VRF Developer Tutorial

Learn how to use SecretPath on EVM to access on-chain verifiable random numbers.

Overview

SecretVRF over SecretPath enables EVM developers to accesson-chain verifiable random numbers at a fraction of the cost and block time of traditional RNG oracles such as ChainlinkVRF. With fewer than 100 lines of code, you will have access to an infinite supply of randomness.

To learn how SecretVRF works underneath the hood, refer to the docshere .

Getting Started

To get started, clone theSecret Labs examples repo :

```
Copy gitclonehttps://github.com/scrtlabs/examples.git
```

EVM Prerequisites

1. Add Polygon Mumbai testnet to Metamask
2. .
3. Fund your Mumbai wallet
4. .
5.

Configuring Environment Variables

cd intoexamples/EVM-snakepath-RNG :

```
Copy cdexamples/EVM-snakepath-RNG
```

Install the node dependencies:

```
Copy npminstall
```

Update theenv file with your EVM wallet private key andInfura API key.

Make sure your Infura API key is configured for Polygon Matic testnet 😎

Upload & Instantiate RandomnessReceiver.sol

Compile your Solidity smart contract:

```
Copy npxhardhatcompile
```

Once the contract is compiled successfully, upload the contract to Polygon testnet:

```
Copy npxhardhatrunscripts/deploy.js--networkpolygon
```

Note the contract address:

```
Copy RandomnessReceiverdeployedto:0x08D05bC52e503C68c38A32c1fA997FB521e614C4
```

Add theRandomnessReceiver contract address to yourenv file:

```
Copy RANDOMNESS_RECEIVER_CONTRACT_ADDRESS="0x08D05bC52e503C68c38A32c1fA997FB521e614C4"
```

Execute RandomnessReceiver.sol

Now that you've uploaded your contract, it's time to set the SecretPath gateway address for Polygon Mumbai and then request on-chain verifiable random numbers!

Gateways are the on-chain smart contracts that handle the broadcasting, receipt, packaging, and verification of messages.

Set Gateway Contract

First, set the gateway address for Polygon Mumbai testnet. You can do this by executingset_gateway.js :

```
Copy npxhardhat--networkpolygonrun./scripts/set_gateway.js
```

This tutorial is for Polygon testnet,, but you can find a list of additional EVM gateway contract addresseshere .

Create Randomness Event Listener

Next, create an event listener so you can listen to when the random numbers that you request have been fulfilled.

Open a new terminal window andcd intoexamples/EVM-snakepath-RNG:

```
Copy cd examples/EVM-snakepath-RNG
```

Then, create the event listener by executingfulfill_randomness_event.js :

```

Copy npxhardhat--networkpolygonrun./scripts/fulfill_randomness_event.js

```

Request Random Numbers

Now it's time to request random numbers! Currently,request_random.js is configured torequest 3 random numbers , but you can update how many numbers you would like to request here (up to 2000 for this example).

Once you have configured how many random numbers you want to request, executerequest_random.js :

```

Copy npxhardhat--networkpolygonrun./scripts/request_random.js

```

Upon successful execution, your terminal will log the following:

```

Copy Currentgasprice:1.500000016gwei Amountofgas:202500002160000 Transactionhash:0x47efe733c6b64a5c65fae68a5fa0f2eb39be107a7d4930325104dfcee36474c2
RandomNumbersrequestedsuccessfully!

```

Navigate to your event listener terminal to see the returned random numbers:

```

Copy Random numbers fulfilled for request ID: 7 Random Numbers:
94412630379044474934232934838909700375960606882138821083837396872559692127250,113337239238407277551866961530595655396141218773986266698805816049961297644274,2742

```

Congrats! You've just used SecretPath to request your first verifiable on-chain random numbers!

If you don't see your random numbers returned, it means that our testnet relayer might have dropped the transaction.See below to learn how to relay your transaction manually.

Execute SecretPath Manually with Polygonscan

To relay your random numbers manually, you can use Polygonscan and Secret.js!

After you executerequest_random.js and have atask_id returned, you can now executequery_secret_network for the giventask_id.

Openquery_secret_network.js and update thetask_id to yourtask_id . Then executequery_secret_network.js :

```

Copy npxhardhat--networkpolygonrun./scripts/query_secret_network.js

```

The query will return info about your transaction for the giventask_id :

```

Copy { source_network:'pulsar-3', task_destination_network:'80001', task_id:'5', payload_hash:'0xad5f42b51c2d755f5427f6373a7398b9b24ba68baa17dc590f05bb83f3e0f940',
result:'0xe4b051f8e4407a7b44a170cfed845b98ba9db0864e2c43eef3009d42c0e5ed05a1f2023d5de167f4f9b2c8646992b65098af109ea076f9e2d128e8975e54dfaa90d1502c126a8a672bccb3c4d69034
packet_hash:'0x5fb0bb5e85357373b84f92b95f41cb404385165d46a58af9470bf13eb2648f7b',
packet_signature:'0x815823bda4562ba7411ec5bf2de492bb377a808c1c0b17a0dfa6f5729c23af222c76e46531cee78e94730fa4a63426eb20de0cfd9389cdc45407e12a1c8ed3d51b',
callback_address:'0x08d05bc52e503c68c38a32c1fa997fb521e614c4', callback_selector:'0x38ba4614', callback_gas_limit:'0x00015f90' }

```

Now, openPolygonscan for the Mumbai proxy contract and then input the returned query info into thepostExecution field:

Once you have entered your transaction info, select "Write" to execute the transaction.

Congrats! You've just used SecretPath to request your first verifiable on-chain random numbers!

Conclusion

Secret VRF offers an innovative and cost-effective solution for EVM developers seeking access to verifiable random numbers. By following this guide, you've successfully set up your environment, deployed theRandomnessReceiver.sol contract, and interacted with the SecretPath network to request and receive random numbers. Dive into the world of decentralized randomness with SecretPath, where security meets simplicity.

Last updated5 days ago