introduction)

Was this helpful? Edit on GitHub Export as PDF

Secret-VRF - on-chain Randomness

An in-depth explanation of Secret VRF, a secure and verifiable random number generator Want to develop with decentralized on-chain randomness? Visit the development examples here!

Introduction

The ability to generate fair and verifiable random numbers on blockchain without compromising security or usability is critical for many decentralized applications. However, generating true randomness is difficult because blockchains rely on consensus mechanisms, such as Proof of Work or Proof of Stake, to reach agreement on the state of the network, and these consensus mechanisms are deterministic and therefore do not provide a source of true randomness. There have been attempts to solve this problem by various methods, such as using random data from external sources (the hash of a previous block, etc) or by using off-chain solutions, such as Chainlink’s VRF oracle, but these solutions are not optimal as they are either deterministic, rely on trusted parties, and/or require additional fees for gas and infrastructure. What is needed is a verifiable random number generator that exists on-chain, and this is now possible across blockchain ecosystems through Secret Network Random Number Generator (RNG).

Why Randomness Matters

The ability to generate true randomness is a fundamental requirement for blockchains. Verifiable randomness guarantees that operations such as the minting of NFTs, on-chain games, and DAOs are equitable and secure. In the case of NFT minting, randomness allows for features such as unordered minting, trait randomization, and identity numbering, all of which are critical for verifying the authenticity and security of NFT collections. In Web3 gaming, components such as gambling, damage calculation, loot boxes, boss drops, etc, rely on randomness to create trust amongst players and ensure that no player has an unfair advantage. And in the case of DAO tooling, randomness is required for features such as wallet initialization, task assigning, unordered voting/liquidations, order book ordering, etc. By using Secret Network RNG in these and other applications, the Interchain can help to ensure the security, transparency, and fairness of blockchain operations, which in turn promotes the growth and adoption of the Cosmos.

How It Works

Secret Network is built on top of Tendermint, a peer-to-peer networking protocol that provides Byzantine-Fault-Tolerance (BFT) and Proof-of-Stake (PoS) consensus. This consensus ensures that all nodes maintain the same blockchain and that all nodes have the ability to propose new blocks. From a high-level standpoint, a new consensus round is:

1. Initiated with a Proposer Node
2. The Proposer selects transactions in the mempool to be included in a new block
3. This proposed block is broadcasted to all nodes (Pre-vote phase) and nodes verify that the block is valid, simultaneously verifying that the Proposer is also valid and sign the pre-vote message
4. If >2/3 of nodes pre-vote for this block, the block is deemed valid
5. Once a block is committed, there is no way to revert it and there is instant finality

The process of block proposal and execution involves a crucial intermediary step known as the "prevote phase." Secret Network RNG creates verifiable on-chain randomness by generating a random number that is attached to each Proposer’s block proposal, and this random number can then be used for on-chain randomness for the current block and exhibits the following properties:

1. The number is generated with sufficient entropy
2. The block proposer cannot predict or influence the entropy generation
3. Outside actors cannot predict random numbers before they are used to compute network transactions

Secret Network achieves this by implementing the following privacy network consensus:

Secret Network Consensus Schema

Secret Network is unique from other blockchains because it utilizes Software Guard Extensions (SGX) to provide a Trusted Execution Environment (TEE) by creating a secure area in memory, called an "enclave," where sensitive data and code can be processed in a way that is isolated from the rest of the blockchain. This means that even if the operating system, other applications, or an attacker were to somehow compromise the security of the blockchain, the data and code within the SGX enclave remains protected.

Secret Network's SGX allows for private computation on-chain and uniquely positions Secret Network to be the privacy provider of the interchain. In order to provide randomness as an interchain service, Secret Network's SGX generates a random number on-chain during the block proposal stage of Tendermint. Block proposers generate a random number inside of the Trusted Execution Environment using RDRAND/RDSEED instruction.

RDRAND and RDSEED are Intel Secure Key technology instructions used in computer processors to generate random numbers

The RDRAND instruction is used to generate random numbers, while the RDSEED instruction is used to generate seed values for cryptographic purposes. Externally influencing RDRAND that is executed inside an SGX enclave is very difficult to do, with no known practical or theoretical attacks that can reduce the resulting entropy of the instructions.

In addition to generating random numbers using SGX, Secret Network adds an additional layer of security to the output of the random number generation by pairing it with an Initial Randomness Seed (IRS). This seed originates from a shared consensus seed among all network nodes, and the random number and IRS are combined using an algorithm called HKDF(rfc5869) to certify that the final number is truly random and cannot be predicted by anyone, even if they were to influence the original random number generation.

Secret Network also includes an additional key that is available to all nodes on the network within the TEE called the Random Encryption Key (REK). The REK is used to encrypt the generated random number using symmetrical encryption, and the encrypted blob (Erand) is attached as a new field to the block proposal. The REK is derived from the consensus seed, so all network nodes can derive it without the need for key-sharing protocols.

In order to make sure the random number generated by the block proposer is secure, a majority of network validators must signal their approval of the proposed block before the encrypted number can be decrypted. Tendermint's pre-vote mechanism allows the blockchain to "signal" that the proposed block has been authorized by a majority of network validators, and guarantees attempted execution of the block. Therefore, the only way to decrypt the Initial Randomness Seed is for nodes to submit a 66% majority of pre commits/commits to the TEE. The TEE will then verify this majority and decrypt the seed, returning it to the caller in its decrypted state, which can be then used for network randomness.

To prevent errors or malicious behavior, the encrypted number will be accompanied by a proof, which is defined so that an SGX enclave can verify that the proposed encrypted random number matches the block height and active set of the current Tendermint block. The proof is created using a one-way-function that takes the encrypted random value, the block height, active validator set, and the Random Encryption Key (known only to network nodes). With the proof key, SGX enclaves can verify that the proof was made within the enclave.

Combatting Potential Attack Vectors

Secret Network's TEE is continuously updated in response to changes to the active set of network validators. In order to avoid the issue of verifying changes to network validators, the active network validators influence the result of the derived random value during the decryption and verification process of the random number. Thus, changes to the active validator set produce a different random number, thereby rendering attacks using modified active validator sets ineffective. In addition, attempts to fork the chain and decrypt the number using a different active validator set are also rendered ineffective.

If a harmful actor were to attempt to fork the chain to predict the next number, the attack is rendered ineffective because the next number is generated randomly by the block proposer's enclave and cannot be predicted. Furthermore, a validator cannot execute blocks without the ability to decrypt the random number, which requires submitting prevotes to do so. If the malicious validator attempts to reuse a previous random number, the validator must submit a proof that links the encrypted random number with the block height and validator set, which it cannot.

To avoid being able to replay previous signatures with a new encrypted value, the encrypted number must be part of the transactions signed by the validators when creating their approval votes. To validate the prevotes, the TEE needs to receive the signatures and the original transactions. As an optimization, validation of the prevotes by Tendermint can be offloaded to the TEE to avoid verifying the same data twice.

Given that the voting power of validators fluctuates constantly and the computation of the active set's hash can be demanding, Secret Network achieves optimization by setting a minimum sensitivity to the changes in voting power. However, this can lead to disparities between the pre-vote weight and the weight calculated by the TEE, so the validation threshold must be lowered for the TEE. For instance, on Secret Network, if a 0.1% sensitivity to voting power changes is established, a prevote threshold of 61% would be necessary to consider the worst-case scenario where 50 validators, who together make up 66% of voting power, submitted prevotes with a 0.1%-ε higher voting power than what the TEE is aware of.

Finally, if a malicious actor were to somehow break Secret Network's SGX and steal the REK it would be inconsequential because random numbers are generated independently for each block, so knowledge of a single random value is of no use without the ability to influence block creation. Even in a scenario where the malicious actor was a validator and could submit random numbers for blocks they propose, it would be ineffective because they do not have access to the consensus seed as stated above.

Conclusion

In conclusion, the Secret Network Random Number Generator (RNG) is a significant breakthrough in the generation of verifiable and fair random numbers on-chain. By leveraging the power of SGX and Tendermint consensus, Secret Network has created a first-in-class on-chain solution that generates random numbers with sufficient entropy, and these numbers are then encrypted, paired with an Initial Randomness Seed, and attached to block proposals where they are decrypted in order to generate true on-chain randomness. Coupled with cross-chain interoperable smart contracts over IBC, Secret Network enables thousands of developers and projects across the Cosmos access to state-of-the-art on-chain RNG.

For documentation on using Secret VRF in a native Secret contractclick here .

Maths

~~~

Copy IRS=HKDF(Consensus Seed,0xdeadbeef) Rand=HKDF(Ordrand, IRS) REK=HKDF(Consensus Seed,<0xffffffff>) Erand=AES-SIVREK(rand) Proof-Key=HKDF(Consensus Seed,<0xaaaaaaaa>) Proof=HKDF(Proof Key, Erand,Active-Set, Block Height)

Seed used by chain=HKDF (DecryptREK(Erand), SHA256active_set)

Where: SHA256active_set=SHA256(PKval1||VP1||PKval2||VP2 ….)

``` Previous Trusted and untrusted data Next Privacy design Last updated10 months ago
~~~