How does Gelato VRF Work?

Gelato VRF (Verifiable Random Function) provides a unique system offering trustable randomness on EVM-compatible blockchains. But what's the magic behind this reliable randomness? Let's see! After reading this page:

- You'll be able to understand the core components of Gelato VRF.
- You'll understand how you can initiate a randomness request.
- You'll be able to navigate the randomness delivery.
- You'll understand how to integrate and utilize Gelato VRF. *

Core Component

Drand : This is the heart of the randomness. Drand is a decentralized randomness beacon, ensuring the unpredictability and unbiased nature of the random numbers provided. To learn more about Drand and how it works, please refer totheir documentation .

Top level Flow

```

Copy +------------------------+ | 1. Contract Deployment | +------------------------+ | v +------------------------+ | 2. Requesting | | Randomness | +------------------------+ | v +------------------------+ | 3. Processing the | | randomness | | event | +------------------------+ | v +------------------------+ | 4. Randomness Delivery| +------------------------+

```

1. Contract Deployment

The smart contract that developers need to interact with is located atGelatoVRFConsumerBase.sol

This contract serves as an interface to the Gelato VRF system, allowing other smart contracts to request and receive random numbers.

1. Requesting Randomness

Inside theGelatoVRFConsumer contract, there's an event namedRequestedRandomness . When a randomness request is made, this event is emitted.

TheRequestedRandomness event serves as a beacon, signaling the Gelato VRF system about the need for a random number. It contains 2 parameters:

- round
- explicitly signifies which Drand round is targeted to fulfill the randomness,
- data
- offers versatility for developers, it can be used to attach any supplementary information or context about the randomness request.
-

```

Copy eventRequestedRandomness(uint256round,bytesdata);

```

1. Processing the Randomness Request

Internally, the system leveragesWeb3 functions to listen for the emittedRequestedRandomness event and to fetch the required random number fromDrand .

1. Delivering Randomness

Composable Callback with Arbitary Data

Internally, the system invokes thefulfillRandomness function in the requesting contract.

Callback Invocation and Data Decoding

The random number (sourced fromDrand) is passed as therandomness parameter to the function. Additionally, thedata parameter can carry any supplementary data provided during the original request or by the Gelato VRF.

Previous Understanding VRF Next Security Considerations Last updated3 months ago On this page *Core Component * Top level Flow * 1. Contract Deployment * 2. Requesting Randomness * 3. Processing the Randomness Request * 4. Delivering Randomness