Smart Contract Test Plan

1. Test Environment Introduction

There are 4 contracts tested in this project, namely DataUsage Contract, Agreement Contract, Log Contract and Verification Contract. Use Web3.js + Truffle to test. Test from three aspects: functional testing, non-functional testing and security testing.

2. Functional testing

The functional testing of this project is divided into unit testing, integration testing and system testing. Unit tests are designed to test whether each function of the contract works normally according to the requirements. Integration testing aims to test whether the interaction between smart contracts and other related system components behave as expected. System testing aims to deploy the tested smart contracts through Metamask to Ethereum public blockchain test networks such as Goerli or Sepolia for testing.

2.1 Unit tests

This project will be unit tested from the following aspects.

- Status check: For example, after the contract is deployed, check whether the initial state is correct, and after the function call, check whether the status has changed and meets expectations. The general status check is to read the view function.
- Event triggering: Basically, key operations in the contract should trigger events for corresponding tracking. In the unit test, test whether the event is triggered and whether the parameters thrown are correct.
- Transaction reset: When testing some unexpected conditions, the transaction should be reset and give the corresponding reason. Use unit tests to check whether the reset and the cause of the error are the same.
- Function Computing: For example, to calculate the return value of a function under different conditions, you need to call a function repeatedly and input different parameters to see if the result matches or meets expectations.
- Full functional test: Test whether the various functions in the smart contract can run normally and meet expectations.
- Test coverage: including code coverage, function coverage and branch coverage.

2.1.1 DataUsage Contract

The main tests included in the DataUsage Contract are shown in Figure 1.

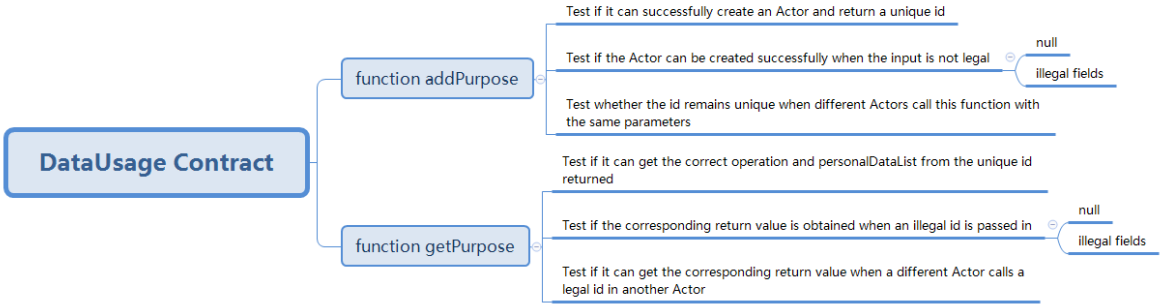

Figure 1. DataUsage Contract Test Plan

1. function addPurpose
2. function getPurpose

2.1.2 Agreement Contract

The main tests included in the Agreement Contract are shown in Figure 2.

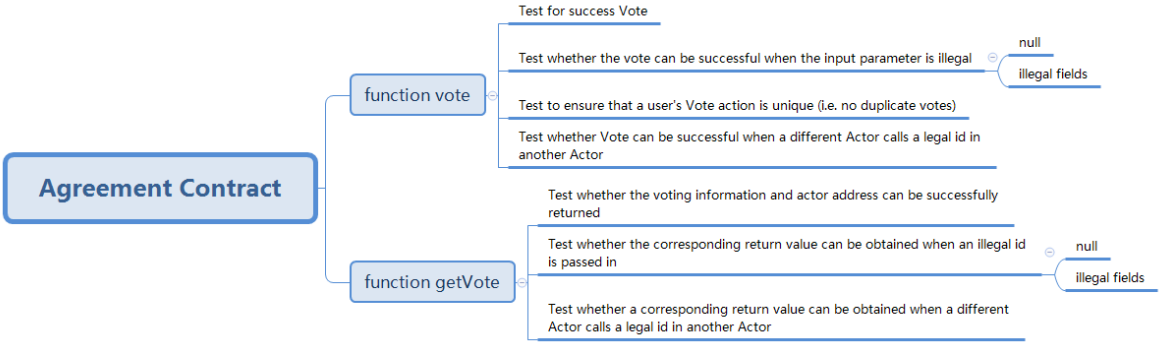

Figure 2. Agreement Contract Test Plan

1. function vote
2. function getVote

2.1.3 Log Contract

The main tests contained in the Log Contract are shown in Figure 3.

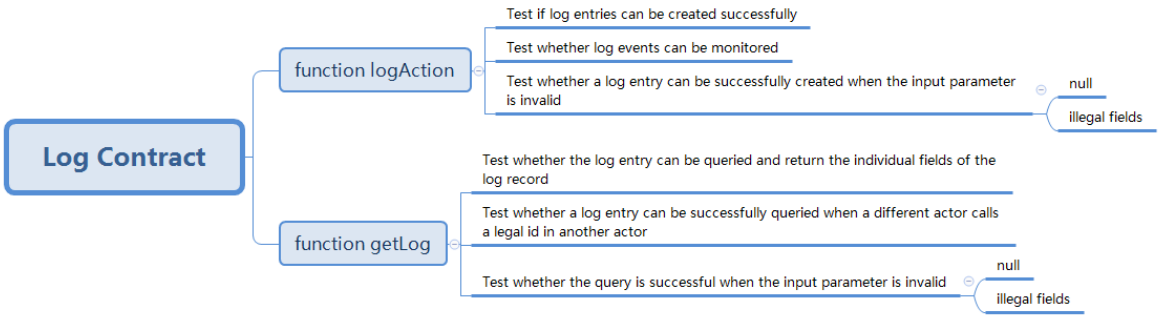

Figure 3. Log Contract Test Plan

1. function logAction
2. function getLog

2.1.4 Verification Contract

The main tests included in the Verification Contract are shown in Figure 4.

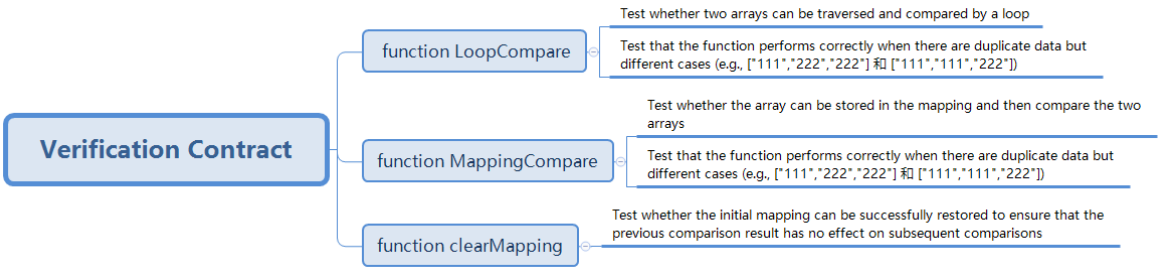

Figure 4. Verification Contract Test Plan

1. function LoopCompare
2. function MappingCompare
3. function clearMapping

2.1.5 Test coverage

2.2 Integration testing

The integration tests in this project are mainly in the Verification Contract, the main tests of which are shown in Figure 5.

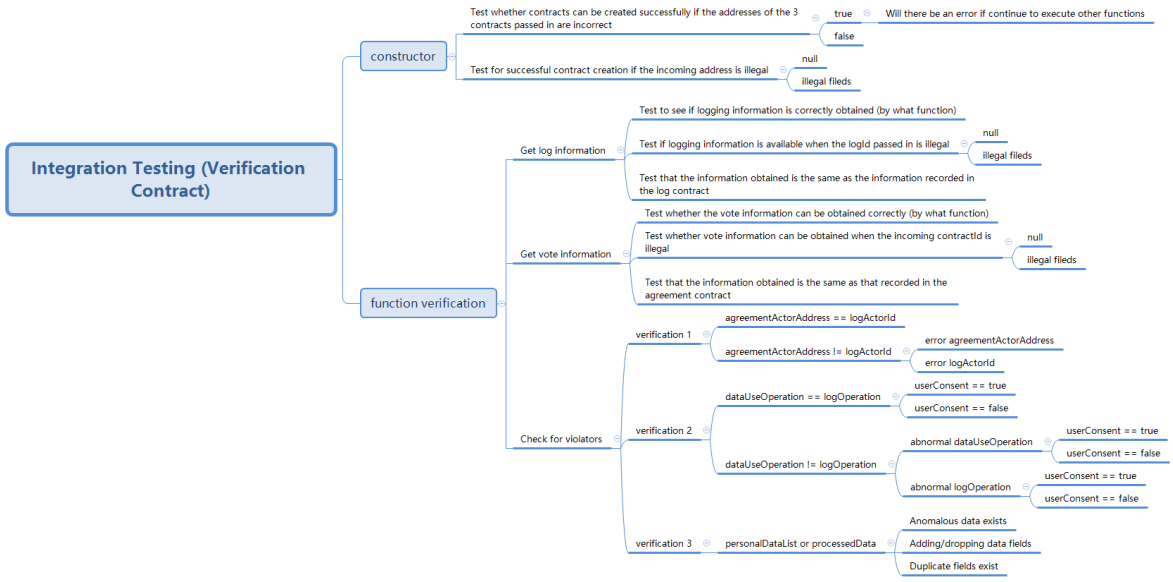

Figure 5. Integration Test Plan

2.2.1 constructor

2.2.2 function verification

1. Get log information
2. Get vote information
3. Check for violators
    - verification 1
    - verification 2

- verification 3

2.3 System testing

3. Non-functional testing

The non-functional test in this project will be tested and elaborated from 5 aspects of usability, scalability, reliability, compatibility and Gas consumption. The Gas consumption test refers to the change of the Gas consumption of the contract when the Actor increases from 1 to 10.

- Ease of use
- Scalability
- reliability
- Gas consumption

4. Security Testing

The non-functional tests in this project will be tested and described in five areas: boundary value testing, re-entry attack prevention, security of encryption algorithms, error handling, and static code security testing. The details of the tests are shown in Figure 6.

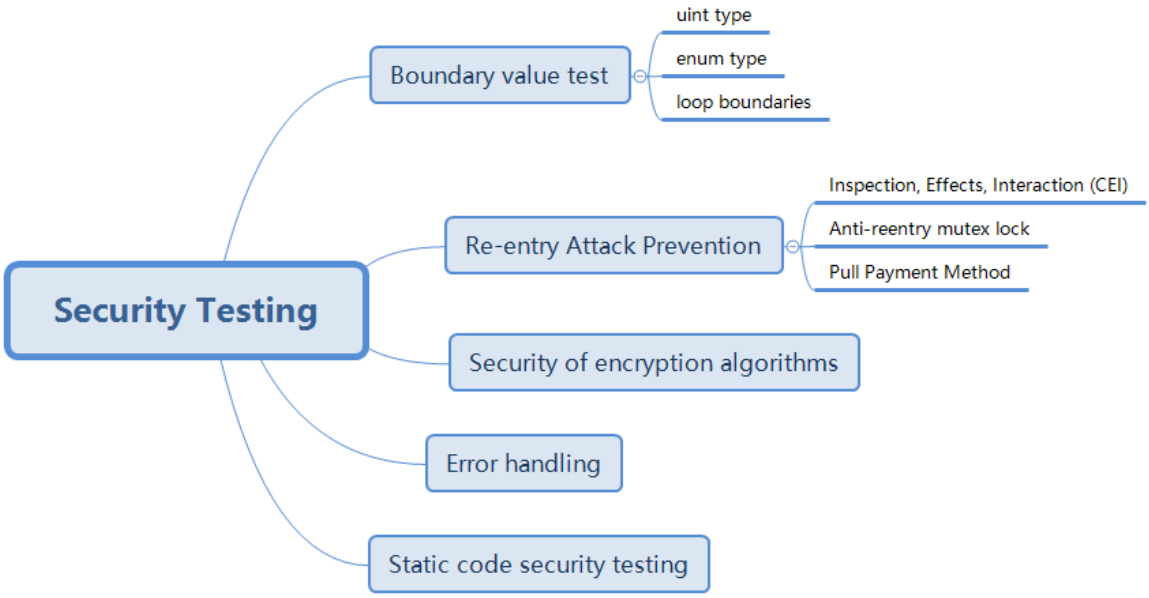

Figure 6. Security Test Plan