TokenScope: Automatically Detecting Inconsistent Behaviors of Cryptocurrency Tokens in Ethereum

Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security

Ting Chen[1], Yufei Zhang[1], Zihao Li[1], Xiapu Luo[2], Ting Wang[3], Rong Cao[1], Xiuzhuo Xiao[1], Xiaosong Zhang[1]

1: University of Electronic Science and Technology of China

2: The Hong Kong Polytechnic University

3: Pennsylvania State University

Background

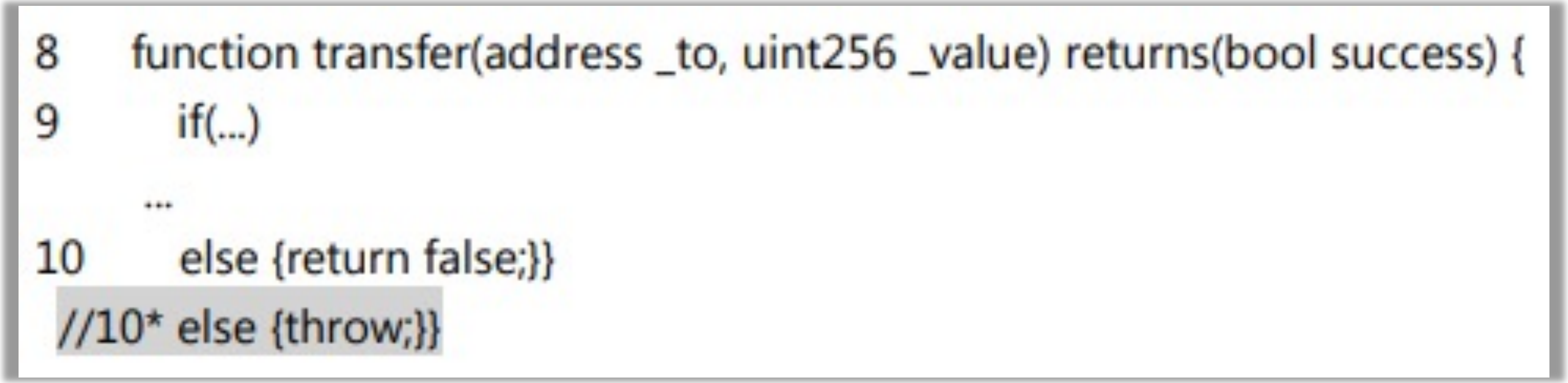
- Ethereum: a famous blockchain
  - Account
    external owned account (EOA) and smart contract account
  - Smart contract
    developed => compiled => deployed => invoked => emit events
  - Transaction
    a message sent by an account
    all historical transactions can be replayed

Background

- Cryptocurrency
  native assets ₿ Ξ and tokens 🪙
- Token
  a smart contract that records the information of token holders and their shares, and supports token activities.
- Token standard
  to regulate the interactions between token contracts and users as well as the third-party tools
- ERC20 standard
  function transferFrom(address _from, address _to, uint256 _value) public returns (bool success), *transfer()*, *event Transfer*

Problem

- Users usually employ third-party tools to manipulate tokens: wallet, exchange markets, blockchain explorers
- These tools interact with tokens through token standard
- Inconsistent!



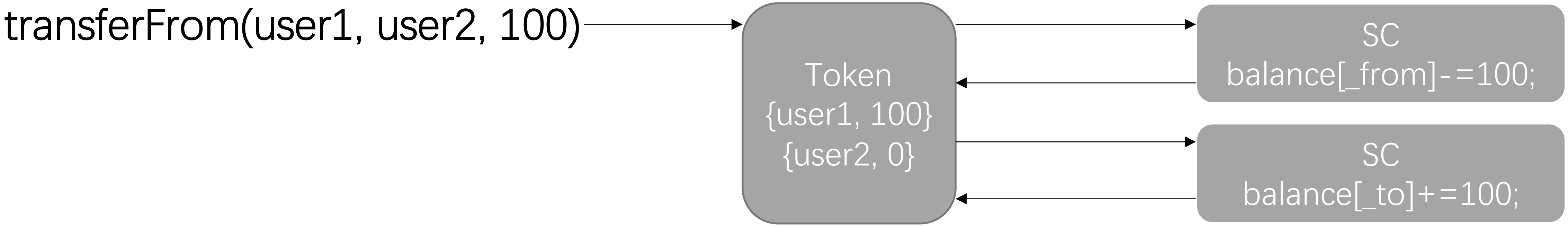
```
8    function transfer(address _to, uint256 _value) returns(bool success) {
9        if(...)
        ...
10       else {return false;}}
 //10* else {throw;}}
```

Automatically Detecting Inconsistent Behaviors of Cryptocurrency Tokens in Ethereum

- TokenScope: automatically detect the inconsistent behaviors by contrasting the information from three different sources.
  - manipulations of core data structures
  - the actions indicated by standard interfaces
  - the behaviors suggested by the standard events

Challenges

1. how to automatically identify the core data structures that store each token holder’ s identifier and balance
2. how to recognize token transfers that are triggered through inter-contract invocations.



transferFrom(user1, user2, 100)

Token
{user1, 100}
{user2, 0}

SC
balance[_from]-=100;

SC
balance[_to]+=100;

Address the challenges

1. how to automatically identify the core data structures that store each token holder’ s identifier and balance
   by exploiting how EVM accesses the data structures
2. how to recognize token transfers that are triggered through inter-contract invocations
   recover the execution traces of token contracts by node instrumentation

Workflow

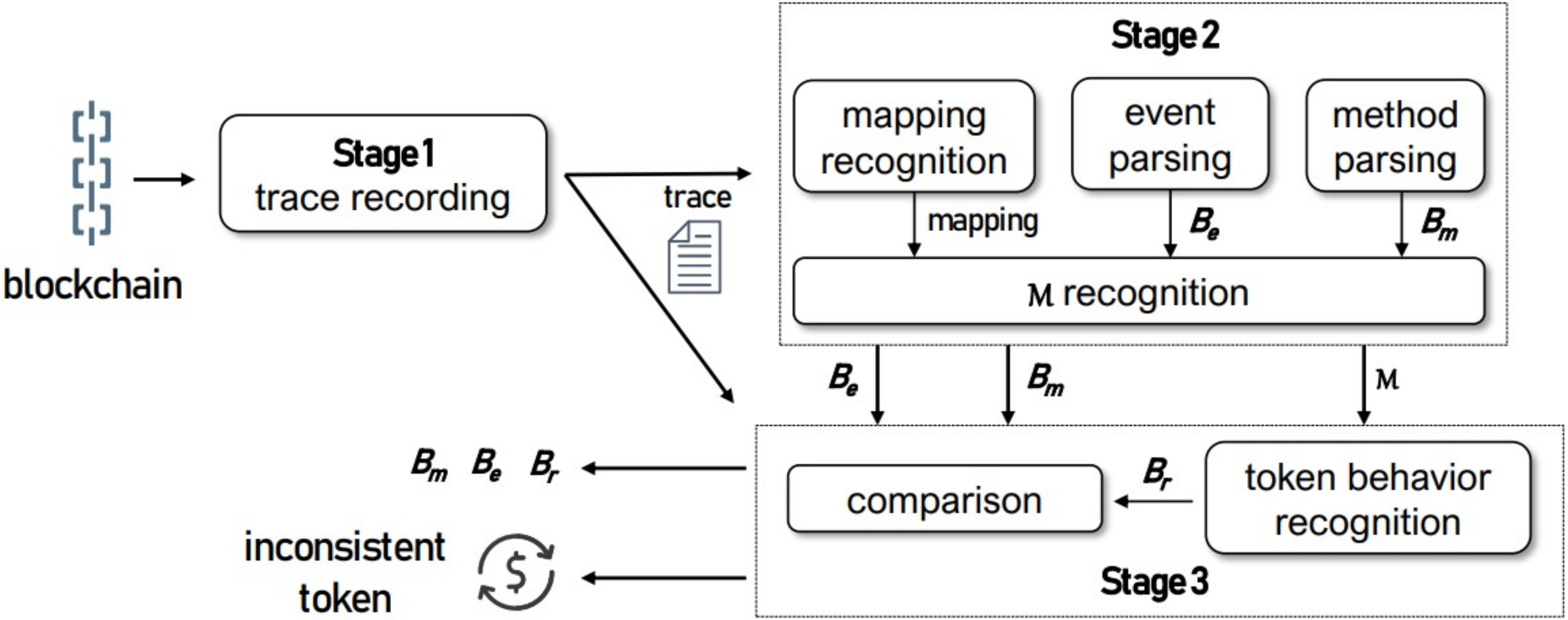
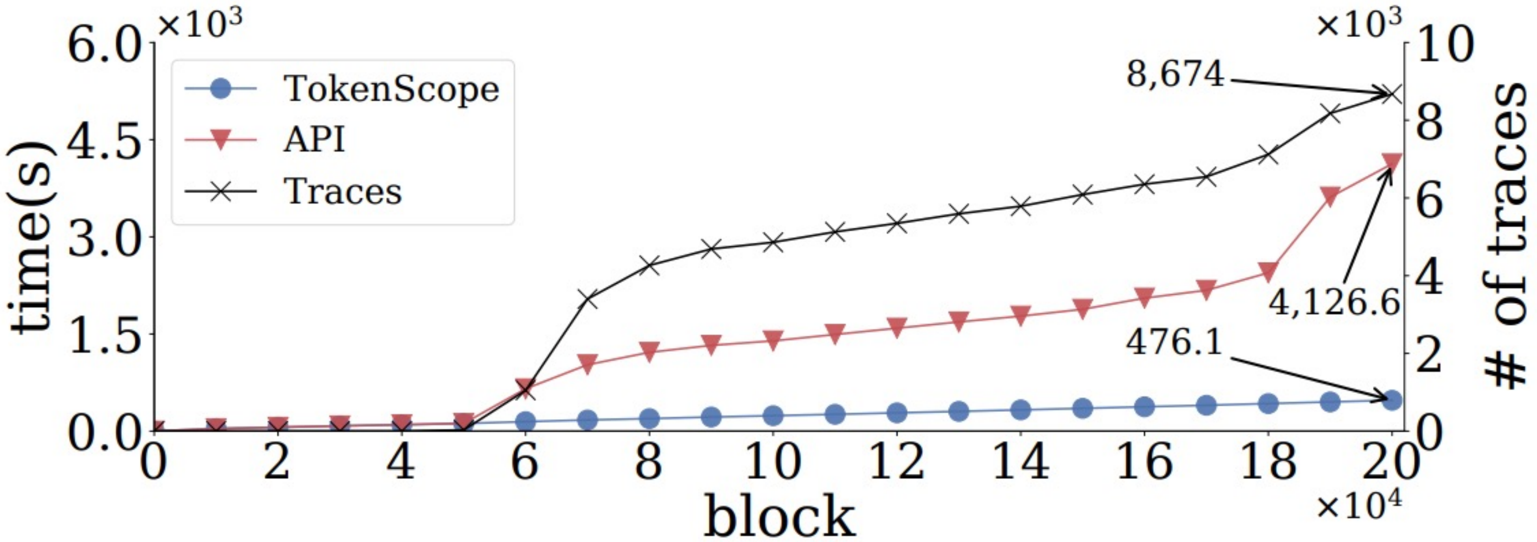


Figure 3: Architecture of TokenScope

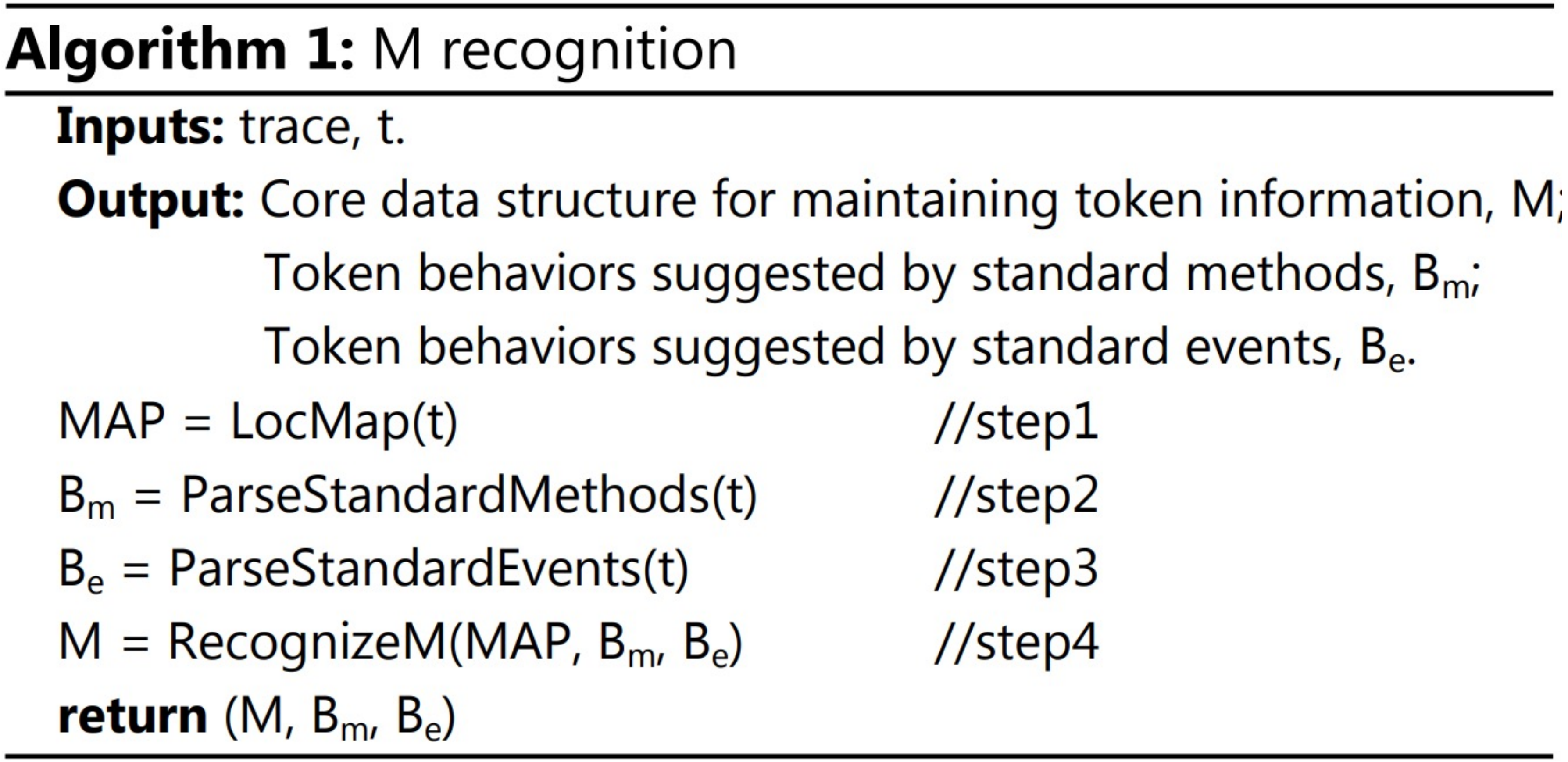
Stage 1: Trace Recording

- Trace: <TxHash, invoked smart contract, data, executed instruction>
- Existing approach: debug.traceTransaction() provided by an Ethereum node
- TokenScope: instruments an Ethereum node to record traces



Stage 2: Locating Core Data Structure

- Basic idea



Algorithm 1: M recognition

```
Inputs: trace, t.
Output: Core data structure for maintaining token information, M;
        Token behaviors suggested by standard methods, $B_m$;
        Token behaviors suggested by standard events, $B_e$.
MAP = LocMap(t)                          //step1
$B_m$ = ParseStandardMethods(t)          //step2
$B_e$ = ParseStandardEvents(t)           //step3
M = RecognizeM(MAP, $B_m$, $B_e$)        //step4
return (M, $B_m$, $B_e$)
```

Stage 2: Locating Core Data Structure

- Step 1: Locating Mapping variables
  - mapping(address => uint256);
  - only less than 1% deployed smart contracts are open-source
  - Idea: exploit how a mapping variable is stored in EVM bytecode and how a mapping variable is manipulated by EVM instructions.

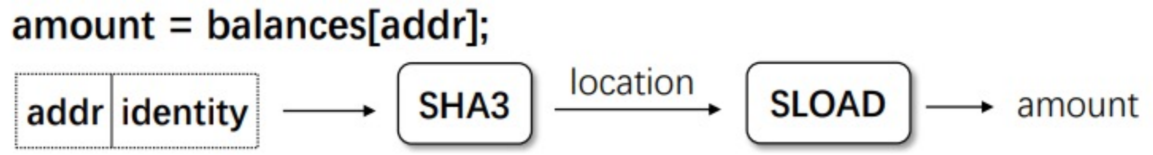


Figure 5: Read *amount* from <key: *addr*, value: *amount*>

  - identify 4 types of mapping variables after manually inspecting 16,248 open-source tokens

Stage 2: Locating Core Data Structure

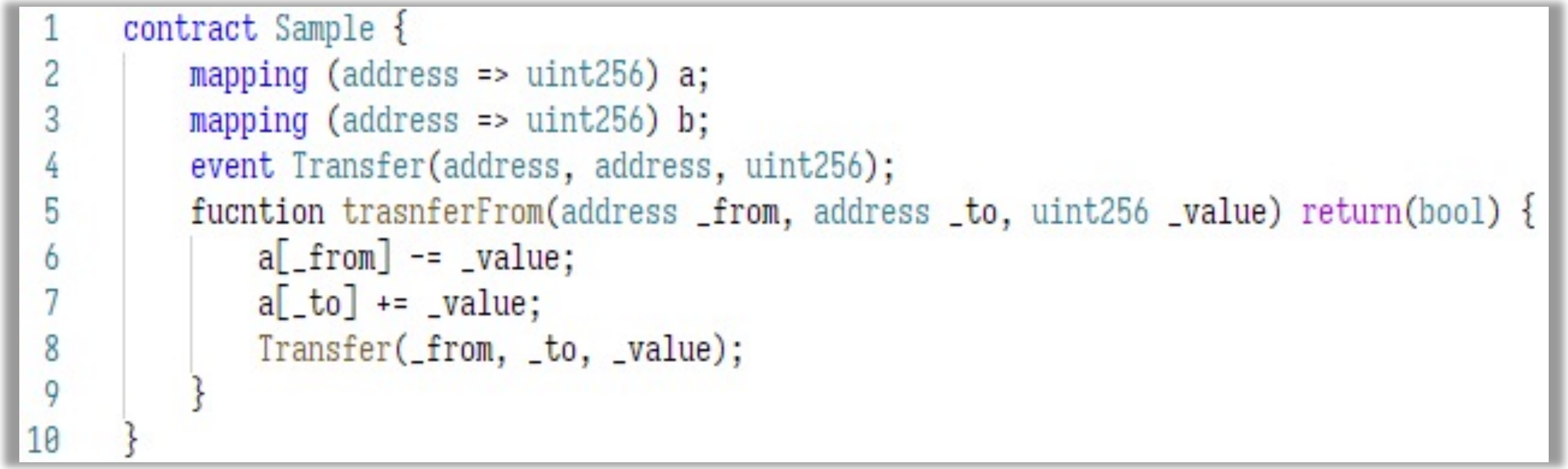
- Step 2: Parsing standard methods
  - *transfer(_to, _value);*
    token: <msg.sender, -value>, <_to, +value>
  - *transferFrom(_from, _to, _value);*
    token: <_from, -value>, <_to, +value>
  - Idea
    - function signature: unique identifier of the function

Stage 2: Locating Core Data Structure

- Step 3: Parsing standard events
  - *Transfer(_from, _to, value);*
    token: <_from, -value>, <_to, +value>
  - Idea
    - Event signature

Stage 2: Locating Core Data Structure

- Step 4: Recognizing the core data structure M
  - Recall step 1: Locating Mapping variables
  - Goal: distinguish the M from the mapping variables
  - Idea: correlate the modification of a mapping variable with the standard interfaces and the Transfer event.



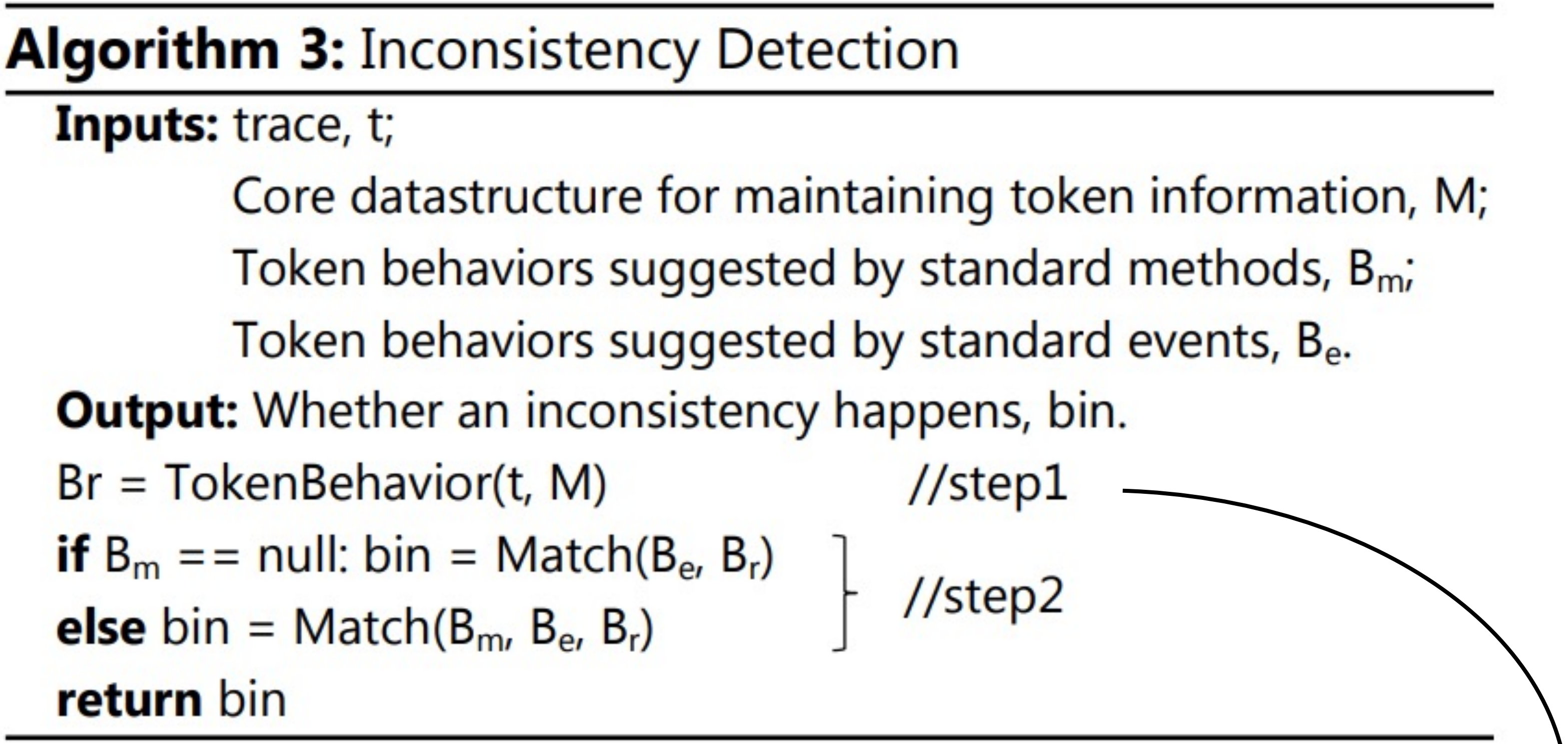
```
contract Sample {
    mapping (address => uint256) a;
    mapping (address => uint256) b;
    event Transfer(address, address, uint256);
    fucntion trasnferFrom(address _from, address _to, uint256 _value) return(bool) {
        a[_from] -= _value;
        a[_to] += _value;
        Transfer(_from, _to, _value);
    }
}
```

a:<_from, ->

<_to, +>

Transfer(_from, _to)

Stage 3: Detecting Inconsistent Behaviors



Algorithm 3: Inconsistency Detection

```
Inputs: trace, t;
        Core datastructure for maintaining token information, M;
        Token behaviors suggested by standard methods, $B_m$;
        Token behaviors suggested by standard events, $B_e$.
Output: Whether an inconsistency happens, bin.
Br = TokenBehavior(t, M)                        //step1
if $B_m$ == null: bin = Match($B_e$, $B_r$)     ]
else bin = Match($B_m$, $B_e$, $B_r$)           ] //step2
return bin
```

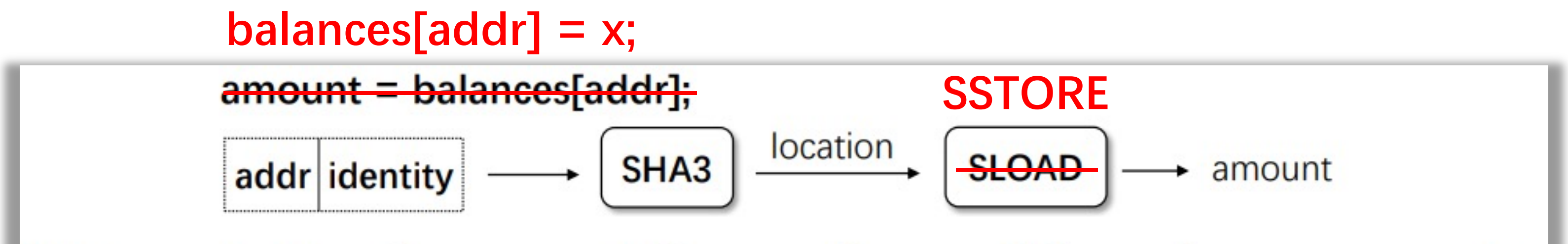


Figure 5: ~~Read~~ ***amount*** **from <key: *addr*, value: *amount*>**

Write

Experiments

- Dataset: 1~6,066,793 blocks (Jul. 30, 2015 to Aug. 1, 2018)

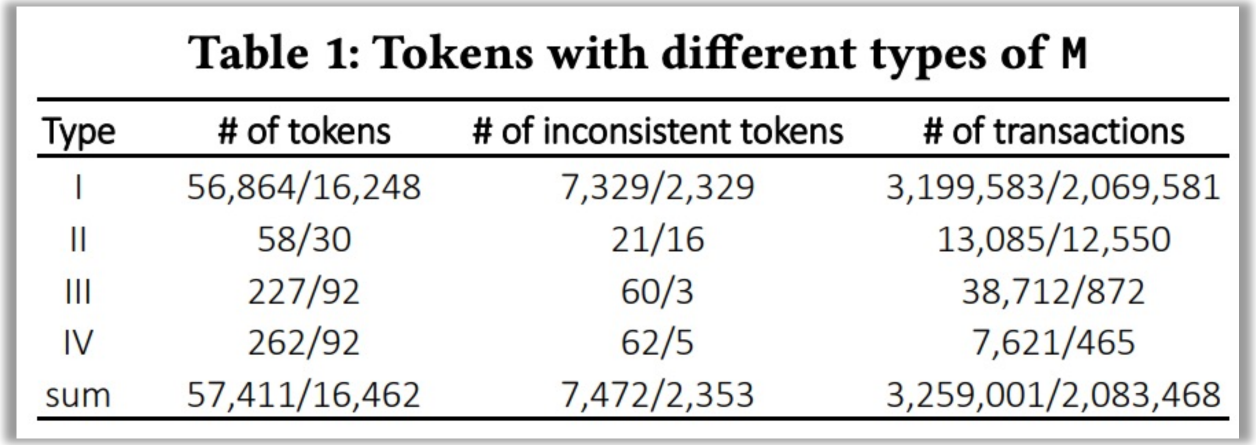


Table 1: Tokens with different types of M

Type	# of tokens	# of inconsistent tokens	# of transactions
I	56,864/16,248	7,329/2,329	3,199,583/2,069,581
II	58/30	21/16	13,085/12,550
III	227/92	60/3	38,712/872
IV	262/92	62/5	7,621/465
sum	57,411/16,462	7,472/2,353	3,259,001/2,083,468

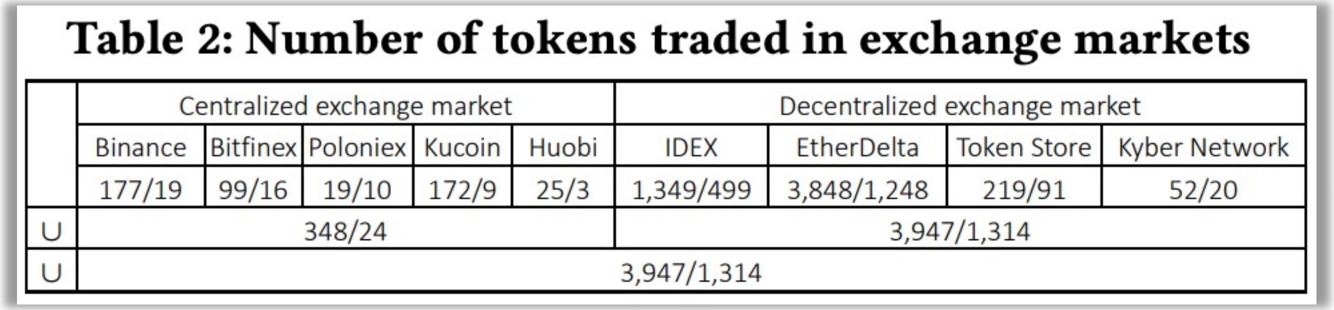


Table 2: Number of tokens traded in exchange markets

	Centralized exchange market					Decentralized exchange market			
	Binance	Bitfinex	Poloniex	Kucoin	Huobi	IDEX	EtherDelta	Token Store	Kyber Network
	177/19	99/16	19/10	172/9	25/3	1,349/499	3,848/1,248	219/91	52/20
∪	348/24					3,947/1,314			
∪	3,947/1,314								

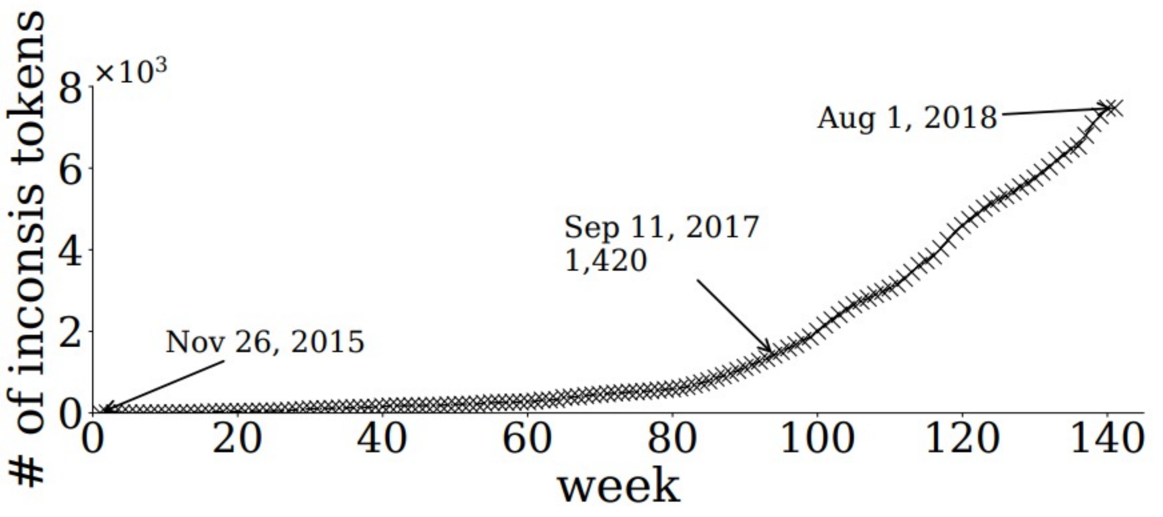
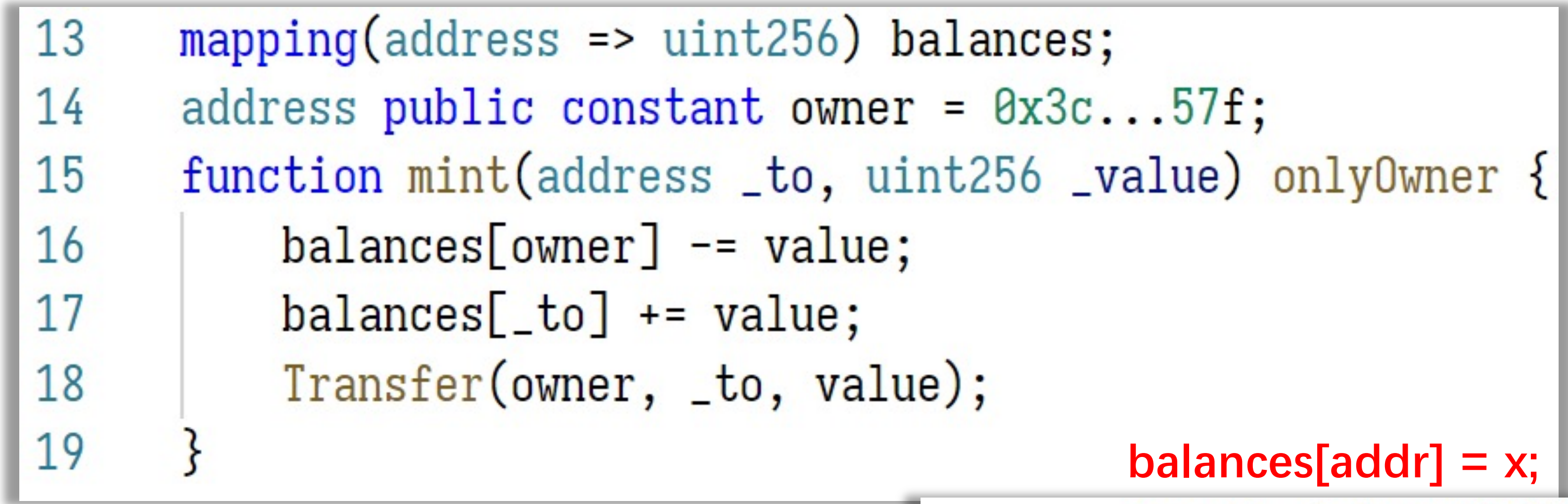


Figure 9: Deployment time of inconsistent tokens

Experiments

- Precision: 99.9%
  manually check all 2,353 open-source inconsistent tokens detected by TokenScope
- only 1 false positive:



```
    mapping(address => uint256) balances;
    address public constant owner = 0x3c...57f;
    function mint(address _to, uint256 _value) onlyOwner {
        balances[owner] -= value;
        balances[_to] += value;
        Transfer(owner, _to, value);
    }
```

balances[addr] = x;

~~amount = balances[addr];~~

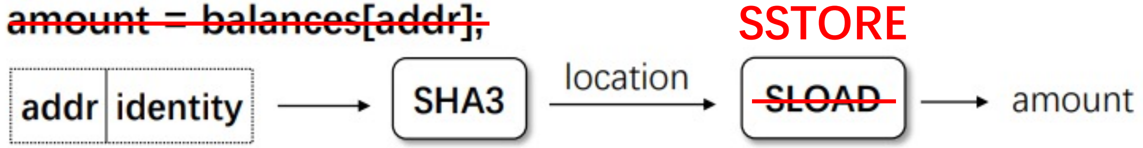
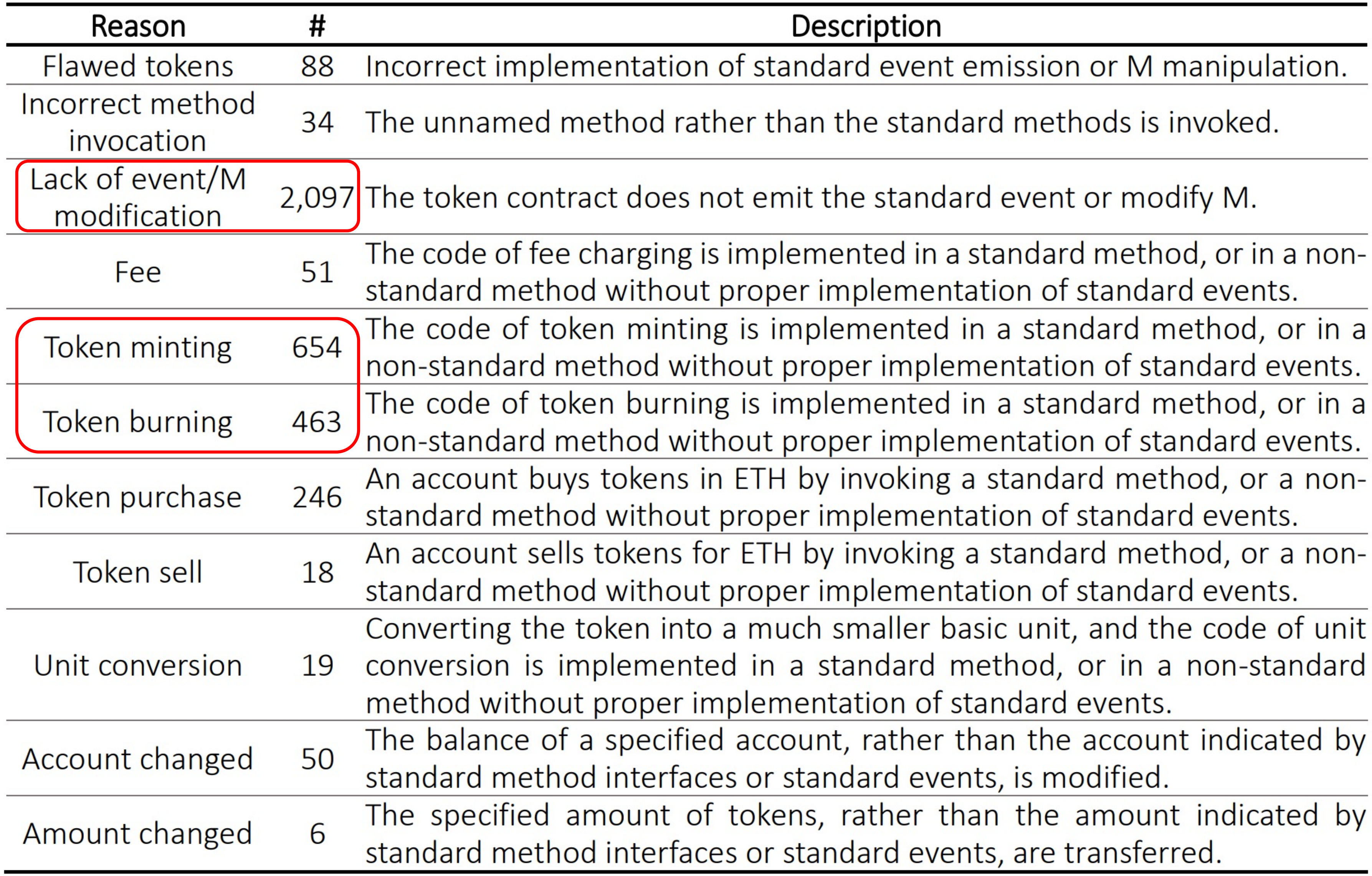


Figure 5: ~~Read~~ *amount* from <key: *addr*, value: *amount*>

Write

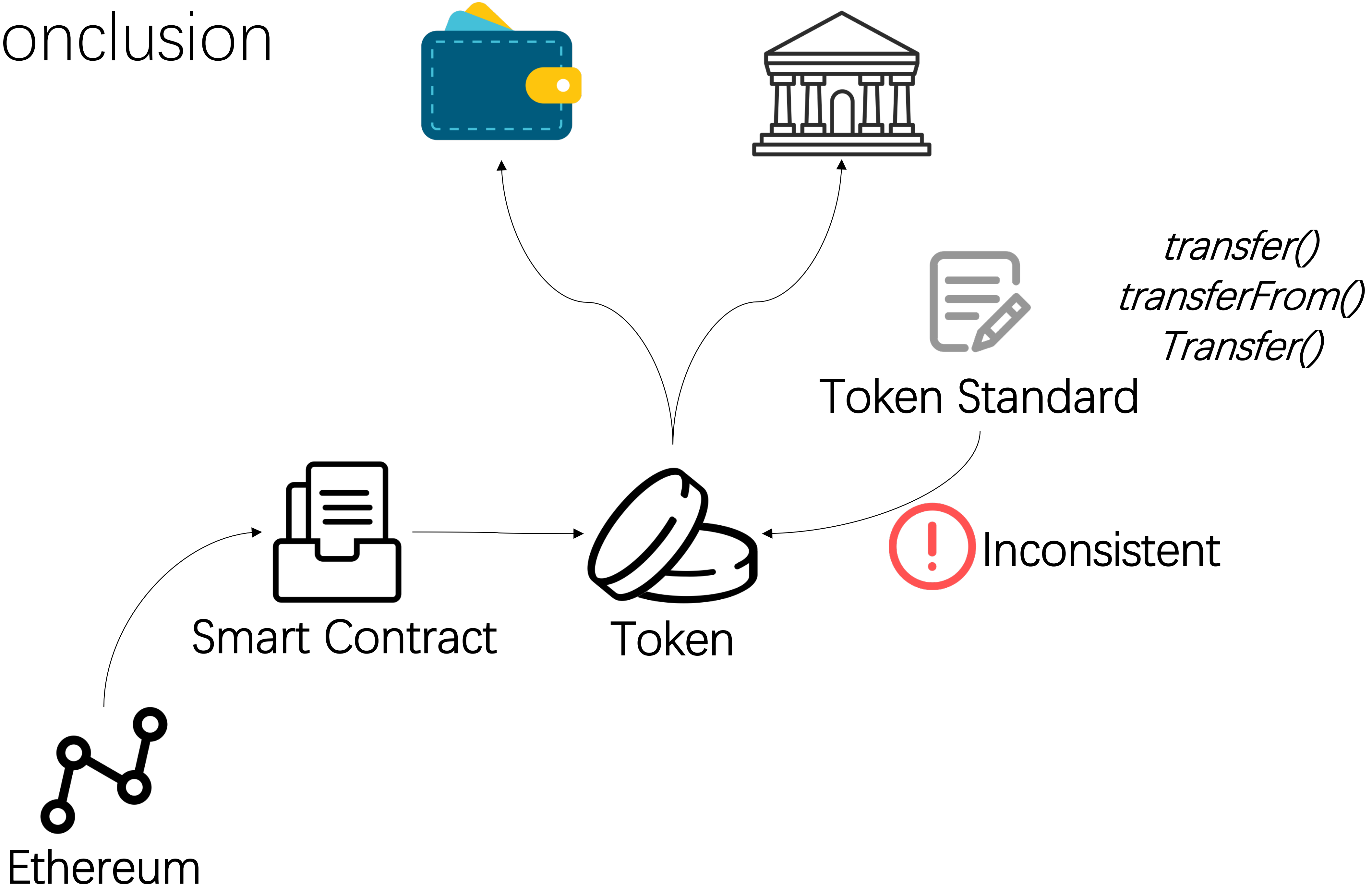
Reasons of inconsistent behaviors

Table 4: 11 major reasons for inconsistency



Reason	#	Description
Flawed tokens	88	Incorrect implementation of standard event emission or M manipulation.
Incorrect method invocation	34	The unnamed method rather than the standard methods is invoked.
Lack of event/M modification	2,097	The token contract does not emit the standard event or modify M.
Fee	51	The code of fee charging is implemented in a standard method, or in a non-standard method without proper implementation of standard events.
Token minting	654	The code of token minting is implemented in a standard method, or in a non-standard method without proper implementation of standard events.
Token burning	463	The code of token burning is implemented in a standard method, or in a non-standard method without proper implementation of standard events.
Token purchase	246	An account buys tokens in ETH by invoking a standard method, or a non-standard method without proper implementation of standard events.
Token sell	18	An account sells tokens for ETH by invoking a standard method, or a non-standard method without proper implementation of standard events.
Unit conversion	19	Converting the token into a much smaller basic unit, and the code of unit conversion is implemented in a standard method, or in a non-standard method without proper implementation of standard events.
Account changed	50	The balance of a specified account, rather than the account indicated by standard method interfaces or standard events, is modified.
Amount changed	6	The specified amount of tokens, rather than the amount indicated by standard method interfaces or standard events, are transferred.

Conclusion



transfer()

transferFrom()

Transfer()

Token Standard

Inconsistent

Smart Contract

Token

Ethereum

Conclusion

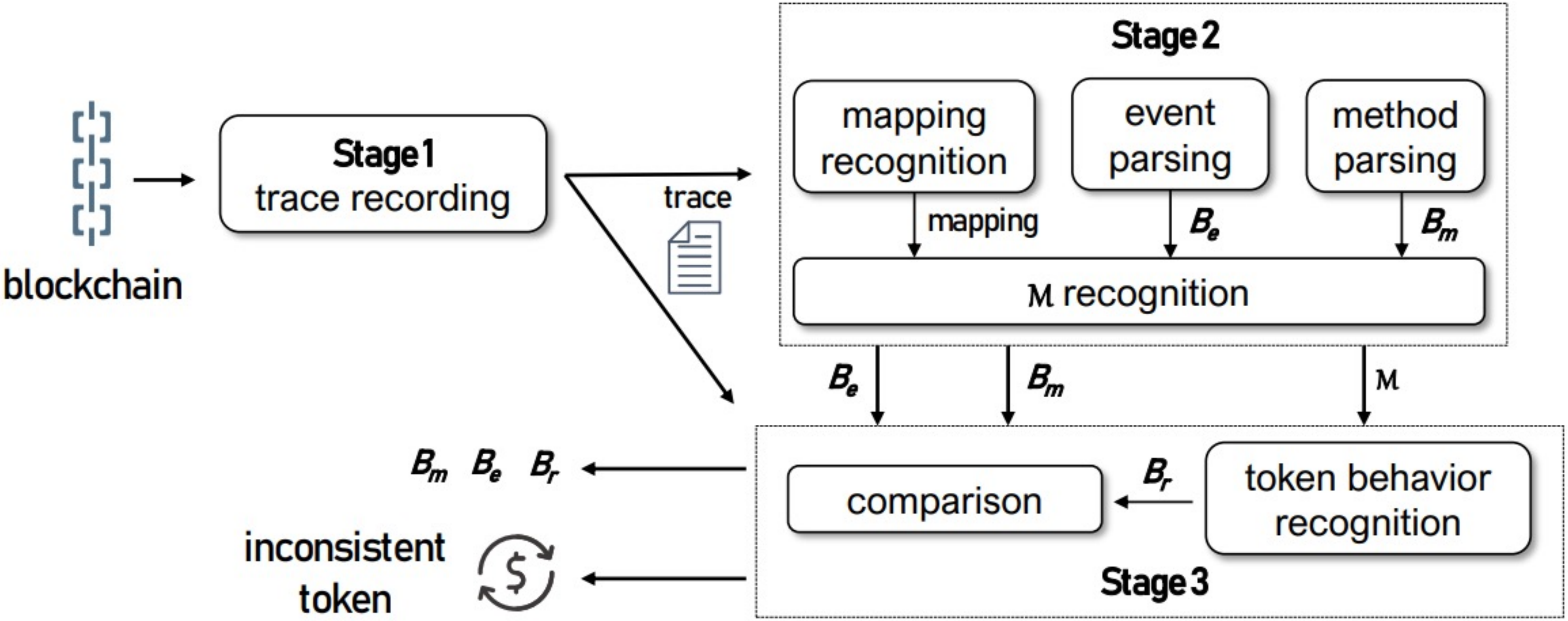


Figure 3: Architecture of TokenScope