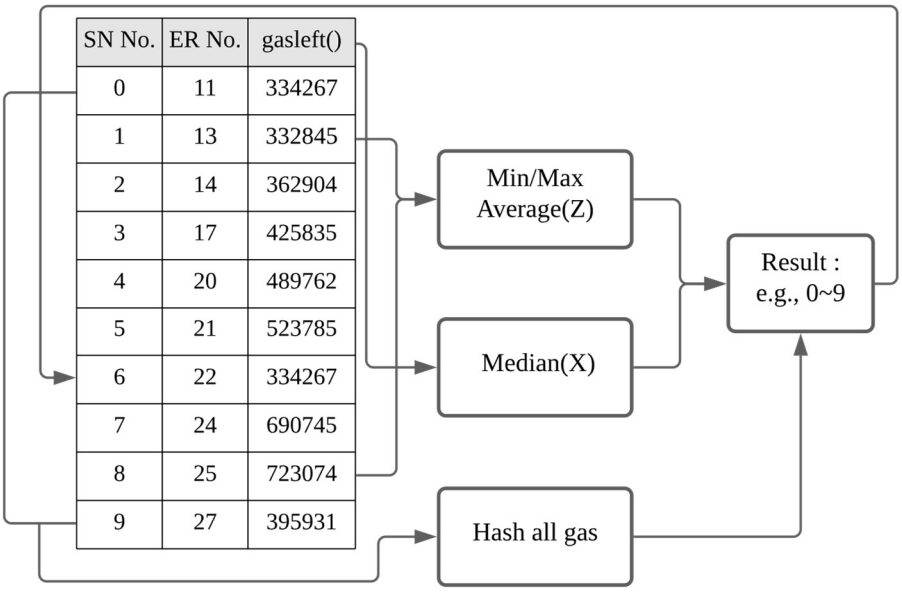
X = List of values sorted order by a Gas fee (gasleft()) in the entropy chain data set.

n = Gas fee assigned to element number in the entropy chain data set.

Choosing which seeds in the entropy chain to generate secure entropy uses the gas fee item of all elements in the chain. The result value derived from the median (X) and average (Z) values generally uses a decimal-based modulo operation. However, A method that nearly never performs division was described in 2018 Fast Random Integer Generation in an Interval by Daniel Lemire with the current state-of-the-art being the arithmetic encoding-inspired 2021 optimal algorithm by Stephen Canon of Apple Inc. The entropy registry is always stored sequentially, but the chain keeps a jumbled order as the computation proceeds. Phase 1 calculates all gas fees, including gas that will not be used. In Phase 1.5, Will calculate all of the initial/middle/last gas fees while the transaction is running.

'Add V1.0.1' - In addition, users can apply it to various methods. The following figure shows an example of deriving enhanced secure entropy by adding the hash values of all gas as noise to the result. This indicates that it can be applied in various ways. Users can use off-chain oracle services or side chains according to their choice or use data from multiple transactions. It is a knotty process to precisely predict the gas of a single transaction. Moreover, the gas fee is 10^-18 units. Therefore, the more linked transactions, it becomes exponentially harder to predict secure entropy. Users should note that mixing additional noise into the secure entropy in a separate transaction makes the entropy predictable. Users must chain the transactions and have them run simultaneously.

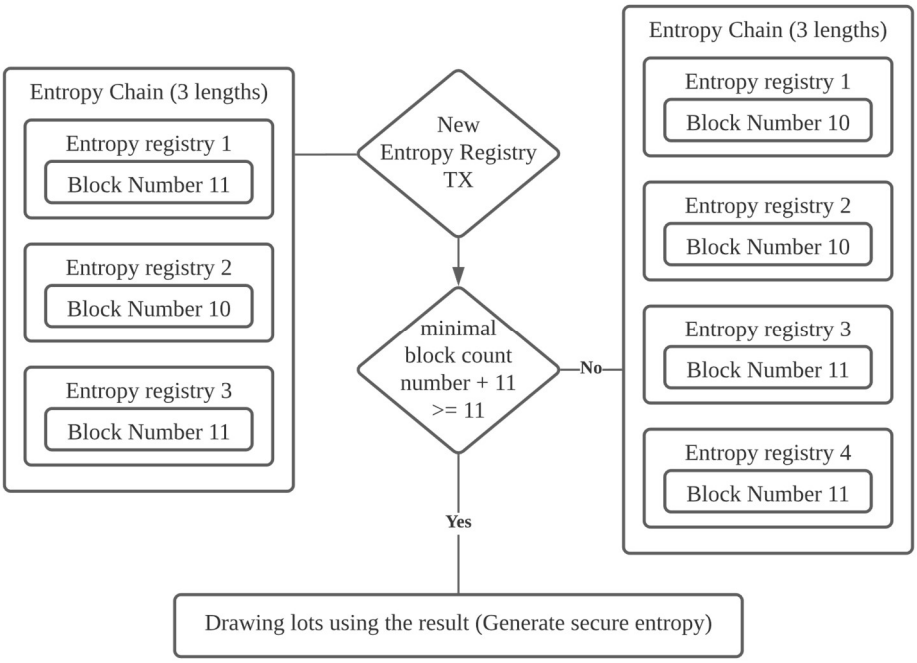
Draw lots for seed that will be mixed into the secure entropy
in the entropy chain



SN No. = Number of elements in the array,
Entropy Chain length (e.g., Ten)
ER No. = Entropy registry number

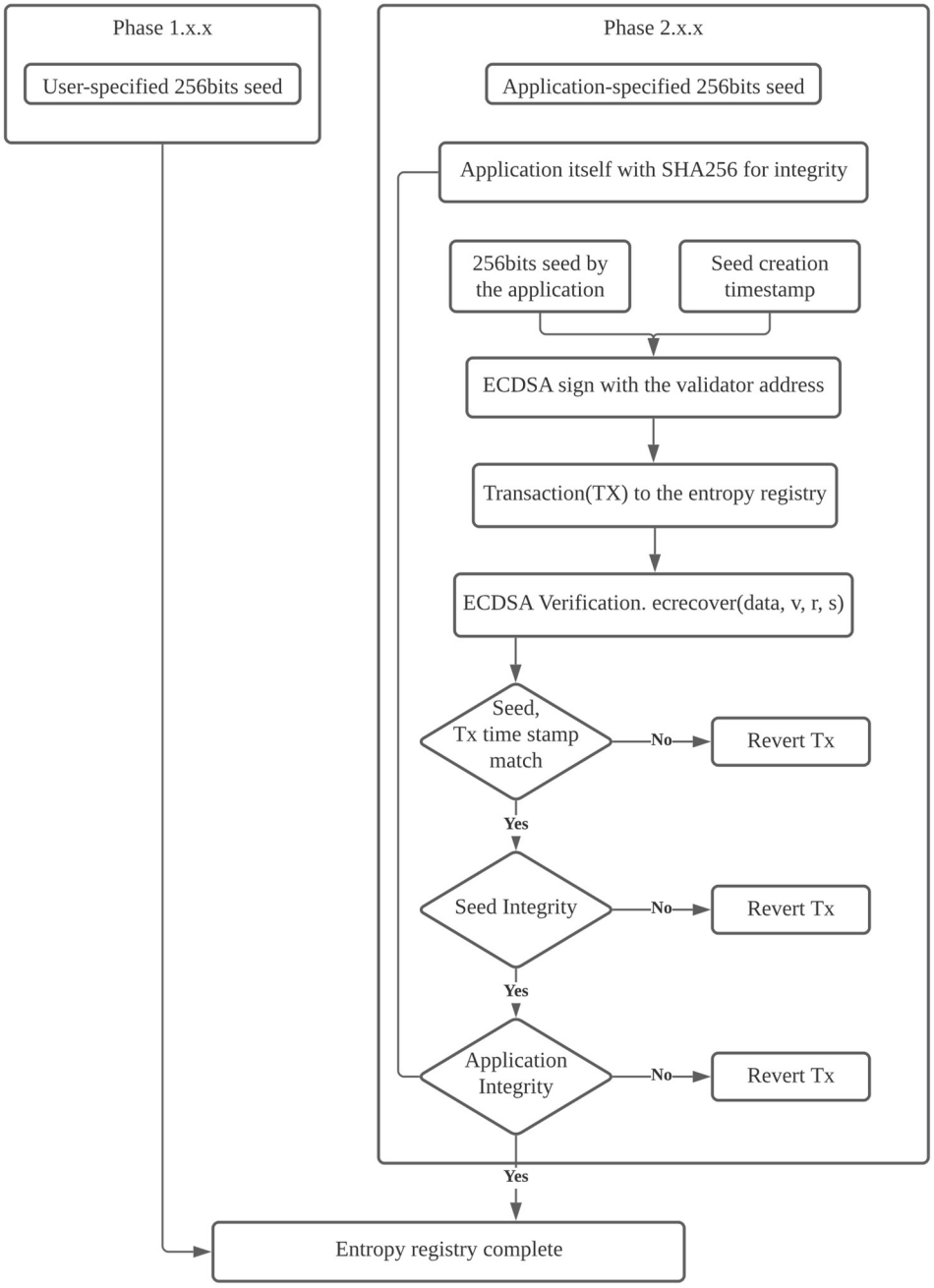
5. Entropy settle block

Even if the Entropy Register Tx is created immediately after the length of the entropy chain is filled, secure entropy generation does not happen.

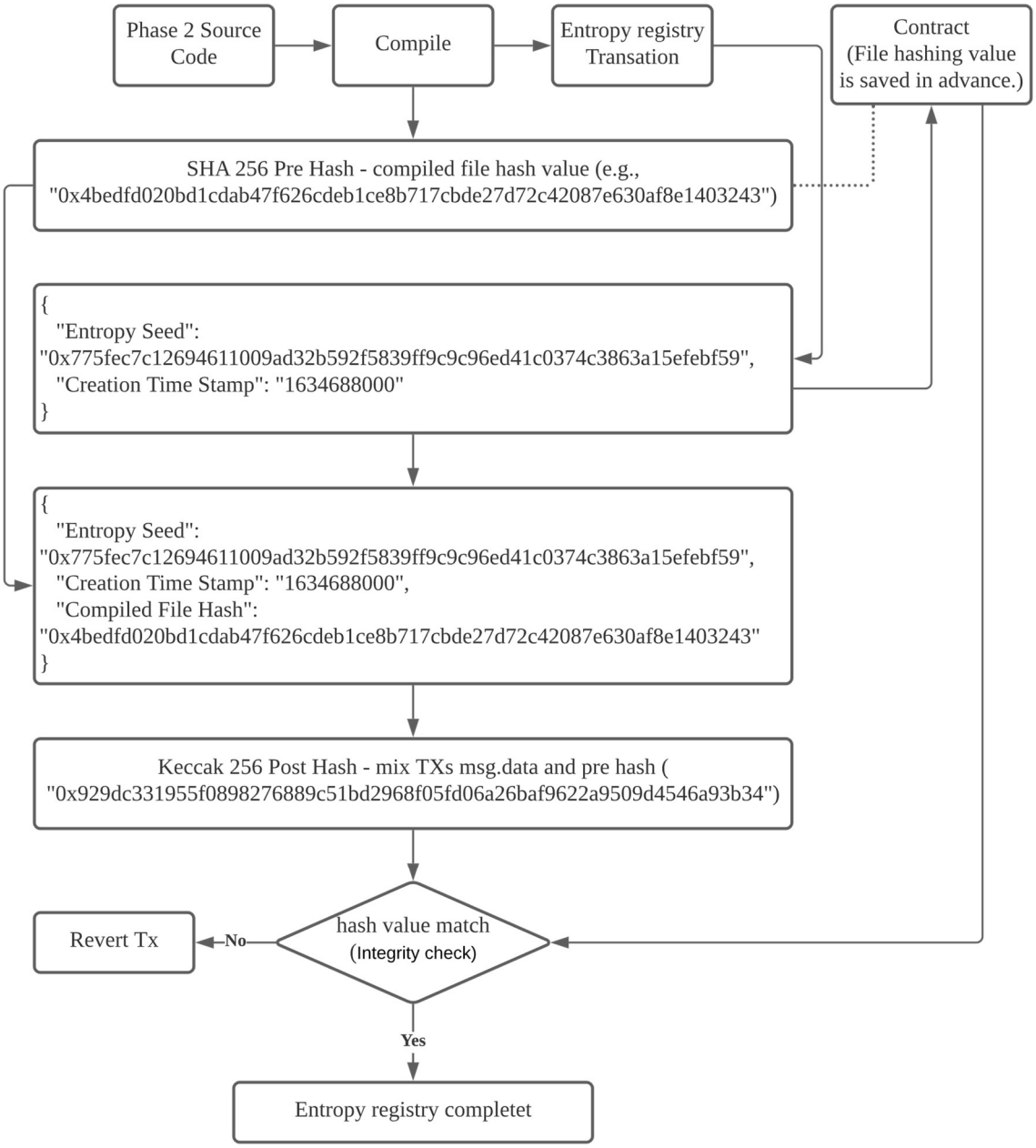


If the block number of the last element of the entropy chain is '11' and the 'minimal block count number' is 1, for secure entropy to be generated, TX must occur in 11+1 blocks or more. The 'minimum number of blocks' is a variable. The higher this value, the higher the contract can obtain entropy reliability, but the user has to wait that much longer. The initial setting is 1, even if the value increases in the future, it should be maintained at an appropriate level considering the user wait time. However, sometimes or always, in the case where more TXs than the length of the entropy chain occurs within blocks less than the 'minimal block count number', the entropy chain accepts all TXs without reverting them. And when the contract draws lots in the entropy chain, it calculates all the elements entered in the entropy chain. It exceeds the set length of the entropy chain, but the longer the chain, the higher the reliability of the entropy, so there is no reason for the contract to reject TXs. Although the length of entropy increases, the order in which users receive secure entropy does not change, and the process proceed sequentially. As a result, the length of the entropy chain is constant and, at the same time, has the feature of a variable.

6. Entropy mining, Phase 1 / Phase 2



The act of users paying gas fees and creating entropy registry TXs are similar to mining, which generates hash power by paying electricity and equipment costs. Mining is generally with CPU, but GPU/ASIC mining is recommended for efficiency. For entropy mining, Phase 2 is recommended rather than Phase 1 for unbiased entropy. Similar to effectively determining which side has the upper hand in rewards for CPU/GPU/ASIC, in entropy mining, Phase 2 gets more rewards than Phase 1. Even if there is Phase 2, Phase 1 also coexists, but since the rewards are within limited resources, Phase 1 may be wither away.



Phase 1 transmits Tx by specifying 256 bits by the user. Users can use the SHA256 hash function as it is. Unpredictability can be achieved with Phase 1 as well, but it cannot be guaranteed that the seed will not be biased, as users may do things like constantly adding +1 to the seed. Phase 2 satisfies both unbiasedness and unpredictability to increase the reliability of secure entropy. Phase 2 is mainly composed of two parts, 1) Entropy seed(256 bits seed by the application) generated by the application directly like OpenSSL, 2) Seed generation and transaction generation occur at the same time(users cannot know which seed was created until the transaction generates). Users need the validator's private key to sign the ECDSA, however the contract doesn't hide this key value. Phase 2 applications are open source. The most important is the process of checking the integrity

of the compiled entropy generation/transfer application.

User transaction consists of 1) Entropy Seed, 2) Time Stamp, and 3) the hash value of the compiled file, 4) A value obtained by hashing all data from 1 to 3 at once. The contract only knows 3) and it is not known which 1), 2), and 4) data will be transmitted. The contract internally checks the integrity by determining whether the hashed value of 1), 2), 3) matches 4) sent by the user.

However, there is a way to pass the verification of the integrity of the contract by decompiling (reverse engineering) the compiled file or manipulating the source code. Surprisingly, the method is very simple. However, the point is that there is no benefit to be gained by executing such an attack. It is self-evident that such an attack while consuming gas fees reduces the value of the contract, and this is a proportional decrease in the actual value of contract rewards that the attacker must obtain. Instead, it is advantageous to transmit the more unbiased entropy seed to the supporter's position rather than the attacker's position. Of course, in any case, the goal of generating unpredictable secure entropy is constantly being achieved.

7. Conclusion

This white paper proposed a reliable secure entropy generation system in blockchain. Entropy seed generation in a deterministic system fundamentally requires that the process is not exposed externally. However, since blockchain is a highly reliable and transparent system, it is complicated to hide these processes. In some cases, contracts can make processes unexposed, but doing so creates entropy reliability and integrity issues. Nevertheless, collecting each entropy seed, forming a group, and waiting for a specific block makes it possible to generate secure entropy while revealing all processes. Moreover, because gas fees are charged differently depending on the network state, drawing lots using aggregates all gas fees increases unpredictability. Entropy can be applied as-is to random bit generators (RGBs, random number generators, RNGs); besides, entropy seeds can also use it for high-dimensional RGBs, digital signature keys, encryption keys, and Probability distributions depending on how cryptographic applications combine. We hope that this experimental project will be useful to users.

Following in the footsteps of Satoshi Nakamoto...

References

[1] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System"

[2] Dr. Gavin Wood Founder, ETHEREUM & PARITY "ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER"

[3] Meltem Sönmez Turan, Elaine Barker, John Kelsey, John Kelsey, Mary L. Baish, Mike Boyle, "NIST Special Publication 800-90B" "Recommendation for the Entropy Sources Used for Random Bit Generation"

[4] Lemire, Daniel, "Fast Random Integer Generation in an Interval". ACM Transactions on Modeling and Computer Simulation.

[5] Stephen Canon, "An optimal algorithm for bounded random integers by stephentyrone · Pull Request #39143 · apple/swift"

[6] From Wikipedia, the free encyclopedia, "Hypergeometric distribution"

[7] From Wikipedia, the free encyclopedia, "Entropy (computing)"

[8] Sam M. Werner, Paul J. Pritz, and Daniel Perez, Imperial College London, United Kingdom "Step on the Gas? A Better Approach for Recommending the Ethereum Gas Price"