

Panel: Source Code Verification



Giacomo Barbieri – Routescan

Kaan Uzdoğan – Sourcify

Kirill Fedoseev – Blockscout

Moderator: Gary Thung – Codeslaw

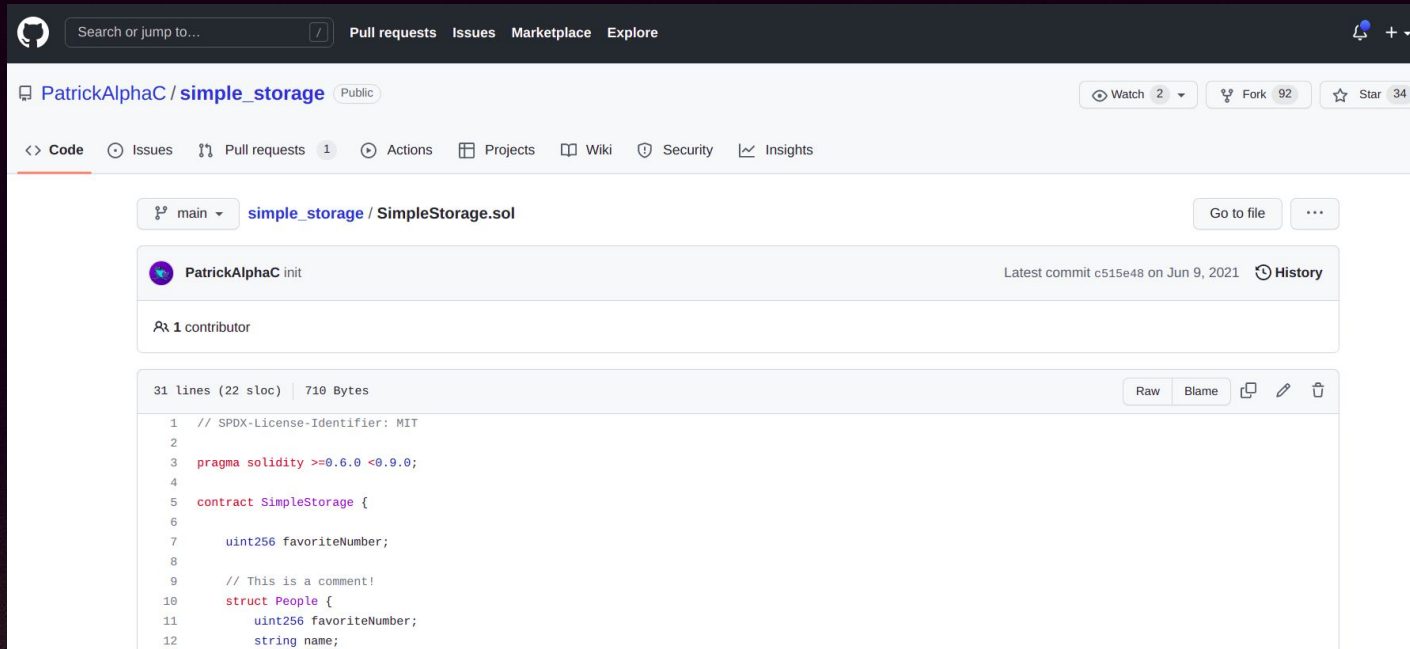
What is “source code verification”

Source Code Verification

- Smart contract code lives in bytecode on the blockchain
- Not human-readable, machine readable

Source Code Verification

- Smart contract code lives in bytecode on the blockchain
- Not human-readable, machine readable

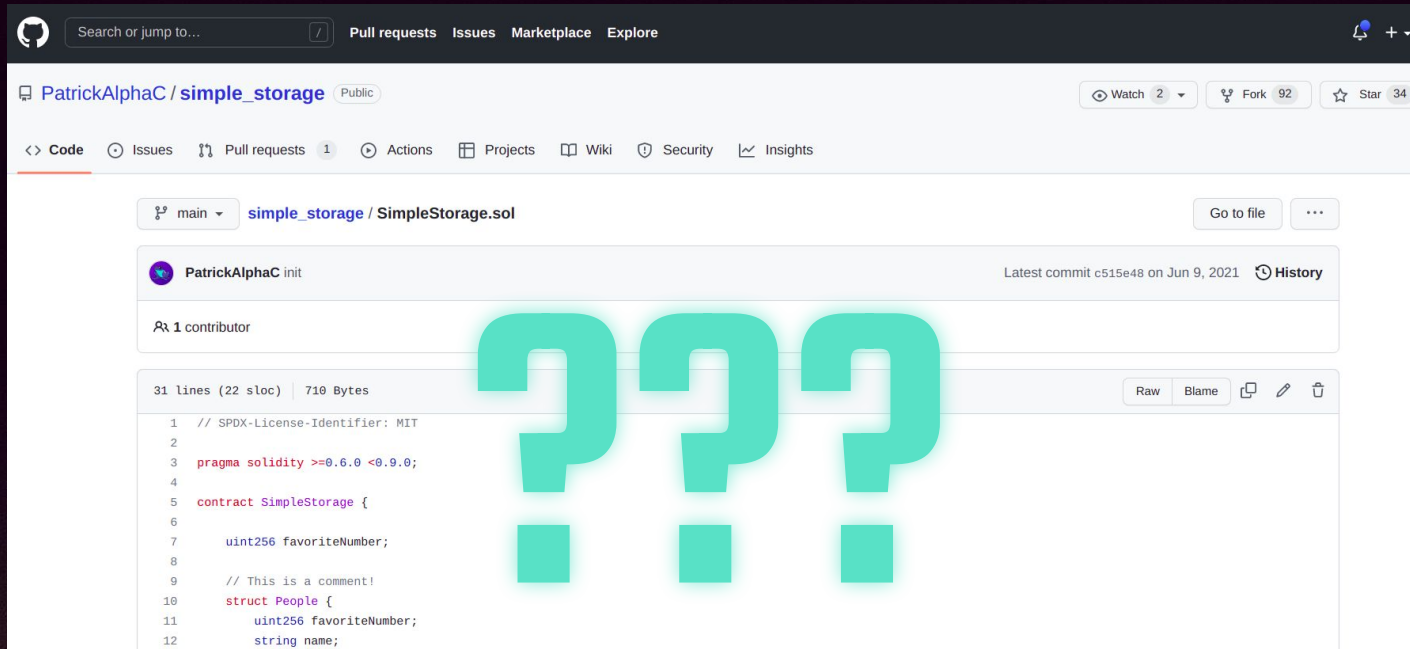


The screenshot shows a GitHub repository page for 'PatrickAlphaC / simple_storage'. The repository is public and has 2 watchers, 92 forks, and 34 stars. The main branch is 'main'. The file 'SimpleStorage.sol' is selected, showing 31 lines of Solidity code (22 sloc) and 710 bytes. The code is licensed under MIT. The code includes a pragma statement for Solidity version, a contract definition for 'SimpleStorage', and a struct for 'People'.

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity >=0.6.0 <0.9.0;
4
5 contract SimpleStorage {
6
7     uint256 favoriteNumber;
8
9     // This is a comment!
10    struct People {
11        uint256 favoriteNumber;
12        string name;
```

Source Code Verification

- Smart contract code lives in bytecode on the blockchain
- Not human-readable, machine readable




The screenshot shows a GitHub repository for 'PatrickAlphaC / simple_storage'. The file 'SimpleStorage.sol' is open, showing Solidity code. A large red '???' is overlaid on the code, indicating a lack of understanding or verification. The code is as follows:

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity >=0.6.0 <0.9.0;
4
5 contract SimpleStorage {
6
7     uint256 favoriteNumber;
8
9     // This is a comment!
10    struct People {
11        uint256 favoriteNumber;
12        string name;
```

Source Code Verification

 MyContract.sol


 Ownable.sol

 ERC20.sol

...

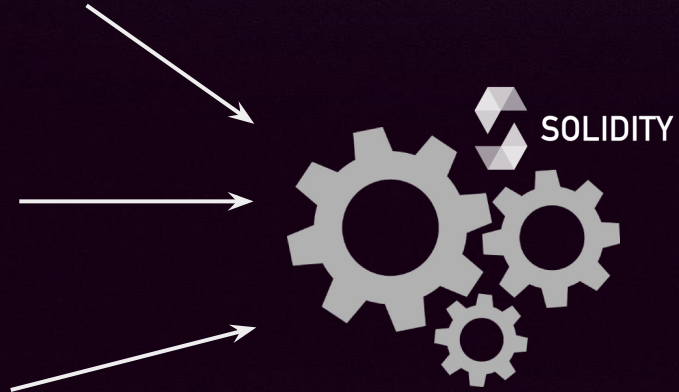
Source Code Verification

 MyContract.sol

 Ownable.sol


 ERC20.sol

...



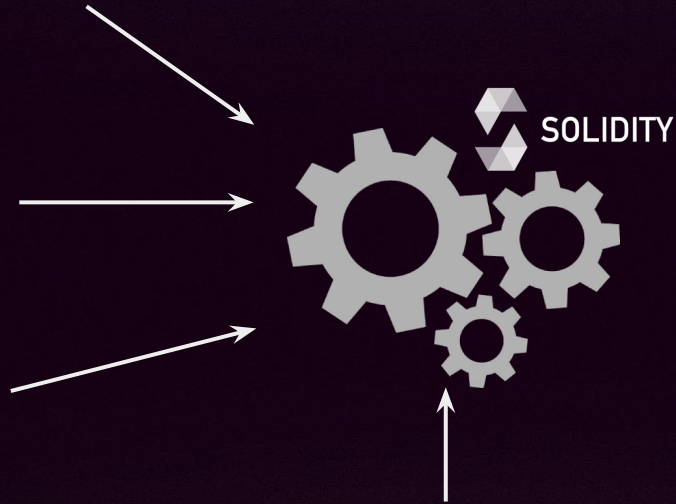
Source Code Verification

 MyContract.sol

 Ownable.sol

 ERC20.sol

...




Compilation Settings

```
· version: "0.8.7+commit.e28d00a7",  
· optimizer: {  
·   enabled: true,  
·   runs: 200  
· },  
· ...
```

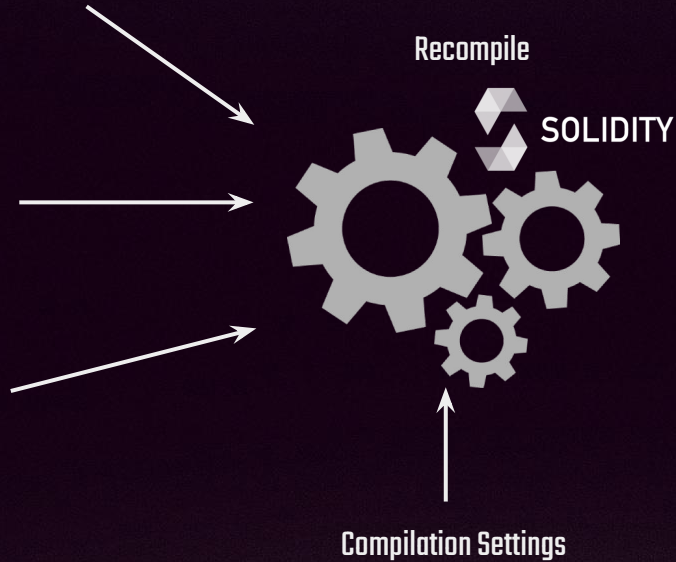

Source Code Verification

 MyContract.sol

 Ownable.sol

 ERC20.sol


...



```
· version: "0.8.7+commit.e28d00a7",  
· optimizer: {  
·   enabled: true,  
·   runs: 200  
· },  
· ...
```

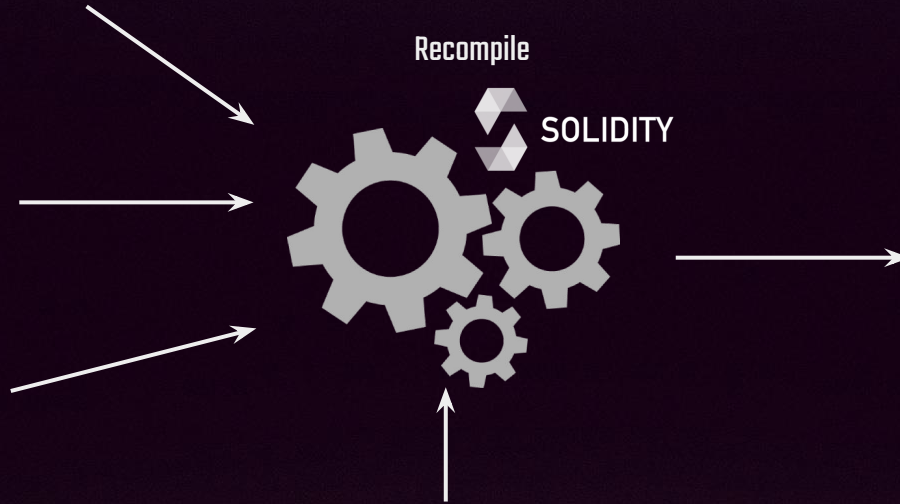
Source Code Verification

 MyContract.sol

 Ownable.sol

 ERC20.sol

...



Compilation Settings

```
· version: "0.8.7+commit.e28d00a7",  
· optimizer: {  
·   · enabled: true,  
·   · runs: 200  
· },  
· ...
```

recompiled bytecode

```
0x60806040526004361062  
00010b5760003560e01c80  
6379ba5097116200009b57  
8063addd50991162000069  
578063addd509914620002  
d1578063d0e30db0146200  
0113578063d4ee1d901462  
000305578063dce0b4e414  
62000327578063f2...
```

Source Code Verification

le

SOLIDITY



recompiled bytecode

```
0x60806040526004361062
00010b5760003560e01c80
6379ba5097116200009b57
8063addd50991162000069
578063addd509914620002
d1578063d0e30db0146200
0113578063d4ee1d901462
000305578063dce0b4e414
62000327578063f2...
```


Source Code Verification



recompiled bytecode

```
0x60806040526004361062
00010b5760003560e01c80
6379ba5097116200009b57
8063addd50991162000069
578063addd509914620002
d1578063d0e30db0146200
0113578063d4ee1d901462
000305578063dce0b4e414
62000327578063f2...
```

`eth_getCode("0x01fDBf...64cc90BB26D0C")`

le

SOLIDITY



Source Code Verification



recompiled bytecode

```
0x60806040526004361062
00010b5760003560e01c80
6379ba5097116200009b57
8063addd50991162000069
578063addd509914620002
d1578063d0e30db0146200
0113578063d4ee1d901462
000305578063dce0b4e414
62000327578063f2...
```

`eth_getCode("0x01f0Bf...64cc90BB26D0C")`



on-chain bytecode

```
0x60806040526004361062
00010b5760003560e01c80
6379ba5097116200009b57
8063addd50991162000069
578063addd509914620002
d1578063d0e30db0146200
0113578063d4ee1d901462
000305578063dce0b4e414
62000327578063f2...
```

le

SOLIDITY



Source Code Verification



recompiled bytecode

```
0x60806040526004361062
00010b5760003560e01c80
6379ba5097116200009b57
8063addd50991162000069
578063addd509914620002
d1578063d0e30db0146200
0113578063d4ee1d901462
000305578063dce0b4e414
62000327578063f2...
```

`eth_getCode("0x01f0Bf...64cc90BB26D0C")`

on-chain bytecode

```
0x60806040526004361062
00010b5760003560e01c80
6379ba5097116200009b57
8063addd50991162000069
578063addd509914620002
d1578063d0e30db0146200
0113578063d4ee1d901462
000305578063dce0b4e414
62000327578063f2...
```

match?

SOLIDITY

le



verifieralliance.org



Code

Issues 1

Pull requests

Actions

Projects

Security

Insights

Settings



database-specs

Public

Edit Pins

Watch 2

Fork 1

master

12 Branches 0 Tags

Go to file

t

Add file

<> Code

About

No description provided.

Readme

Activity

Custom

12 stars

2 watching

1 fork

Report repository

Releases

No releases published

[Create a new release](#)

Packages

No packages published



kuzdogan

Merge pull request #18 from verifier-alliance/rimrahimov/refactor



04e01f8 · 2 months ago

69 Commits



.github/workflows

refactor: add new lines

2 months ago



json-schemas

Update library identifiers to be fully qualified names. Also ...

4 months ago



tests

refactor: rename compilation_artifacts, creation_code_art...

2 months ago



.gitignore

refactor: add new lines

2 months ago



README.md

initial commit

last year



database.sql

refactor: rename compilation_artifacts, creation_code_art...

2 months ago



requirements.txt

refactor: add new lines

2 months ago



README



database-specs

verified_contracts ▾	
id ▾	BIGSERIAL NN
created_at ▾	timestampz NN
updated_at ▾	timestampz NN
created_by ▾	varchar NN
updated_by ▾	varchar NN
deployment_id	uuid NN
compilation_id	uuid NN
creation_match	bool NN
creation_values	jsonb
creation_transformations	jsonb
creation_metadata_match	bool
runtime_match	bool NN
runtime_values	jsonb
runtime_transformations	jsonb
runtime_metadata_match	bool

compiled_contracts ▾	
id ▾	uuid NN
created_at ▾	timestampz NN
updated_at ▾	timestampz NN
created_by ▾	varchar NN
updated_by ▾	varchar NN
compiler	VARCHAR NN
version	VARCHAR NN
language	VARCHAR NN
name	VARCHAR NN
fully_qualified_name	VARCHAR NN
compiler_settings	jsonb NN
compilation_artifacts	jsonb NN
creation_code_hash	bytea NN
creation_code_artifacts	jsonb NN
runtime_code_hash	bytea NN
runtime_code_artifacts	jsonb NN

compiled_contracts_sources ▾	
id ▾	uuid NN
compilation_id	uuid NN
source_hash	bytea NN
path	varchar NN

sources	
source_hash ▾	bytea NN
source_hash_keccak	bytea NN
content	varchar NN
created_at ▾	timestampz NN
updated_at ▾	timestampz NN
created_by ▾	varchar NN
updated_by ▾	varchar NN

contract_deployments ▾	
id ▾	uuid NN
created_at ▾	timestampz NN
updated_at ▾	timestampz NN
created_by ▾	varchar NN
updated_by ▾	varchar NN
chain_id	numeric NN
address	bytea NN
transaction_hash	bytea NN
block_number	numeric NN
transaction_index	numeric NN
deployer	bytea NN
contract_id	uuid NN

contracts ▾	
id ▾	uuid NN
created_at ▾	timestampz NN
updated_at ▾	timestampz NN
created_by ▾	varchar NN
updated_by ▾	varchar NN
creation_code_hash	bytea NN
runtime_code_hash	bytea NN

code ▾	
code_hash ▾	bytea NN
created_at ▾	timestampz NN
updated_at ▾	timestampz NN
created_by ▾	varchar NN
updated_by ▾	varchar NN
code_hash_keccak	bytea NN
code	bytea



Sourcify Docs

Running Sourcify

Verification

Contract Repository

Sourcify Database

File Repositories

repo.sourcify.dev

API

Monitoring Service

Contract Metadata

Migration from Filesystem to Database

Requesting Chain Support

Supported Chains

Glossary

F.A.Q.

Additional Resources

Download

We dump the whole database daily in [Parquet](#) format and upload it to a Cloudflare R2 storage. You can access the manifest file at <https://export.sourcify.dev> ([.dev](#) redirects to [.app](#) domain, which also belongs to Sourcify). The script that does the dump is at [sourcifyeth/parquet-export](#).

[export.sourcify.dev](#) will redirect to a `manifest.json` file:

▶ `manifest.json`

You can download all the files and use a [parquet](#) client to query, inspect, or process the data.

1. Download the manifest file (`-L` to follow redirects):

```
curl -L -O https://export.sourcify.dev/manifest.json
```

2. Download all the tables listed in the manifest:

```
jq -r '.files | keys[] as $k | .[$k[]]' manifest.json | xargs -I {} curl -L -O https://e
```

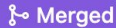
For example you can install the `parquet-cli` to do basic inspection:

```
brew install parquet-cli  
  
parquet meta compiled_contracts_0_5000.parquet
```



Remix - Unified Verification Plugin

Contract verification plugin #5221



Merged

Aniket-Engg merged 100 commits into `ethereum:master` from `sourcifyeth:contract-verification-plugin` last month



Conversation 2



Commits 100



Checks 1



Files changed 61



manuelwedler commented on Sep 25 · edited

Member



This PR adds a generalized contract verification plugin. Contracts written / compiled in Remix can be verified at Sourcify, Etherscan and Blockscout at the same time. It basically adds all the features the separate Etherscan and Sourcify plugins had and makes them obsolete.

Features:

- Verify contracts and their proxies on Sourcify, Etherscan and Blockscout by providing the source code to their APIs
- Check the verification status via receipts that are stored in local storage
- Lookup up the source code of any address and add it to Remix, if the respective contract is verified on one of the three verifiers

The plugin isn't enabled by default yet, as I wasn't sure how to do it.

We had to make changes to the `compiler-artefacts.ts` to be able to get the compiler input JSON of contracts. Some more info about this here: [sourcifyeth#1](#)



CONTRACT VERIFICATION



Verify Receipts Lookup Settings



Verify compiled contracts on different verification services



Chain

Ethereum Testnet Holesky (17000)



Contract Address

0x2738d13E81e30bC615766A0410e7



Contract Name

Storage - contracts/1_Storage.sol



☐ The deployed contract is behind a proxy

Verify on:

☒ **Sourcify**

<https://sourcify.dev/server>

☒ **Etherscan**

<https://api-holesky.etherscan.io>

☒ **Blockscout**

<https://eth-holesky.blockscout.com>

☒ **Routescan**

<https://api.routescan.io/v2/network/tes...>

Verify



// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

/**

* @title Storage

* @dev Store & retrieve value in a variable

* @custom:dev-run-script ./scripts/deploy_with_ethers.ts

*/

contract Storage

uint256 number;

/**

* @dev Store value in variable

* @param num value to store

*/

function store(uint256 num) public { 22514 gas

number = num;

}

/**

* @dev Return value

* @return value of 'number'

*/

function retrieve() public view returns (uint256){ 2410 gas

return number;

}



• ethers.js
• sol-gpt <your Solidity question here>

Type the library name to see available commands.

0



Listen on all transaction

Creation Bytecode vs Runtime Bytecode

- In the compiler output:
 - "bytecode" (=creation bytecode)
 - "deployedBytecode" (=runtime bytecode)

```
{
  "contracts": {
    "Storage.sol": {
      "Storage": {
        "evm": {
          "bytecode": {
            "object": "6080604052348015600e575f80fd5b5060a580601a5f395ff3fe6080604052348015600e575f80fd5b506036057361d146048575b5f80fd5b5f5460405190815260200160405180910390f35b6057605336600f0fd5b503591905056fea26469706673582212200c503f43b526fa24028cad6d4b26f7134a6392ef515f",
            "deployedBytecode": {
              "object": "6080604052348015600e575f80fd5b50600436106030575f3560e01c80632e64cec11460345780636060405180910390f35b605760533660046059565b5f55565b005b5f602082840312156068575f80fd5b1a24028cad6d4b26f7134a6392ef515855de9da213ea641832c964736f6c634300081a0033",
            }
          }
        }
      }
    }
  }
}
```

```
{
  "contracts": {
    "Storage.sol": {
      "Storage": {
        "evm": {
          "bytecode": {
            "object": "6080604052348015600e575f80fd5b5060a580601a5f395ff3fe6080604052348015600e575f80fd5b50600436106030575f3560e01c80632e64cec11460345780636057361d146048575b5f80fd5b5f5460405190815260200160405180910390f35b605760533660046059565b5f55565b005b5f602082840312156068575f80fd5b503591905056fea26469706673582212200c503f43b526fa24028cad6d4b26f7134a6392ef515855de9da213ea641832c964736f6c634300081a0033",
          },
          "deployedBytecode": {
            "object": "6080604052348015600e575f80fd5b50600436106030575f3560e01c80632e64cec11460345780636057361d146048575b5f80fd5b5f5460405190815260200160405180910390f35b605760533660046059565b5f55565b005b5f602082840312156068575f80fd5b503591905056fea26469706673582212200c503f43b526fa24028cad6d4b26f7134a6392ef515855de9da213ea641832c964736f6c634300081a0033"
          }
        }
      }
    }
  }
}
```


Compiled Creation Code

```
0x6080604052348015600e575f80fd5b506101188061001c5f395ff3fe6080604052348015600e575f80fd5b50600436106026575f3560e01c80636057361d14602a575b5f80fd5b6039603536600460b6565b603b565b005b60405163cad0899b60e01b815260048101829052602a602482015273__$9e30dc62696a7b2364588369d98712b6df__$9063cad0899b90604401602060405180830381865af4158015608f573d5f803e3d5ffd5b505050506040513d601f19601f8201168201806040525081019060b1919060cc565b5f5550565b5f6020828403121560c5575f80fd5b5035919050565b5f6020828403121560db575f80fd5b505191905056fea264697066735822122036cbbd47009d2c233e50a23cc7be08b002c152e65efdd1c9d17a1ccd70d26cab64736f6c634300081a0033
```



match?

On-chain Creation Code

```
0x6080604052348015600e575f80fd5b506101188061001c5f395ff3fe6080604052348015600e575f80fd5b50600436106026575f3560e01c80636057361d14602a575b5f80fd5b6039603536600460b6565b603b565b005b60405163cad0899b60e01b815260048101829052602a602482015273513ddfe9cdfcc551a4088bf956477ce7d5f2887e9063cad0899b90604401602060405180830381865af4158015608f573d5f803e3d5ffd5b505050506040513d601f19601f8201168201806040525081019060b1919060cc565b5f5550565b5f6020828403121560c5575f80fd5b5035919050565b5f6020828403121560db575f80fd5b505191905056fea264697066735822122036cbbd47009d2c233e50a23cc7be08b002c152e65efdd1c9d17a1ccd70d26cab64736f6c634300081a0033
```

Compiled Creation Code

```
0x6080604052348015600e575f80fd5b506101188061001c5f395ff3fe6080604052348015600e575f80fd5b50600436106026575f3560e01c80636057361d14602a575b5f80fd5b6039603536600460b6565b603b565b005b60405163cad0899b60e01b815260048101829052602a602482015273__$9e30dc62696a7b2364588369d98712b6df__$9063cad0899b90604401602060405180830381865af4158015608f573d5f803e3d5ffd5b505050506040513d601f19601f8201168201806040525081019060b1919060cc565b5f5550565b5f6020828403121560c5575f80fd5b5035919050565b5f6020828403121560db575f80fd5b505191905056fea264697066735822122036cbbd47009d2c233e50a23cc7be08b002c152e65efdd1c9d17a1ccd70d26cab64736f6c634300081a0033
```



match?

On-chain Creation Code

```
0x6080604052348015600e575f80fd5b506101188061001c5f395ff3fe6080604052348015600e575f80fd5b50600436106026575f3560e01c80636057361d14602a575b5f80fd5b6039603536600460b6565b603b565b005b60405163cad0899b60e01b815260048101829052602a602482015273513ddfe9cdfcc551a4088bf956477ce7d5f2887e9063cad0899b90604401602060405180830381865af4158015608f573d5f803e3d5ffd5b505050506040513d601f19601f8201168201806040525081019060b1919060cc565b5f5550565b5f6020828403121560c5575f80fd5b5035919050565b5f6020828403121560db575f80fd5b505191905056fea264697066735822122036cbbd47009d2c233e50a23cc7be08b002c152e65efdd1c9d17a1ccd70d26cab64736f6c634300081a0033
```

Compiled Creation Code

```
0x6080604052348015600e575f80fd5b506101188061001c5f395ff3fe6080604052348015600e575f80fd5b50600436106026575f3560e01c80636057361d14602a575b5f80fd5b6039603536600460b6565b603b565b005b60405163cad0899b60e01b815260048101829052602a602482015273__$9e30dc62696a7b2364588369d98712b6df$__9063cad0899b90604401602060405180830381865af4158015608f573d5f803e3d5ffdf5b505050506040513d601f19601f8201168201806040525081019060b1919060cc565b5f5550565b5f6020828403121560c5575f80fd5b535919050565b5f6020828403121560db575f80fd5b505191905056fea264697066735822122036cbbd47009d2c233e50a23cc7be08b002c152e65efdd1c9d17a1ccd70d26cab64736f6c634300081a0033
```

116 bytes

On-chain Creation Code

```
0x6080604052348015600e575f80fd5b506101188061001c5f395ff3fe6080604052348015600e575f80fd5b50600436106026575f3560e01c80636057361d14602a575b5f80fd5b6039603536600460b6565b603b565b005b60405163cad0899b60e01b815260048101829052602a602482015273513ddfe9cdfcc551a4088bf956477ce7d5f2887e9063cad0899b90604401602060405180830381865af4158015608f573d5f803e3d5ffdf5b505050506040513d601f19601f8201168201806040525081019060b1919060cc565b5f5550565b5f6020828403121560c5575f80fd5b5035919050565b5f6020828403121560db575f80fd5b505191905056fea264697066735822122036cbbd47009d2c233e50a23cc7be08b002c152e65efdd1c9d17a1ccd70d26cab64736f6c634300081a0033
```


Compiled Creation Code

```
0x6080604052348015600e575f80fd5b506101188061001c5f395ff3fe6080604052348015600e575f80fd5b50600436106026575f3560e01c80636057361d14602a575b5f80fd5b6039603536600460b6565b603b565b005b60405163cad0899b60e01b815260048101829052602a602482015273513ddfe9cdfcc551a4088bf956477ce7d5f2887e9063cad0899b90604401602060405180830381865af4158015608f573d5f803e3d5ffd5b505050506040513d601f19601f8201168201806040525081019060b1919060cc565b5f5550565b5f6020828403121560c5575f80fd5b5035919050565b5f6020828403121560db575f80fd5b505191905056fea264697066735822122036cbbd47009d2c233e50a23cc7be08b002c152e65efdd1c9d17a1ccd70d26cab64736f6c634300081a0033
```

On-chain Creation Code

```
0x6080604052348015600e575f80fd5b506101188061001c5f395ff3fe6080604052348015600e575f80fd5b50600436106026575f3560e01c80636057361d14602a575b5f80fd5b6039603536600460b6565b603b565b005b60405163cad0899b60e01b815260048101829052602a602482015273513ddfe9cdfcc551a4088bf956477ce7d5f2887e9063cad0899b90604401602060405180830381865af4158015608f573d5f803e3d5ffd5b505050506040513d601f19601f8201168201806040525081019060b1919060cc565b5f5550565b5f6020828403121560c5575f80fd5b5035919050565b5f6020828403121560db575f80fd5b505191905056fea264697066735822122036cbbd47009d2c233e50a23cc7be08b002c152e65efdd1c9d17a1ccd70d26cab64736f6c634300081a0033
```

Transformations

[database-specs](#) / [json-schemas](#) /

Transformations

A "transformation" on bytecode is a necessary change in the bytecode string to achieve the exact (on-chain) bytecode from a base bytecode. These changes don't effect the functionality of the bytecode. It includes:

- Constructor Arguments (only creation code)
- Immutable Variables (only runtime code)
- Libraries
- CBOR Auxdata
- Call Protection (only runtime code)

The transformations (where in the code and the reason) are stored in the `creation_` or `runtime_transformations` columns and the inserted values are stored in the `creation_` or `runtime_values` columns.

<https://github.com/verifier-alliance/database-specs/tree/master/json-schemas>

Example:

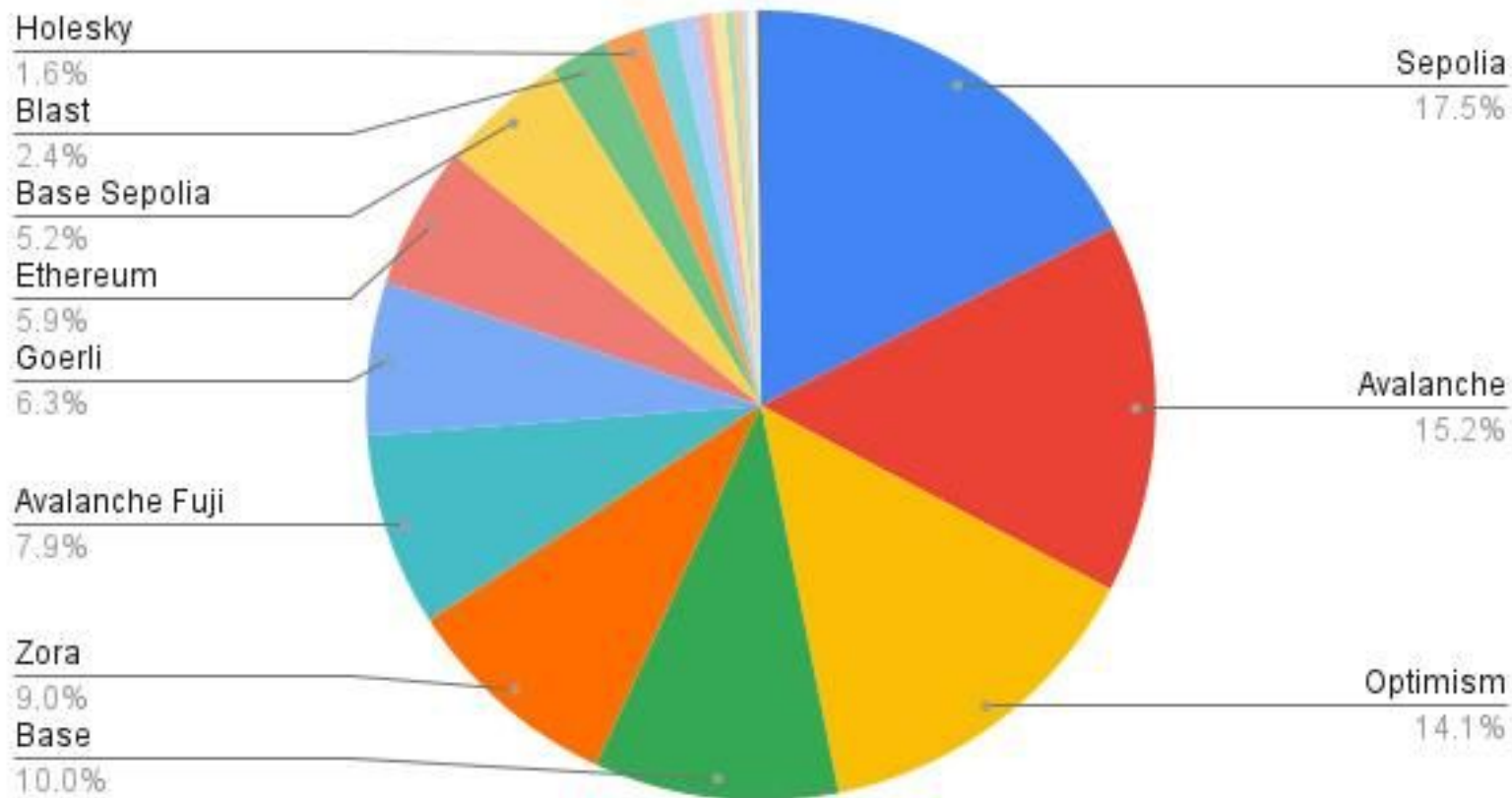
runtime_transformations :

```
[
  {
    "id": "20",
    "type": "replace",
    "offset": 137, // in bytes
    "reason": "immutable"
  },
  {
    "id": "0",
    "type": "replace",
    "offset": 1002,
    "reason": "auxdata"
  }
]
```

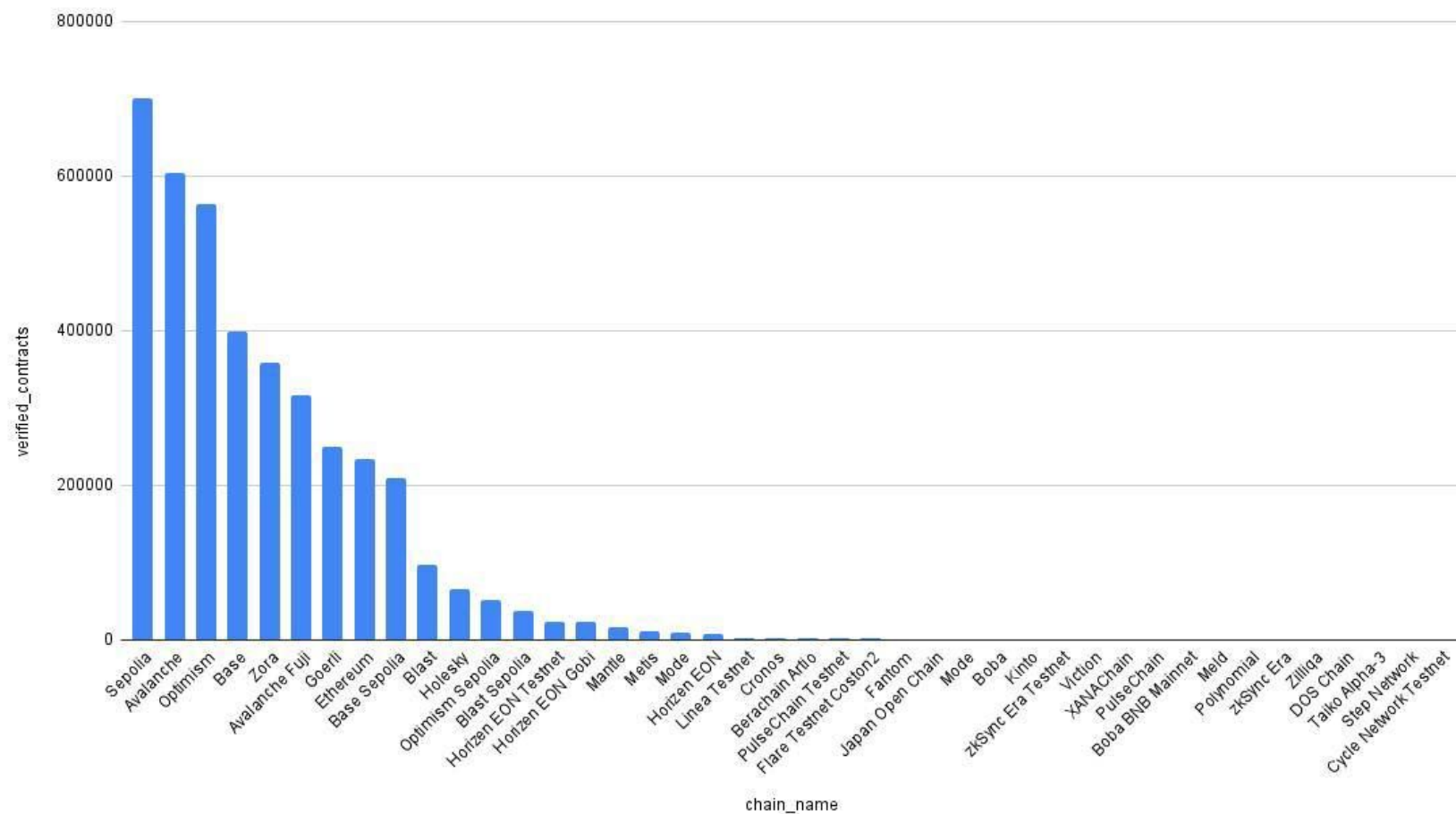
runtime_values :

```
{
  "immutables": {
    "20": "0x000000000000000000000000b5d83c2436ad54046d57cd48c00d619d702f3814"
  },
  "cborAuxdata": {
    "0": "0xa26469706673582212205817b060c918294cc8107069c4fd0a74dbfc35b0617214043887fe9d4e17a4a864736f6c634300081;"
  }
}
```

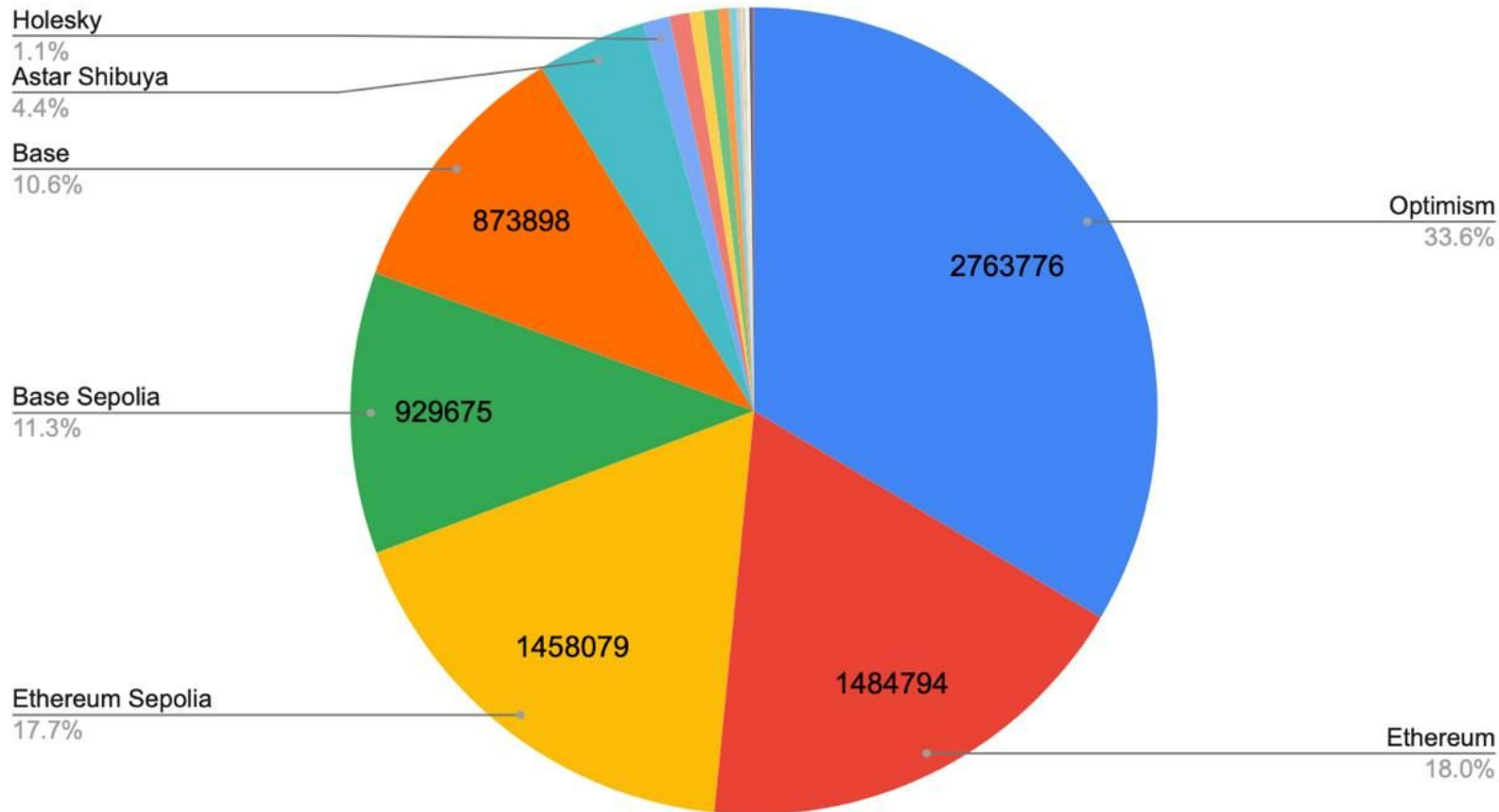

Verified Contracts vs. Network (source: Routerscan)



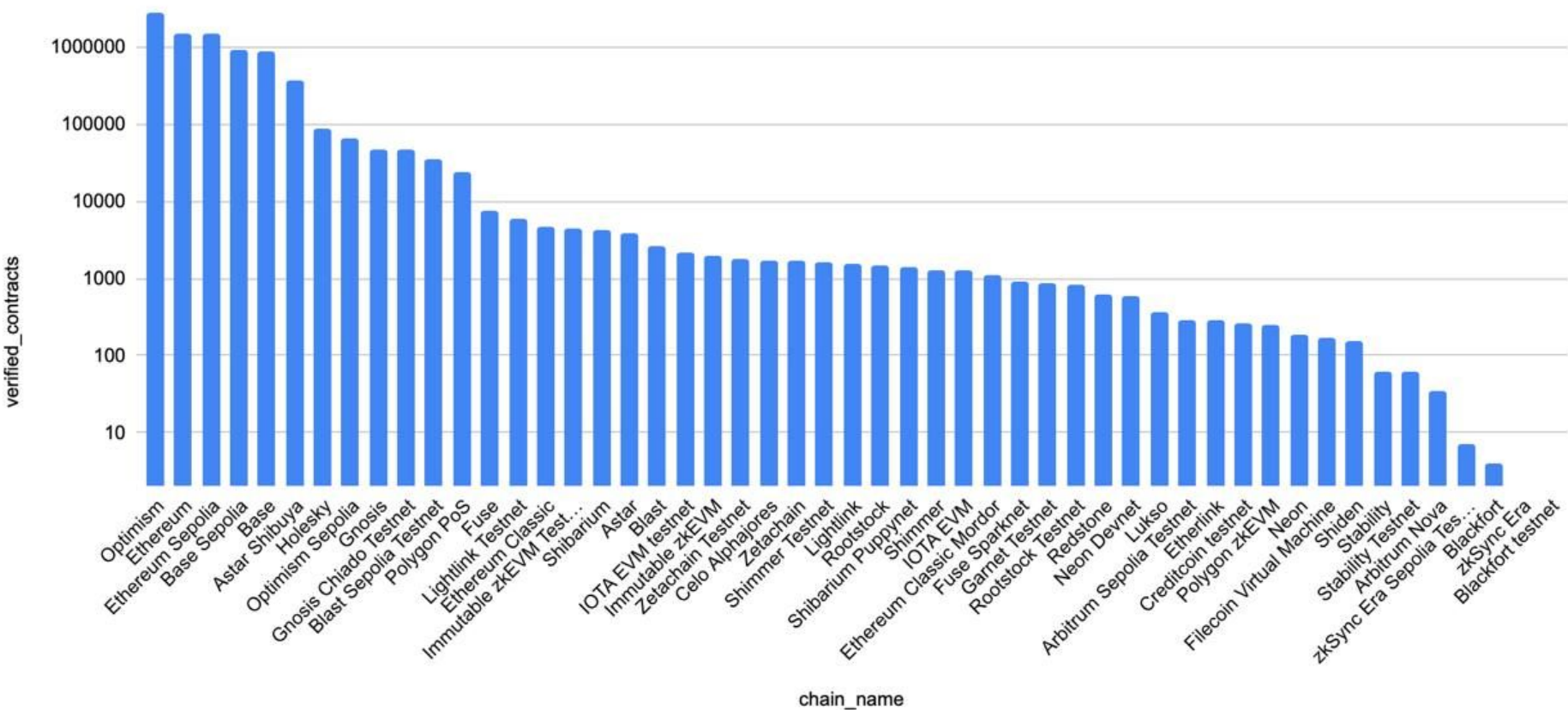
Verified Contracts vs Network (Source Routsacan)



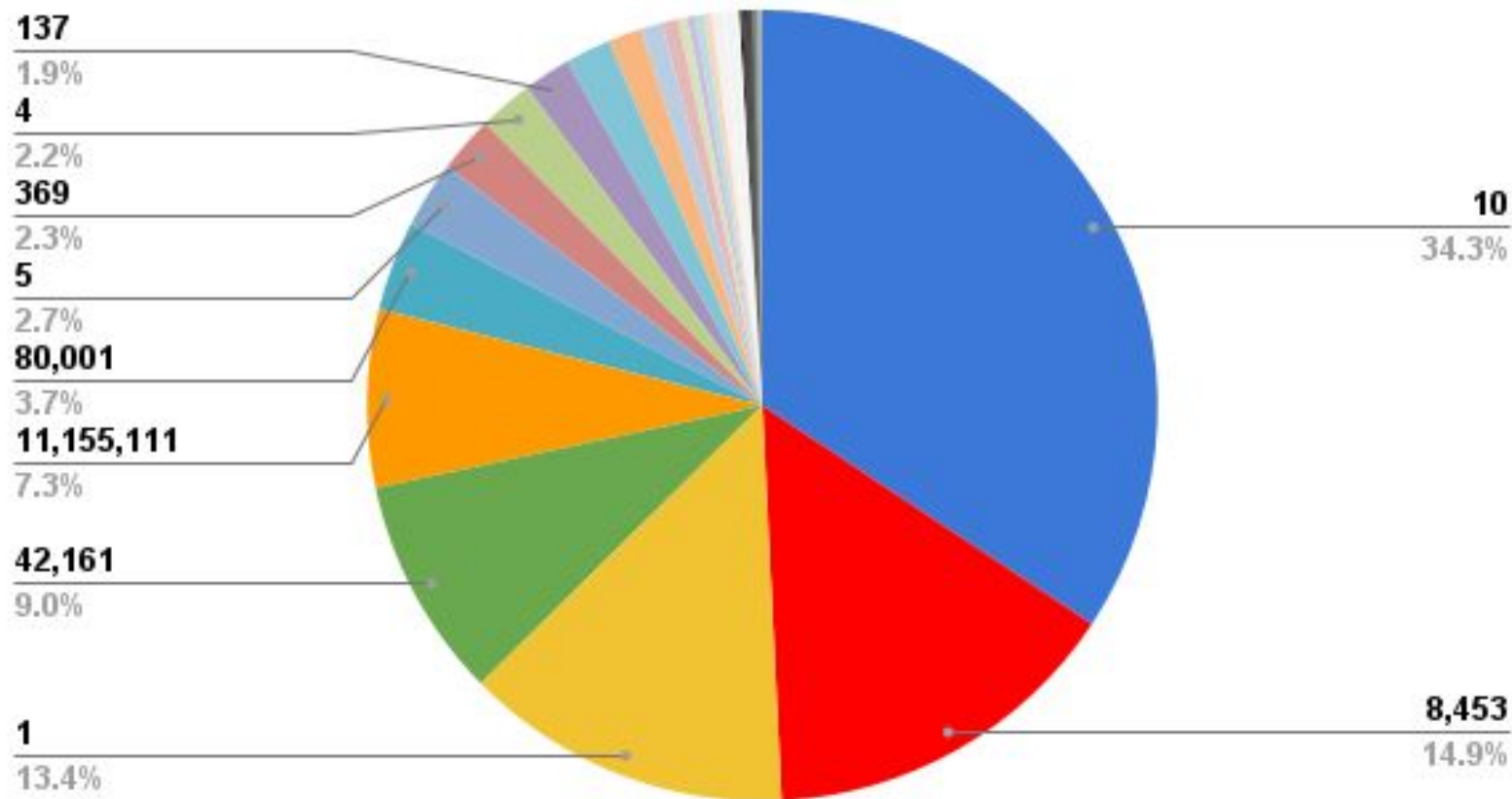
Verified Contracts vs. Network (source: Blockscout)



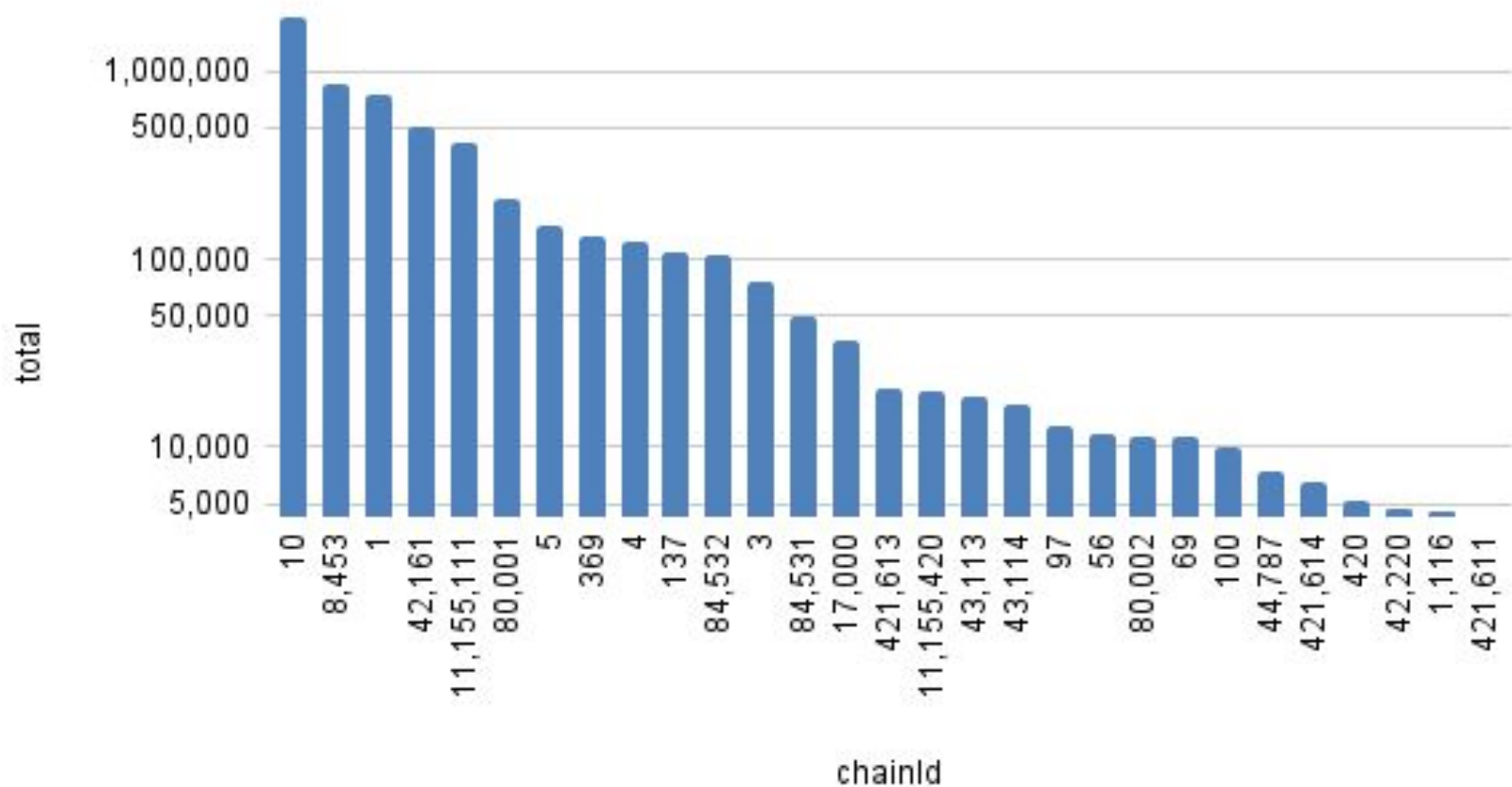
Verified Contracts vs. Network (source: Blockscout)



Verified Contracts vs. Network (source: Sourcify)



Verified Contracts vs. Network (source: Sourcify)

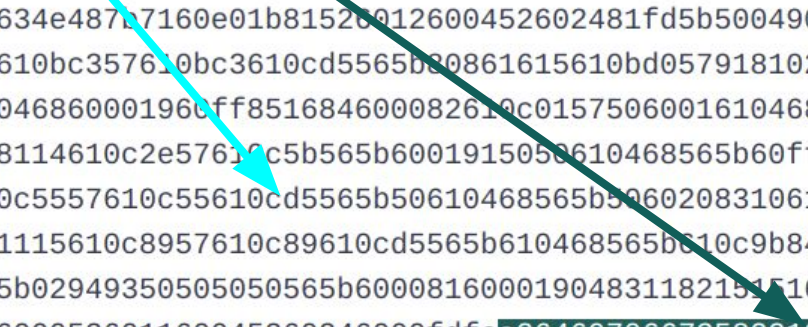


Source Code Verification

Partial match = bytecodes match

Full / Perfect Match = **bytecode** + **metadata** match (compilation fingerprint)

Contract Bytecode



```
919050565b600082610b9a57634e487b7160e01b81526012600452602481fd5b500490565b80825b6001808611610bb157
50610bdc565b818704821115610bc357610bc3610cd5565b80861615610bd057918102915b9490941c938002610ba2565b
94509492505050565b600061046860001960ff851684600082610c0157506001610468565b81610c0e5750600061046856
5b8160018114610c245760028114610c2e57610c5b565b6001915050610468565b60ff841115610c3f57610c3f610cd556
5b6001841b915084821115610c5557610c55610cd5565b50610468565b5060208310610133831016604e8410600b841016
1715610c8e575081810a83811115610c8957610c89610cd5565b610468565b6010c9b8484846001610b9f565b8086048211
15610cad57610cad610cd5565b02949350505050565b6000816000190483118215151615610cd057610cd0610cd5565b50
0290565b634e487b7160e01b600052601160045260246000fdfea264697066735822122078b530288f1cffe879bb7d9062
e904deb3aa9b9c8d27ea4ecafa987583c18a6e64736f6c63430008010033
```

15610cad57610cad610cd5565b02949350505050565b6000816000190483118215151615610cd057610cd0610cd5565b50
0290565b634e487b7160e01b600052601160045260246000fdfe**a264697066735822122078b530288f1cffe879bb7d9062**
e904deb3aa9b9c8d27ea4ecafa987583c18a6e64736f6c63430008010033

Metadata Hash (decoded)

ipfs://QmWTqspM5B1quNvdhXbS6TbXzyLZ5cUGHnTV8ZWJPqrQqj

<https://playground.sourcify.dev>