

RANDAO-based RNG: Last Revealer Attacks in Ethereum 2.0 Randomness and a Potential Solution

Do Hai Son
dohaison1998@vnu.edu.vn
VNU Information Technology
Institute
Hanoi, Vietnam

Tran Thi Thuy Quynh
quynhttt@vnu.edu.vn
University of Engineering and
Technology
Hanoi, Vietnam

Le Quang Minh
quangminh@vnu.edu.vn
VNU Information Technology
Institute
Hanoi, Vietnam

ABSTRACT

Ethereum 2.0 is a major upgrade to improve its scalability, throughput, and security. In this version, RANDAO is the scheme to randomly select the users who propose, confirm blocks, and get rewards. However, a vulnerability, referred to as the ‘Last Revealer Attack’ (LRA), compromises the randomness of this scheme by introducing bias to the Random Number Generator (RNG) process. This vulnerability is first clarified again in this study. After that, we propose a Shamir’s Secret Sharing (SSS)-based RANDAO scheme to mitigate the LRA. Through our analysis, the proposed method can prevent the LRA under favorable network conditions.

CCS CONCEPTS

• Security and privacy → Privacy-preserving protocols;

KEYWORDS

Last revealer attack, Ethereum, Shamir’s Secret Sharing, Verifiable Delay Function

1 INTRODUCTION

Launching in 2016, Ethereum rapidly rose to rank 2nd in terms of crypto market capitalization [4] due to its superior features, e.g., smart contract, lower block confirmation time, and so forth. Hence, this network is not only a digital asset but also a decentralized platform for other impactful applications [14, 10]. However, an intensive number of applications brings the need to scale the Ethereum network and reduce the energy used for mining [8] through the old consensus mechanism, i.e., Proof-of-Work (PoW). Thus, in 2018, the Ethereum Foundation introduced the Ethereum 2.0 road map [9] to upgrade from PoW to Proof-of-Stake (PoS) using Casper consensus [3]. In 2020, the first phase on the road map, named ‘Beacon chain’, was officially merged into the Ethereum mainnet. After three years, the first phase is done with the highlight, which is a fully PoS-based Ethereum network.

The move from PoW to PoS means the mining process is replaced by stakers, who staked their money to become a validator. Any validator has the chance to propose a new block, a.k.a proposer, and get a bounty of ETH (Ethereum native token). In this work, we focus on the weakness of the random process to select the proposer in Ethereum. According to Eth2.0 specs, this proposer is randomly chosen by the RANDAO scheme [9] (Random decentralised autonomous organisation). Generally, RANDAO is based on CRS (Commit reveal scheme) [2]. In CRS, each member commits their value to a group, and the final number is a combination of these values. However, in [6], the author indicated a vulnerability of the RANDAO scheme named LRA. This weakness allows attackers

to bias the RNG output. In [2], Vitalik (Ethereum Founder) pointed out that if an attacker had 36% of the total staked money, he would gain control of the Ethereum network. This means the attacker manipulates the proposers always to be his validators. That leads to the consequence, i.e., he has the ability to validate any invalid blocks. The LRA is similar to the 51% attack in Ethereum 1.0, but a massive amount of money replaces the overwhelming computational power.

To deal with this vulnerability, the author in [1] proposed the VDF (Verifiable Delay Function) algorithm. This method prevents any validator from finding the final random number before the reveal phase. Following that, other studies [11, 15, 7] proposed their versions of VDF, i.e., simple VDF, efficient VDF, and continuous VDF, respectively. The Ethereum Foundation confirmed the VDF version, named minimal VDF [5], which will be used in the Ethereum mainnet after phase 2 on the road map. However, the drawback of VDF is that this algorithm requires a specific hardware named ‘Rig’. Although trustful organizations centrally control this device, it has lost the decentralized properties of blockchain technology. In the worst case, none of ‘Rig’ is available, the VDF is turned down, and then Ethereum will switch back to using the RANDAO scheme with the LRA weakness. In this work, we propose to use the SSS [13] algorithm for the RANDAO process on Ethereum or other Ethereum variants. SSS is an old but effective algorithm for securely sharing a secret. Thus, SSS-based RANDAO can provide a controllable security level to select proposers randomly. In Table 1, we explain several Ethereum terms used in the following sections.

Our main contribution in this paper is to propose an SSS-based RANDAO scheme for Ethereum 2.0. In this study, our proposal is a preliminary idea with some security analysis to deal with the LRA attack on Ethereum.

The remainder of the paper is organized as follows. In Section 2, the brief about LRA attack in RANDAO-based RNG is presented. In Section 3, we propose to use Shamir’s Secret Sharing algorithm for randomly chosen proposers on Ethereum. Finally, Section 4 concludes the paper.

Table 1: Ethereum terms used in the paper

Term	Description
Validator	Instead of ‘miner’ in PoW
Proposer	The chosen validator to propose blocks
Committee	A set of validators to validate the proposed blocks
Effective Balance	The current balance of a validator
Slot	Fixed at 12 seconds and equal to a block
Epoch	Consists of 32 slots

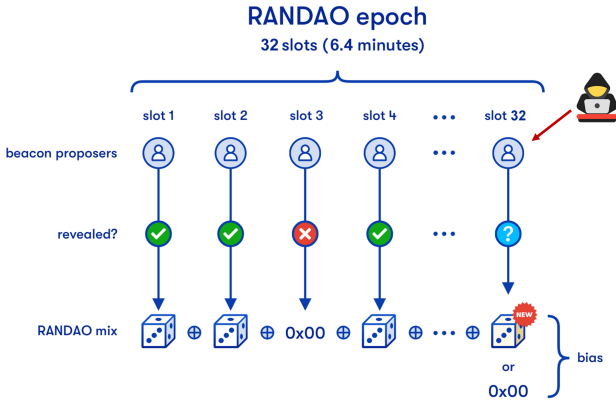


Figure 1: Last revealer attacks on Ethereum.

2 LAST REVEALER ATTACKS IN RANDAO-BASED RNG

First, we briefly present the RANDAO-based RNG algorithm of Ethereum with an illustration in Fig. 1. On Ethereum, the set of proposers and committees in $(j + 2)$ -th epoch is selected from the RANDAO-based results at (j) -th epoch. At the first slot in j -th epoch, the proposer signs its BLS private_key on the array of current epoch number (j) and fixed DOMAIN_RANDAO ('0x02000000'). The output of this process is a digital signature (r_1) in size of 256 bits that any validator can verify by the public_key of the first proposer. This sequence of bits is included in the first slot in j -th Epoch. Similarly, the second proposer computes its 256-bits (r_2). However, it does not immediately add this sequence to the block but computes

$$r_2 = r_1 \oplus r_2, \quad (1)$$

where \oplus is the 'XOR' operator. The second proposer puts the computed r_2 into its block. In the last slot, the sequence is as follows:

$$r_{32} = \oplus_i r_i, \quad 1 \leq i \leq 31. \quad (2)$$

In the absence of a proposer case, the sequence is preserved as xor with a series of zeros. The final sequence r_{32} together with DOMAIN_BEACON_PROPOSER ('0x00000000') and current epoch number (j) makes an array called 'seed32'. This array is passed through a hash function and returns an array of 32 elements. The final array is one of the two inputs to randomly select the set of proposers for $(j+2)$ -th epoch. The latter input is a set of all activated validators in the network at the first slot of j -th epoch. The inputs are put into a deterministic function to select the set of validators. Note that there is no randomness in this final validator selection function. Hence, the randomness only comes from the 'seed32' array, which is contributed by a set of proposers.

In the LRA, suppose the attacker is the proposer holder in the 32-th slot. At this stage, he already has information on the set of activated proposers from the first slot and the mixed sequence (r_{31}) from the 31 slots previously. Thus, he has two options, i.e., commit his digital signature as usual or pretend to be absent to keep the r_{31} . This is a bias in the RNG process. Note that only the last proposer can have this bias because the before validators are infeasible to predict the after's digital signature. That leads to the name 'Last

revealer attack'. Furthermore, if the attacker has more than one proposer in slots at the tail of an epoch, e.g., h proposers, he will have a total of 2^h options to decide whether to propose or not propose a block. Consequently, these 2^h choices may introduce biases in the selection of proposers and committees for the $(j+2)$ -th epoch.

3 SHAMIR'S SECRET SHARING-BASED RANDAO

In this section, we present our proposal for SSS-based RNG to prevent the LRA attack. After that, the security analysis and limitations of the proposed scheme are provided.

3.1 Proposal

Our proposal is to apply the SSS algorithm to replace the CRS in RANDAO, as mentioned in the previous section. SSS [13] threshold scheme based on polynomial interpolation over finite fields. There are three definitions of SSS (n, m) , as follows:

- Secret (s_k): is the secret information (e.g., number, text, etc.) that needs to be shared securely.
- Share: is a part of the secret. For example, from an s_k , we have m shares, i.e., $s_{k1}, s_{k2}, \dots, s_{kn}, \dots, s_{km}$.
- Threshold (n): is the minimum number of shares required to recover the original secret (s_k).

As shown in Fig. 2, at the first slot, the proposer still computes its digital signature (r_1), then divides r_1 into 31 shares ($m = 31$) corresponding to the rest of proposers in this epoch, i.e., $s_{11}, s_{12}, \dots, s_{131}$, respectively. These shares are encrypted using asymmetric key schemes, e.g., RSA [12], so only the validator identified by its public key can reveal the corresponding share. The first proposer then packs these shares into its block and broadcasts the block to the network. The rest of the proposers also do the same in their slots. At the last proposer, what it has is 31 shares (maximum), i.e., $s_{131}, s_{231}, \dots, s_{3131}$. Using these shares, the last validator cannot recover any digital signature (r_i) from the rest of the proposers. This implies that it is not feasible for this validator to manipulate the RNG process of RANDAO. After 32 slots (an epoch), there is the reveal phase, when validators broadcast their decrypted shares. Each secret from i -th proposer (s_i) requires at least n shares (n proposers broadcast their $s_{ki}, 1 \leq i \leq n$, decrypted shares) to recover. The unrecoverable r_i , due to not having enough shares, can be treated as absent proposers, as RANDAO did. RANDAO scheme selects proposers and committees for the $(j+2)$ -th epoch from recovered digital signatures and the set of activated validators. The reveal and mixing phase in SSS-based RANDAO occurs after an epoch has been completed and may lead to a delay in the timeline of Ethereum. Inspired by minimal VDF [5], the reveal and mixing phase can be conducted in parallel as a pipeline process.

3.2 Security analysis

Assume that t, h is the number of proposers who joined SSS-based RANDAO and the number of dishonest proposers in an epoch, respectively. In the first case, as follows:

$$t \geq n \quad \text{and} \quad h < n, \quad (3)$$

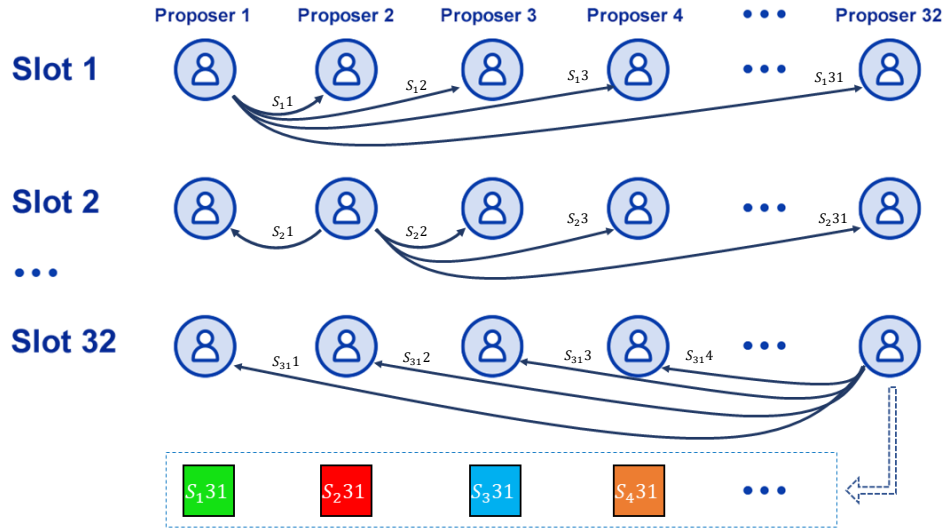


Figure 2: SSS-based RANDAO.

The proposed method can recover all secrets, and the attacker cannot predict the digital signatures of other honest proposers. Thus, the LRA is prevented. In the second case, as follows:

$$t < n, \tag{4}$$

This is a bad case when we do not have enough shares to recover any secret. This leads to no 'seed32' array, the original RANDAO mechanism will break, and a backup solution is needed for this case. In the third case, when attackers have a few proposers in an epoch, as follows:

$$t \geq n \text{ and } h \geq n, \tag{5}$$

Attackers have enough shares to recover secrets and compute the 'seed32' array before starting the reveal phase. This leads to an attack similar to LRA, but the attackers' proposers do not need to be at the tail of the epoch.

The controllable property of the proposed scheme is that we can adapt the threshold n to guarantee the randomness of RANDAO-based RNG. However, it is a trade-off because a higher n means more proposers must be present in the reveal phase.

4 CONCLUSION

In this work, we proposed to use an SSS-based RANDAO scheme instead of CRS to mitigate the LRA. We first represented the RANDAO-based RNG of Ethereum and pointed out its weakness. The SSS algorithm is applied to RANDAO and shows its advantages and limitations. Our study is still in its preliminary stages and requires further analysis to estimate the amount of stake attackers need to control the mechanism we have proposed.

REFERENCES

[1] Dan Boneh et al. "Verifiable delay functions". In: *Annual international cryptology conference*. Springer, 2018, pp. 757–788.

[2] Vitalik Buterin. *RANDAO beacon exploitability analysis, round 2*. URL: <https://ethresear.ch/t/randao-beacon-exploitability-analysis-round-2/1980> (visited on 10/20/2023).

[3] Vitalik Buterin et al. "Combining GHOST and Casper". In: (2020), pp. 1–38. arXiv: 2003.03052. URL: <http://arxiv.org/abs/2003.03052>.

[4] CoinMarketCap. *Cryptocurrency Prices, Charts And Market Capitalizations*. URL: <https://coinmarketcap.com> (visited on 10/20/2023).

[5] Justin Drake. *Minimal VDF randomness beacon*. URL: <https://ethresear.ch/t/minimal-vdf-randomness-beacon/3566> (visited on 10/07/2023).

[6] Paul Dworzanski. *A Note On Committee Random Number Generation, Commit-Reveal, And Last-Revealer Attacks*. URL: <https://ethresear.ch/t/limiting-last-revealer-attacks-in-beacon-chain-randomness/3705> (visited on 10/10/2023).

[7] Naomi Ephraim et al. "Continuous verifiable delay functions". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2020, pp. 125–154.

[8] Peter Fairley. *Ethereum Plans to Cut Its Absurd Energy Consumption by 99 Percent*. 2019. URL: <https://spectrum.ieee.org/computing/networks/ethereum-plans-to-cut-its-absurd-energy-consumption-by-99-percent> (visited on 10/19/2023).

[9] Ethereum Foundation. *Ethereum 2.0 Specifications*. URL: <https://github.com/ethereum/eth2.0-specs> (visited on 10/07/2023).

[10] Shi-Yi Lin et al. "A survey of application research based on blockchain smart contract". In: *Wireless Networks* 28.2 (2022), pp. 635–690.

[11] Krzysztof Pietrzak. "Simple verifiable delay functions". In: *10th innovations in theoretical computer science conference (ITCS 2019)*. California, USA, 2019.

[12] R. L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Commun. ACM* 21.2 (1978), pp. 120–126.

- [13] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (1979), pp. 612–613.
- [14] Do Hai Son et al. "An Effective Framework of Private Ethereum Blockchain Networks for Smart Grid". In: *2021 International Conference on Advanced Technologies for Communications (ATC)*. 2021, pp. 312–317.
- [15] Benjamin Wesolowski. "Efficient verifiable delay functions". In: *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Darmstadt, Germany, 2019, pp. 379–407.