Working Paper : Ethereum Blockhashes and Commitment mechanism as a on-chain provably fair, tamper-resistant, no limit stake, R.N.G. for P.o.W. and P.o.S.

Vincent ELI - a.k.a. WhySoS3rious

October 2, 2016

Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.
John Von Neumann, 1951.

God, grant me the serenity to accept the things I cannot change, Courage to change the things I can, And wisdom to know the difference.
Reinhold Niebuhr, 1934

Abstract

This draft paper describes a solution to the pseudo R.N.G. problem for public blockchains such as Ethereum.

I construct a source of pseudo R.N.G. that can provide a result at each block of the chain and which does not rely on any (oracle-based) external source of R.N.G.. The result of the pseudo R.N.G. relies on a future blockhash mixed with a commit/reveal mechanism of a seed from a trusted peer. This scheme makes any miner cheating moot since the impact of any change to the blockhash is not known at the time the miner computes the block on which the R.N.G. relies. This results holds no matter the stake on the R.N.G. result and is thus valid for Proof of Work and Proof of Stake blockchains.

Finally, we consider the fact that there remains one potential source of tampering : the seed provider could be colluding with a significant miner/validator. Yet the seed provider can be monitored via statistical analysis and also constrained to the use a specific mechanism of seed/secret production based on a BIP39 HD key pair generation (solving potential hash collision exploit). More importantly, we show how the collusion problem can be easily solved via a case-based approach tailored to the incentive structure to which this R.N.G. is applied.

1 Introduction :

Blockchains such as Bitcoin and Ethereum provide an innovative way to look at the computer-based pseudo Random Number Generation (R.N.G.) problem. The number of p2p nodes and the amount of computations solved every second (almost 2 billion of GigaHash/s for Bitcoin at the time of writing) generate a unique source of publicly available data that constitute a unique opportunity to build a highly secure and fast worldwide pseudo R.N.G..

I provide a solution via the use of blockhashes. In the first section I present blockhashes as a uniform, unpredictable but with significant limitations, source of RNG. The main problem is posed by potential tampering of the blockhash by a block's miner/validator. In the next section I propose the addition of a commit reveal mechanism to the blockhash R.N.G. as a solution solve this problem. I then discuss and provide a proof of concept of this first solution to a fully on-chain and secure source of pseudo R.N.G..

2 Blockhashes as a source of pseudo R.N.G.

In a blockchain (PoW or PoS), every block is validated and afterwards verified through a unique 64 chars hexadecimal number (bytes32). This hexadecimal can be used to provide a different number at each block once we convert it to an integer and then do some modulo operation to determine the odds.

Yet, we must assure ourselves of 3 important aspects before we can chose to use these blockhashes as a source of pseudo R.N.G. : R.N.G. soundness, using future blockhashes and potential tempering by the miners.

2.1 Blockhashes as a uniform and memoryless source of pseudo R.N.G.

First of all before we try to secure blockhashes as a source of pseudo R.N.G., we need to make sure they are trully a good source of pseudo randomness. In other words, we need to be able to generate an uniform distribution (ex : a six-sided dice with equiprobable sides) and it should not be predicable in any (time feasible) way.

This analysis has been already conducted in the literature for Bitcoin blockhashes and is provided in [BCG15]. Bitcoin blockhashes provide over the long run an uniform distribution and there are no autocorrelation no matter the lag parameter used.

I conducted the same analysis on the Ethereum blockchain using the first 140 000 blocks of the live chain. Informal results can be found in this blog article [Why16c]. They confirm the adequate generalization of results from the bitcoin's blockchain to ethereum's blockchain regarding blockhashes distribution and its memoryless nature.

2.2 The importance of using future hashes

A blockhash is generated at each block and we now know that this generation gives a uniform distribution. But once this blockhash is generated, its value becomes a public data available to anyone that can access the blockchain at that time. So past blockhashes, i.e. that are available at the time someone sends a transaction on the network, can't be an unpredictable source of R.N.G. since they are known by anyone.

Ethereum smart contract allow to build easily an on chain software that is able to compute at each block the right bet to any game that uses blockhashes of past blocks. In other words, if an application relies on a past blockhash as a source of R.N.G., this randomness is moot and can be solved easily by an ethereum smart contract. More details are provided in this blog post by M. Swende [Swe15].

2.3 The miner cheating and incentive problem

Blockhashes are the result of a miner work in a Proof of Work model or the object of the stake of a validator in a Proof of Stake model. If the designated miner is able to know the impact of the blockhash he is about to publish on a pseudo R.N.G. source, the miner could decide not to publish a block if the result is not favorable to him/her. Another (or possibly the same) miner will find a block which will yield a different blockhash and thus a different R.N.G. result. This places the miners in a unique position towards any application that uses blockhashes as a source of R.N.G. [1]. Indeed the miner basically faces a new decision problem every time he mines a block that is used for R.N.G. by some (or several applications). Yet any application can devise an incentive structure that makes the miner not able to cheat profitably. But this can't be fixed and needs to rely on variable settings because the stake on each blockhashes is relative to the whole ecosystem of application. A new application can change the whole incentives for the other applications. Computations and simulations were done in some of my previous work for Ethereum R.N.G. . This post [Why16b] analyzes the incentive problem of miner cheating in Ethereum PoW blockchain and show that under 3% of mining power, cheating is not profitable in any kind. Yet above this value, if the bet size is high enough, miner cheating can become profitable. For instance on a roulette game, a number bet strategy with cheating becomes profitable in the long run (compared to the tx fees and the house edge) if the miner has 3% mining power and the bet size is at least 23 ethers, hence a payout of 828 ether.

A proof of concept implementation of such on chain provably fair R.N.G. can be found in the following Ethereum smart contract [Why16a]

Solutions can also be devised to monitor the mining pool that could potentially cheat and such strategy would also impact negatively the long term revenue of their farmers. Yet this solution will never be entirely satisfactory since each application does not have the control over its R.N.G. domain and the environment could change drastically if a second application uses the exact same source of R.N.G. as the first one. Another important issue is the adaptation of the incentive system to a PoS environment where the reward mechanisms for block validators is entirely different. A switch to PoS of the Ethereum blockchain could force any of the discussed applications to reduce a lot its allowed stake size.

[1]note that some have explored the possibility of using several blockhashes to change the incentive problem of the miner and make the impact of each miner less important. Yet this technique is not successful because the last miner is able to change the result of the whole sequence of blockhashes used. And he can do so in the same extent as if we used only this blockhash as a source of R.N.G.

3 Solution : a fully secure, no limit stake, provably fair on chain R.N.G.

In this section I provide a solution to the miner cheating and incentive problem by introducing a commit/reveal mechanism of an encrypted seed by a trusted peer. I start by describing the mechanism of the solution, continue by discussion its characteristics and then implement it in a proof of concept ethereum smart contract code.

3.1 Overcoming the incentive problem : a 2 step commit/reveal mechanism to get rid of the miner cheating problem

The whole miner cheating problem comes from the fact that the miner are able to compute the impact of a blockhash on the pseudo R.N.G. result. It should not be too complex to solve this problem. We can for instance think of a commit and reveal mechanism from a trusted peer such that the miner will not be able to know the difference between a favorable and an unfavorable blockhash. This remove altogether the decision problem for the miner and solves the incentive problem.

3.1.1 The process and pseudo R.N.G

- Step 1 : A trusted peer commits an encrypted seed to a smart contract I propose to use sha3(bytes32 seed, bytes32 secret)
- Step 2 : A stake (or bet) is committed by a third party on the pseudo R.N.G. (sophisticated version : for extra security the third party could also provide an encrypted pair of seed/secret)
- Step 3 : The block which includes the step 2 transaction is validated/mined A blockhash is now available for this stake or bet. (note that this blockhash was not known at step 2.)
- Step 4 : Once the step 3 blockhash has been committed by the miners/validators, the trusted peer reveals the seed and the secret in clear form to the smart contract. The smart contract computes the pseudo R.N.G. result and solves the bet placed at step 2 by using :

  uint(sha3(blockhash, seed)) % (odds+1)

  (in the sophisticated version : the third party staker, should also reveal his seed/secret at this step, and the R.N.G. result will be based on $sha3(blockhash, seed_1, seed_2)$).

3.1.2 Characteristics

To the best of my knowledge this pseudo R.N.G. mechanism provides the first fully on chain provably fair R.N.G. that allows an unlimited stake size and does not rely on any third party oracle. It is also very fast since it is theoretically able to provide a different result at each block (scaling with block time). With this respect it provides a faster and more secure solution than the clever RANDAO solutions that requires at least 6 blocks. [Ran16] Also it is more gas effective and requires no additional stake of R.N.G. providers. This solution has the merit of not making the different applications of the blockchain interdependent since they can rely on different seeds. Finally, since the incentive problem is solved, **the proposed R.N.G. is fully transposable to a Proof of Stake framework.**

3.2 A proof of concept smart contract implementation

```
contract Wss_RNG
{
    bytes32 encrypted_seed;
    bytes32 revealed_seed;
    bytes32 revealed_secret;

    uint odds;

    uint bet_blocknumber;
    uint bet_input;
    bool bet_made;
    uint bet_result;
    string log;

    function Wss_RNG()
    {

       odds=100;
       //example seed
       //made by secret = "0x006ee0953c7Bb75CA6De27f58b5512390B1C4d6a"
       // seed="0x7cB57B5A97eAbe94205C07890BE4c1aD31E486A8"
       encrypted_seed="0x903e88616c7587ec063b5c79f4f5aad24de25f1e149a87edeeb964cbfbf70b38";
    }

    function bet(uint integer_0_to_100)
    {
        if (bet_made) throw;
        if (integer_0_to_100>=0 || integer_0_to_100 <= 100)
        {
            bet_input = integer_0_to_100;
            bet_blocknumber=block.number;
            bet_made = true;
        }
    }

    function reveal(bytes32 rev_seed_, bytes32 rev_secret_)
    {
        //verify that bet has been made
        if (!bet_made) throw;
        //verify that the block of the bet has been mined.
        if (block.number>bet_blocknumber)
        {
            //verify if the revealed seed and secret match with encrypted committed hash.
            if ( sha3(rev_seed_,rev_secret_)!=encrypted_seed) throw;
            revealed_seed=rev_seed_;
            revealed_secret=rev_secret_;
        }
        else
        {
            throw;
        }

        //The EVM only stored the last 256 blockhashes
        // After that it’s too late to reveal and an alternative resolution should be
            provided here
        if (block.number>bet_blocknumber+250) throw;

        //compute pseudo RNG
        bet_result=uint(sha3(revealed_seed,block.blockhash(bet_blocknumber)))%(odds+1);
        //solve bet
        if (bet_input == bet_result)
        {
            log="You won";
        }
        else
        {
            log="You lose";
        }

    }

}
```

4 The collusion problem and its resolution

4.1 Monitoring the seed provider

The only caveat of this R.N.G. is if the trusted peer becomes malicious and gathers enough computational power, the miner cheating problem would apply to this peer. This is hence a drastic improvement over the previous solutions to miner cheating based on incentives because now we have a single potential source of tempering with the results instead of several. Note that the seed provider can be anyone, and we could even allow for several seed providers or a mixture of seeds to avoid full information by any peer. But in a single trusted peer mechanism, the seed provider needs to be incentivized to remain an impartial provider of seeds and a few solutions can also help to guarantee this aspect. The first enhancement is provided by requiring the seed provider to follow a hierarchical deterministic pattern of seed/secrets generation based on a master mnemonic bip 39 seed. At each round of R.N.G. the seed provider has to use the right index to generate a new key pair. At the end of each period (say a month), the master key can be revealed and the seed generation verified. The second one resides again in statistical monitoring and reputation external incentives of the trusted peer.

4.2 Solving the problem altogether : A case-based approach tailored to the incentive structure

Finally a mechanism can be built where several peers are used as seed providers which can remove trust altogether. This mechanism should guarantee that every peer is incentivized to reveal its seed and that no peer is in a position to collude profitably with a miner.

There is no general solution to this problem since it really depends on the exact incentive structure where the R.N.G. is used for a stake.

Let me provide here a simple solution that applies to a betting game. The model is composed of a casino house that offers a simple european roulette game. The casino house bankroll is owned by a single actor which has a (financial and reputational) incentive to keep its game running over the long run. Any interruption of service or non resolution of a bet, would cause a significant harm to this casino. The model also contains a third party agent, the player, who is willing to commit a bet against the casino.

If both players commit an encrypted seed, as explained in the sophisticated version of my R.N.G. protocol, I claim that this process would be fully tamper proof and trustless. The collusion problem is now solved since even if the casino is working with a large miner/validator, it will not be able to tamper with the results since he is not able to know the impact of the blockhash in advance. Indeed the R.N.G. depends on $sha3(blockhash, seed_1, seed_2)$ and $seed_2$ is controlled by the player. Hence the players are guaranteed to be facing a fully provably fair R.N.G. and no trust is required.

Both the casino and the player are incentivized to reveal their seeds. The player has already committed his bet and would lose his wager if he does not reveal. The casino would lose his reputation and business if he does not reveal.

The only remaining source of tampering would be the case where the casino is also the player and a significant miner[2]. But then, I claim there is no problem. In this very unlikely case, casino would be stealing and cheating against itself, which is not a problem in this incentive structure.

5 Conclusion

I described the pseudo R.N.G.problem in a blockchain environment and provided what I think to be the first form of fully on chain R.N.G. that is tamper-proof trustless while allowing limit-less staking. The casino / player model described in the last section provides an innovative solution to full on chain R.N.G. and removes any problematic incentive from the process.

This paper is a work in progress and any comments or criticism are welcome.

Contact me on reddit : https://www.reddit.com/user/WhySoS3rious/

[2]Similar to this situation, is the one where the casino server have been hacked and the casino seed is known.

References

[BCG15] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. On bitcoin as a public randomness source. Cryptology ePrint Archive, Report 2015/1015, 2015. https://eprint.iacr.org/2015/1015.

[Ran16] Randao. Github, 2016. "https://github.com/randao/randao".

[Swe15] Martin Swende. Ethereum smart contract security : Breaking the house. Author Blog, 2015. "http://martin.swende.se/blog/Breaking_the_house.html".

[Why16a] WhySoS3rious. Ethereum rouleth proof of concept, solidity code. Github, 2016. "https://github.com/Bunjin/Rouleth/blob/master/rouleth.sol".

[Why16b] WhySoS3rious. A provably fair roulette : note on random number generation and miner cheating. Github, 2016. "https://github.com/Bunjin/Rouleth/blob/master/Provably_Fair_No_Cheating.md".

[Why16c] WhySoS3rious. Rouleth.com : A new paradigm for gambling : Decentralized, account-less and provably fair ! Author Blog, 2016. http://whysos3rious.com/index.php/2016/07/22/rouleth-com-a-new-paradigm-for-gambling-decentralized-account-less-and-provably-fair/.