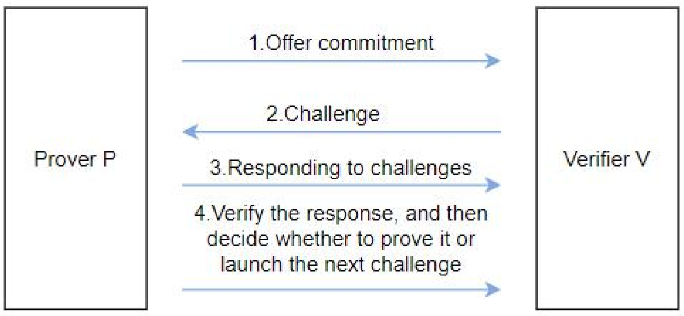
Decentralized red envelope

1 Overview

Red envelopes are a unique traditional Chinese culture, which has evolved into an indispensable part of daily communication. With the export of culture, it has also been more and more popular abroad. Traditional red envelope generation methods are usually based on deterministic randomness algorithms, such as WeChat red envelopes, but they cannot fundamentally achieve truly open and credible randomness; applying red envelopes in a decentralized blockchain can also promote users Conduct transactions to achieve the effect of increasing the activity of the platform. Here, we propose a method for generating decentralized red envelopes based on verifiable random functions.

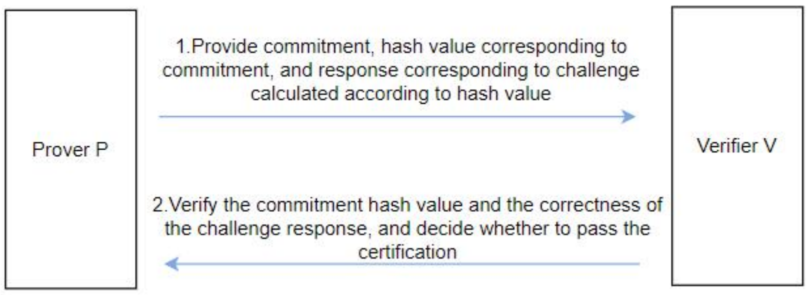
2 Zero-knowledge proof

Zero-knowledge proof means that the prover can convince the verifier that a certain assertion is correct without providing any useful information to the verifier. It has correctness, completeness and zero-knowledge. Zero-knowledge proofs can be divided into interactive and non-interactive according to their interactivity. In interactive zero-knowledge proofs, the prover P needs to provide a promise in advance to wait for the verifier V to initiate a challenge, and then respond to the challenge verification according to the random value in the challenge. Person V, through multiple rounds of challenges, until the prover P proves that the probability of the proof is large enough to recognize the proof. This kind of interactive zero-knowledge proof requires both parties to establish communication and conduct multiple rounds of interaction, so the communication efficiency is low, and it cannot meet the situation that the two parties cannot communicate.


Prover P
1.Offer commitment
2.Challenge
3.Responding to challenges
4.Verify the response, and then decide whether to prove it or launch the next challenge
Verifier V

The non-interactive zero-knowledge proof can solve the problems in the interaction well. Through the Fiat-Shamir transformation, the interactive proof can be transformed into the non-interactive proof, which utilizes the randomness of the hash function result, and the prover P can The hash calculation result of the promised data is used as a random number sequence to generate the challenge and the response corresponding to the challenge. Compared with interactive proof, non-interactive reduces the repeated challenge and response process. The prover only needs to send the data once, and the verifier can Verify by yourself without the need to communicate again.

In verifiable random functions, an important feature is the application of non-interactive zero-knowledge proofs to make the final generated results verifiable.


Prover P
1.Provide commitment, hash value corresponding to commitment, and response corresponding to challenge calculated according to hash value
Verifier V
2.Verify the commitment hash value and the correctness of the challenge response, and decide whether to pass the certification

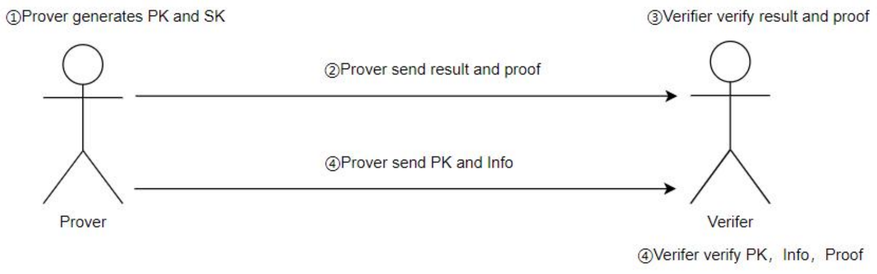
3 Verifable Random Functions

Verifiable Random Function (VRF) is an encryption scheme that maps the input to a verifiable pseudo-random output. Its final output result is a random number. Because the function execution process produces the corresponding non- With interactive zero-knowledge proof, the verifier can determine the legality of the random number through the public key. The verifiable random function has verifiability, uniqueness and randomness.

A basic verifiable random function should include four sub-functions: VRF_HASH, VRF_Proof, VRF_P2H, and VRF_Verfiy. The prover uses VRF_HAS and VRF_Proof to encrypt the information with the private key SK to generate the encrypted result and proof, and send the result and proof to The verifier, the verifier can verify the Proof through VRF_P2H to check whether the Proof is generated based on the result, if the result can be derived through this function, the prover needs to send its public key PK and the original text before encryption to the verifier, the verifier Verify that all parameters are correct through VRF_Verify.

The specific agreement process is as follows:

1. The prover generates a pair of secret keys, PK and SK;
2. The prover calculates result = VRF_HASH(SK,info);
3. The prover calculates proof = VRF_Proof(SK,info);
4. The prover submits the result and proof to the verifier;
5. The verifier calculates whether result = VRF_P2H(proof) is established, if yes, continue, otherwise abort;
6. The prover submits PK and info to the verifier;
7. The verifier calculates True/False = VRF_Verify(PK,info,proof), True means verification passed, False means verification failed.

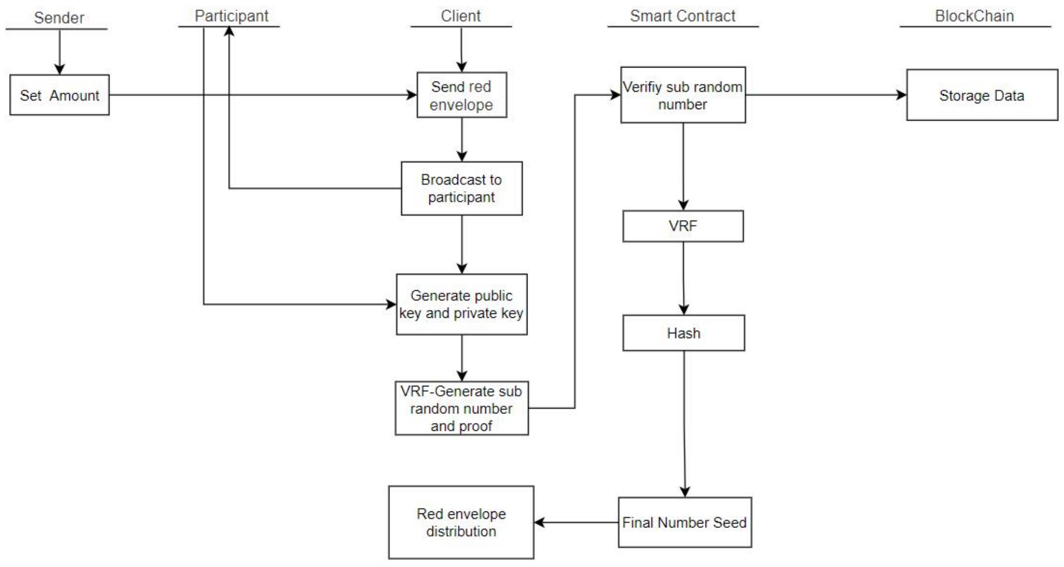

①Prover generates PK and SK
③Verifier verify result and proof
②Prover send result and proof
④Prover send PK and Info
Prover
Verifer
④Verifer verify PK，Info，Proof

4 Decentralized red envelope generation scheme based on verifiable random function

In order to provide a publicly verifiable red envelope generation, we propose a red envelope generation scheme based on a verifiable random function. This scheme can make the final generated red envelope amount public and credible through a verifiable random function, and people can verify it through the data stored on the chain, which solves the decentralization problem of traditional red envelopes.

In this scheme, when the user receives the red envelope, he needs to generate his own public key, private key, and initial random number in advance. The initial random number can be generated off-chain through some pseudo-random generation methods, such as hashing the time of receipt The user sends the user's sub-shares to the smart contract on the chain by running the verifiable random function with the initial random number. All users need to send their own sub-shares to the contract within the specified time, and the contract will perform the share calculation according to the VRF mechanism. Verification, users who fail the verification will be kicked out of the red envelope generation process; when all users upload the sub-shares successfully, the smart contract will perform VRF calculations on all the shares again, which can ensure that the execution of the smart contract is also publicly verifiable; The result of the final VRF operation is hashed to obtain the random seed generated by the red envelope, and the random seed can be allocated through the prescribed logic.

The specific program process is as follows:


Sender
Participant
Client
Smart Contract
BlockChain
Set Amount
Send red envelope
Verifiy sub random number
Storage Data
Broadcast to participant
VRF
Generate public key and private key
Hash
VRF-Generate sub random number and proof
Red envelope distribution
Final Number Seed