Secret VRF for IBC with IBC-Hooks

Verifiable on-chain random number generator for the entire Cosmos.

Secret VRF (Verifiable Random Function)

This VRF tutorial uses IBC Hooks, as introduced in IBC v3.4.0 . If your chain is on a previous IBC version, see the SecretVRF tutorial using proxy contracts here . Secret VRF is a provably fair and verifiable on-chain random number generator (RNG) that enables smart contracts to access random values without compromising security or usability. Coupled with cross-chain interoperable smart contracts using IBC hooks , Secret Network enables developers and projects across the Cosmos access to state-of-the-art on-chain RNG.

Use Secret VRF to build reliable smart contracts for any application that relies on unpredictable outcomes:

- NFT Minting
- : Utilize randomness for features like unordered minting, trait randomization, and identity numbering, enhancing the authenticity and security of NFT collections.
- Web3 Gaming
- : Apply randomness in gambling, damage calculation, loot boxes, and boss drops to build trust among players by ensuring fairness and preventing any player from having an unfair advantage.
- DAO Operations
- : Employ randomness for wallet initialization, task assigning, unordered voting/liquidations, and order book ordering, facilitating equitable and secure decentralized governance and operations.
-

Learn more about how SecretVRF works in-depth here .

Getting Started

To use SecretVRF on any IBC-enabled chain with IBC hooks, all that is required is:

1. An IBC transfer channel between Secret Network and the consumer chain that receives randomness
2. .
3. Uploading the RNG Consumer Contract to your chain and directing it to your chain's transfer channel
4.

You can look up existing transfer channels between Secret Network on a block explorer such as Ping here. Existing IBC connections with Secret Network

Requesting Randomness

Git clone the IBC hooks randomness repository:

```

Copy gitclonehttps://github.com/writersblockchain/ibchooks-secretVRF.git

```

Update the SECRET_TRANSFER_CHANNEL_ID and the CHAIN_TRANSFER_CHANNEL_ID to the channel-id for your IBC-enabled chain:

```

Copy //Juno constSECRET_TRANSFER_CHANNEL_ID:&str="channel-8"; constCHAIN_TRANSFER_CHANNEL_ID:&str="channel-48";

```

For this example, we request randomness on Juno,but you can request randomness on any IBC-compatible chain that has a transfer channel established with Secret Network. Once you have updated the transfer channels, compile the contract:

```

Copy make build-mainnet-reproducible

```

Upload the compiled contract:

```

Copy junod tx wasm store consumer-side/artifacts/secret_ibc_rng_consumer_side_proxy.wasm --from--chain-id juno-1 --node https://rpc.juno.chaintools.tech:443 --gas 1200000 --gas-prices 0.075ujuno

```

Upon successful upload , acode_id is returned:

```

Copy {"key":"code_id","value":"4210"}]}]}]"

```

Instantiate the contract with the returnedcode_id :

```

Copy junod tx wasm instantiate 4210 '{}' --label RNG-IBC-JUNO --no-admin --from--chain-id juno-1 --node https://rpc.juno.chaintools.tech:443 --gas 200000 --gas-prices 0.075ujuno

```

Upon succesful instantiation , acontract_address is returned:

```

Copy [{"key":"_contract_address","value":"juno1srwcjsaslt9ewujg6wcpcwv08lsrsn6rx6ja5mqx4qngqjh8cugqt73c8m"},

```

Now that you've instantiated your randomness contract, all that's left is to request randomness! Simply executerequest_random:

```

Copy junod tx wasm execute "juno1srwcjsaslt9ewujg6wcpcwv08lsrsn6rx6ja5mqx4qngqjh8cugqt73c8m" '{"request_random":{"job_id":"1"}}' --from--chain-id juno-1 --node https://rpc.juno.chaintools.tech:443 --gas 200000 --gas-prices 0.075ujuno --amount 1ujuno

```

A transaction hash will be returned:

```

Copy txhash:92D5BDA1344529B58EAD5A0068A807632F46BCCCF05CF10E67211F9CBBD2A74B

```

You can update thejob_id string to any string of your choosing Navigate to the recently received transactions for your contract:

And then view the magic of cross-chain randomness with IBC hooks 😍:

```
```

Copy {"amount":"1","denom":"transfer/channel-8/ujuno","memo":"{\"wasm\": {\"contract\": \"juno1srwcjsaslt9ewujg6wcpcwv08lsrsn6rx6ja5mqx4qngqjh8cugqt73c8m\", \"msg\": {\"receive_random\": {\"job_id\": \"1\", \"randomness\": \"bjFHP7rrLwP4f6fpGpeTt5+N1zPiTO+y7da7Rl9kHzk=\", \"signature\": \"y+Kwu0T2gwRwDZGCdDPzGrm6hE2S2UZF1e1jm47pv85pdRdP7HdOfl6T+VKfAE4hPxSWJ5LBcTSNZ+b0KTe0xQ==\"}}}}","receiver":"juno1srwcjsaslt9ewujg6wcpcwv08lsrsn6rx6ja5mqx4qngqjh8cug

```

Congrats! You've just sent a verifiable on-chain randombyte with SecretVRF .

Conclusion

Secret VRF revolutionizes blockchain applications by providing a secure and verifiable source of randomness, critical for fairness in NFT minting, gaming, and DAO operations. Its seamless integration with IBC hooks enables cross-chain interoperability, allowing developers across the Cosmos ecosystem to build more reliable and elegant smart contracts.

If you have any questions, join ourdiscord and a Secret developer will assist you ASAP.

Last updated2 months ago On this page *Secret VRF (Verifiable Random Function) * Getting Started * Requesting Randomness * Conclusion

Was this helpful? Edit on GitHub Export as PDF
```