

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.

Are you the contract creator? [Verify and Publish](#) your contract source code today!

[Decompile Bytecode](#)
[Switch to Opcodes View](#)
[Similar Contracts](#)
[Decompile in Dedaub](#)
[Decompile in ethersvm.io](#)

[illegible]

Code

[Read Contract](#)

Write Contract

 View Storage

Similar Match Source Code

ⓘ This contract matches the **deployed Bytecode** of the Source Code for Contract [0xB4e16d01...1Ec28C9Dc](#)

① The constructor portion of the code might be different and could alter the actual behaviour of the contract

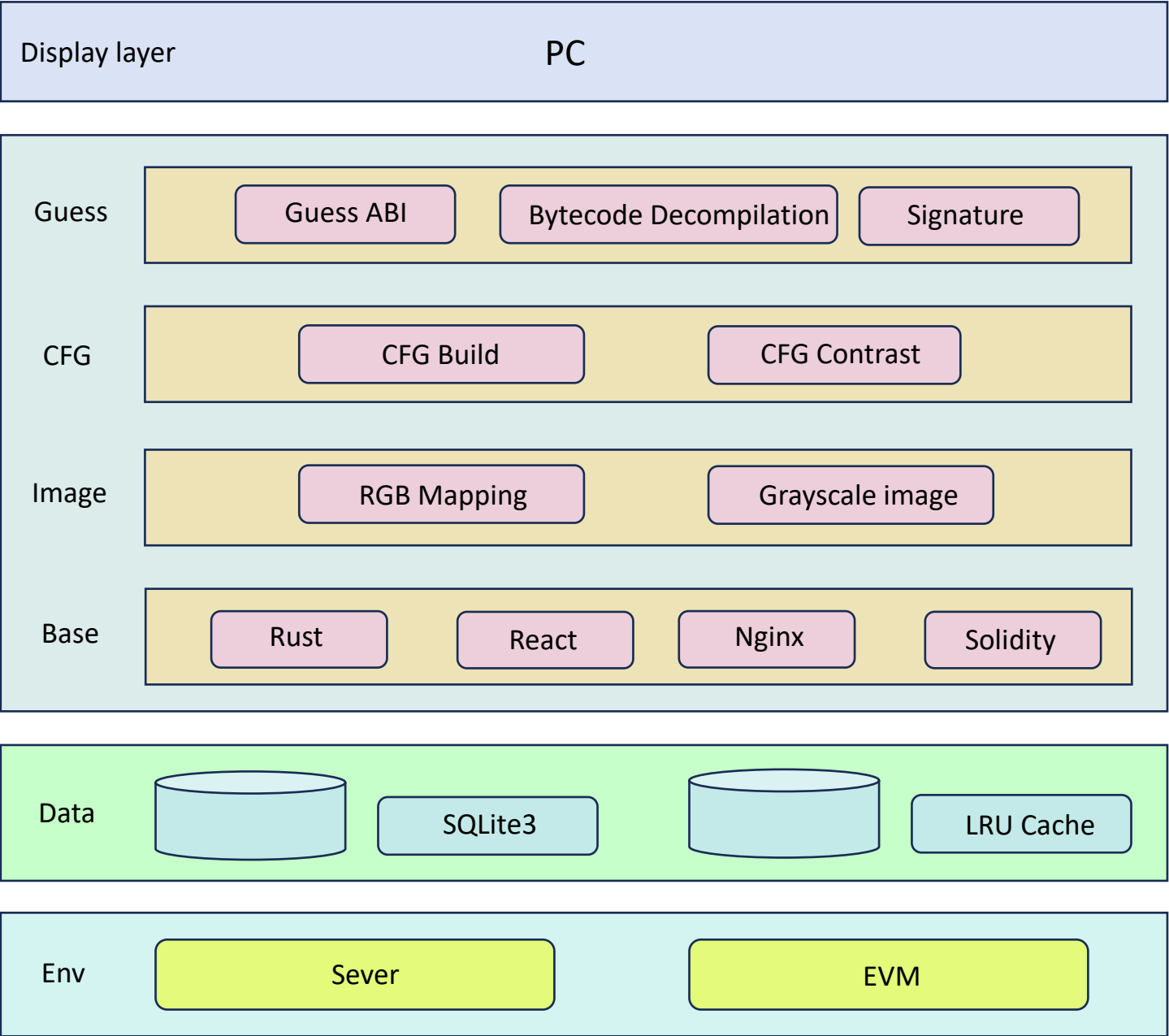


Feature Description

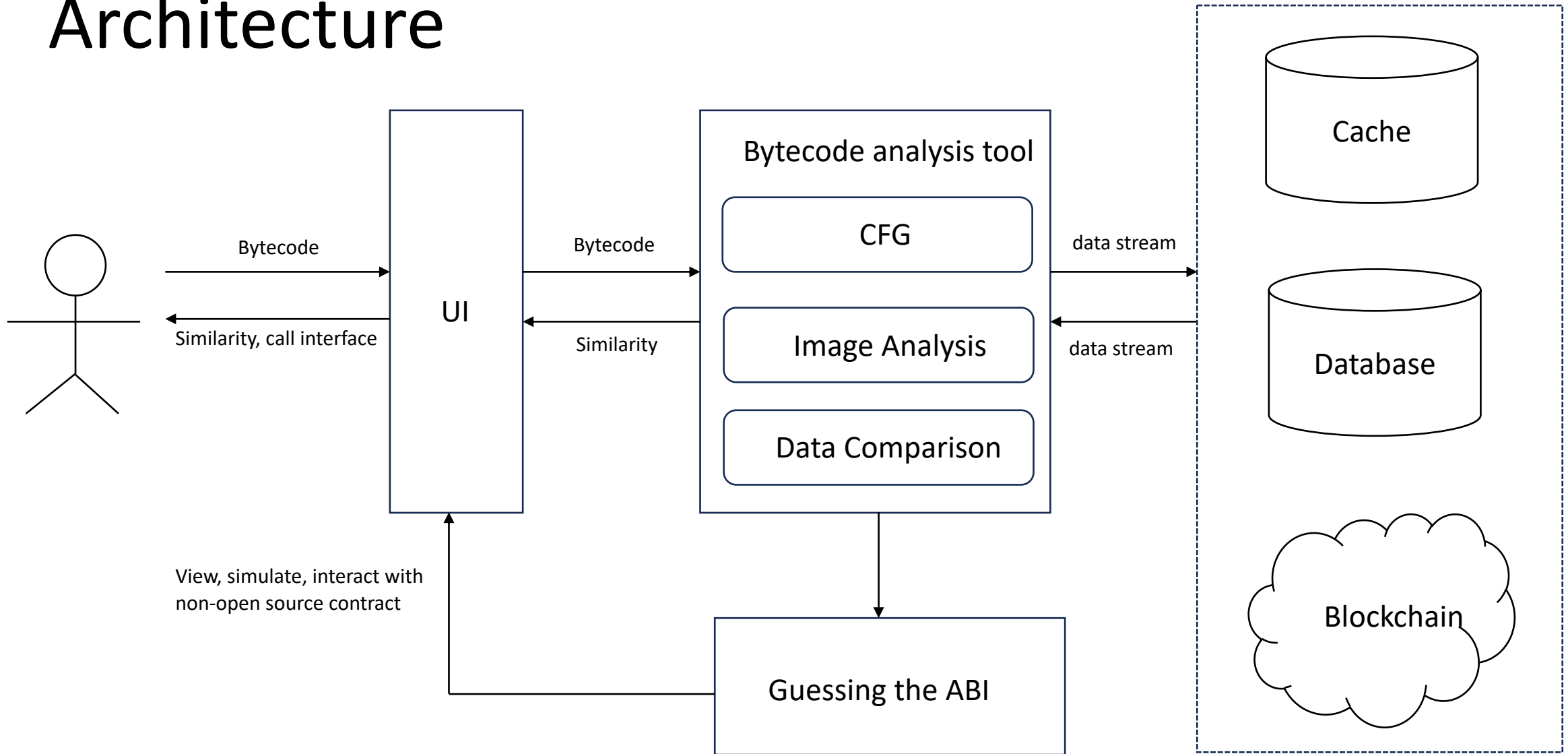
Design and implement a bytecode analysis platform that can decompile non-open source contracts, enabling:

- 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

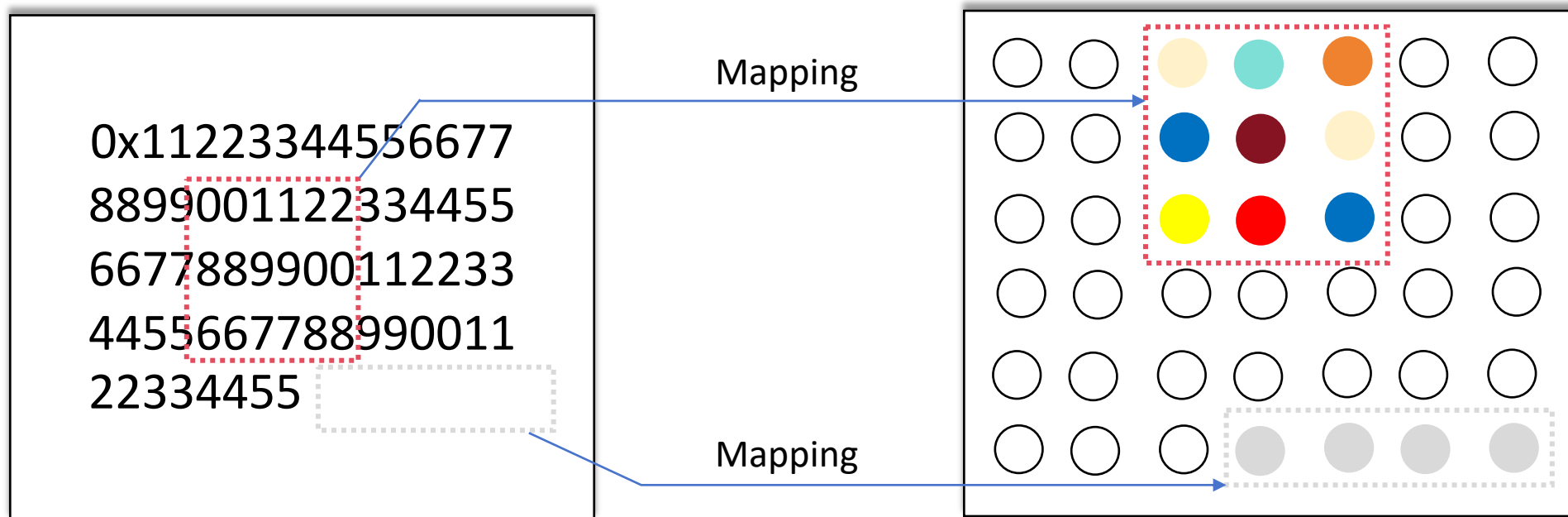


Architecture



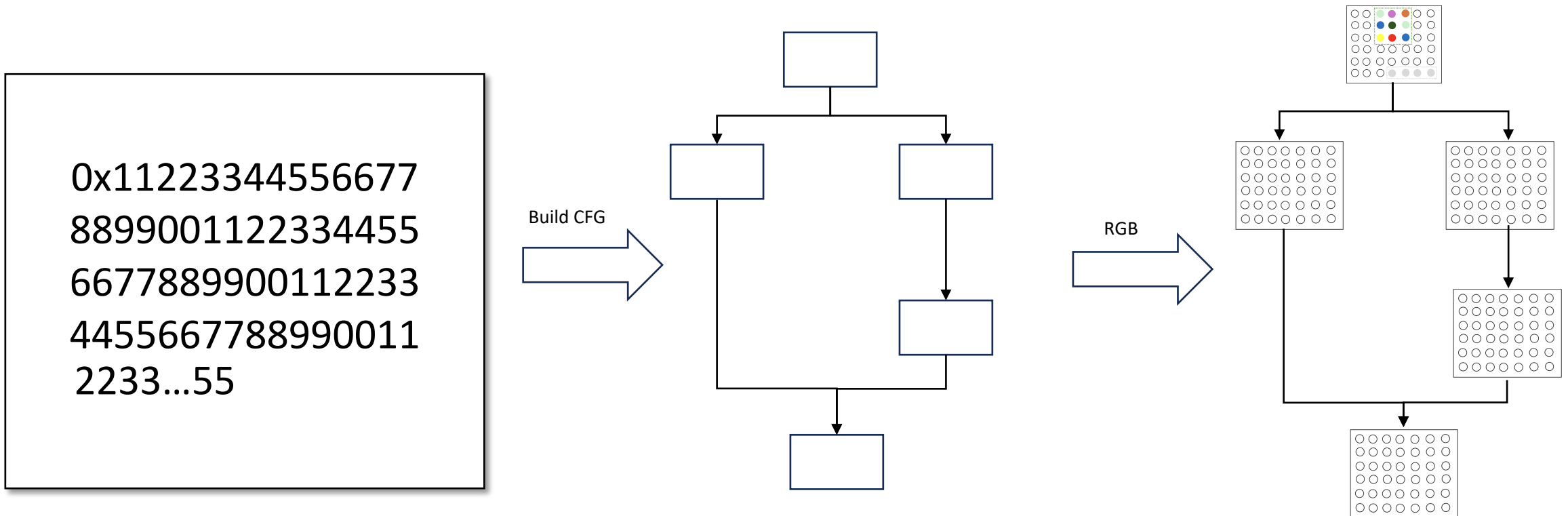
Bytecode Analysis - Assembling Overall Bytecode

- 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.



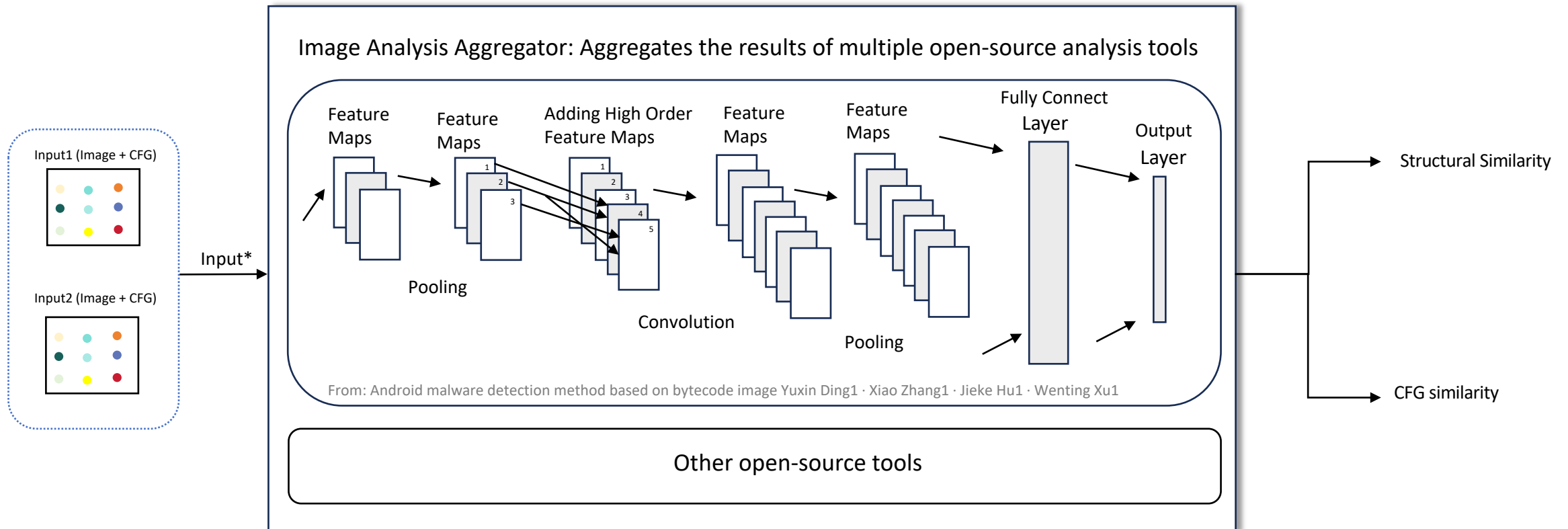
Bytecode Analysis - CFG Builder

- 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.



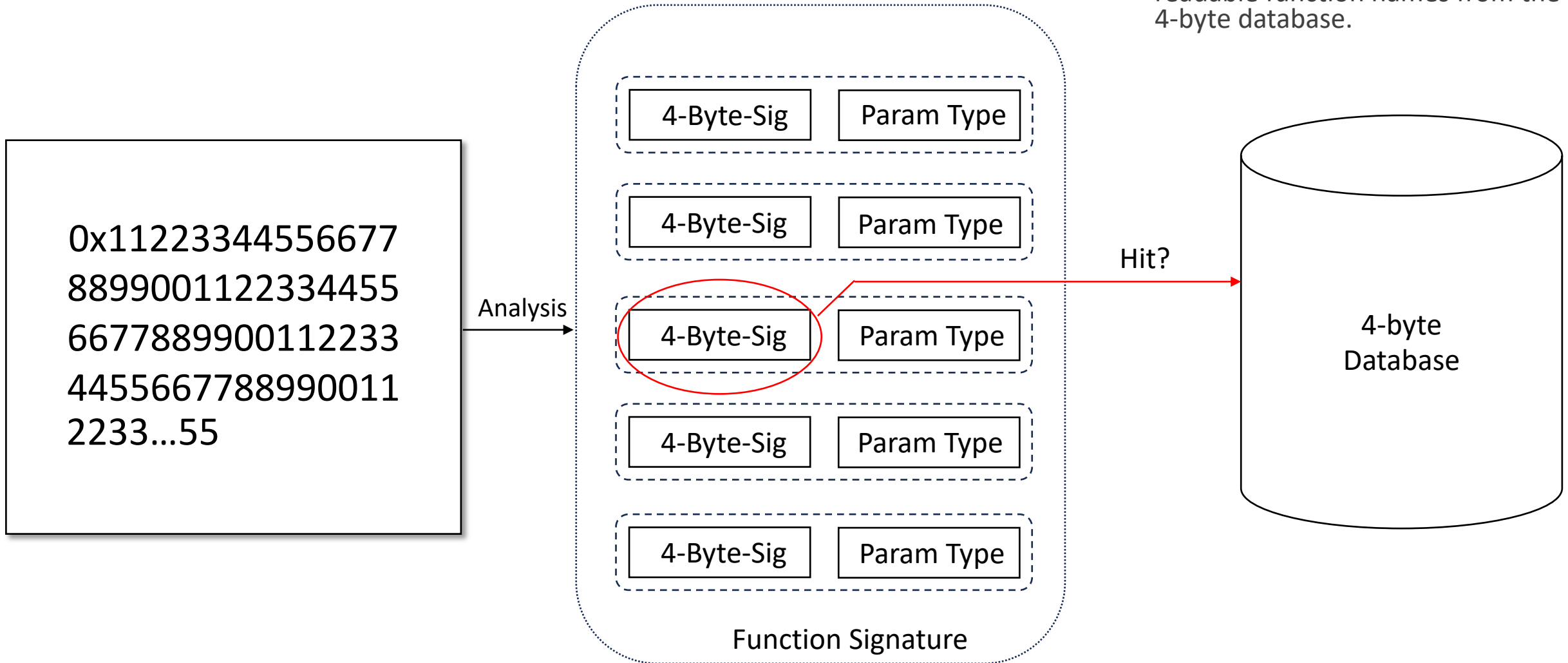
Bytecode Analysis - Image Analyzer

- 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.



Front-End

Current blockchain, wallet address, website language switch



Switch mode

Switch blockchain

Enter contract address

Front-End

Copy CFG result details

Similarity results

平均: 96%

• CFG 相似度比较 ?

▶ 显示 CFG 结果

0390f3	0390f3
Node-6 100%	Node-6 100%
80633fb5c1cb1461007757	80633fb5c1cb1461007757
Node-7 86%	Node-7 86%
5b610091600480360381019061008c91906101fa565b5f6020828403 121561020f57	5b610091600480360381019061008c9190610201565b5f6020828403 121561021657
Node-8 99%	Node-8 99%
5b5f61021c848285016101e6565b5f813590506101f4816101d0565b 6101d98161019b565b5f819050919050565b81146101e357	5b5f610223848285016101ed565b5f813590506101fb816101d7565b 6101e0816101a2565b5f819050919050565b81146101ea57
Node-9 100%	Node-9 100%
5f5ffd	5f5ffd
Node-10 57%	Node-10 57%
5b50565b92915050565b91505092915050565b6100df565b805f8190	5b50565b92915050565b91505092915050565b6100df565b80600181

Detailed Information

Opcode 1

```
000000 JUMPDEST
000001 PUSH2 0091
000004 PUSH1 04
000006 DUP1
000007 CALLDATASIZE
000008 SUB
000009 DUP2
00000a ADD
00000b SWAP1
00000c PUSH2 008c
00000f SWAP2
000010 SWAP1
000011 PUSH2 01fa
000014 JUMP
000015 JUMPDEST
000016 PUSH0
```

Opcode 2

```
000000 JUMPDEST
000001 PUSH2 0091
000004 PUSH1 04
000006 DUP1
000007 CALLDATASIZE
000008 SUB
000009 DUP2
00000a ADD
00000b SWAP1
00000c PUSH2 008c
00000f SWAP2
000010 SWAP1
000011 PUSH2 0201
000014 JUMP
000015 JUMPDEST
000016 PUSH0
```

Close

Detailed opcode comparison
for each pair of Blocks

Horizontal comparison of each
pair of blocks in CFG

Front-End

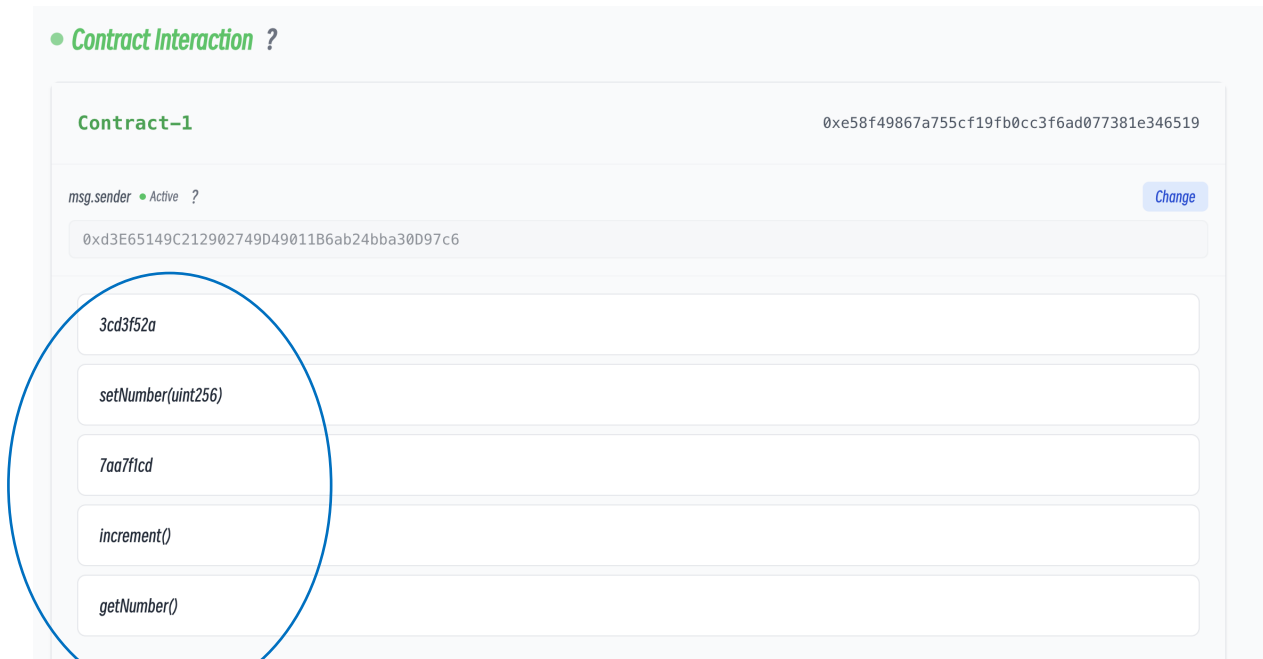
• 反汇编

The image shows a disassembler interface with two side-by-side windows, '字节码_1.hex' and '字节码_2.hex'. Both windows display a list of instructions with their addresses. A red box highlights the 'POP' instruction at address 000010 in both windows. A red arrow points from the text 'Compare each opcode horizontally' to the highlighted area.

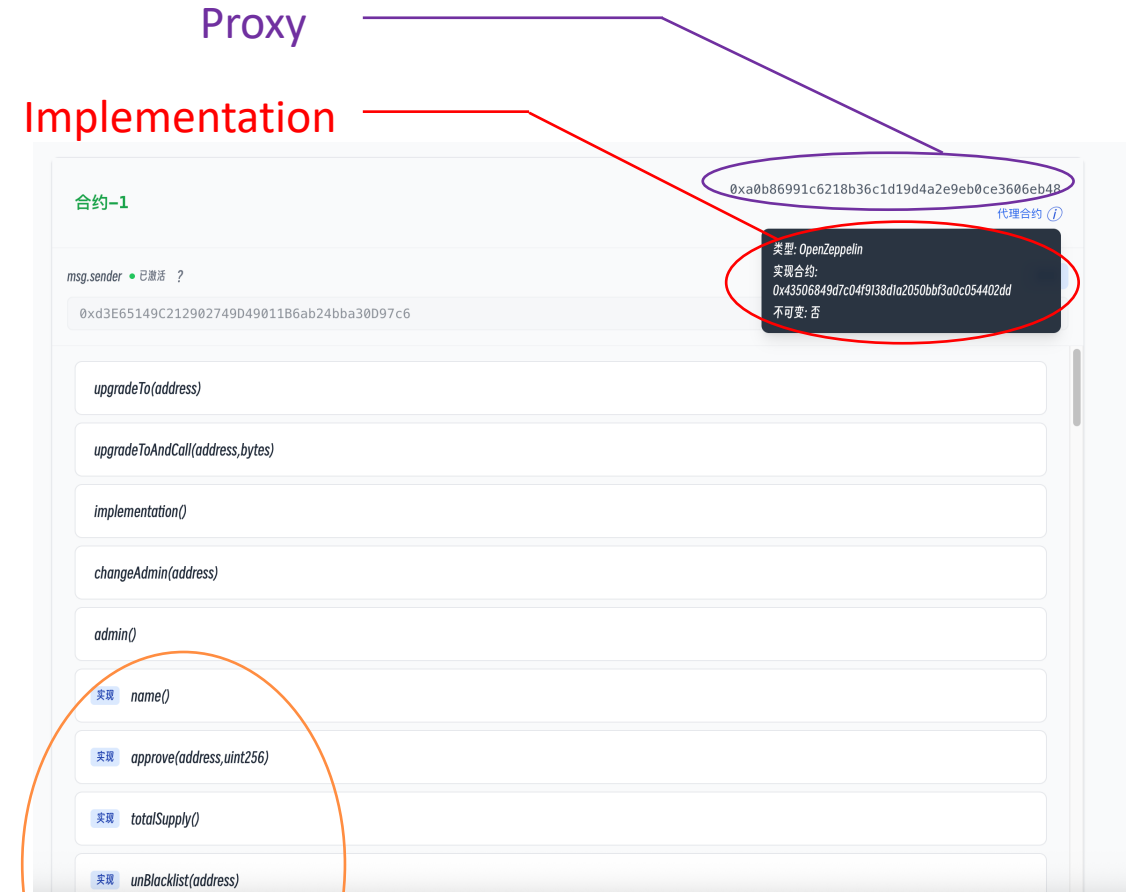
Address	Instruction (字节码_1.hex)	Instruction (字节码_2.hex)
000000	PUSH1 80	PUSH1 80
000002	PUSH1 40	PUSH1 40
000004	MSTORE	MSTORE
000005	CALLVALUE	CALLVALUE
000006	DUP1	DUP1
000007	ISZERO	ISZERO
000008	PUSH2 000f	PUSH2 000f
00000b	JUMPI	JUMPI
00000c	PUSH0	PUSH0
00000d	PUSH0	PUSH0
00000e	REVERT	REVERT
00000f	JUMPDEST	JUMPDEST
000010	POP	POP
000011	PUSH1 04	PUSH1 04
000013	CALLDATASIZE	CALLDATASIZE
000014	LT	LT
000015	PUSH2 0055	PUSH2 0055
000018	JUMPI	JUMPI
000019	PUSH0	PUSH0
00001a	CALLDATALOAD	CALLDATALOAD
00001b	PUSH1 e0	PUSH1 e0
00001d	SHR	SHR
00001e	DUP1	DUP1
00001f	PUSH4 3cd3f52a	PUSH4 3cd3f52a
000024	EQ	EQ
000025	PUSH2 0059	PUSH2 0059
000028	JUMPI	JUMPI
000029	DUP1	DUP1
00002a	PUSH4 3fb5c1cb	PUSH4 3fb5c1cb
00002f	EQ	EQ
000030	PUSH2 0077	PUSH2 0077
000033	JUMPI	JUMPI
000034	DUP1	DUP1

Compare each opcode horizontally

Front-End



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



Implementation - Functions

Front-End

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

7aa7f1cd

● arg0 (uint256)

0xff

查看

模拟

写入

{

"type": "Execute",

"txHash": "0xa52814376d4d24e7623fb215c2fd8dc566b4abb63d67aa752ed4dc5e94460e95",

"receipt": {

"_type": "TransactionReceipt",

"blockHash": "0xc4913b2c12c863768a353c4f223bcc778e0137c46ff56067ee5cd726b6bc0bf9",

"blockNumber": 8137038,

"contractAddress": null,

"cumulativeGasUsed": "4159731",

"from": "0xd3E65149C212902749D49011B6ab24bba30D97c6",

}

}

Call the function 3cd3f52a()

3cd3f52a

查看

模拟

写入

[

{

"paramType": "address",

"decodedResult": "0x00ff"

},

{

"paramType": "int256",

"decodedResult": "0xff"

},

{

"paramType": "uint256",

}

]

Check the function 3cd3f52a()

3cd3f52a

查看

模拟

写入

Traces:

[2409] 0x99C0645CdF652Df9052E68050dEC5546416Eb3df::3cd3f52a()

└─ [Return] 0x00ff

Transaction successfully executed.

Gas used: 23473

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