Quanta lottery white paper draft v0.4.5

Arman Ospan, Krzysztof Skrzypski and Chris Hatch

February 22nd, 2017

Abstract

This document describes the first implementation of a fully automatic and trustless Decentralized Autonomous Lottery Game using smart contracts on the blockchain.

1 Introduction

Decentralized Autonomous Lottery Game built on the Ethereum blockchain can become a major drive of change in the global lottery industry. The lottery industry holds a 29% share of the global gambling revenue [1], and is facing challenges due to the growth of the digital economy. The worldwide demand for more transparent, fair services grows along with the advance of technologies that enable the change. The distributed, decentralized nature of blockchain technology is very promising for financial technology innovation, and, in particular, for solving some existing problems of the lottery segment. Ensuring fairness of the game itself and fair distribution of funds, along with licensing and accessibility are among the most common problems that lotteries face.

A decentralized application built on the Ethereum blockchain with the use of smart contracts may be completely open-source, can operate with a high degree of autonomy, and its data and records are cryptographically stored in a public, decentralized blockchain. Available online and borderless, such a system will solve several challenges of traditional lotteries.

Random number generation (RNG) method that is secure and fair, and makes manipulation and fraud impossible is necessary to develop the proposed lottery. Currently a 100% secure method is not yet possible with the current version of Ethereum decentralized protocol [2], but this problem can be solved in the upgraded versions of the Ethereum blockchain or other cryptocurrency platforms. We suggest an RNG method on top of Ethereum protocol that is automated and makes manipulation practically impossible (computationally and financially infeasible).

Further work will eventually create a next generation, truly incorruptible and automated lottery system. Different new possibilities may open up in the future, such as upgrading and migrating the lottery to a new blockchain using DAO voting, or running the lottery on many blockchains, connecting lotteries from different blockchains to create even larger prize pool.

2 Global lottery industry

The global lottery industry grew from $187.1 billion in 2004 to $284.3 billion in 2014, representing a compound annual growth rate (CAGR) of 4.3% over ten years. In 10 years from 2004 to 2014, lottery sales have declined only once, evidencing the stability of the business. More than 64% of global lottery sales come from Europe and North America [3].

At the same time, with growing penetration of the Internet and mobile phone adoption online and mobile lotteries are growing everywhere. The total revenue of mobile lotteries in 2013 is estimated to be $34 billion, while 44% of the amount is generated in Europe and North America. The pace of technology adoption can be even faster in the developing markets.

Up until recently, there were mostly B2C lottery operators. With the advent of online and mobile lotteries, there are now more B2B companies offering online lottery platforms. Quanta can operate as a global blockchain protocol, but can also be offered to governments of specific countries as a solution to ensure organizing a fair lottery.

Globalization, economic growth of the developing world and information technology progress are the factors that will accelerate changes in the lottery industry, especially in the emerging markets. Global lottery industry is moving closer to cross-border, high technology format of lotteries, and blockchain technology can be the stepping stone needed for this innovation.

3 Challenges faced by lottery industry

Blockchain technology may solve trust and fairness issues faced by the lottery industry. As all centralized services, lotteries have the issues of trust. Some of the most common problems of the lottery industry are described below.

Fairness

Ensuring fairness of the game remains the biggest challenge. Lottery players may question the possibility of manipulation or fraud, e.g. just several possible concerns:

a) is the ticket and lottery real?
b) is the money really accumulated in one pool and used for social causes?
c) is RNG truly random and secure?
d) is the jackpot winner real?
e) is the prize paid on time?

Famous manipulation cases

- **Hot Lotto fraud scandal (2010)**
  Eddie Tipton, the former information security director of the US Multi-State Lottery Association, installed software code that allowed him to manipulate random number generator and predict winning numbers on specific days of the year and win a $14.3 million jackpot in 2010. Tipton was convicted and sentenced to ten years imprisonment in 2015.
- **The 1980 Pennsylvania Lottery scandal (also known as the Triple Six Fix)**
  It was a plot to rig the Daily Number, a three-digit game the Pennsylvania Lottery offers. All of the balls in the three machines, except those numbered 4 and 6, were weighted, meaning that the drawing was almost sure to be a combination of those digits. The scheme was successful in that 666 (one of the eight combinations that the conspirators were hoping for) was drawn on April 24, 1980; however, the unusual betting patterns alerted the authorities. The chief participants in the plot were sent to prison, and most of the fraudulently acquired winnings were never paid out.

Distribution of funds

Lotteries serve as funding for charity and other social projects. However, in many cases, especially in countries with high levels of corruption, fair distribution of funds can be questioned by the players.

Availability

Users from other countries can not easily take part in some of the biggest lotteries in the United States and are limited to playing smaller local lotteries due to the smaller size of their domestic market, underdevelopment of the lottery industry at home or strict regulation. New online solutions that allow such players to buy the U.S. lottery tickets have recently gained some popularity, but the service charges, as well as risks of fraud and mismanagement, remain high.

Regulations

In some regions around the world lotteries are not legal, and in most other regions only governments and affiliated organizations can obtain a license to operate a lottery business. Corruption schemes that can also affect distribution of funds may take place in different regions.

4 Advantages of Quanta

Due to introduction of blockchain technology, Quanta is the first step to the new generation of global lotteries. Current market capitalization of all cryptocurrencies is already equivalent to 0.016% of the world's GDP in 2015 according to the International Monetary Fund. If the same ratio is applied to the lottery market, the global cryptocurrency lottery market size can be as big as $45 million in 2016, and the addressable market will continue to grow along with the adoption of cryptocurrencies across the world. Major improvements that a mass decentralized lottery will bring are described below:

- **Introduction of a borderless, officially licensed lottery**
  The lottery will be easy to access for any Internet user able to buy a cryptocurrency. Quanta protocol will obtain a license in one of the leading cryptocurrency or online gaming jurisdictions (e.g. Isle of Man). The protocol will be tested and approved to ensure the randomness of results in the lottery. Quanta Co. Ltd. will obtain lottery licenses in relevant markets.
- **Up to 85% rate of winning pool money**
  In traditional lotteries only 45-60% of revenue is distributed back to users as prizes. 25-40% can be the government tax or collection for public needs, and 15% cover the costs, profit of sellers or organizers. With the use of blockchain a global online lottery can be created where **up to** 85% of the revenue is distributed back to ticket buyers. Remaining 15% of the revenue can be distributed to a special pool for social causes, profit of the operator/sellers and shareholder dividends. Quanta will start with a 70% winning pool rate and will work on minimizing lottery costs that will eventually lead to higher prize pool for lottery players.
- **RNG method that is fully automatic and it is practically impossible to manipulate it**
  Random number generation in Quanta will take place on Ethereum and involve commitment scheme based method, the idea came from paper "A random zoo: sloth, unicorn, and trx" by Arjen K. Lenstra and Benjamin Wesolowski [31] and is called RANDAO++ [33] on Ethereum platform. To successfully manipulate RNG and match it with a hash number of a previously purchased ticket is a practically impossible task. The

adversary will have to be able to perform time taking *sloth* calculation based on SHA-3 hashing function on a single device that surpasses e.g. 10-100 times the capabilities of the fastest available implementation. He would have to be in control of one of the seeds, e.g. we can use Bitcoin blockhash as a last source of seed, which means that the adversary would have to be able to control a significant part of the Bitcoin mining power to be able to gain one chance to manipulate the lottery outcome. An adversary would have to hold a number of tickets that is big enough to have a higher chance of matching the jackpot in alternative result of RNG with a previously purchased ticket. Apart from technical infeasibility of the task, the amount of investment required to ensure a high chance of successful manipulation can exceed to the size of jackpot and will disincentivize an effort of manipulation.

5 Random number generation (RNG)

True randomness doesn't occur in the physical world, there is always a cause and effect, however it is very difficult to make observations of the rules that triggered the random event and extremely difficult to collect data at desired levels to explain what happened. Randomness may occur in the digital world and programming, where e.g. a deterministic function with sufficient source of entropy as a seed with a delay (e.g. by solving computationally hard problem) can generate unpredictable random numbers.

What is a random number?

A random number is generated as a set of possible, equally probable values. The values are uniformly distributed and it is not possible to predict future values based on present or past ones. In statistics uniform distribution is the distribution where for each member of the family, all intervals of the same length on the distribution's support are equally probable. For example it is equally probable to toss a coin, each time probability of tossing head or tails is always equal to 50%. Statistically the chance to toss heads four times in a row is equal to 6.25%. The ssame rule applies to randomness, bit sequences can occur in statistically acceptable intervals.

Random and pseudorandom numbers have many uses in science, art, statistics, cryptography, gaming, gambling, and other fields. They are necessary for cryptographic applications, e.g. common cryptographic algorithms are using keys that must be generated in random fashion. Random number generators used in cryptography or to determine winners of the digital lotteries may need to meet stronger requirements than for use in other applications. The output of random number generator must be unpredictable and uniformly distributed.

There are two general types of random number generators: physical random number generators (RNG) and software pseudorandom number generators (PRNG), both can be used as a source of randomness for gambling applications.

RNG

This generator uses a nondeterministic source of entropy along with a function that is processing input data and returning processed output. The source of entropy comes from statistically random noise signals. It typically consists of some physical quantity, such as the noise in an electrical circuit, thermal noise or the quantum effects in a semiconductor. These processes are in theory completely unpredictable. The main problem with this approach on the

decentralized blockchain is the difficulty of proving that random number is fresh and not manipulated.

PRNG

This generator uses one or more inputs called seeds and generates pseudorandom number as an output of deterministic function, it may use RNG as a source of seed. It is not possible to get a truly random number using pseudorandom number generator, for a certain input PRNG will always generate the same output. The seed itself needs to be nondeterministic and unpredictable. Quanta lottery will use SHA-3 (Keccak) hashing function as a PRNG, more detailed description below.

RNG for online lottery on the blockchain

There are many problems to consider when choosing RNG method for lotteries and gambling websites. There is a need for a secure and verified source of entropy as well as a cryptographically secure pseudorandom number generator. Decentralized blockchain requires a deterministic algorithm which given a random binary sequence as an input, outputs pseudorandom bit sequence, a binary sequence that is indistinguishable from random. Currently there is no blockchain that could provide secure source of randomness on the protocol level.

The target blockchain for Quanta lottery development is not yet specified, at the current stage of technology stack development the prototype of Quanta will be based on the Ethereum blockchain [2]. Ethereum comes with a built-in "Turing complete" programming language called Solidity for developing smart contracts, the concept that was introduced by Nick Szabo [6][7] Solidity source code is compiled to the bytecode that runs on EVM (Ethereum Virtual Machine) and is executed by all nodes working on the decentralized network. Currently Ethereum has the second-largest market capitalization among all existing cryptocurrencies, therefore it is a good target platform for a lottery with a big prize pool.

In the near future the lottery could be deployed on a much more advanced platform, as other gaming blockchains such as CARDANO [12] or Peerplays [13] are being developed. Hawk will provide on chain privacy and has already proven (though not released) implementations sealed auction and rock, paper, scissors implementations [41]. Tor Project is working on a decentralized random number generator [10] and there are few other solutions being developed at the time this paper is written. Ethereum itself has plans to implement on-protocol randomness in one of the future releases, however there is no official statement from Ethereum on that matter yet [23].

Currently secure RNG could not be done on the Solidity programming language level as there is no existing implementation of randomness generator on the Ethereum protocol level. The best known solution to deal with the problem of secure RNG on Ethereum blockchain is using *sloth* and *unicorn* algorithms proposed by Arjen K. Lenstra and Benjamin Wesolowski in paper [31] and named RANDAO++ [32][33] on Ethereum.

Randomness testing

One of the requirements to obtain a lottery license is to pass randomness tests of the RNG method. Quanta lottery will use SHA3 for hashing the inputs to output pseudo random number. At least one of the input seeds has to meet requirements for a uniformly distributed random number. The commitment scheme requires at least one seed with sufficient entropy

input from an honest participant to generate an unpredictable random number. Previously generated random number will also be used as one of the seeds. More seeds will be fetched from RANDAO++ players or external oracles to fetch e.g. blockhashes from Bitcoin or other cryptocurrencies.

With NIST Statistical Test Suite [28] we have analyzed large quantity of Ethereum block hashes concatenated and transformed into a bitstream of 400000 bits, which is equal to 400000 coin tosses. Ethereum block hashes as well as outputs from SHA-3 Keccak are passing all NIST statistical tests. Keccak as PRNG is described in the following "*SHA-3 Keccak as PRNG*" section.

Randomness requirements for an online lottery

- SHA-3 Keccak and Quanta RNG pass all NIST statistical randomness tests [28]
- SHA-3 Keccak analysis [37][38] shows that it is a perfect candidate for PRNG

SHA-3 Keccak security

SHA-3 Keccak is a cryptographic hashing algorithm widely used on Ethereum blockchain. The SHA-3 calculation can be easily done one way and it is extremely difficult to reverse it. For a given capacity c, Keccak is claimed to stand any attack up to complexity $2^{c/2}$, SHA3-256 can stand any attack up to 2^{128} [15] To crack 128 bit key size by brute force it takes approximately 1.02 x 10^{18} years, more than the age of the universe [16]. Keccak can be also used as *sloth*, a slow-timed hash function described in [31].

SHA-3 Keccak collisions attacks [17][18][19]

- Keccak has 24 rounds
- Collisions were found in reduced, 4-round Keccak-256 within a few minutes
- Near-collisions were found in reduced, 5-round Keccak-256 within a few hours
- Attacks on the 9 round version of Keccak are already close to brute force complexities in terms of number of compression function calls

SHA-3 Keccak performance

- It is inherently difficult to parallelize Keccak on the GPU. The main reason is the very nature of the sponge construction and a very sequential process [26]
- Hardware implementations of Keccak ASIC and FPGA are able to process hashing up to 43986 Mbit/s [34]

SHA-3 Keccak as PRNG

SHA-3 can easily be translated into a PRNG in a nearly black-box way [37][38]. The minimum security requirement is that a Keccak hashing function has accumulated sufficient entropy material. The output bits from Keccak are indistinguishable from random and pass all NIST statistical tests for randomness [28]. RNG based on Keccak can use the previous random number as one of the seeds for a new random number. To provide sufficient source of entropy each cycle of generating a new random number has to be reseeded with user's random input or external oracles. One of the requirements of the lottery is that the generated bits can not be predicted, reseeding hashing function with external oracles provides sufficient level of

unpredictability of the generated outcome. Keccak function can generate uniformly distributed bits even if some parts of the seeds data are a low source of entropy due to imbalance of or correlations between bits. Keccak is also collision-resistant and can produce random bit output from any variable length input.

All of the above characteristics are making Keccak a perfect candidate for building PRNG, supplying sufficient source of entropy and producing pseudo-random bits as output.

RNG on the blockchain

Until today there is no blockchain with a built-in, secure RNG feature. There are several gaming blockchains under development such as CARDANO [12] or Peerplays [13] that have it in their plans. Ethereum also plans to have a built-in RNG feature. There are several possible ways to generate random number on the blockchain, most of the solutions are not secure enough for a lottery with a big jackpot prize. There are a few possible ways to solve this problem using commitment schemes.

Possible solutions

- Using block hash - only if there is no money involved or the money at stake is lower than miner reward e.g. as of July 10th, 2016 block mining rewards are 5 ETH (US$55) on Ethereum platform and 12.5 BTC (US$8,109) on Bitcoin platform. An adversary who possesses some reasonable hashing power will have an incentive to manipulate the RNG outcome if he gains profits by not submitting his mined block.
- Using timestamp - a naive approach, as calculation is done globally there is an acceptable time gap that miners can alter when they publish a block
- Using external oracles - problem with random number freshness, correctness and verification of the services, trust is on third party supplier [42]
- Using commitment schemes - the best known approach
- Using blockchain protocol level secure RNG - the most desirable solution for any gaming blockchain. Developers will be able to call random() function to receive uniformly distributed stream of random bits from the blockchain itself.

RANDAO, the two-round and three-round commitment scheme

Commitment scheme is an algorithm that lets the number of participants reach consensus on random number generation. It is one of the best random number generation methods on a decentralized network. RANDAO idea is based on a two-round commitment scheme, and the algorithm is secure if there is at least one honest player in the game and each player has a deposit equal to the jackpot prize. Detailed description of the two round RANDAO algorithm proposed by Youcai Qian can be found here [8].

RANDAO can be improved by adding one more synchronization round. This allows us to create a three round commitment scheme that can be combined with an additional, public source of randomness. The idea comes from existing works on proofs of knowledge and non-malleable commitment schemes [9][22].

The first round of RANDAO

Double hashed seeds are collected from participating players within a specific time period. Players will submit valid hashes of the hashes of the secrets: SHA3(SHA3(X_n)). Secrets will be defined as 32 bytes random strings, the first round will be completed only if a minimum number of players join RANDAO game. Secrets will be double hashed with SHA3 algorithm and passed to Ethereum RANDAO contract along with ETH used as a deposit. The deposit will be paid out only if the player reveals the secret during the third round. During the first phase participants have no knowledge about other players' secrets, they only know double hashed secrets. Players will reveal their secrets during the third phase, and the contract will verify if the revealed secrets match double hashes of the secrets submitted during the first round. It will be possible to submit only one double hashed secret per one player account.

The second round of RANDAO

Collecting hashes of the seeds SHA3(X_n). This phase allows separation and synchronization between the first and the third RANDAO round. At this phase an additional, public seed can be provided as an input to RNG. It will act as an additional layer of security in the RNG algorithm. It can be an external random oracle [20], Bitcoin block hash or Ethereum block hash generated by miners. Public seed will be the first known value that will be a part of RNG. In case the adversary controls all of the players of RANDAO game he will have to manipulate that public seed, either by controlling external Oracle or having enough influence of Ethereum or Bitcoin mining process.

This layer is required to provide synchronization between the first and the third rounds. In the second round only one participant is required to reveal hashed secret SHA3(X_n) to provide required synchronization. Once the second round is commenced, the game goes to the third round where all players must reveal all secrets.

Purpose of the second round is synchronization between rounds that provides the possibility to add one more source of randomness. Solidity contract can not perform any operations on its own, it has to be called from the outside. There are few different ways to achieve synchronization. We could create automatic script that could trigger RANDAO contract status changes. It could use Ethereum Alarm Clock contract [25] or the game could be simply synchronized by the players on their own, by adding one extra round to RANDAO.

The third round of RANDAO

Collecting valid secrets X_n. At this stage players can not change double hashed secrets committed in the first round. All players who participated in the first round and submitted SHA(SHA3(X_n)) have to submit their secrets X_n within a specified time limit. Random number will be calculated from concatenated secrets as a seed for hashing function

$$Random_{RANDAO} = SHA3(X_1,X_2,...X_n)$$

The contract will check if all of the secrets are equal to double hashed secrets submitted during first RANDAO phase.

RNG method requires that all of the users have to submit the secrets, it will not be possible to submit subsets of the secrets as it would provide possible way for manipulation of the outcome. Each player will receive the deposit back after successfully submitting his secret X_n. If one or more players fail to reveal their secrets their deposits will be confiscated and RNG will fail. Confiscated deposits will be added to the lottery jackpot. There will be a fixed prize for a successfully generated random number that corresponds to submitted deposits. If RNG fails,

the deposit value will grow with each RANDAO game until a random number is successfully generated. It is also possible to finalize RNG with a subset of revealed secrets.

To make RANDAO game fully secure and fair each of the players will have to pay a deposit equal to the jackpot prize, this is the biggest flaw of the algorithm.

Without deposits equal to jackpot prize last RANDAO player or players who collude may have an incentive to withhold their secrets to manipulate RNG. The last RANDAO player could calculate whether his submission could get a jackpot prize or not and decide to withhold his secret if doing so is more profitable. Colluded players could choose the most favourable subset of secrets to get the best random number. RANDAO can also fail if one or more secrets is not revealed, in that case the last player can push RNG to failure by not revealing his secret and getting another chance in the next RNG round.

There are several ideas of how to work around this problem.

Trusted dealer and issuers

The RNG algorithm uses a trusted dealer idea to collect all of the secrets from users, hash them and use trusted issuers to sign them. Signatures will be used to generate random number. The trusted dealer and issuers idea is described in a document introduced by Frank Stajano and Richard Clayton “Cyberdice: peer-to-peer gambling in the presence of cheaters” [10]. This idea eliminates the problem with high deposits in RANDAO game albeit introduces trusted parties which is not a perfect solution for the trustless Quanta lottery.

Quanta secret

Another way to mitigate the problem of malicious players in RANDAO game who will try denial of service attack by not revealing his secret is to make sure that a trusted party will be the last to reveal the secret.

Additional layer of security could be provided by a two-round commitment scheme carried out by only one player, trusted Quanta dealer. Each lottery round would be initialized by submitting hash of the Quanta secret SHA3(X_{Quanta}), lottery will be finished by Quanta X_{Quanta} secret reveal. This will guarantee that no external player will try to cheat by not revealing his secret during RANDAO game. Quanta dealer will have to submit deposit while starting each lottery round. The deposit value could be equal up to 10% of the predicted jackpot value and it could generate 5% profit deducted from the total jackpot value after revealing Quanta secret. The only way to finish the lottery round will be through Quanta secret reveal. This is still not a sufficient solution to lower deposit values and trustless lottery. Lottery Quanta dealer could have incentive to play RANDAO game with his players and decide to not reveal the secret in unfavourable situation and lead to RNG restart before revealing Quanta secret.

Board of Directors

Quanta secret could be distributed among few entities within the company, trusted agents [40], also called Board of Directors (BOD) [21]. BOD would provide a way to enable timed release crypto. BOD members would generate their numbers in advance and they would initialize the lottery by committing hashes of their secrets. The lottery will be drawn after all members of BOD would reveal their secrets. Single entity won’t be able to cheat without knowing other’s secrets. This idea will increase security but it will provide a new problem of securely storing and retrieving few secrets. It will be also impossible to prove that BOD doesn’t

collude on RNG. BOD is not sufficient as a solution for low deposits in RANDAO game for trustless RNG.

Weak encryption

Weak encryption mentioned by Frank Stajano and Richard Clayton in “Cyberdice: peer-to-peer gambling in the presence of cheaters” [10] could be a good solution to lower RANDAO deposit values significantly. SHA-3 Keccak could be used in example commitment scheme with non-interactive zero-knowledge proof where players reveal commitments, data and witness, keeping *blinding_factor* as a secret:

commitment = SHA3(blinding_factor || witness || data)

Witness could be used to confirm that *blinding_factor* has some characteristics e.g. 10 character long. This solution could prove that *blinding_factor* could be broken by brute force attack in some reasonable time frame with a normal PC. If any RANDAO player does not reveal his *blinding_factor* it would be possible to break it by brute force. RANDAO reveal round would last for 1 minute and time to break *blinding_factor* security would need to be long enough to withstand a brute force attack from any supercomputer for a minimum period of time. This task may be not possible to accomplish as brute force key search is trivially parallelizable, i.e. N computers make the computation run N times faster. It is also not possible to predict exact time of the calculation, actual running time can be larger or smaller depending on the key searching algorithm used.

SHA-3 hashing function does not support ZKP functionality and a witness would have to be implemented into that function. It could also be possible to use different hashing function with witness support.

The RANDAO three rounds commitment scheme as RNG

To control RNG the adversary would have to control two layers of security: he would need to control all players of RANDAO game and to have enough power to affect Ethereum block mining process. Adversary would need to have x percent hashpower of Ethereum network to have x percent chance of being able to gain one chance to manipulate the lottery outcome. He would also have to buy substantial number of lottery tickets to gain advantage by doing malicious mining. If RANDAO will synchronize external random Oracle [20] e.g. Oraclize [11] instead of blockhash, adversary would have to be in control of that mechanism.

RANDAO would be secure against DoS attack with high deposit values but that could bring a risk that no one will risk to play the game when deposits will become extremely high or there would be one adversary controlling all of the players in the game, that would bring additional risk that the adversary could collude with miners or external oracle service provider.

The final solution

Final solution will use time-lock puzzle introduced in paper [39], a *sloth* [31], the slow hashing function to simplify RANDAO commitment scheme to only one round. Non-interactive zero-knowledge proof needs further research and for now it will be substituted by an easier to implement challenge - response algorithm. More research will be done on square roots

extraction in finite fields, the calculation is computationally expensive and it is very easy to verify the results.

A *sloth* and *unicorn* [31] namely RANDAO++ on Ethereum [33][32]

RANDAO++ is a random number generation method for Ethereum proposed by Vitalik Buterin, the idea was first introduced by Arjen K. Lenstra and Benjamin Wesolowski in paper [31]. In RANDAO++ users submit values to the contract. These values are inputs to an off-chain computation. The computation is difficult enough that it can not be computed before the user submissions complete. This ensures that an attacker can not pick a number to manipulate the result in his favor. Following user submissions a verification round takes place where a result for the calculation can be posted and later challenged if the result is believed to be incorrect.

RANDAO++ key differences with RANDAO

- No commit and reveal phase, in RANDAO++ users submit values only one time
- No issue with users withholding reveals
- Requires off chain computation of the random number
- Requires at least 1 honest user to submit the correct result of the computation
- Requires a verification mechanism (challenge and response [36] or ZK-Snarks) to ensure that the result is correct

Randao++ contract functionality

Randao++ contract provides 3 major functions:

- collect values for input to the off chain random number computation
- collect the proposed computation result
- provide a result verification mechanism in case the result is wrong (Vitalik suggested a challenge / response or a ZK-Snark mechanism. Below we discuss only the challenge / response mechanism.)

Challenge and response verification mechanism

This mechanism has been implemented in the RanDAOPlus implementation on github. See the contracts at [32].

Appendix A shows full working examples of this with a number of possible scenarios. A short overview of the process is described here:

It starts after all user values have been submitted:

- A user completes the computation and submits a deposit and the result (submitProposal)
- Other users run the computation and compare the result
- If a user believes the proposed result is wrong they submit a challenge with a deposit and the value they think is correct (newChallenge)
- This starts a round of challenge and response between proposer and challenger:
    - The goal of this process is to find the “transition point”. The point in the chain of values in the calculation where the values went from being correct to incorrect.
    - This is done by binary search where on each iteration the proposer provides the value at the current point in the tree and the challenger says if this value is correct or not.

    - If the challenger says it is correct the current position is moved to the right - the direction of the final result value
      If the challenger says it is incorrect the current position is moved to the left - the direction of the first calculation (made from the input value which is known by everyone)
    - When the transition point is reached we are at a place where there is agreement that the i-th value is correct. The contract can calculate (i+1)-th value and compare it with a value the proposer says is at that position. This comparison determines if the challenge is successful or not.
- If the challenge succeeds the challenger's result replaces the proposed result

Gas Costs

The contract's challenge response mechanism requires multiple calls from proposer and challenger to respond() and challenge() respectively. For binary tree search this takes O(log(n)) steps. If there are 1,000,000 steps in the calculation then the verification will take 20 steps. After this the contract does a single SHA-3 call to compare at the transition point.

So the gas costs should not be high. Of course the total gas consumed must be much lower than the deposits for proposal and challenge.

The number of steps could be reduced by descending the tree by factors greater than 2. This needs experimentation to get gas cost estimates and achieve a good balance.

Off Chain Computation

Off chain functionality is required to calculate the random number (this is similar to a Proof of Work calculation). The time taken to complete the computation must be longer than the time allowed for users to submit values to the contract. For example if RANDAO++ allows submissions for 20 blocks (about 5 minutes) then the computation must take longer than 20 blocks to complete. Ideally twice that. This should hold true even for the largest miners and supercomputers.

Who computes this?

- Any Ethereum user who wants to make the calculation and get a reward for doing it.
- In case of the Lottery, Quanta would also compute this to ensure the value is correct.

Algorithm

User submitted values are combined with the latest ethereum block hash and sequentially hashed as follows:

r = SHA3^diff(SHA3^diff(SHA3^diff(x[0] + blockHash), x[1] + blockHash), ... x[n] + blockHash)

Where *r* is the random number and *diff* is the difficulty adjusted depending on the number of user inputs and estimates of time to complete the calculation. Difficulty is calibrated with timings from previous recent calculations.

Storage

To facilitate the challenge response mechanism, every step in the off-chain computation must be stored as any value may be called upon by the challenge. Required storage will be very large.

If for example we determine 1,000,000,000 hashes a second is the fastest computation speed (it's probably higher) than the computation would produce 29.8GB of data per/second (32 bytes x 1,000,000,000).

Some recent Keccak hardware implementations can provide s starting point for calculations [34]

Further Research on RANDAO++

- Fully automated RNG
    - Add registered RANDAO++ users and add reputation system
        - Replace human participants with external oracles during submission phase, human inputs could be use as optional source of RNG seeds
    - Replace human verifiers with automated bots (miners) as a puzzle solvers/verifiers
        - Users could register their custom virtual miners
- All bot miners could submit results and make confirmations to the first submitted correct value, fastest confirmation will get better chance to win 1st reward, confirmations will also receive rewards
- The paper "A random zoo: sloth, unicorn, and trx" [31] details an approach on a new slow-timed hash function based on modular square roots to build functions of fixed computation time, described in [35]
- Further research has to be done on Non Interactive Zero-knowledge proofs that are possibly a very good replacement for an interactive challenge and response algorithm
- Possible RNG improvements and research on fast RNG for gaming purposes
- Random number generation could be additionally seeded by commitment scheme and puzzle that could be solved by human miners [40]. Human players could solve a CAPTCHA-like puzzle, a puzzle that is easy for a human to solve, but difficult for a computer. SHA-3 timelock function could be running in parallel to guarantee time delay and it could be reseeded with outputs from human miners e.g. before 1 minute of *sloth* runtime will pass,. This could get a lot of attention if there would be challenging and interesting game based on human puzzles. It could be a part of TV show.

Potential Security Concerns

Submitted values all come from one source, or all sources are colluding

The calculation is a function of all submitted values, a block hash and the difficulty which determines how many times SHA-3 is called. So a miner who controls all input values and a block hash could control the final result.

How to mitigate this:

- encourage many users to participate and submit values by providing rewards for participation and making participation in Randao++ easy (provide an easy to use Dapp in Mist and on the Web [metamask.io])

- (optional and not necessary if the previous point is covered) inclusion of Ethereum block hash could be further strengthened by using a Bitcoin block hash via Bitcoin Relay as well. For example we could replace SHA-3^difficulty(x[0] + block[N].hash) with SHA-3^difficulty(x[0] + block[N].hash + bitcoinBlockHash[now])

Quanta PoW puzzle will be solved too fast

Somebody is able to complete the off chain computation before all numbers are submitted. For example cryptocurrency miners, NSA, anyone with access to Supercomputer power.

How to mitigate this:

- Ensure difficulty is high enough that even these actors could not compute it in time
- Use Quanta RNG bots running on the most powerful ASIC and FPGA devices [34]
- A short user submit round and long verification round will allow verification without powerful hardware.

Security improvements over RANDAO

- The problem of the last revealer withholding their secret does not occur
- No one can calculate potential random number results before submission round completes
- Simpler contract interaction for participants – only one submit

Alternative methods

- RANDAO with low deposits
- Secure PoW/PoS protocol that makes sure that the first miner who found suitable nounce number is registered, even if he decided to not publish block
- Secure way of using blockchain miners that would provide randomness from QRNG [30] hardware devices

6 Lottery system

Participants will be able to buy lottery tickets (make bids) by sending specific amount of Ether to seller addresses or directly to the Quanta smart contract. Most processes after this transaction will be automated.

According to the regulation of each country, local corporations can be established where it is possible to obtain a license. These local corporations will sell QNT tokens that will become a valuable digital asset and will be available for trade and exchange on several platforms.

Holders of a certain amount of QNT tokens will be able to set up seller (agent) smart contracts to sell lottery tickets. According to agents' sales performance, bonus QNT tokens will be issued to them by token manager smart contract. QNT token buyers will have to pass KYC (know your customer) process in order to be able to complete the purchase. After the system launch token manager smart contract will manage all issues related to QNT tokens.

The key element of the lottery service that is random number generation is automated and free from manipulation, but nevertheless, agents enabled by QNT tokens are very important in making the lottery spread and alter the industry. Agents activity will also be automated by introducing special smart contract addressed that will be managed by respective owners.

Likewise, in countries where obtaining a lottery license is not yet possible, governments may operate a domestic lottery service by utilizing the Quanta global lottery smart contract, supporting smart contracts and secondary non-blockchain systems. Quanta Co. Ltd. will provide all necessary support as a service in this scenario. In this case, seller smart contracts will connect to a national lottery smart contract instead of Quanta's global address. These local corporations or governments will conduct KYC procedures with buyers of QNT tokens.

Another smart contract acting as a global public pool will be controlled by QNT holders and used for charity and other public welfare causes after the proposals are approved by voting.

QNT token system

Local corporations will be set up to sell QNT tokens in each country. Governments or public organizations may also use the system to operate a domestic lottery.

The initial sale of QNT tokens will take place before the launch of Quanta. Holders of QNT tokens will regularly receive dividends in ETH or other currency specified by account owners on an automated basis. QNT will be implemented a tradable token on the Ethereum blockchain, compatible with some Ethereum wallets or smart contracts.

QNT issuance

- A portion of QNT tokens will be sold in several pre-sales or crowd-sales rounds
- QNT holders will share dividends from the revenue
- QNT holders can sell lottery tickets and get bonus tokens depending on sales results
- Final terms of QNT will be announced after launch

Lottery ticket sales

The price of 1 lottery ticket is planned to initially be around 0.08 ETH (equivalent to $0.96 as of June 29th, 2016). Each ticket is randomly assigned a unique number. Users are able to specify the amount of tickets that they want to purchase. Users pay all related gas costs. Alta Wallet and Alta Bridge can also be used to allow holders of any cryptocurrency to buy tickets with minimal effort.

Tickets are sold until the minimum sales target of 2,000,000 tickets or 160,000 ETH is reached (equivalent to $1,896,000 as of June 29th, 2016). Upon achieving the target lottery ticket sales will continue for one week after which the draw and distribution of prizes/dividends will take place.

Revenue distribution[1]

1. 70-79% will be paid out to lottery winners (prize pool)
2. Around[2] 1% - 10% commission will be earned by seller smart contracts
3. Around[2] 5% will be proportionately paid out to QNT token holders as direct dividends
4. Up to[2] 15% will be the operator’s margin (we will strive to further reduce operational costs to lower it to less than 10%, potentially increasing the prize pool

[1] Ethereum gas costs will be estimated shortly before the launch.

[2] Exact rates of commission or dividends and additional pool contribution are subject to change after further analysis.

ratio to 85%)

Draw

After ticket sales finish the actual draw utilizing RANDAO method for RNG will start.

Prize Tiers

In case of 2,000,000 tickets sold total revenue of the the lottery will reach US$1,896,000 with the prize pool of US$1,440,960.

Prize	Prize pool	Winners (example numbers)	Prize value in USD
1st prize	40%	0 or 1	$576,384
2nd prize	20%	15	$19,212
3rd prize	20%	50	$5,763
4th prize	5%	150	$480
5th prize	5%	450	$160
6th prize	10%	20%	$0.36

Table 1. Prize tiers

Next cycle

After distribution of prizes and dividends a new ticket selling cycle starts automatically.

Seller smart contract

After a user transfers ETH to the address, 8.5% is transferred to the randomly chosen consolation prize winners (20% of all tickets) and agent smart contract keeps 1-10% of every ticket's price depending on the amount of sales (editable smart contract logic). The remaining amount is transferred to the main (global or national) lottery smart contract with the data of remaining 80% of tickets.

Main lottery smart contract

Main lottery smart contract address will receive 81.5% to 90.5% of the initial lottery revenue. 5% of the initial revenue will be transferred to the public pool (managed by the government). 4% will be distributed between all QNT holders as dividends. 66.5% to 75.5% will be distributed between lottery prize winners. 5% will be transferred to the QNT holders lottery smart contract.

The smart contract will conduct the final draw using a random number generator and the data of tickets that have advanced to the final draw.

Public pool smart contract

The amount received (5% of the initial revenue) can be transferred to a government or other public organization's account according to specific rules. It can be managed by the lottery community in a decentralized way.

QNT holders lottery smart contract

5% of the initial revenue is received by this smart contract and distributed between the separate lottery pool winners. All QNT token holders will participate and receive tickets according to the proportion of QNT in possession. The concept of this special lottery will be revised.

Token manager smart contract

Lottery tokens will be held in their owners' cryptocurrency wallets (e.g. Alta Wallet). Both token manager smart contract and a wallet will update the number of tokens (token manager - after sales incentives; wallet - after the trade of tokens) and exchange the updated information with each other. Token manager smart contract will send the latest information to the wallet after issuing new tokens to sellers. The wallet will send the latest information to the token manager smart contract after token trade between users.

Tokens registration

Tokens will be sold at local corporations in each country. The token holders' data will be input and updated in the token manager smart contract (owner's ETH address, number of tokens, seller smart contract address).

Managing tokens

Lottery tokens will become an asset in Alta Wallet, a multi-asset HD wallet developed by CARDANO Labo. Both token manager smart contract and Alta Wallet will update the number of tokens (token manager - after sales incentives; Alta Wallet - after the trade of tokens) and exchange the updated information with each other. Token manager smart contract will send the latest information to Alta Wallet after issuing new tokens to sellers. Alta Wallet will send the latest information to the token manager smart contract after token trade between users.

Collecting seller smart contracts sales stats

Token manager will get sales stats and issue new QNT tokens to the sellers according to their performance (updating number of tokens according to smart contract logic and sales stats received from seller smart contract).

Reference for paying dividends

Token manager will provide QNT holders' data (ETH address, number of tokens held) to the main lottery smart contract.

Reference for token holders lottery

Token manager will provide QNT holders' data (ETH address, number of tokens held) to the token holders lottery smart contract.

References

1. Global Gambling Report 11th edition, 2015 http://www.gbgc.com/publications/global-gambling-report/
2. Ethereum White Paper https://github.com/ethereum/wiki/wiki/White-Paper
3. Union Gaming Research, “Global Gaming Technology”, 2015 http://www.uniongaming.com/wp-content/uploads/2012/07/GamingTech.pdf
4. Guy Zyskind, Oz Nathan and Alex ’Sandy’ Pentland. “Enigma: Decentralized Computation Platform with Guaranteed Privacy”, 2015 http://enigma.media.mit.edu/enigma_full.pdf
5. “Ethereum: Past, Present, & Future”, Blockchain Roadmap, Zurich Meetup, 2016 https://blockchainconsulting.expert/slides/ethereum-pastpresentfuture.pdf
6. Nick Szabo, “Smart Contracts” http://szabo.best.vwh.net/smart.contracts.html
7. Nick Szabo, “The Idea of Smart Contracts” http://szabo.best.vwh.net/smart_contracts_idea.html
8. https://github.com/randao/randao/blob/master/README.en.md
9. Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi and Ivan Visconti “Concurrent Non-Malleable Commitments (and More) in 3 Rounds” https://eprint.iacr.org/2016/566.pdf
10. Frank Stajano and Richard Clayton “Cyberdice: peer-to-peer gambling in the presence of cheaters” http://www.cl.cam.ac.uk/~fms27/papers/2008-StajanoCla-cyberdice.pdf
11. http://docs.oraclize.it/#introduction
12. http://cardano.io/index_e.html
13. Jonathan Baha’i and Michael P. Maloney “Peerplays A Provably Fair Blockchain-Based Gaming Platform” http://www.peerplays.com/Peerplays_Whitepaper.pdf
14. “Tor Project is working on a web-wide random number generator” http://boingboing.net/2016/05/26/tor-project-is-working-on-a-we.html
15. Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche "The Keccak sponge function family" http://keccak.noekeon.org/
16. http://www.eetimes.com/document.asp?doc_id=1279619
17. Itai Dinur, Orr Dunkelman and Adi Shamir "Practical Collisions in Round-Reduced Keccak" http://fse2012.inria.fr/SLIDES/75.pdf
18. Itai Dinur, Orr Dunkelman and Adi Shamir "New attacks on Keccak-224 and Keccak-256" http://eprint.iacr.org/2011/624.pdf
19. Donghoon Chang, Arnab Kumar, Pawell Morawiecki and Somitra Kumar Sanadhya "1st and 2nd Preimage Attacks on 7, 8 and 9 Rounds of Keccak-224,256,384,512" http://csrc.nist.gov/groups/ST/hash/sha-3/Aug2014/documents/chang_paper_sha3_2014_workshop.pdf
20. https://en.wikipedia.org/wiki/Random_oracle
21. “Decentralized Peer to Peer Game Assets Platform, Integration with Third Party Games using Smart Contract” https://dacplay.org/pdf/BitSharesPlayWhitePaper_EN.pdf
22. Marc Fischlin and Roger Fischlin "Efficient Non-Malleable Commitment Schemes" http://www.iacr.org/archive/crypto2000/18800414/18800414.pdf
23. Vitalik Buterin “Hard Problems of Cryptocurrency 2015” https://github.com/ethereum/wiki/wiki/HPOC_2015
24. Frank Stajano and Richard Clayton “Cyberdice: peer-to-peer gambling in the presence of cheaters” http://www.cl.cam.ac.uk/~fms27/papers/2008-StajanoCla-cyberdice.pdf

25. Ethereum Alarm Clock http://www.ethereum-alarm-clock.com/
26. Pierre-Louis Cayrel, Gerhard Hoffmann and Michael Schneider “GPU Implementation of the Keccak Hash Function Family” http://www.sersc.org/journals/IJSIA/vol5_no4_2011/11.pdf
27. BTC Relay http://btcrelay.org/
28. Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, San Vo and Lawrence E Bassham III “A statistical test suite for random and pseudorandom number generators for cryptographic applications” Revision 1a http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf
29. What is the Q in QRNG? http://www.idquantique.com/wordpress/wp-content/uploads/What-is-the-Q-in-QRNG.pdf
30. ID Quantique White Paper, Version 3.0 http://marketing.idquantique.com/acton/attachment/11868/f-004b/1/-/-/-/-/Quantum%20RNG%20White%20Paper.pdf
31. Arjen K. Lenstra and Benjamin Wesolowski “A random zoo: sloth, unicorn, and trx” 2015 https://eprint.iacr.org/2015/366.pdf
32. RanDAOPlus https://github.com/tjade273/RanDAOPlus
33. Vitalik Buterin “Could Ethereum do this better?” https://www.reddit.com/r/ethereum/comments/4mdkku/could_ethereum_do_this_better_tor_project_is/d3v6djb
34. “SHA-3 Hardware Implementations” http://ehash.iaik.tugraz.at/wiki/SHA-3_Hardware_Implementations
35. R. L. Rivest, A. Shamir and D. A. Wagner “Time-lock puzzles and timed-release crypto” https://cryptoexperts.github.io/million-dollar-curve/specifications/2016-02-01_trap-me-if-you-can.pdf
36. Christian Reitwiessner “From Smart Contracts to Courts with not so Smart Judges” https://blog.ethereum.org/2016/02/17/smart-contracts-courts-not-smart-judges/
37. Peter Gazi and Stefano Tessaro “Provably Robust Sponge-Based PRNGs and KDFs” https://eprint.iacr.org/2016/169.pdf
38. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche “Spongebased pseudo-random number generators” http://www.iacr.org/archive/ches2010/62250031/62250031.pdf
39. Ronald L. Rivest,Adi Shamirand David A. Wagner “Timelock puzzles and timedrelease Crypto”, 1996 https://people.csail.mit.edu/rivest/pubs/RSW96.pdf
40. Jeremiah Blocki and Hong-Sheng Zhou “Designing Proof of Human-work Puzzles for Cryptocurrency and Beyond” February 16, 2016 http://eprint.iacr.org/2016/145.pdf
41. Ahmed Kosba and Andrew Miller and Elaine Shi and Zikai Wen and Charalampos Papamanthou “Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts” July 4, 2016 http://eprint.iacr.org/2016/675.pdf
42. Cecile Pierrot and Benjamin Wesolowski “Malleability of the blockchain’s entropy” 2016 https://eprint.iacr.org/2016/370.pdf

Appendix A - Challenge Response working Example

For some background on this process see the section “The Power of Binary Search” in the referenced article “From Smart Contracts to Courts with not so Smart Judges”.

Calculation Sequence

The calculation sequence used in the scenarios below will use a small set of 8 values to demonstrate the logic. The actual implementation will have billions of 32 byte hashes produced by SHA-3 but the logic is the same.

Let’s say the 8 steps in the computation resulted in the following 8 value sequence using some function f and the initial input to the function is ‘10’:

99	22	55	44	33	88	66	11

So the 1st value ‘99’ is from f(10), the 2nd ‘22’ is from f(99) and so on … and the final result is let’s say 100 after an f(11).

Scenario 1 - Unsuccessful Challenge on Correct Proposal

In this scenario an honest user has performed and stored the 8 steps of the calculation above and has sent this to the contract as the proposed correct value.

A challenger comes along and tries to disprove it. We’ll see that so long as we can respond to the challenge we can prove we are correct. There are 3 variations described. In the first the challenger claims every value the proposer sends is false:

- Challenger calls the contract with the deposit amount required to challenge and the value he is claiming is the correct result ‘111’:
    - randaoPlus.newChallenge(111, {value: depositAmount})
- Contract calls on proposer to show the computed value at the half way point (determined by binary search logic (0 + 8) / 2 = 4)
- Proposer responds with the value at position 4
    - randaoPlus.respond(44)
- The challenger must respond now to say if he agrees the value at this position is correct or not. He responds false:
    - randaoPlus.challenge(false)
- If this value is false then the remainder of the tree after 4 must also be false. So we continue examining only the first half of the sequence. Using binary search logic we’ll start looking at half way in this remaining part of the list (0 + 4) / 2 = 2
- Proposer responds with the value at 2:
    - randaoPlus.respond(22)
- Challenger says it’s false
    - randaoPlus.challenge(false)
- Position now moves to half of the remainder of the sequence again which is position 1 ((0 + 2) / 2). But now the distance between the start and the end 0 - 1 is only 1. So the contract can do a compare. The user sends the value for position 1:
    - randaoPlus.respond(99)

- The contract knows the input is 10 and it knows the function f so it checks:
    - f(10) == 99

  It does so the challenge is unsuccessful. The challengers deposit goes to the proposer and the challenge is complete.

Now let's look at a variation of an unsuccessful challenge on a correct proposal. This time the challenger keeps sending true instead of false for each challenge and instead of arriving at the first value in the sequence we arrive at the end where the result of the full computation is. It goes like this:

1. randaoPlus.newChallenge(111, {value: depositAmount})
2. randaoPlus.respond(44) // position 4
3. randaoPlus.challenge(true)
4. randaoPlus.respond(88) // position 6
5. randaoPlus.challenge(true)
6. randaoPlus.respond(11) // position 8

We've reached the end so we can compare with the proposed result:

1. f(11) == 100 // contract proves proposer correct
2. f(11) != 111 // and the challenger incorrect

Finally what if we end up not at the end or the beginning because the challenger agrees with some responses but not others:

1. randaoPlus.newChallenge(111, {value: depositAmount})
2. randaoPlus.respond(44) // position 4
3. randaoPlus.challenge(true)
4. randaoPlus.respond(88) // position 6
5. randaoPlus.challenge(false)
6. randaoPlus.respond(33) // position 5

We've reached the end so we can compare with the response at position 4. Because both the challenger and proposer agreed that this value was correct:

1. f(44) == 33 // contract proves proposer correct

Scenario 2 - Successful Challenge on an Incorrect Proposal

In this scenario an incorrect proposal is sent then an honest user challenges this successfully identifying the "transition point".

Let's say the correct calculation is still the above but the proposer submits the following instead which has a final result of 222:

99	22	55	44	45	56	67	87

We can see it matches the correct sequence up to the fourth value and then after that it is incorrect.

The challenge would be like this:

1. randaoPlus.newChallenge(100, {value: depositAmount})
2. randaoPlus.respond(44) // position 4
3. randaoPlus.challenge(true) // challenger agrees 44 is at position 4
4. randaoPlus.respond(56) // position 6 now called on
5. randaoPlus.challenge(false) // doesn't match the correct sequence
6. randaoPlus.respond(45) // position 5

Now the contract can compare:

1. f(44) != 45

So the challenge is successful. Again because both parties agreed 44 was at 4 so the contract takes that as true. The transition point was successfully identified at position 5.

Scenario 3 - Successful Challenge with Incorrect Proposal on an already Incorrect Proposal

In this scenario an incorrect proposal is sent then an honest user challenges this successfully identifying the "transition point".

Now let's assume the same incorrect proposal from Scenario 2. A challenger comes along and challenges is it with another incorrect value 333. It might go like this:

1. randaoPlus.newChallenge(333, {value: depositAmount})
2. randaoPlus.respond(44) // position 4
3. randaoPlus.challenge(true) // challenger agrees 44 is at position 4
4. randaoPlus.respond(56) // position 6 now called on
5. randaoPlus.challenge(false) // challenger says false and it IS incorrect
6. randaoPlus.respond(45) // position 5 called

Now again the contract declares the challenge successful even though the new proposal is also incorrect:

1. f(44) != 45

At this point the contact needs an honest user to come along as was the case in Scenario 2 and disprove this new proposal. Once a user does that they will always be able to defend with Scenario 1.