A next-generation impartial space system

Introductory papers on Square Queue are continually maintaining and upgrading and are introduced with the release.

There are many different types of games, such as sports, video, boards, and card games. Square Queue does not think of games as just entertainment, but as a world and 'space' that reflects reality. In general, the goal is to have an entertaining experience by engaging the user in a ‘space’ where a particular world view and rules are applied. These kinds of 'spaces' are innumerably derived. Still, fundamentally the reason why users participate in and play 'spaces' must be based on the belief that particular worldviews and rules operate precisely as they are seen and operate transparently. However, many systems require the intervention of a referee or operator who can be called the creator or manager of a space, which inevitably causes the rules of the game to be arbitrarily interpreted or manipulated by third parties. In 'real space,' resources such as energy, time and substance interact with various physical laws and become the core of 'real space' as the natural law. Even if the virtual space created by someone, there is a resource that can be compared with the resources of the ‘real space’ and all of these must be made up of laws that are considered to be constant, transparent and taken for granted. In particular, the natural world has an ecosystem based on probabilities that no one can predict. Therefore, the probability derived from natural law or its nondeterministic system can be unparalleled reliability, but the probability derived from the deterministic system is difficult to reliability. Thus, the ‘space’ of games must also establish transparency of operation and reliability of probability. Square Queue aims to open the Core of the Core Backend services to non-centralized systems in existing game system logic and future system logic to establish transparency and introduce reliable random values to provide the integrity of random probability. The game of the future will become a ‘space’ that can go beyond simple entertainment and become a realizing high-level society, and if transparency, credibility, and integrity are established, it will create many added values we have not yet imagined.

Introduction to Transparent system

Transparent systems have the same meaning as transparent information disclosure. Transparent information must not only disclose the data itself but also how the data is processed. However, some information may be inherent in worth only when it is closed. Therefore, information that should be disclosed, and information that should not be disclosed should be distinguished. The subject(in general, user) who uses the information assigns a true or false value to the information depending on whether it is reliable/unreliable, sure/unsure, correct/inaccurate. In self-explanatory results processing, the user may think that the value of the information is ‘True,’ but doubt about ‘True’ or ‘False’ for any ambiguous or nontransparent processing and results. Suspicious processes can be classified as requiring third-party(person) judgment and intervention or using random values that are predictable or not transparent. Even if operators and gachas in computer games are fair, it is certain that they are less reliable than transparent systems where all information is disclosed. Even if the operators and gachas in computer games claim that they are fair, it is certain that they are less reliable than transparent systems where all information is disclosed. A truly transparent system, therefore, requires random data resources that cannot be manipulated by third parties as a stakeholder, based on unpredictable entropy to minimize the debate about "True" and "False." Moreover, all rules and results must be handled automatically without anyone's intervention.

Automatic Process & Result

A system in which action for a goal begins to be perceived, followed by a process, and a result; Deterministic systems that generate clear results by internal rules without external control in these systems after entering values can be referred to as automatic processing & result systems. In other words, it is a system that requires no control and intervention, so all process routines and results must be disclosed to the user and convinced of the user. Besides, it must have a strong shield against attackers with malicious intent on the system. Square Queue decided that Ethereum Smart Contracts were the best match for this system. All code, processing, results, and history are stored and published, as well as a dominant defense against attacks.

Random Resource

Coin flip and dice, which is also used in real-world games, can produce random results in the simplest non-deterministic way. However, it is apparent that difficult to extract random values in a non-deterministic way because a computer system, including Ethereum smart contracts, is one of the deterministic Turing machines. Therefore, the computer uses a pseudo-random number generation algorithm that adds entropy to the deterministic algorithm to mimic true-random numbers. At any rate, pseudo-random numbers are determined by algorithms, so they have a specific pattern and method, and therefore are predictable. In the modern cryptology field, the key is to create pseudo-random numbers that are hard to predict or manipulate, so it use algorithms such as CSPRNG(Cryptographically secure, PRNG, deterministic random bit generator, DRBG). Therefore, it may have sufficient reliability as the random number used in the game using "CSPRNG." But, the game developer or Stakeholders have the authority to decide whether or not to apply the generated random number to a game and to extract the desired number from the CSPRNG. In particular, in the case of the commit-reveal random number output method, if someone can specify 'reveal (Even if an automatic process that mixes the time, number, weather, etc into entropy.)' for anything, its random number is not safe for any algorithm in the world.

True random number generator, TRNG

True random numbers are not some kind of algorithms, but devices that generate random numbers using the entropy of physical phenomena such as temperature, photoelectricity, and quantum phenomena, which can be accomplished using special hardware random number generators. Observing the actual physical phenomena makes it impossible to predict what will happen. Besides, the algorithm can't be figured out, so no one can't predict the random value even if using an unimaginable powerful computing device. Since the act of observing physical phenomena becomes a physical act that affects physical behavior, the act of predicting random numbers, and the act of observing itself are quite tricky. However, even with this powerful advantage, the developer has full control over how to extract the desired number from the generated random number. Even with TRNG, it must either disclose the entire process of how to apply for the generated random number or prove that the value of the random number is intact without manipulation.

Square Queue Random data source

Principle of Random Numbers in Square Queue:

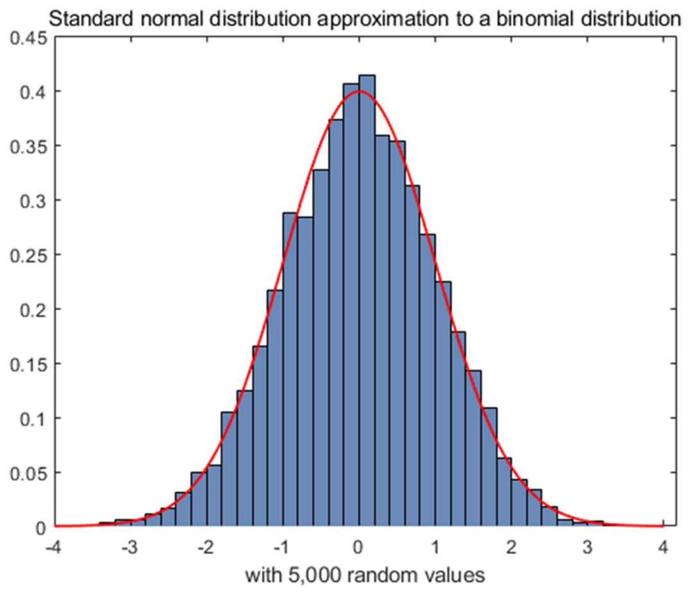
1. Random numbers should be generated by third parties or third-party devices without any stakeholder or game operator intervention.
2. Need to self-certify that random number generation is not manipulated and predictable.
3. Demonstrate that all processes from random number generation and processing to final result introduction are published as code and executed as-is.

Square Queue decided to get a random data source from provable.xyz (oraclize). The Provable

Random Data Source leverages the Ledger Proof and a custom application to generate unbiased random numbers and provide them on demand to blockchain and non-blockchain based applications. The Ledger Proof leverages both the code attesting and the device attesting features to attest to any third-party that the applications developed by Provable are running in a TEE of a true Ledger device. The Ledger Proof is currently used by the Provable Random Data Source to provide untampered entropy to smart contracts, blockchain and non-blockchain applications.

Square Queue reward system

From a player's point of view, there are two consequences: win and lose. Although there is a result of a tie, it can be included in the category of win or loss if there is a result predicting that a result will be a tie and there is a reward for it. In probability theory and statistics, Win/loss can be described as the binomial distribution with parameters n and p is the discrete probability distribution of the number of successes in a sequence of n independent experiments, each asking a win-loss, and each with its own Boolean-valued outcome: success/fail/yes/no/true/false/win/loss. To put it very directly from the player's point of view, if a player rolls a dice ten times, the most straightforward kind in the game, and count the number three, this is a binomial distribution with n = 10 and p = 1/6. Binomial(n, p), then for sufficiently large n, The binomial distribution is generally considered to be a pretty good approximation for the standard normal distribution. Thus, the game is performed numerous times instead of just once, so in each independent trial, each trial has probability and eventually approximately the standard normal distribution.



$$\mathcal{F}(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \; : \; \mu = 0, \sigma = 1$$

The normal distribution is the most important probability distribution in statistics because it fits many natural phenomena. For example, heights, blood pressure, measurement error, and IQ scores follow the normal distribution. The normal distribution has many different shapes depending on the parameter values. However, the standard normal distribution is a special case of the normal distribution where the mean(μ) is 0 and the standard deviation(σ) is 1. This distribution is also known as the z-distribution. After you standardize your data, you can place them within the standard normal distribution. In this manner, standardization allows you to compare different types of observations based on where each observation falls within its own distribution. In this respect, This means that if a reasonable and reliable random number is provided without the involvement of a developer or operator, it can offer an impartial gameplay result to anyone. Square Queue concluded that if it could derive the inverse function of the standard normal distribution, it could give the player a reasonable reward based on a mathematical probability field:

$$\mathcal{F}(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \; : \; \mu = 0, \sigma = 1 \; \rightarrow \; \mathcal{F}(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

$$\mathcal{F}(z) \times \sqrt{2\pi} = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \times \sqrt{2\pi} \; \rightarrow \; \sqrt{2\pi}\, y = e^{-\frac{z^2}{2}}$$

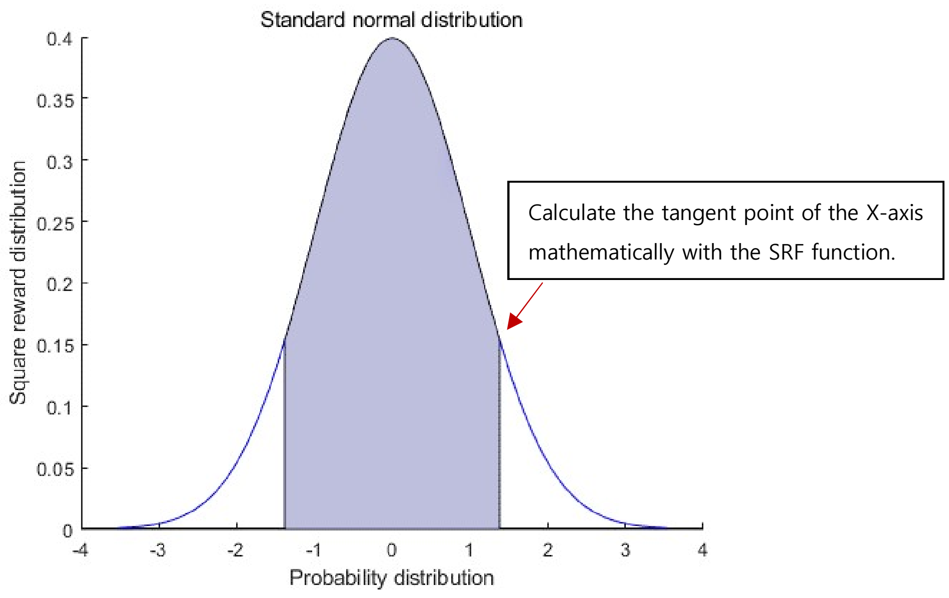
$$\sqrt{2\pi}\, y \times \log e = e^{-\frac{z^2}{2}} \times \log e \; \rightarrow \; \log e \sqrt{2\pi}\, y = -\frac{z^2}{2}$$

$$(-2) \times \log e \sqrt{2\pi}\, y = (-2) \times -\frac{z^2}{2} \; \rightarrow \; -2 \ln \sqrt{2\pi}\, y = z^2$$

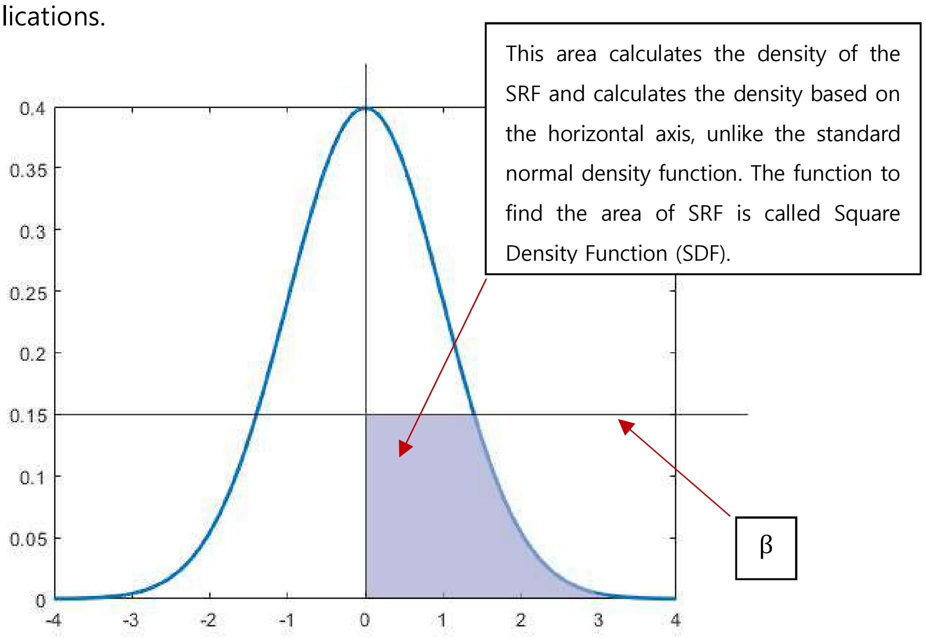
$$\sqrt{-2 \ln \sqrt{2\pi} y} = |z| \; \rightarrow \; |\boldsymbol{y}| = \sqrt{\boldsymbol{-2 \ln \sqrt{2\pi} z}}$$

In code : sqrt(-2*(log(sqrt(2*pi)*(z))))

Square Queue will designate this function as the 'Square distRibution Function' (SRF), which can be used to determine the mathematically impartial player's reward. Square Queue is a platform that gradually expands from the simplest game, so for the purposes of this white paper, it will use dice, one of the simplest game types. Since the dice have a number of six cases, the probability is $p =$ $= 1/6$, and a z can be used to obtain a specific area (probability p) obtained by integral the standard normal distribution. The opposite can also be achieved using the standard normal distribution table.



SRF can be used as a complement to the standard normal distribution table, making the proof of equations more certain. Moreover, the integration to calculate the SRF density can be used in a large number of applications.



Square Density Function (SDF)

$$\int_{0}^{\beta=\frac{1}{\sqrt{2\pi}}r} \sqrt{-2\ln\sqrt{2\pi}z}\,dz \ : \ (0 \le r \le 1)$$

SDF can also be used to calculate reward rates in games. The following table shows the reward rate for using one dice game. x15 percent reward bonus is a gift to users in the Square Queue.

n= (6(one dice) x 2) -1	SDF	0.5(100%)->1(100%) Multiply by 2	SDF rate (inverse of 1/SDFx2)	x 15%
$\beta = \frac{1}{\sqrt{2\pi}} \times \frac{1}{n}$	0.093688	0.187376	5.336868	6.14x
$\beta = \frac{1}{\sqrt{2\pi}} \times \frac{2}{n}$	0.166346	0.332691	3.005790	
$\beta = \frac{1}{\sqrt{2\pi}} \times \frac{3}{n}$	0.228871	0.457741	2.184641	2.51x
$\beta = \frac{1}{\sqrt{2\pi}} \times \frac{4}{n}$	0.283803	0.567605	1.761788	
$\beta = \frac{1}{\sqrt{2\pi}} \times \frac{5}{n}$	0.332318	0.664636	1.504584	1.73x
$\beta = \frac{1}{\sqrt{2\pi}} \times \frac{6}{n}$	0.375031	0.750063	1.333222	
$\beta = \frac{1}{\sqrt{2\pi}} \times \frac{7}{n}$	0.412235	0.824470	1.212901	1.39x
$\beta = \frac{1}{\sqrt{2\pi}} \times \frac{8}{n}$	0.443967	0.887934	1.126210	
$\beta = \frac{1}{\sqrt{2\pi}} \times \frac{9}{n}$	0.469983	0.939965	1.063869	1.22x
$\beta = \frac{1}{\sqrt{2\pi}} \times \frac{10}{n}$	0.489545	0.979089	1.021358	
$\beta = \frac{1}{\sqrt{2\pi}} \times \frac{11}{n}$	0.500000	1.000000	1.000000	

Square Queue Token (SQ coin)

Smart Contracts and Provable(Oraclize) / SDF(Square Density Function) can be used to create a system that produces impartial processing and results in various spaces like games and virtual society. But furthermore, the user wants ‘somethings’ to have a higher probability of winning than other users, even if they invest more resources and even if it's not impartial. 'Somethings' are usually systems such as levels, items, or gachas that allow users to play a higher role than other users and have higher winning odds. For example, a two-dice system, users can increase their odds by using the Square Queue Token when in a face-off between players.

- A player predicts what will be the sum of the two dice and sends Ether to join the game. (The number of cases two dice is 36.)
- A player can only select one prediction number by default but can submit up to 10 prediction numbers using the Square Queue Token.
- Square Queue Tokens sent to game contracts are discarded and cannot be reused.

The gameplay method that combines Ethereum and Square Queue Token introduce a select without a replacement system that uses the Hypergeometric probability density function to set the number of additional prediction transfers using Tokens to -1 of the previous Tokens. If the previous Token Bonus Play(Send more than one predicts numbers using Token.) count is zero, it proceeds to zero as it is. In probability theory and statistics, the hypergeometric distribution is a discrete probability distribution that describes the probability of x successes (random draws for which the object drawn has a specified feature) in r draws, without replacement, from a finite population of size n that contains exactly k objects with that feature, wherein each draw is either a success or a failure. In contrast, the binomial distribution describes the probability of x successes in r draws with replacement.

$$y = \mathcal{F}(x|n,k,r) = \frac{\binom{k}{x}\binom{n-k}{r-x}}{\binom{n}{r}}, \qquad \binom{\alpha}{\beta} = \frac{\alpha!}{\beta!\,(\alpha-\beta)!}$$

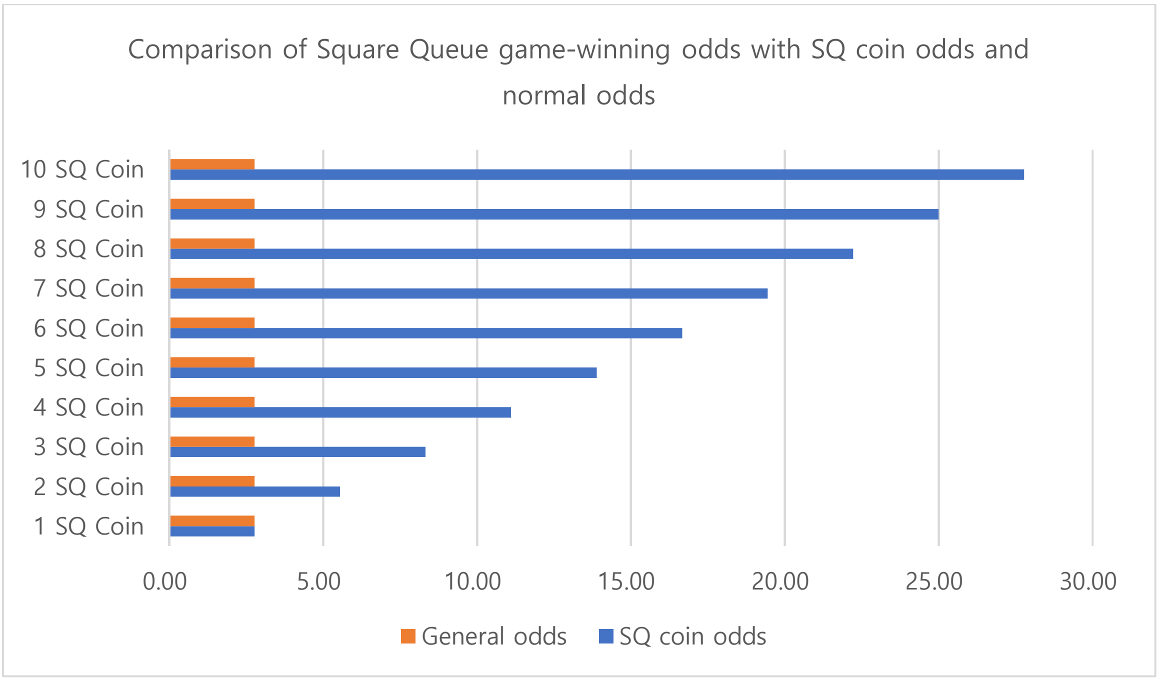
n = Number of total cases : 6x6=36

k = Number of Success in cases : 1

r = Number of draws available: 0~10 (It can increase the number of draws to 1 or more with SQ coin only.)

x = The probability that will k be drawn.

Using the hypergeometric function, it can see how the Square Queue Token can reasonable rules upon 'spaces'.



	1	2	3	4	5	6	7	8	9	10	X10
Coin odds	97.22%	94.44%	91.67%	88.89%	86.11%	83.33%	80.56%	77.78%	75.00%	72.22%	
common	2.78%	5.56%	8.33%	11.11%	13.89%	16.67%	19.44%	22.22%	25.00%	27.78%	

The SQ coin input ratio may change (i.g .: x10). Decimal point units are also applied. The graph and table above are values obtained by calculating the hypergeometric distribution function and applied to the Square Queue Token (referred to as SQ coin). And SQ coin has the following constraints.

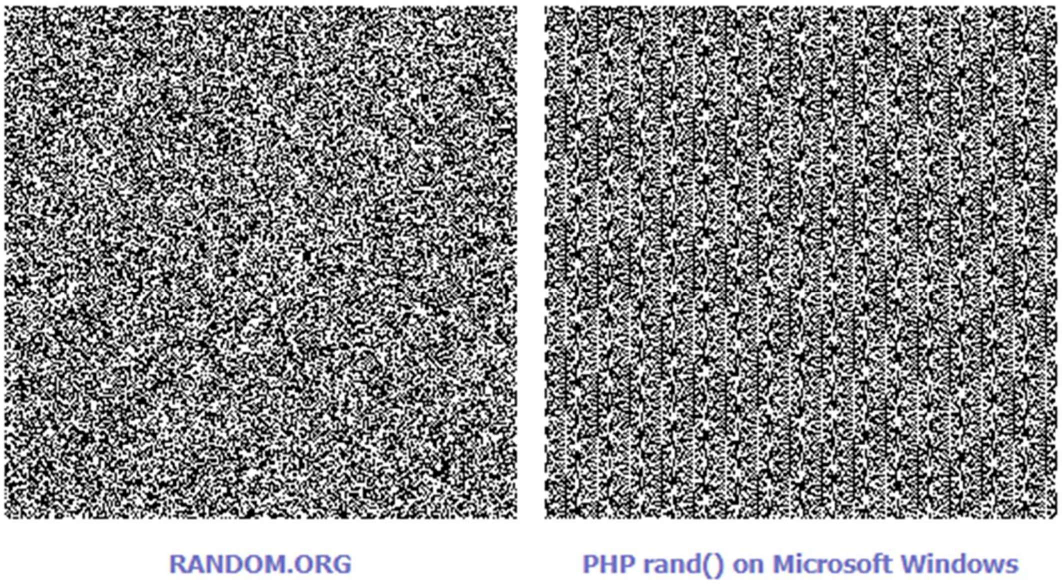
- Number of times r was increased use SQ Coin in the previous round (1 ~ 10) = $\mathrm{p}r$
- Block Number(with $r = 0$) = $\mathrm{b}r$
- To submit in the current round $r = cr$
- $1 \leq cr \leq \mathrm{p}r - 1$ must be satisfied. (If 10 in the previous round and 3 in the current round, a player can only submit 1-2 in the next round)
- cr may be used when $\mathrm{B}r + 200 \leq$ Currunt Block Number. (It may be adjusted according to the future Ethereum block time.)

As stated, even if you have SQ coins, a player can't always apply it to r to increase player game's odds. However, due to the nature of SQ Coins that can be a uint of account, medium of exchange, divisible, fungible and transfer, players will be able to join the game with multiple Ethereum Accounts and always increase the odds of the game. To prevent some negative behavior of this kind, the process is as follows:

- A player sends to the contract a designated Ether, a number, and an SQ coin to designate r. r is zero if no SQ coin is specified.
- The contract checks whether all players have participated. It then determines whether the game of the corresponding round can consume SQ Coin using 'Provable random data resource' and refunds the SQ Coin to players if the round is not available for SQ Coin.

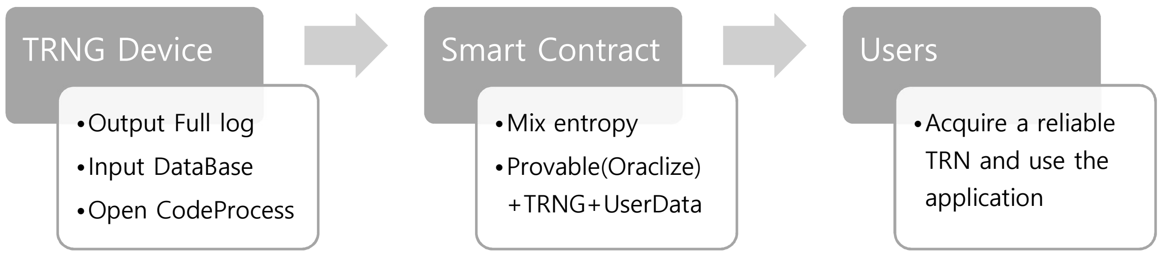
SQ Token on TRNG (SQ coin)

Deep learning requires vast quantities of data and powerful computing power to reach a high level of the realm. This is a firm requirement and unfortunately, we cannot make any exceptions. In order to perform high-level deep learning, data pre-processing, zero centering, and data normalization must be preceded. In fact, it is also complicated to meet precisely this. After meeting all these conditions, deep learning will repeat the process of assigning millions of billions and even countless parameters to a function. In a small network, random values widely known for parameter values may be used. However, the role of random numbers in large-scale self-learning becomes significant at a high level.

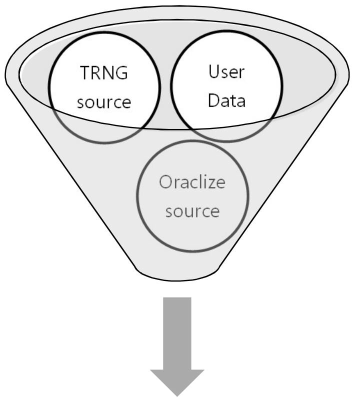


Ref: http://boallen.com/random-numbers.html

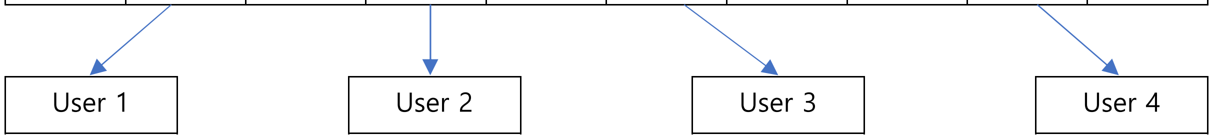
The left is TRNG (True-random number generator), and the right is PRNG (pseudo-random number generator). It is difficult to get the desired result if deep learning uses PRNG, which performs a lot of iterative calculations. PRNGs have a clear pattern, and the higher the resolution, the sharper. Deep learning on the blockchain is difficult to implement compared to the centralized platform, and there is a problem of the cost-to-benefit ratio. However, considering not only the blockchain itself but also decentralized supercomputing, the blockchain platform still has an attractive future for deep learning. The problem that must be solved in advance must be indiscriminately supporting TRNG on the blockchain. Even if TRNG is ported to the blockchain, it is difficult to gain users' trust in the data source. It must increase users' credibility by proving to users that random values have been generated on the TRNG device.



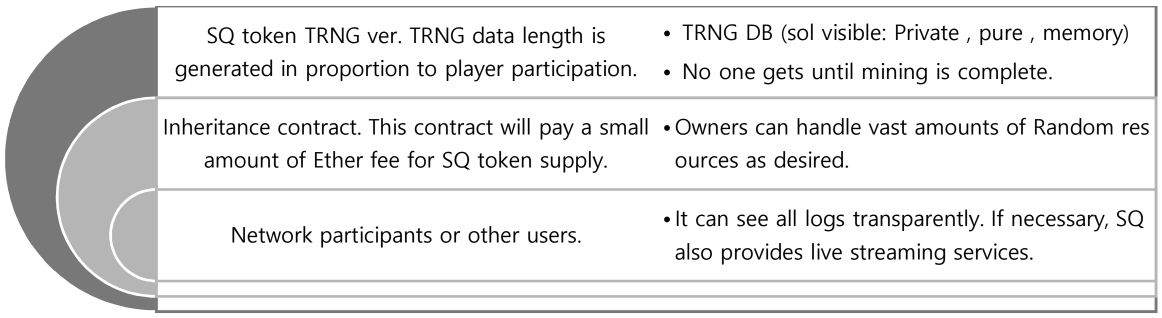
TRNG's unique ID, output value, and processing should be provided as a whole log and can be viewing. If necessary, a live streaming service should be provided. In order to gain user trust, all data generated from the TRNG node must be provided transparently. Square Queue transparently discloses all data provided by TRNG.



e98ea	4084	0008	a110	06f2	16f6	064f	19ed	e13c	ccc7
0436e1	d004f6	66fb	bf548	de347	7d19c	fbb4e	48de3	bb4e7	004f6



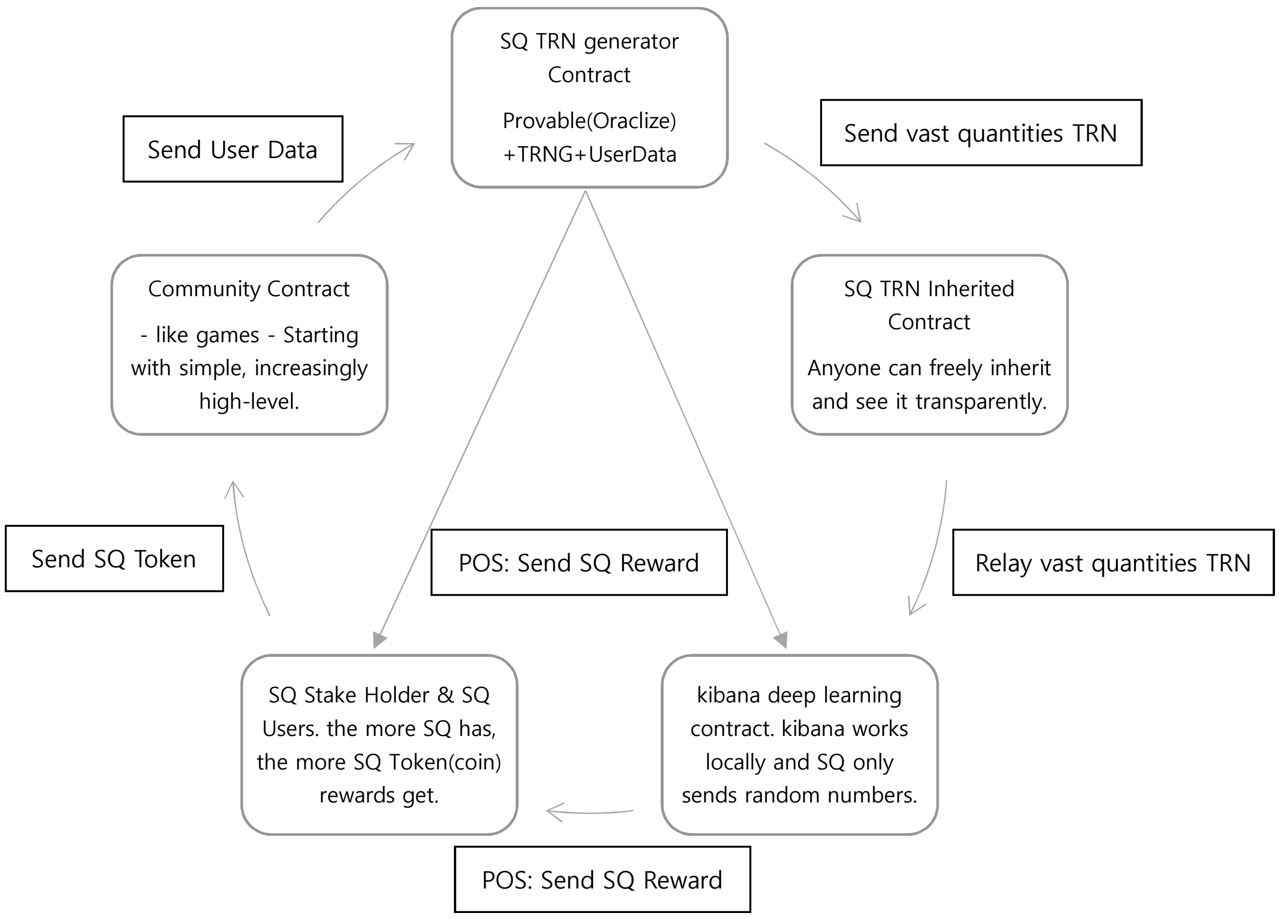
Mix all the entropy with SHA-3 and distribute it to users. In fact, it is rare for users to need an entire string of length 64. Only the length of the string required by the user is returned. The core of SQ Token is to receive user-specific data in order to avoid being suspected of a reliable random number by TRNG and Oraclize alone. That is part of SQ games. The user's data is a unique entropy such as public key, Tx time, and play number. The user should inherit and use the SQ contract, not the way to callback the SQ contract. This saves gas and allows vast quantities of random values to be provided.



SQ token TRNG ver. TRNG data length is generated in proportion to player participation.	• TRNG DB (sol visible: Private , pure , memory) • No one gets until mining is complete.
Inheritance contract. This contract will pay a small amount of Ether fee for SQ token supply.	• Owners can handle vast amounts of Random resources as desired.
Network participants or other users.	• It can see all logs transparently. If necessary, SQ also provides live streaming services.

The essence of the SQ random distribution service is that it can generate vast quantities of data for deep learning and use it with very little gas. SQ's own deep learning service uses kibana service and will be released along with the yellow paper.

SQ token supply chain (Max Supply = 50,000,000)


SQ TRN generator Contract
Provable(Oraclize) +TRNG+UserData
Send User Data
Send vast quantities TRN
Community Contract
- like games - Starting with simple, increasingly high-level.
SQ TRN Inherited Contract
Anyone can freely inherit and see it transparently.
Send SQ Token
POS: Send SQ Reward
Relay vast quantities TRN
SQ Stake Holder & SQ Users. the more SQ has, the more SQ Token(coin) rewards get.
kibana deep learning contract. kibana works locally and SQ only sends random numbers.
POS: Send SQ Reward

The SQ Reward System will be operated after Two-Dice and the token swap is performed first. The reward rate will be released along with the yellow paper. The token allocation of SQ team is limited to within 1%, and there is no Token Sale other than IEO.

- 1/2 of SQ tokens of Community Contract (Game and etc,.) are automatically burning out. The other 1/2 is transferred to the SQ TRNG contract with "User Data", and will be rewarded to the SQ Stake Holder.
- Kibana deep learning system itself consumes of SQ token (coin).
- Theoretically, the SQ token can converge close to the Max supply, but never reaches the Max supply.

Square Queue Multi-player system

Two Dice game is Square Queue's most basic multi-play game, using both SDF and SQ coin and following the probability of Hypergeometry Distribution. Players make a transaction to participate in the game in the Square Queue Contract, and the Contract records the player's transaction until N blocks pass. When the N block passes and the player joins the contract, the player joins the round as the final player, and at the same time, whether the game consumes SQ coin will be determined through Provable Resource. If the game does not consume SQ coins, refund all SQ coins to all players. The fees are paid by Square Queue. After the game is drawn, the winners and losers are divided and the losers lose all Ether which participated in the game and winners get all Ether of losers proportionally based on SDF(At this time, the SDF calculates the winner's Ether amount 'ratio' proportionally and divides the winnings use 'ratio'). By monitoring the Transaction of the Contract, players can see how many Ether has been put into which number before the game is revealed by Provable. Thus, players can participate in the small amount of Ether with the number of games with the least amount of Ether submitted and seek high Ether dividends at low loss rates. In order to avoid the vulnerability of the contract, the Contract calculates the proper allocation to all players based on the normal distribution / SDF, and hypergeometric distribution when the transaction of the last player in block N-1 occurs. If it is determined that proper dividends are not available, the Contract will refund all Ether and SQ coins to the players and the game in that round will be discarded. The player who created the last transaction in blocks N and N-1 will consume more gas than others to settle the game. It is an excellent choice to avoid the last gas by joining a round earlier than other players.

Conclusion

The comments and CODE of Square Queue smart contract are written in plain language and easy CODE for those who are not familiar with programming languages. Please read the source code for information on how to handle each process that is not on this white paper, and various other useful information. There may be a way to point of view the Square Queue as a game, but to be clear, the Square Queue aims to create an impartial 'space,' which is the area where all user activity is possible. Square Queue designates the beginning of 'space' as a 'game' but will ultimately develop it towards 'virtual society' and hope that the impartial laws & rules of the virtual society will be projected into the real world. And from the bottom of the heart, users should play Square Queue games in moderation. Please note that the following table shows the consecutive one-dice games as a hypergeometric cumulative distribution.

	1/6	2/6	3/6	4/6	5/6	6/6
0	0.1667	0.3333	0.5000	0.6667	0.8333	1.0000
1	0.3056	0.5556	0.7500	0.8889	0.9722	1.0000

2	0.4213	0.7037	0.8750	0.9630	0.9954	1.0000
3	0.5177	0.8025	0.9375	0.9877	0.9992	1.0000
4	0.5981	0.8683	0.9688	0.9959	0.9999	1.0000
5	0.6651	0.9122	0.9844	0.9986	1.0000	1.0000
6	0.7209	0.9415	0.9922	0.9995	1.0000	1.0000
7	0.7674	0.9610	0.9961	0.9998	1.0000	1.0000
8	0.8062	0.9740	0.9980	0.9999	1.0000	1.0000
9	0.8385	0.9827	0.9990	1.0000	1.0000	1.0000
10	0.8654	0.9884	0.9995	1.0000	1.0000	1.0000
11	0.8878	0.9923	0.9998	1.0000	1.0000	1.0000
12	0.9065	0.9949	0.9999	1.0000	1.0000	1.0000
13	0.9221	0.9966	0.9999	1.0000	1.0000	1.0000
14	0.9351	0.9977	1.0000	1.0000	1.0000	1.0000
15	0.9459	0.9985	1.0000	1.0000	1.0000	1.0000
16	0.9549	0.9990	1.0000	1.0000	1.0000	1.0000
17	0.9624	0.9993	1.0000	1.0000	1.0000	1.0000
18	0.9687	0.9995	1.0000	1.0000	1.0000	1.0000
19	0.9739	0.9997	1.0000	1.0000	1.0000	1.0000
20	0.9783	0.9998	1.0000	1.0000	1.0000	1.0000
21	0.9819	0.9999	1.0000	1.0000	1.0000	1.0000
22	0.9849	0.9999	1.0000	1.0000	1.0000	1.0000
23	0.9874	0.9999	1.0000	1.0000	1.0000	1.0000
24	0.9895	1.0000	1.0000	1.0000	1.0000	1.0000
25	0.9913	1.0000	1.0000	1.0000	1.0000	1.0000
26	0.9927	1.0000	1.0000	1.0000	1.0000	1.0000
27	0.9939	1.0000	1.0000	1.0000	1.0000	1.0000
28	0.9949	1.0000	1.0000	1.0000	1.0000	1.0000
29	0.9958	1.0000	1.0000	1.0000	1.0000	1.0000