Implementing VRF into any EVM Contract

Got improvements or suggestions on how to improve SecretVRF or this tutorial ? Please ask in the Secret Network Telegram or Discord. For a detailed demonstration, you can watch our video tutorial available here:

After we've gone through an extensive example on how our example contract works, here's how to implement SecretVRF via SecretPath in your own contract in 4 easy steps:

Importing the Interface

First, import theISecretVRF interface into your Solidity Contract:

```

Copy ///@noticeInterface of the VRF Gateway contract. Must be imported. interfaceISecretVRF{ functionrequestRandomness(uint32_numWords,uint32_callbackGasLimit)externalpayablereturns(uint256requestId); }

```

Set the SecretVRF gateway contract

Second, set your gateway address to the Secret VRF Gateways that you can find EVM Testnet and EVM Mainnet . You only need to make sure that your contract knows the correct SecretVRF Gateway address, for example:

```

Copy ///@noticeVRFGateway stores address to the Gateway contract to call for VRF addresspublicVRFGateway;

addresspublicimmutableowner;

constructor() { owner=msg.sender; }

modifieronlyOwner() { require(msg.sender==owner,"UNAUTHORIZED"); _; }

///@noticeSets the address to the Gateway contract ///@param_VRFGateway address of the gateway functionsetGatewayAddress(address_VRFGateway)externalonlyOwner{ VRFGateway=_VRFGateway; }

```

Call the SecretVRF Gateway contract

Now, we implement the function that calls the SecretVRF Gateway on EVM. Note that youhave to pay an extra amount of your gas token asCallbackGas:

```

Copy ///@noticeEvent that is emitted when a VRF call was made (optional) ///@paramrequestId requestId of the VRF request. Contract can track a VRF call that way eventrequestedRandomness(uint256requestId);

///@noticeDemo function on how to implement a VRF call using Secret VRF functionrequestRandomnessTest(uint32_numWords,uint32_callbackGasLimit)externalpayable{ // Get the VRFGateway contract interface ISecretVRF vrfContract=ISecretVRF(VRFGateway);

// Call the VRF contract to request random numbers. // Returns requestId of the VRF request. A contract can track a VRF call that way. uint256requestId=vrfContract.requestRandomness{value:msg.value}(_numWords,_callbackGasLimit);

// Emit the event emitrequestedRandomness(requestId); }

```

The callback gas is the amount of gas that you have to pay for the message coming on the way back. If you do pay less than the amount specified below, your Gateway TX will fail:

```

Copy ///@noticeIncrease the task_id to check for problems ///@param_callbackGasLimit the Callback Gas Limit

functionestimateRequestPrice(uint32_callbackGasLimit)privateviewreturns(uint256) { uint256baseFee=_callbackGasLimit*block.basefee; returnbaseFee; }

```

Since this check is dependent on the currentblock.basefee of the block it is included in, it is recommended that you estimate the gas fee beforehand and add some extra overhead to it. An example of how this can be implemented in your frontend can be found in thisexample and here:

```

Copy //Then calculate how much gas you have to pay for the callback //Forumla: callbackGasLimit*block.basefee. //Use an appropriate overhead for the transaction, 1,5x = 3/2 is recommended since gasPrice fluctuates.

constgasFee=awaitprovider.getGasPrice(); constamountOfGas=gasFee.mul(callbackGasLimit).mul(3).div(2);

```

Wait for the callback

From here, the SecretVRF Gateway will take care of everything, just wait 1-2 blocks for Gateway to provide you the random number by getting it from the Secret Networks on chain VRF and do the callback.

The SecretVRF gateway contract willalways call the contract that called the VRF contract (usingmsg.sender) with the function selector bytes0x38ba4614 , which is the function:

```

Copy functionfulfillRandomWords(uint256requestId,uint256[]calldatarandomWords)external

```

Now, the SecretVRF Gateway contract will verify the validity of the call and when all checks pass, it will call this function. In this case, we just emit a log as an example to finish our demo. Emitting a log isnot obligatory and optional.

```

Copy eventfulfilledRandomWords(uint256requestId,uint256[] randomWords); ///@noticeCallback by the Secret VRF with the requested random numbers ///@paramrequestId requestId of the VRF request that was initally called ///@paramrandomWords Generated Random Numbers in uint256 array functionfulfillRandomWords(uint256requestId,uint256[]calldatarandomWords)external{ // Checks if the callback was called by the VRFGateway and not by any other address require(msg.sender==address(VRFGateway),"only Secret Gateway can fulfill");

// Do your custom stuff here, for example: emitfulfilledRandomWords(requestId,randomWords); }

```

Last updated5 days ago On this page * Importing the Interface * Set the SecretVRF gateway contract * Call the SecretVRF Gateway contract * Wait for the callback

Was this helpful? Edit on GitHub Export as PDF