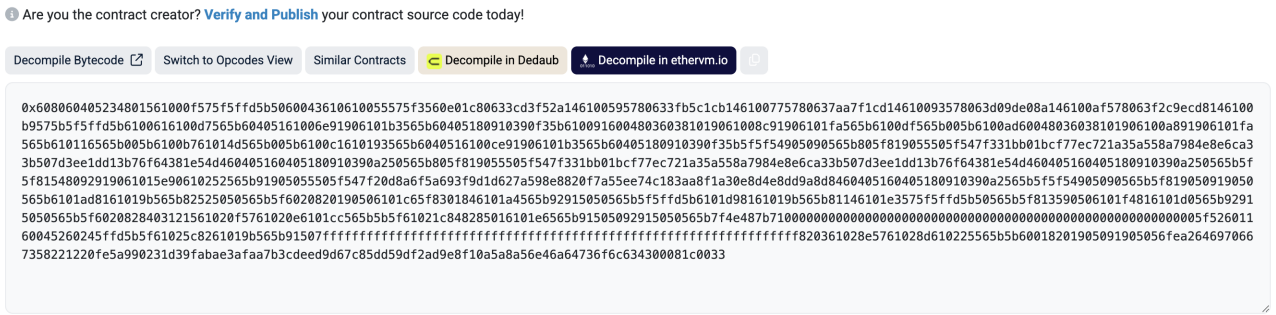
Design and Implementation of an Smart Contract Bytecode Analysis System Based on CFG and Image Analysis Technology

LEVI_104

2025-06

Background

- Ethereum, as the second generation of public blockchain technology, not only inherits the decentralized characteristics of Bitcoin but also introduces the concept of smart contracts.
- However, interacting with Ethereum smart contracts may pose potential dangers, especially in financial transactions and asset transfers. If errors occur or they are maliciously exploited, the consequences could be very serious.
- Smart contracts are not always open source, and users inevitably may interact with non-open source smart contracts. Currently, there is no simple and easy-to-use platform to parse non-open source contracts, and it is impossible to call these non-open source contracts. As shown in the left figure below.
- At the same time, there is no simple and easy-to-use platform to compare the similarity between two contracts. As shown in the lower right figure.

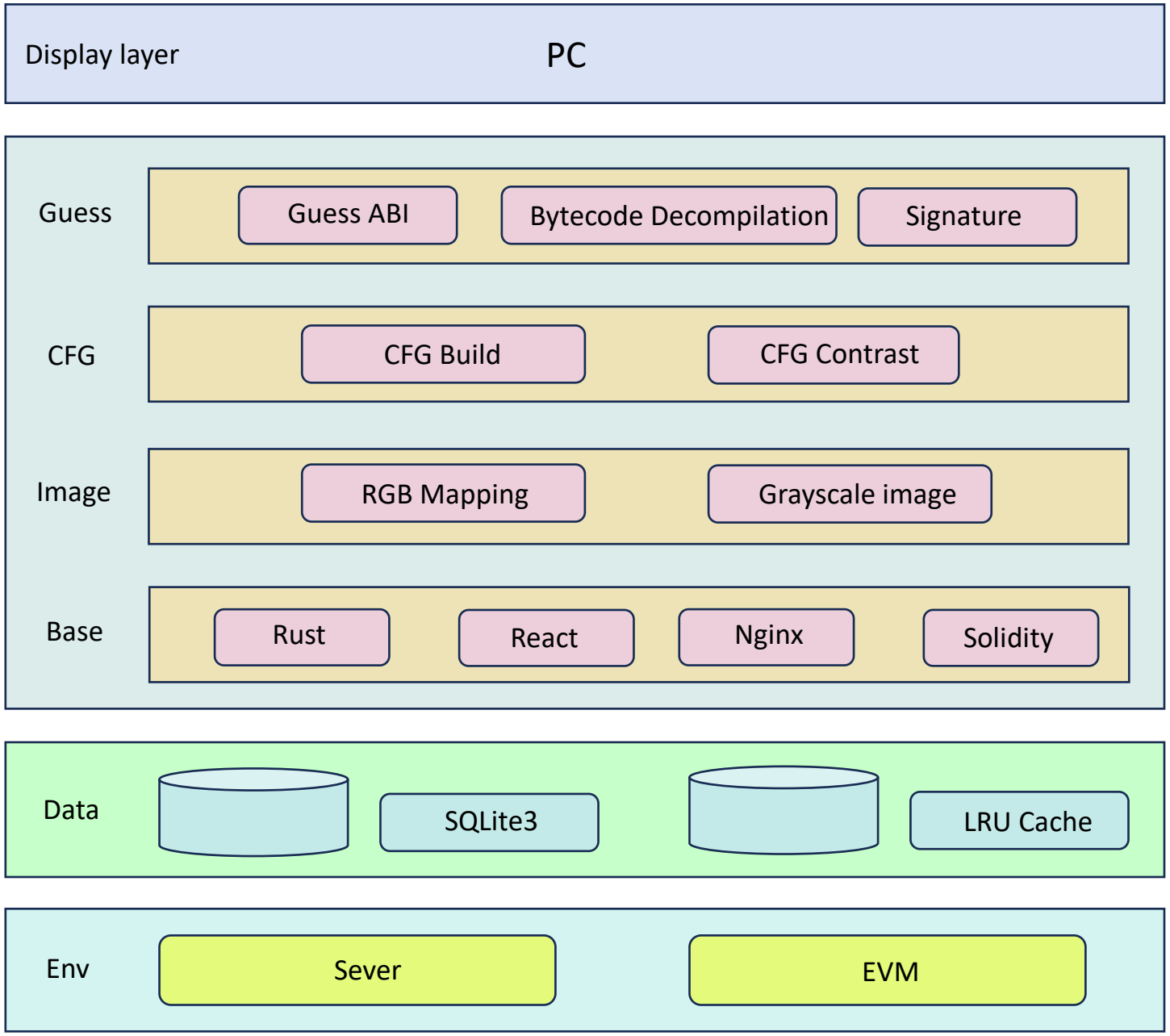


Feature Description

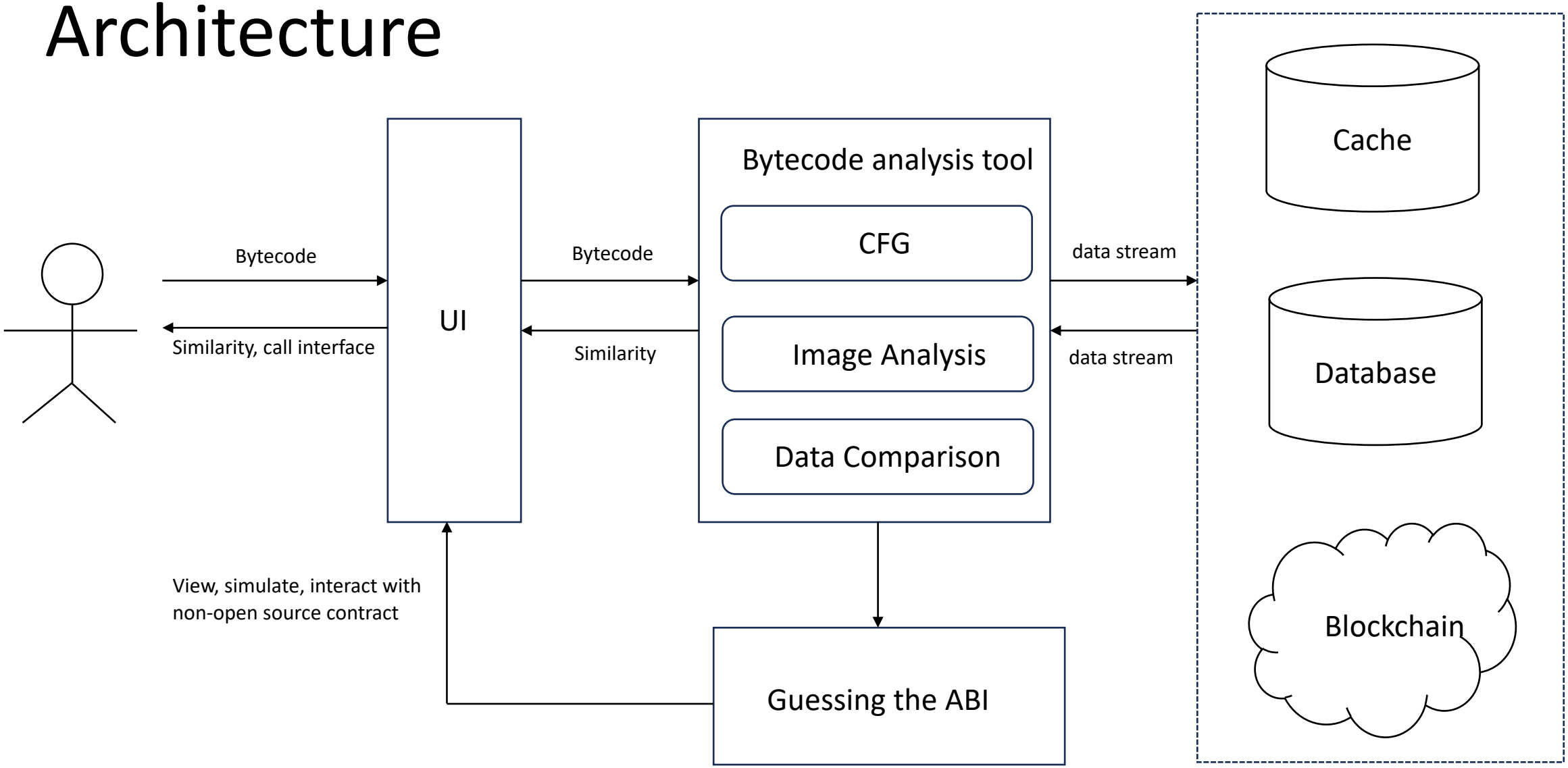
Design and implement a bytecode analysis platform that can decompile non-open source contracts, enabling:

- Viewing, simulating, and calling non-open source smart contracts
- Comparing the similarity between two contracts
- Decompiling bytecode into opcodes
- Supporting all proxy patterns
- Good user experience, friendly to users of all levels

System hierarchy

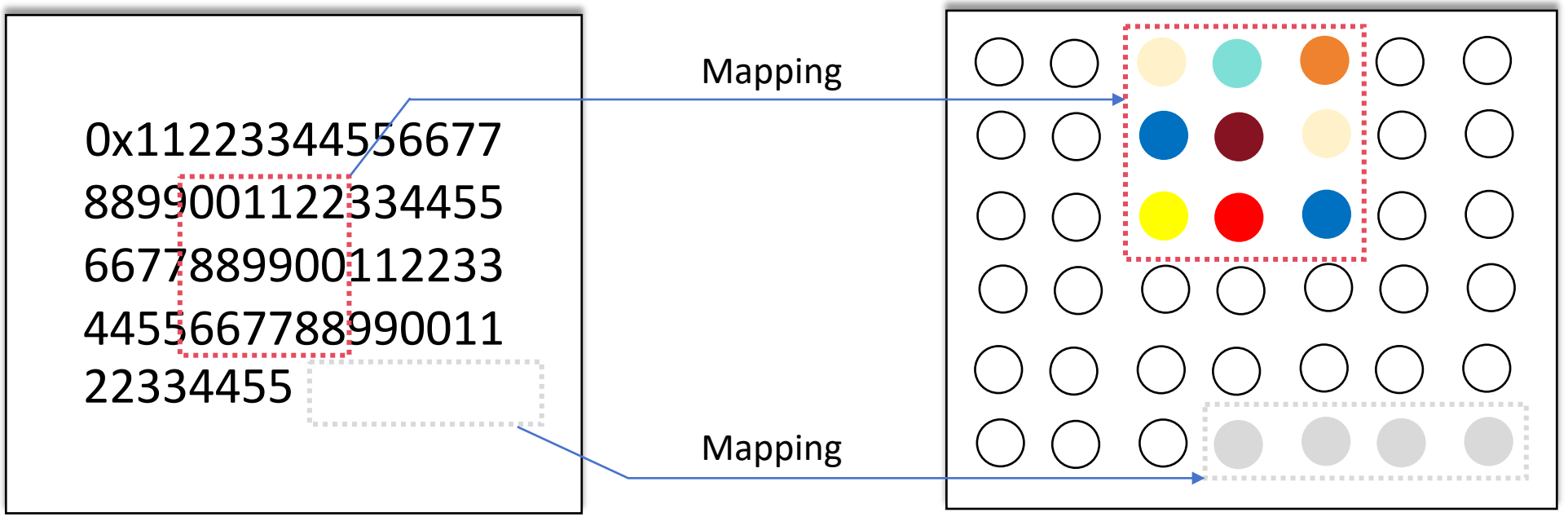


Architecture



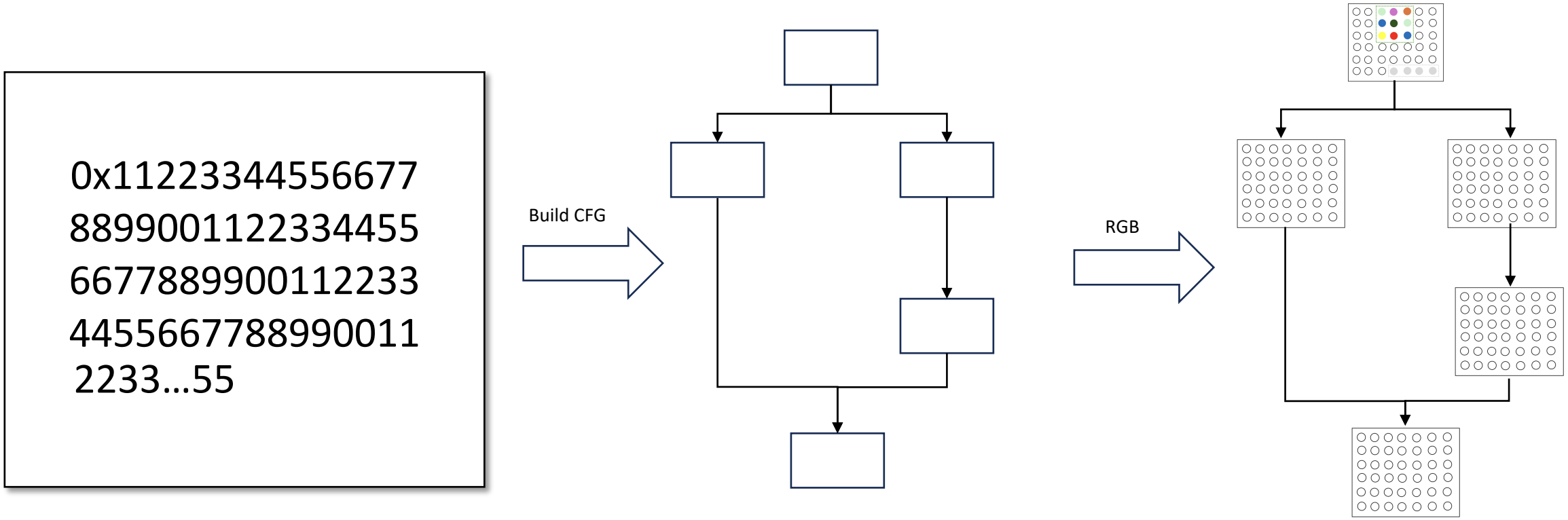
Bytecode Analysis - Assembling Overall Bytecode

- We convert each byte in the bytecode to an RGB color, representing a pixel in the image.
- Bytecodes of different lengths are mapped to rectangular images of the same size, with the missing parts of shorter bytecodes filled with 0 bytes.



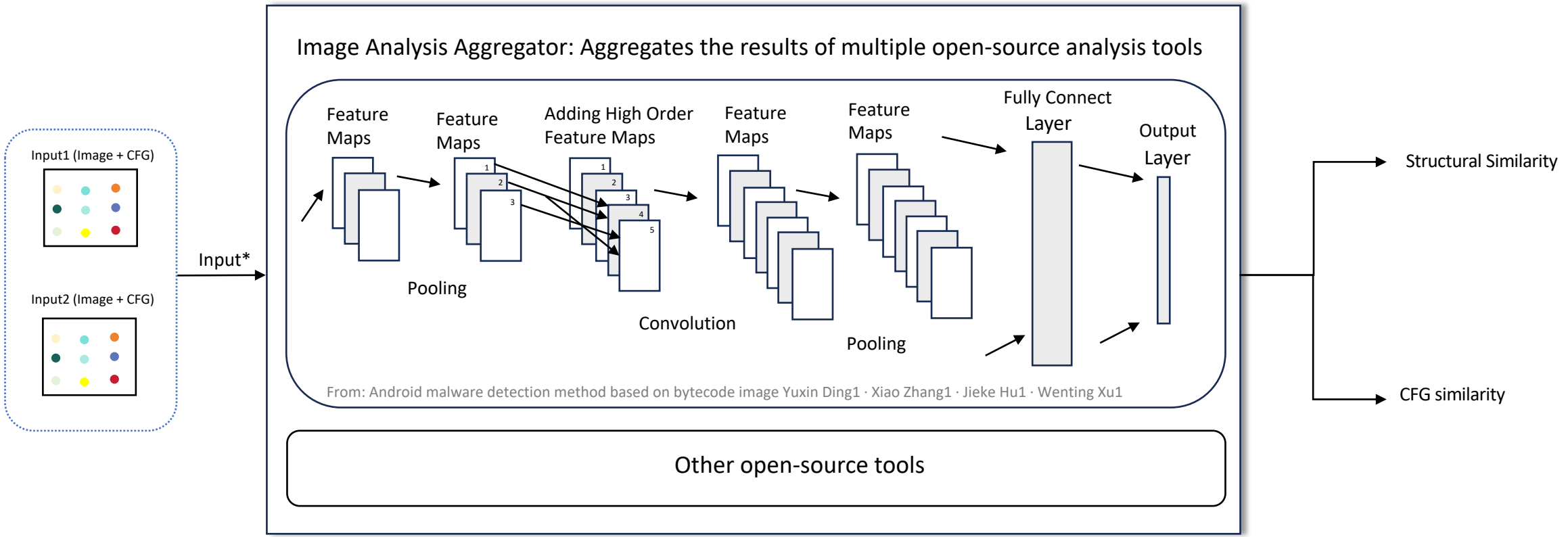
Bytecode Analysis - CFG Builder

- Bytecode is divided into CFG (Control Flow Graph) according to the compiler rules. For example, in EVM, the core logic focuses on opcodes such as Jump, Return, Stop, Revert, etc.
- Each block in CFG can be transformed using the same method as RGB image conversion.



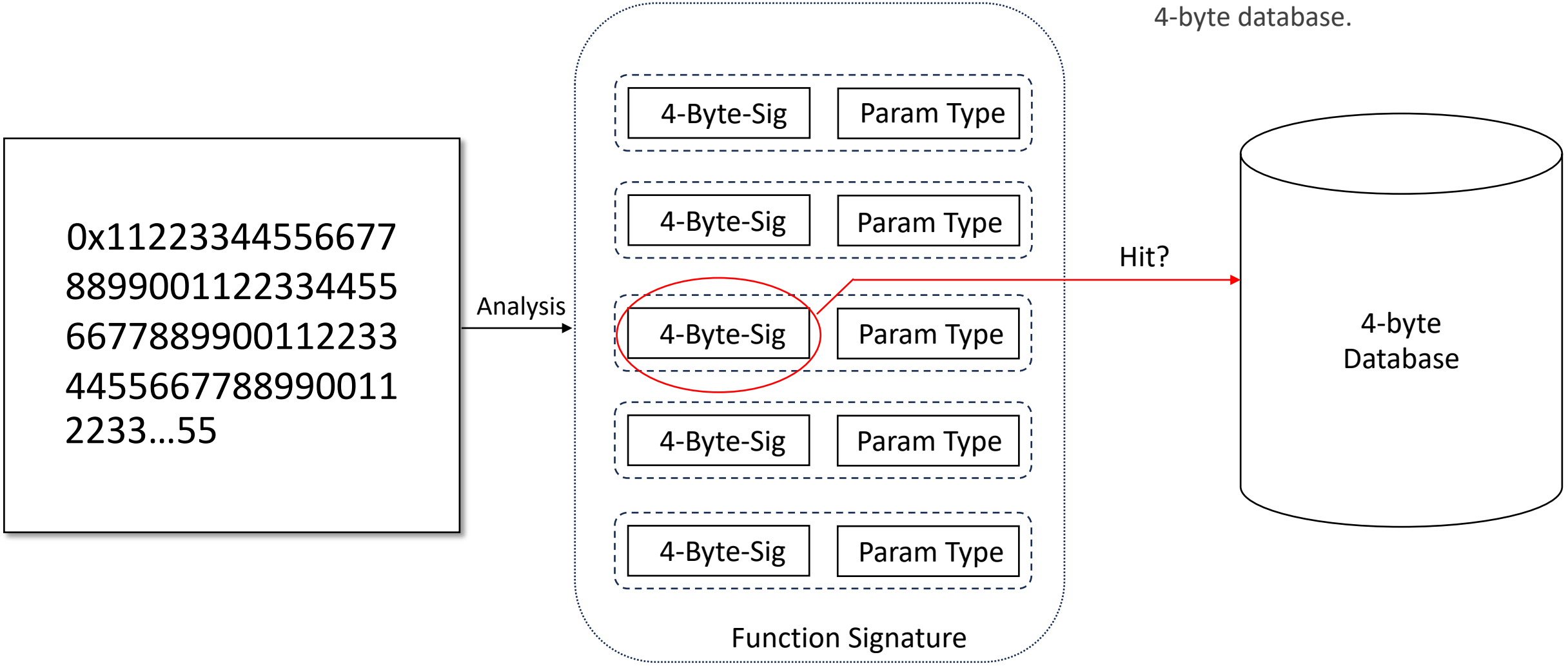
Bytecode Analysis - Image Analyzer

- The image analyzer receives two inputs as parameters, each input is a tuple: (image, CFG)
- Input the parameters into the image analysis aggregator, after analysis by multiple open-source tools, take the average of the results to obtain structural similarity and control flow similarity, as the final similarity parameter reserve.



Bytecode Analysis - ABI Guessing

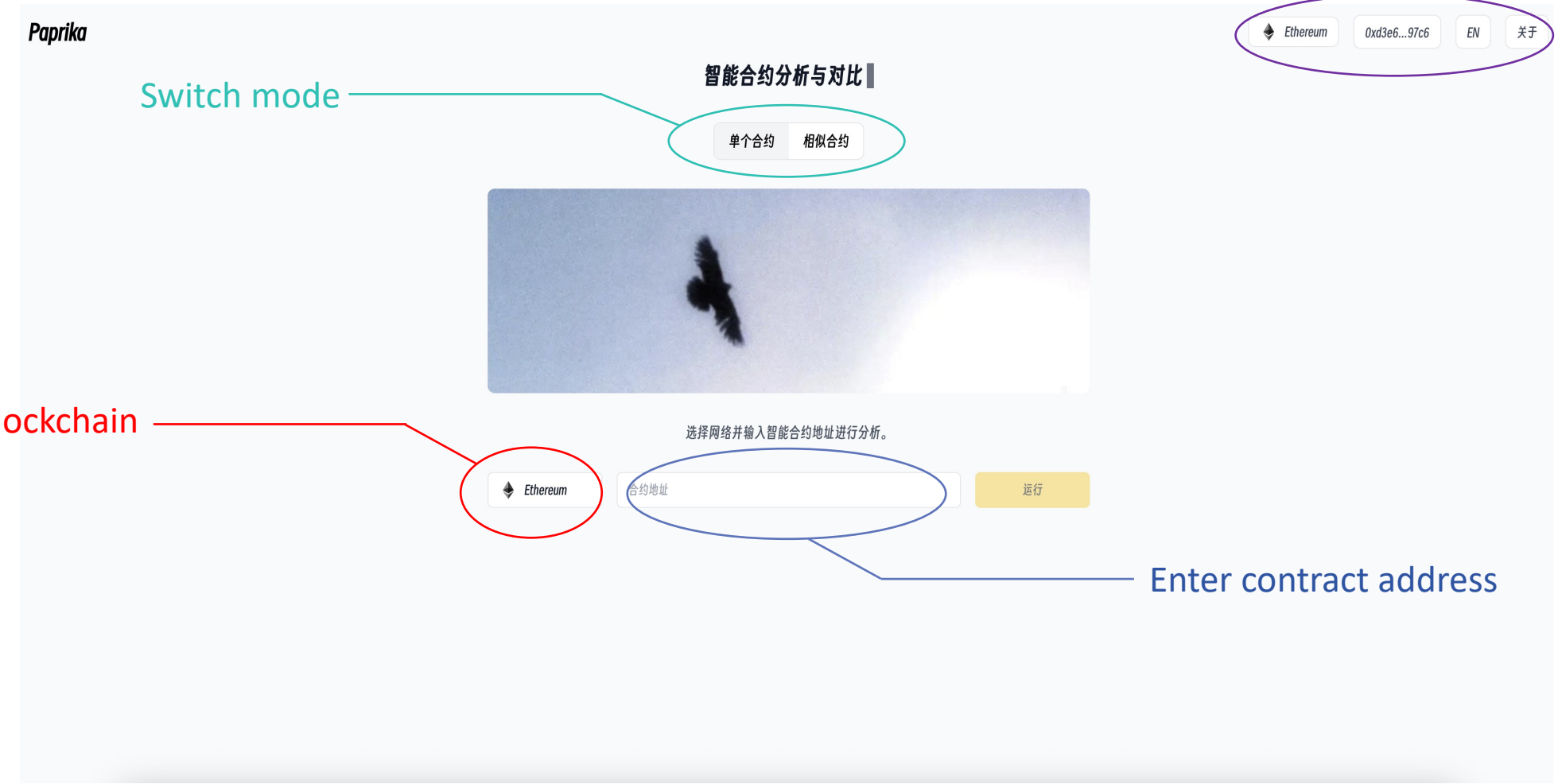
- We only need a four-byte signature and parameter type to piece together calldata, and then interact.
- In order to make the function names on the UI more meaningful, we will try to obtain human-readable function names from the 4-byte database.



* 4-byte database, a database that stores a large number of original function names and their corresponding 4-byte signatures: https://www.4byte.directory/

Front-End

Current blockchain, wallet address, website language switch



Switch mode

Switch blockchain

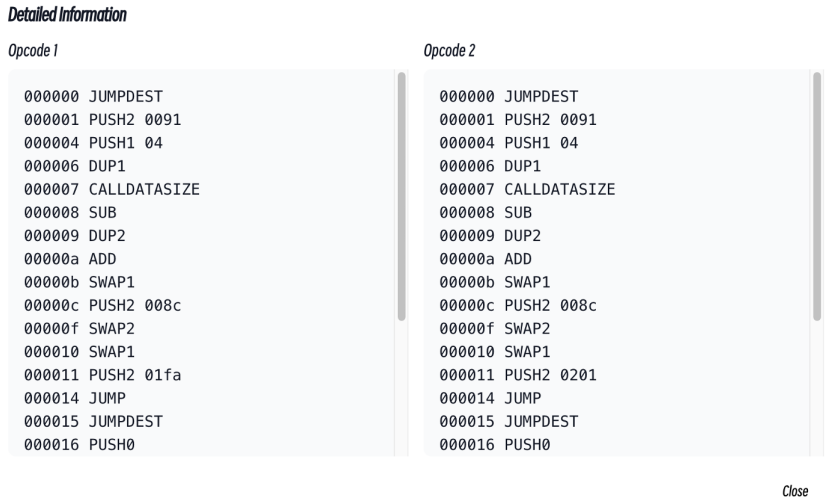
Enter contract address

Front-End

Copy CFG result details

Similarity results

Horizontal comparison of each pair of blocks in CFG



Detailed opcode comparison for each pair of Blocks

Front-End

反汇编

字节码_1.hex

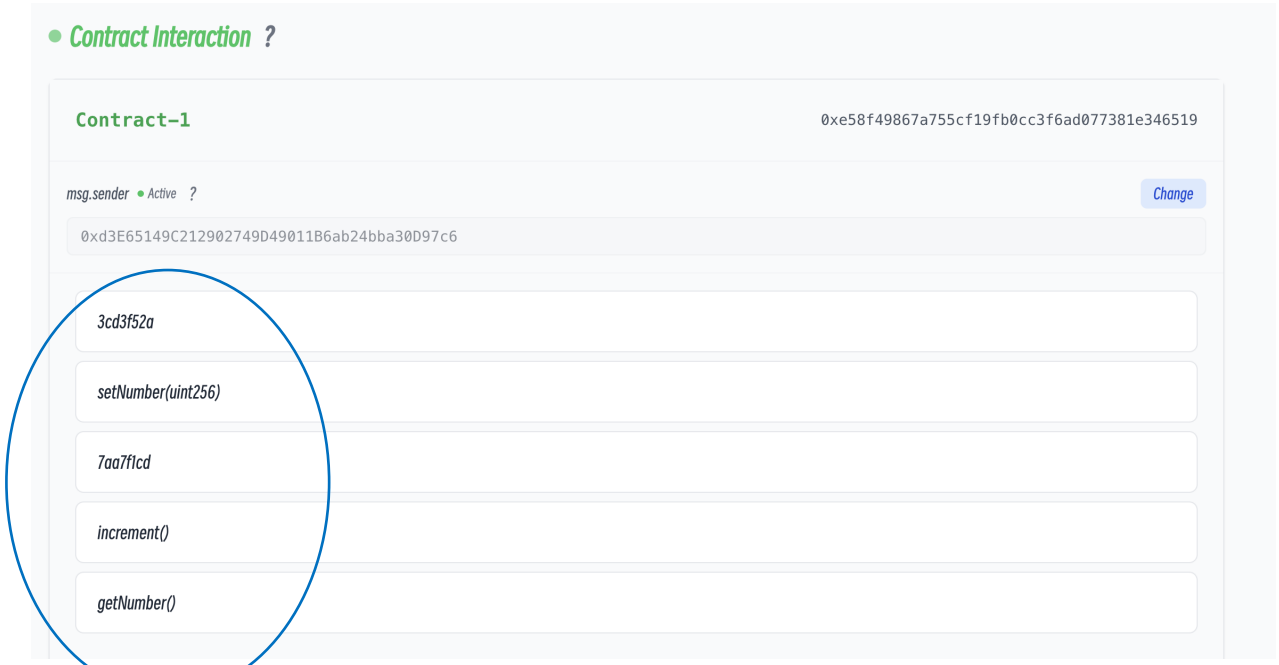
```
000000  PUSH1 80
000002  PUSH1 40
000004  MSTORE
000005  CALLVALUE
000006  DUP1
000007  ISZERO
000008  PUSH2 000f
00000b  JUMPI
00000c  PUSH0
00000d  PUSH0
00000e  REVERT
00000f  JUMPDEST
000010  POP
000011  PUSH1 04
000013  CALLDATASIZE
000014  LT
000015  PUSH2 0055
000018  JUMPI
000019  PUSH0
00001a  CALLDATALOAD
00001b  PUSH1 e0
00001d  SHR
00001e  DUP1
00001f  PUSH4 3cd3f52a
000024  EQ
000025  PUSH2 0059
000028  JUMPI
000029  DUP1
00002a  PUSH4 3fb5c1cb
00002f  EQ
000030  PUSH2 0077
000033  JUMPI
000034  DUP1
```

字节码_2.hex

```
000000  PUSH1 80
000002  PUSH1 40
000004  MSTORE
000005  CALLVALUE
000006  DUP1
000007  ISZERO
000008  PUSH2 000f
00000b  JUMPI
00000c  PUSH0
00000d  PUSH0
00000e  REVERT
00000f  JUMPDEST
000010  POP
000011  PUSH1 04
000013  CALLDATASIZE
000014  LT
000015  PUSH2 0055
000018  JUMPI
000019  PUSH0
00001a  CALLDATALOAD
00001b  PUSH1 e0
00001d  SHR
00001e  DUP1
00001f  PUSH4 3cd3f52a
000024  EQ
000025  PUSH2 0059
000028  JUMPI
000029  DUP1
00002a  PUSH4 3fb5c1cb
00002f  EQ
000030  PUSH2 0077
000033  JUMPI
000034  DUP1
```

Compare each opcode horizontally

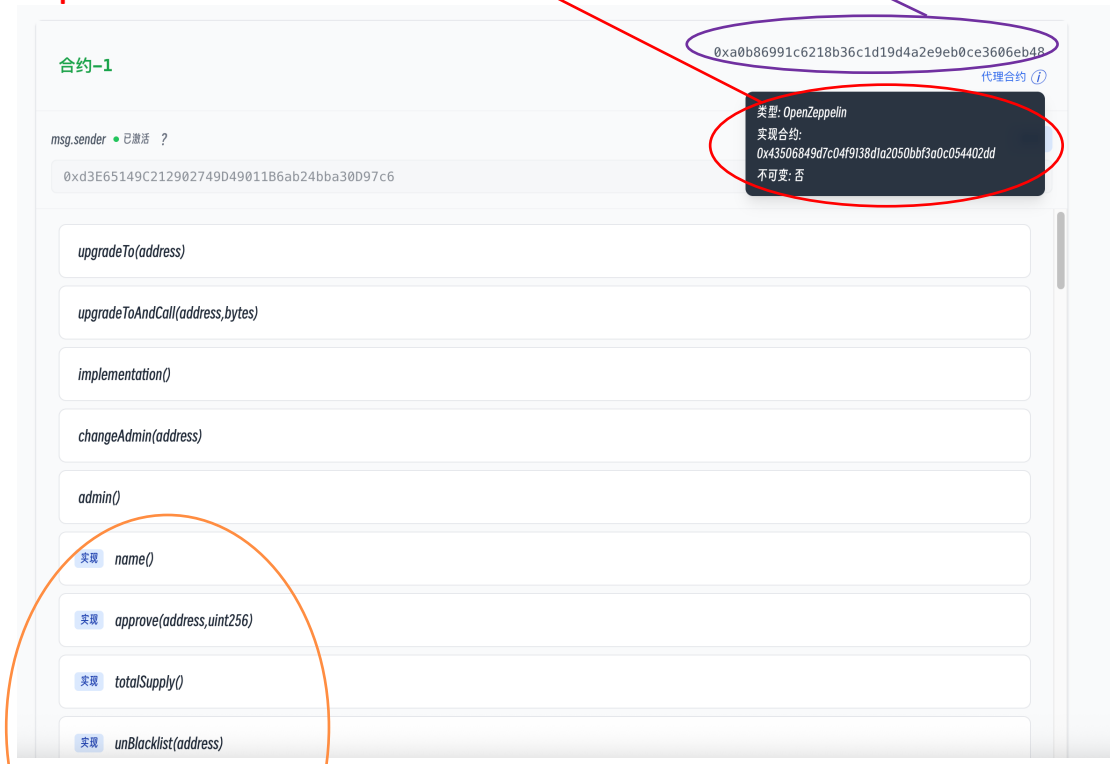
Front-End



The contract is not open source, but the contract interaction interface can still be obtained

Proxy

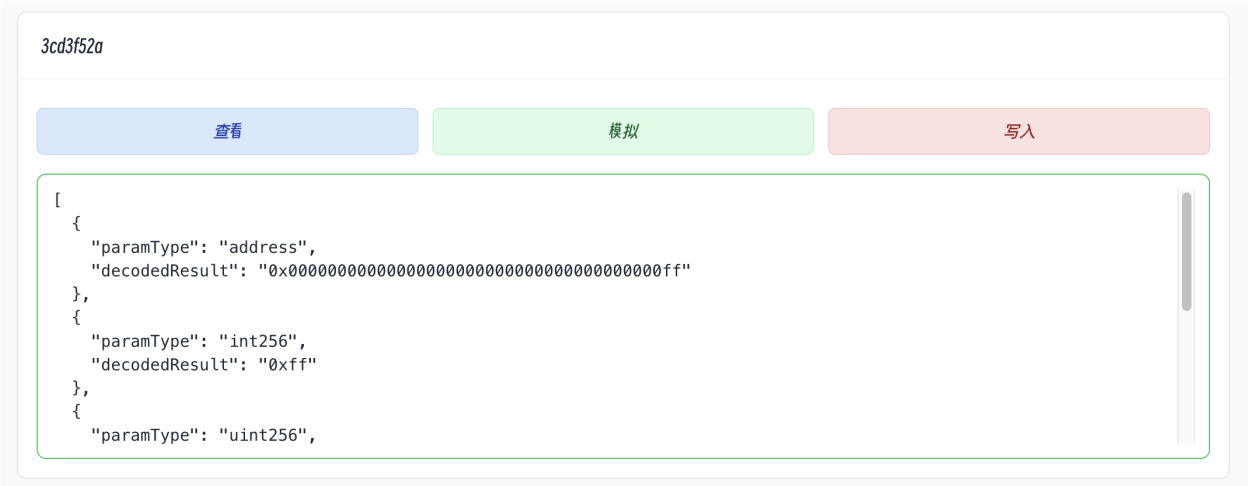
Implementation



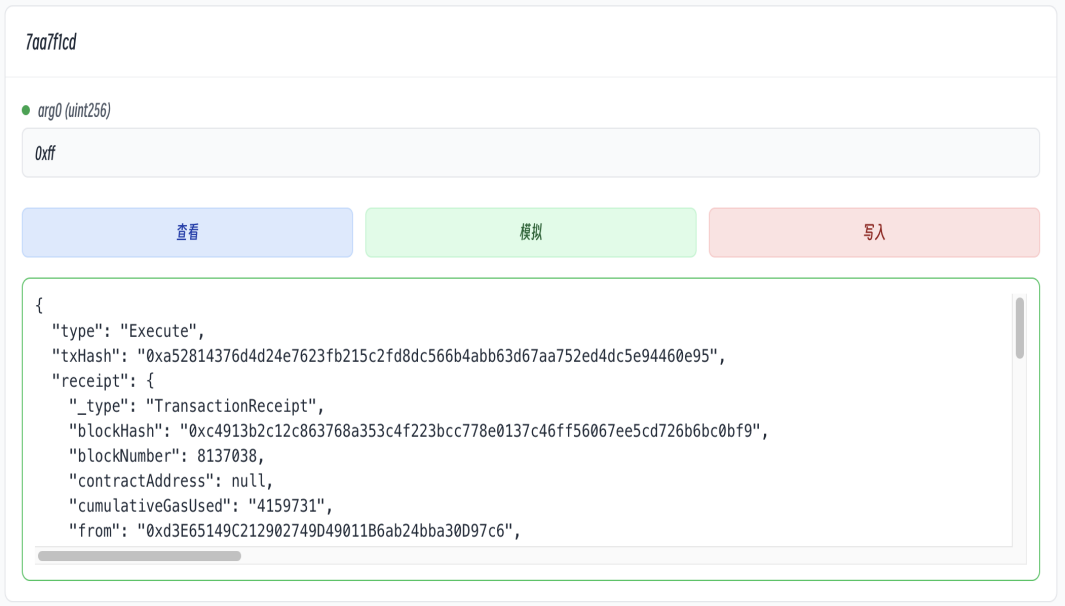
Implementation - Functions

Front-End

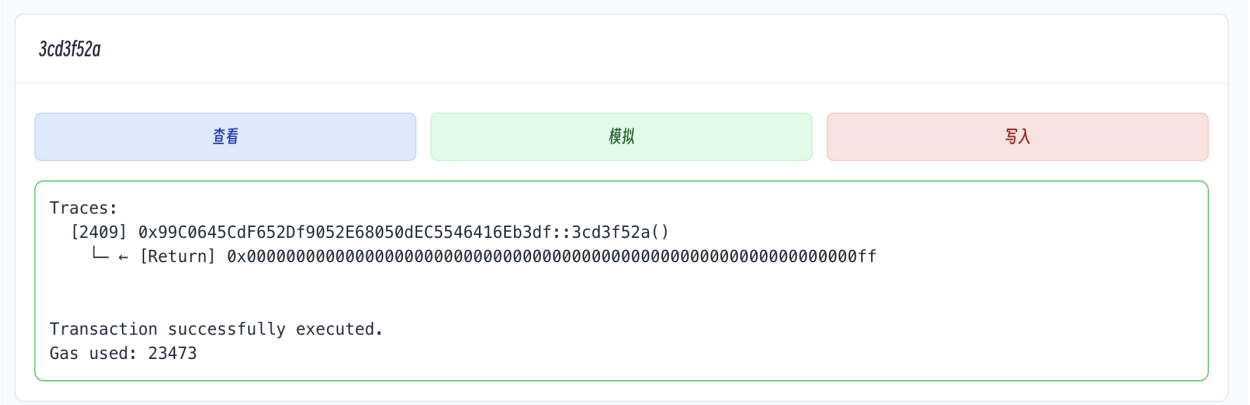
Analyze the non-open source contract 0xe58f49867a755cf19fb0cc3f6ad077381e346519 on the Ethereum-Sepolia test network.



Check the function 3cd3f52a()



Call the function 3cd3f52a()



Simulate 3cd3f52a() function

Summary

- This system has a simple and easy-to-use front-end page that enables users of all levels to understand how to use it
- This system can view, simulate, and call the functions of non open source contracts, enhancing the transparency of blockchain
- This system can compare the similarity between two smart contracts at the bytecode level
- This system supports all proxy modes to quickly locate and implement contracts