TRAIL OF BITS

Automatic Bug-Finding for the Blockchain

EkoParty 2017

Who are we?

- Felipe Manzano, felipe@trailofbits.com
- Josselin Feist, josselin@trailofbits.com
- Trail of Bits: trailofbits.com
  - We help organizations build safer software
  - R&D focused: We use the latest program analysis techniques

Plan

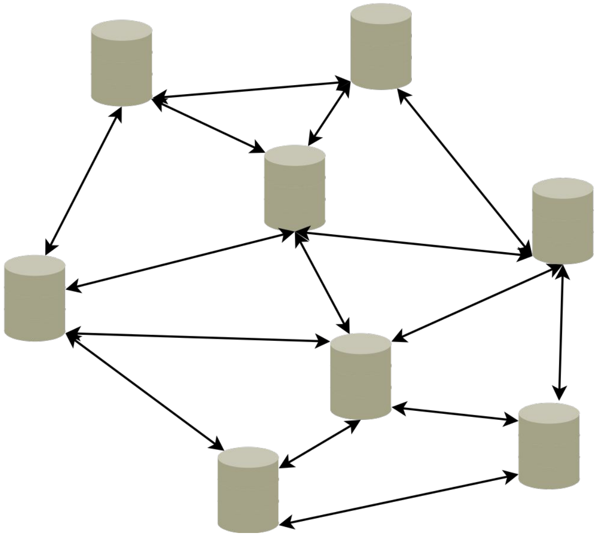
- Ethereum Blockchain
- Smart Contract Design
- Smart Contract Vulnerabilities
- Manticore

Our contribution: **Symbolic Execution on Smart Contracts**

The Ethereum Blockchain

Blockchain

- **Distributed data:** All participants store all the data
- **Decentralized consensus:** Everyone agrees on the data



Blockchain Application

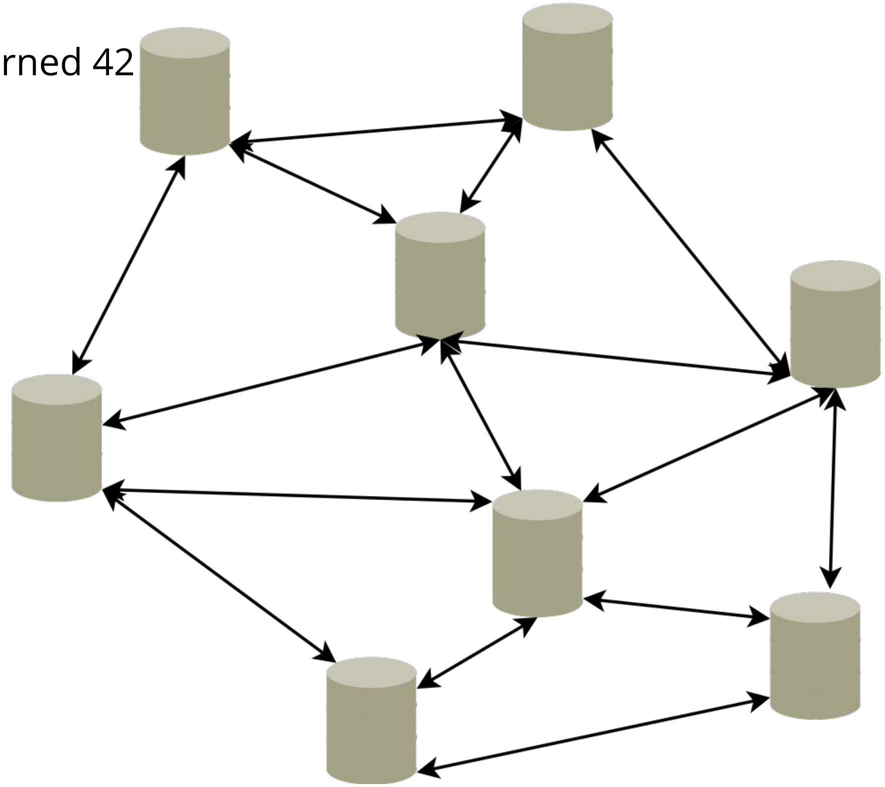
- **Bitcoin (2009):** First digital currency using blockchain
  - Solved the double spending problem

- **Ethereum (2015):** Extended blockchain to run apps
  - Store & execute code

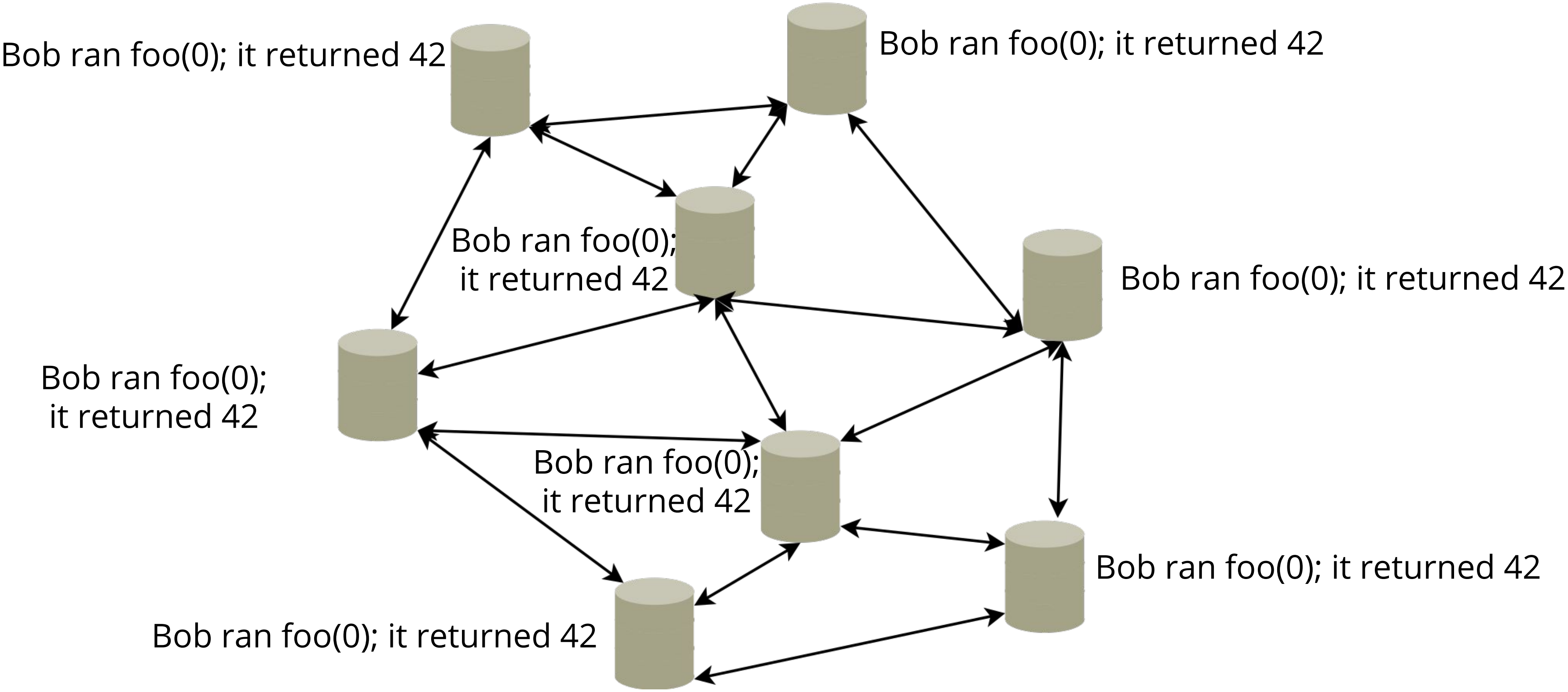
Bitcoin: distributed database => Ethereum: distributed VM

Decentralized Application

Bob ran foo(0); it returned 42



Decentralized Application



Bob ran foo(0); it returned 42
Bob ran foo(0); it returned 42
Bob ran foo(0); it returned 42
Bob ran foo(0); it returned 42
Bob ran foo(0); it returned 42
Bob ran foo(0); it returned 42
Bob ran foo(0); it returned 42
Bob ran foo(0); it returned 42

Smart Contracts

- **Smart Contracts:** Applications that run on Ethereum
  - Everyone executes and verifies it
  - Decentralized: nobody can stop or secretly modify data
  - **=> Ensures strong properties on your application**

Smart Contract Usage

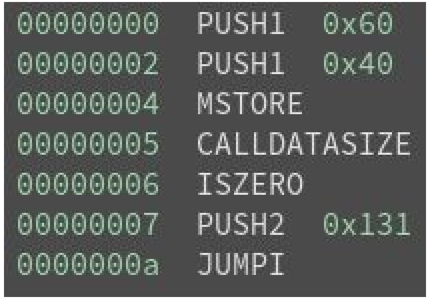
- Digital currency is one example of an application
  - ICO, Crowdfunding system
  - Game (ex: Poker, lotteries, ..)
  - ...

- Already a **lot** of money invested into smart contracts
  - Tezos ICO: $200 million
  - Bancor ICO: $153 million

Smart Contract Design

Ethereum Design

- Ethereum runs EVM bytecode



```
00000000  PUSH1  0x60
00000002  PUSH1  0x40
00000004  MSTORE
00000005  CALLDATASIZE
00000006  ISZERO
00000007  PUSH2  0x131
0000000a  JUMPI
```

- VM with <150 opcodes, only 1 register (PC), stack-based
- Calling a function = making a transaction
- Each transaction has a cost (gas), paid in ethers
- Bytecode cannot be updated (!)

Solidity

- Smart contracts are typically written in **Solidity**
  - High-level language in “Javascript style”
  - Contracts organized as a set of methods
  - State = contract variables + balance (# ethers)

Solidity: Example

```
pragma solidity 0.4.16; // Compiler version
contract Bank{                     // There are bugs, don't use this contract
    mapping(address => uint) private balances;

    function Bank(uint initial_supply) public {
       balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

Solidity: Example

```
pragma solidity 0.4.16; // Compiler version
contract Bank{                              // There are bugs, don't use this contract
    mapping(address => uint) private balances;

    function Bank(uint initial_supply) public {
       balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

Solidity: Example

```
pragma solidity 0.4.16; // Compiler version
contract Bank{                                   // There are bugs, don't use this contract
    mapping(address => uint) private balances;

    function Bank(uint initial_supply) public {
       balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

Solidity: Example

```
pragma solidity 0.4.16; // Compiler version
contract Bank{                     // There are bugs, don't use this contract
    mapping(address => uint) private balances;

    function Bank(uint initial_supply) public {
       balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

Solidity: Example

```
pragma solidity 0.4.16; // Compiler version
contract Bank{                     // There are bugs, don't use this contract
    mapping(address => uint) private balances;

    function Bank(uint initial_supply) public {
       balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

Solidity: Example

```
pragma solidity 0.4.16; // Compiler version
contract Bank{                     // There are bugs, don't use this contract
    mapping(address => uint) private balances;

    function Bank(uint initial_supply) public {
       balances[msg.sender] = initial_supply;
    }
    function transfer(address to, uint val) public {
        balances[msg.sender] -= val;
        balances[to] += val;
    }
    function balanceOf(address user) public constant returns (uint){
        return balances[user];
    }
}
```

State variable

Constructor

Public function

Constant function
(gas-free)

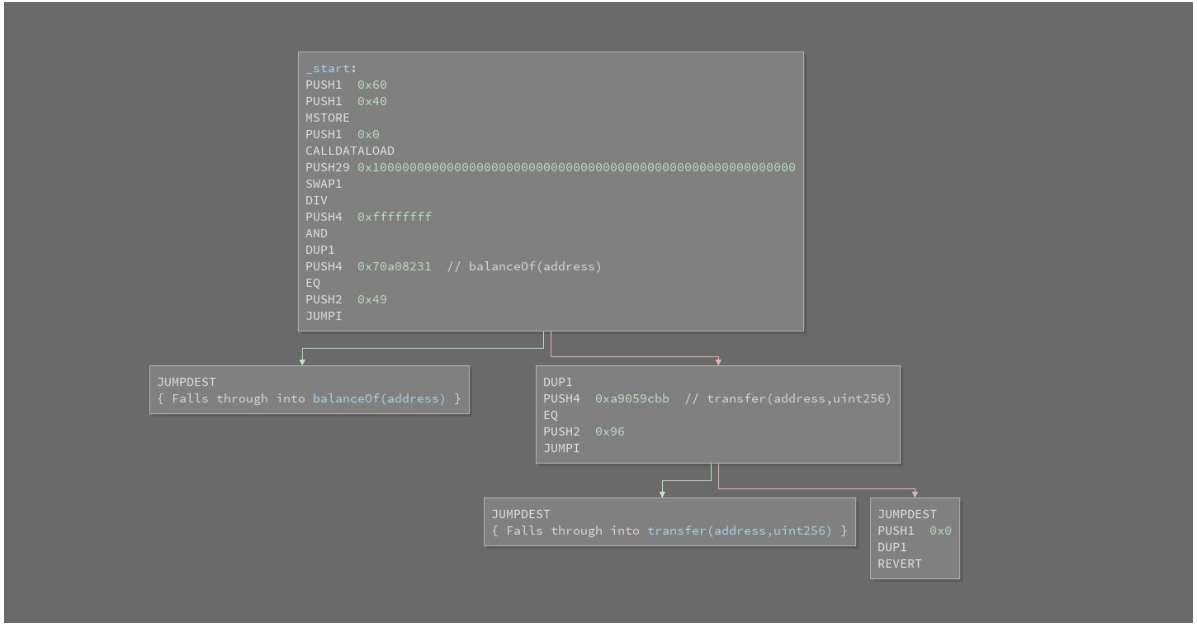
Transaction

- Among other, a transaction has: From/to/data
- Data holds: Function name and parameters
    - Function name: 4 bytes of keccak256(signature)
        - Ex: 'transfer(address,uint256)' => 0xa9059cbb
    - Parameters can be padded with 0 bytes according the size

transfer(0x41414141, 0x42) =

```
    0xa9059cbb0000000000000000000000000000000000000000000000000
    0000000414141410000000000000000000000000000000000000000000000
    0000000000000000042
```

Demo



```
_start:
PUSH1  0x60
PUSH1  0x40
MSTORE
PUSH1  0x0
CALLDATALOAD
PUSH29 0x100000000000000000000000000000000000000000000000000000000
SWAP1
DIV
PUSH4  0xffffffff
AND
DUP1
PUSH4  0x70a08231  // balanceOf(address)
EQ
PUSH2  0x49
JUMPI
```

```
JUMPDEST
{ Falls through into balanceOf(address) }
```

```
DUP1
PUSH4  0xa9059cbb  // transfer(address,uint256)
EQ
PUSH2  0x96
JUMPI
```

```
JUMPDEST
{ Falls through into transfer(address,uint256) }
```

```
JUMPDEST
PUSH1  0x0
DUP1
REVERT
```

Smart Contract Vulnerabilities

Smart Contract Security

- Vulnerabilities in smart contracts have already cost a lot

- **Parity Wallet:** $30 million (could have been a lot worse)
- **DAO Hack:** $150 million (led to a hard fork)

Smart Contract Vulnerabilities

- “Classic” vulnerabilities:
  - Integer overflow/underflow
  - Race condition

- Logic vulnerabilities / errors in the design
  - Harder to find, but deadly

Reentracy Vulnerability

- The DAO ($$$)

```
function withdrawBalance(){
    // Send the balance to the caller.
    // If the caller is a contract, call the fallback function
    if( ! (msg.sender.call.value(userBalance[msg.sender])() ) ){
        throw;
    }
    // Empty the balance
    userBalance[msg.sender] = 0;
}
```

- Use of the fallback function to call the caller
  - call `withdrawBalance` from a malicious contract
  - `withdrawBalance` calls the fallback function of the malicious contract
  - The fallback function calls a second time `withdrawBalance`
  - Repeat n times => withdraws n times the original deposit

Improperly restricted functions

- Parity Wallet
  - Widely used library for storing ethers
  - Built by Gavin Wood, formerly CTO of Ethereum Foundation

- Key function was public, should have been callable only once
  - End result: Anyone can become the owner of the contract

Other Examples

- KingOfTheEtherThrone: Calls to external function not tested -> excepted compensation could be not send
- GovernMental: Uses new address[](0);, which cleans the internal storage, but iterates over all the index -> fees too costly to be executed
- Rubixi : Constructor with incorrect name, anyone could become the owner (and calls to send() are never checked)
- Rock-Paper-Scissor : Data was not hidden
- FirePonzi: Mistype between payoutCursor_Id and payoutCursor_Id_
- The Run: Uses the current timestamp as random number; but timestamp can be manipulated

Logic vulnerabilities are hard to find

- What is a vulnerability in a contract?
  - It depends on the contract purpose!

- A user ends with more ethers than invested, is it a bug?
  - Yes, if the contract is a paid service
  - No, if the contract is a lottery

Smart Contract Symbolic Execution

Manticore - EVM

- A symbolic execution engine for EVM
- All possible contract paths are explored
- Supports multiple contracts and transactions
- Produces examples transactions that fail
- API for generic instrumentation

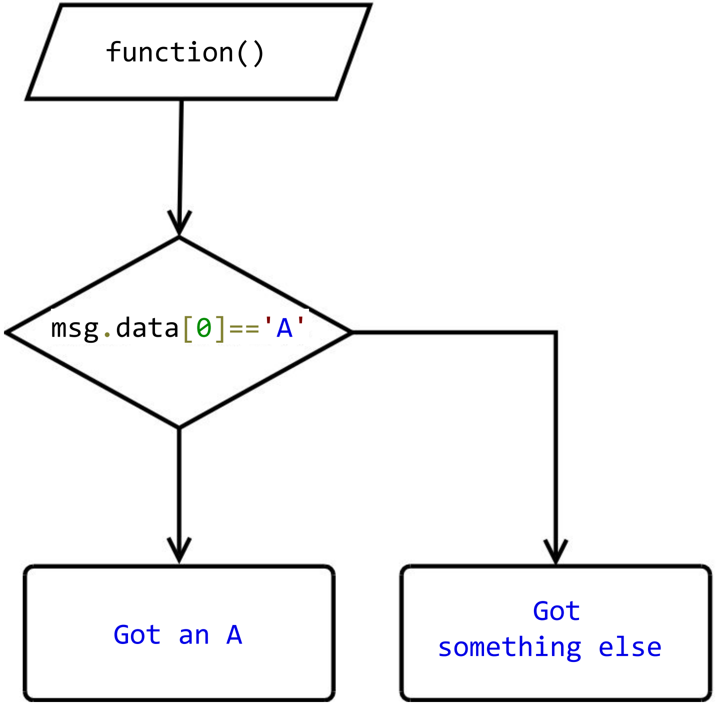
Manticore - EVM - How

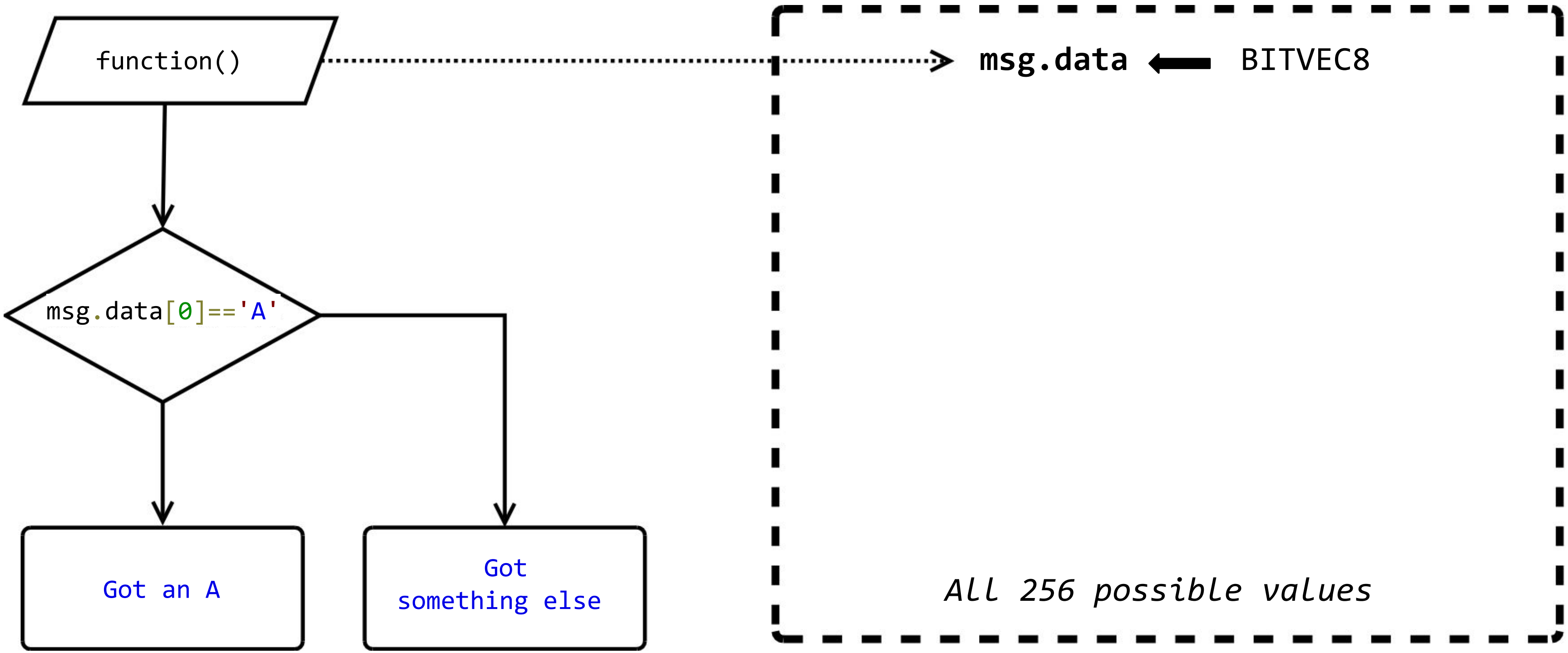
- Transaction inputs are considered symbolic
- Emulated instructions build expressions
- At a conditional jump, analysis is forked
- A set of input constraints is maintained
- An SMT solver is queried to solve these constraints

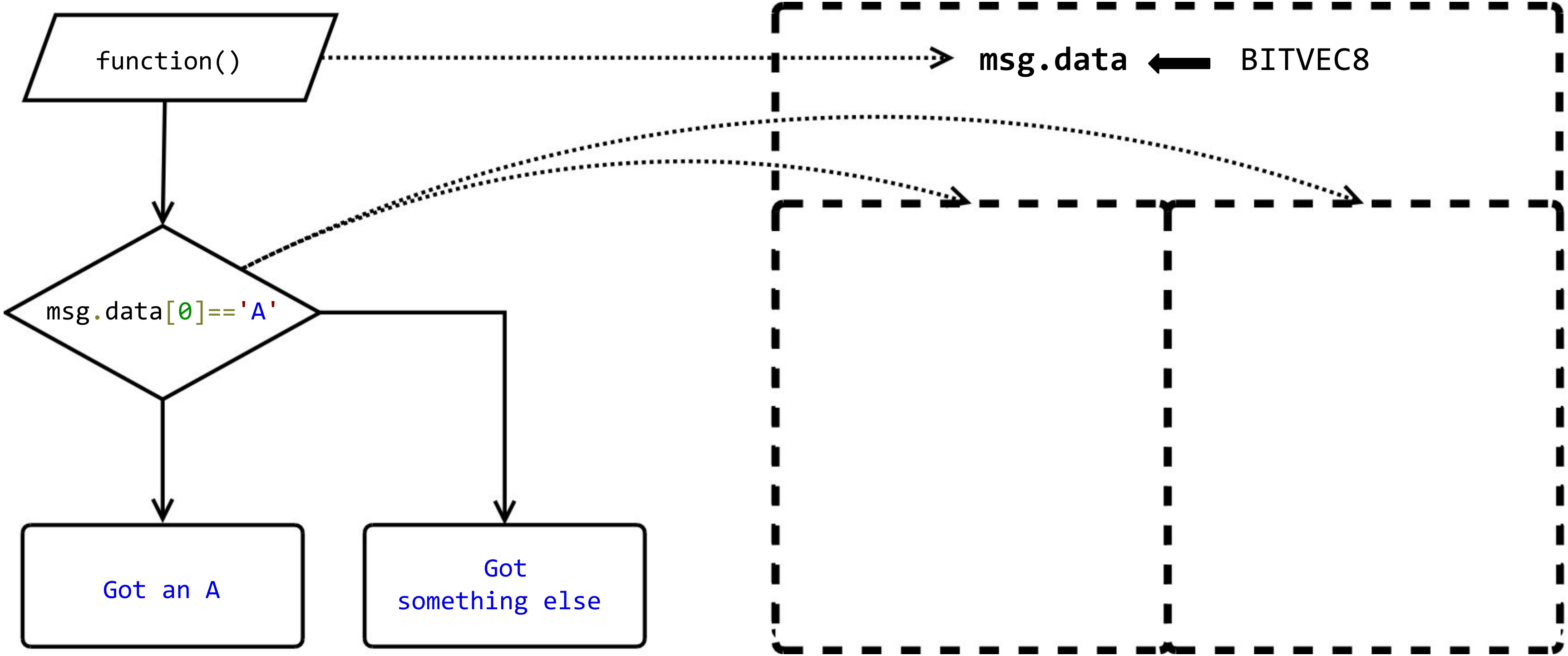
Toy Contract

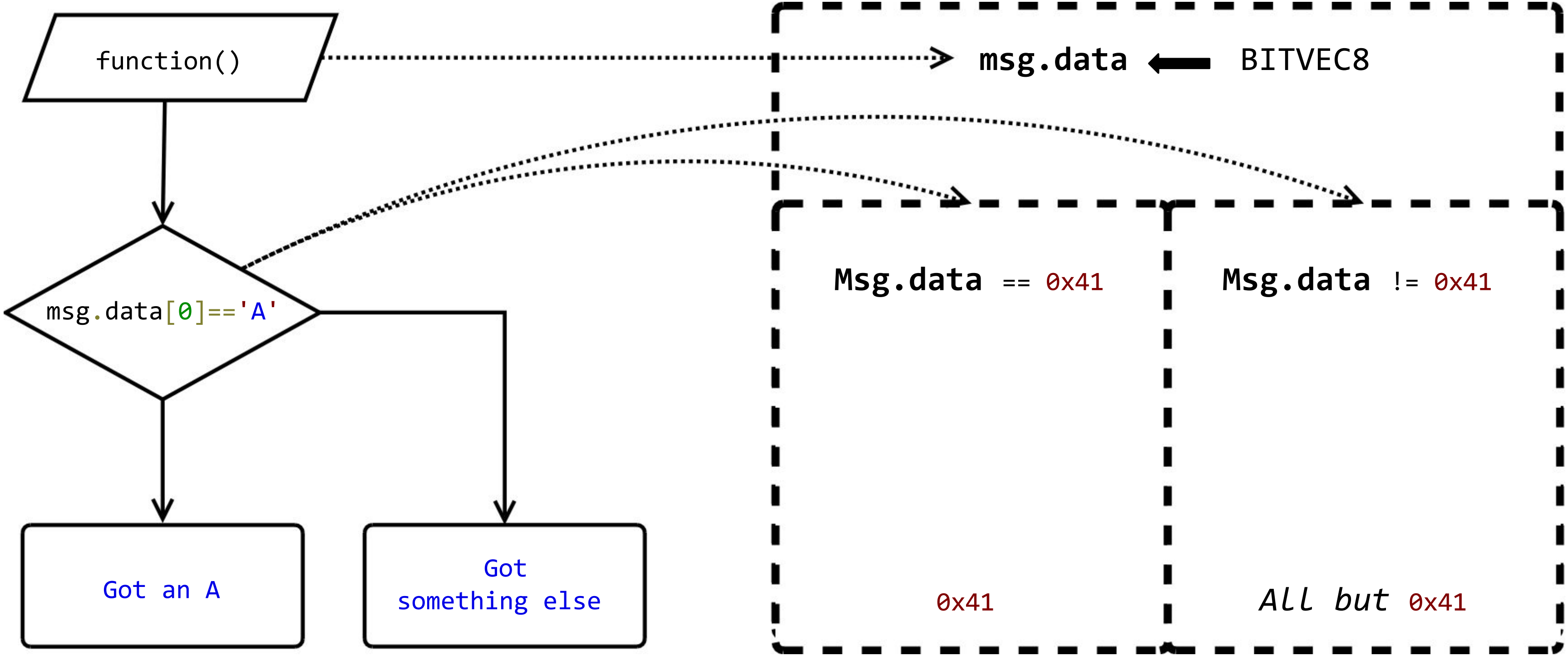
```
contract Simple {
    event Log(string);

    function() payable {
        if (msg.data[0] == 'A') {
            Log("Got an A");
        }else{
            Log("Got something else");
        }
    }
}
```

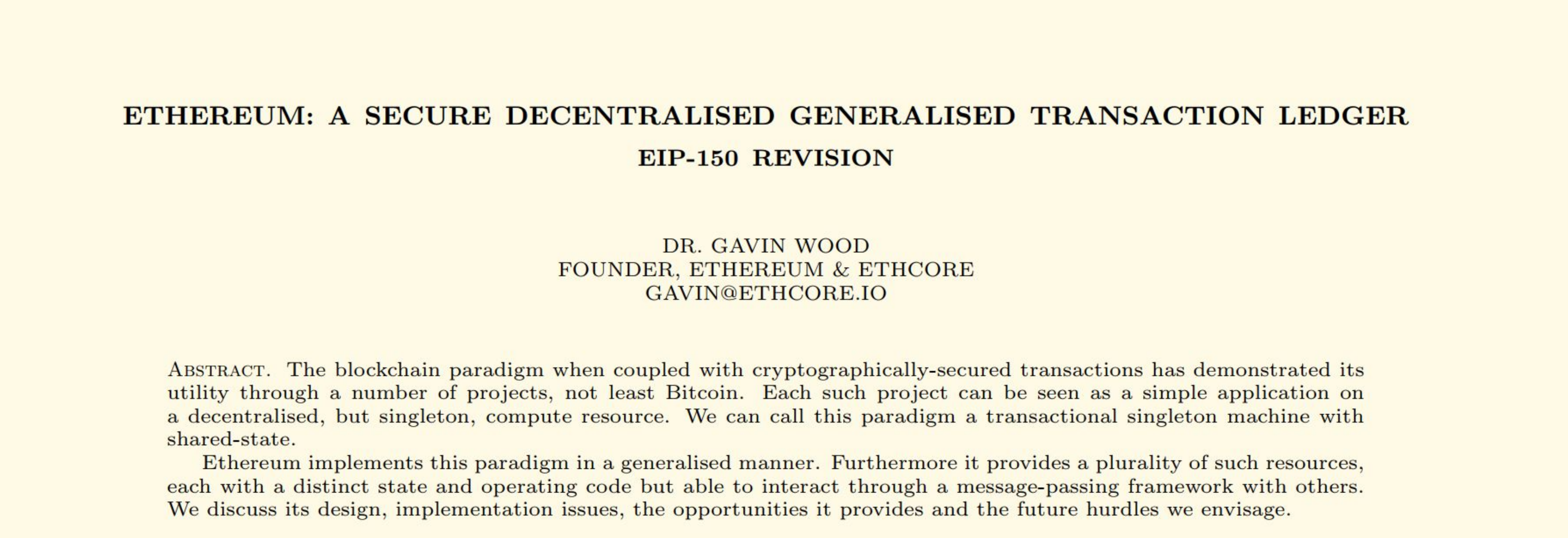

function()
msg.data[0]=='A'
Got an A
Got
something else


function()
msg.data[0]=='A'
Got an A
Got something else
msg.data
BITVEC8
All 256 possible values


function()
msg.data[0]=='A'
Got an A
Got
something else
msg.data
BITVEC8


function()
msg.data[0]=='A'
Got an A
Got something else
msg.data
BITVEC8
Msg.data == 0x41
Msg.data != 0x41
0x41
All but 0x41

The Yellow Paper



ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER

EIP-150 REVISION

DR. GAVIN WOOD
FOUNDER, ETHEREUM & ETHCORE
GAVIN@ETHCORE.IO

ABSTRACT. The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, not least Bitcoin. Each such project can be seen as a simple application on a decentralised, but singleton, compute resource. We can call this paradigm a transactional singleton machine with shared-state.

Ethereum implements this paradigm in a generalised manner. Furthermore it provides a plurality of such resources, each with a distinct state and operating code but able to interact through a message-passing framework with others. We discuss its design, implementation issues, the opportunities it provides and the future hurdles we envisage.

http://gavwood.com/paper.pdf

Instructions

```
STOP, ADD, MUL, SUB, DIV, SDIV, MOD, SMOD, ADDMOD, MULMOD,
EXP, SIGNEXTEND, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR,
NOT, BYTE, SHA3, ADDRESS, BALANCE, ORIGIN, CALLER,
CALLVALUE, CALLDATALOAD, CALLDATASIZE, CALLDATACOPY,
CODESIZE, CODECOPY, GASPRICE, EXTCODESIZE, EXTCODECOPY,
BLOCKHASH, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY,
GASLIMIT, POP, MLOAD, MSTORE, MSTORE8, SLOAD, SSTORE, JUMP,
JUMPI, GETPC, MSIZE, GAS, JUMPDEST, PUSH, DUP, SWAP, LOG,
CREATE, CALL, CALLCODE, RETURN, DELEGATECALL, BREAKPOINT,
RNGSEED, SSIZEEXT, SLOADBYTES, SSTOREBYTES, SSIZE,
STATEROOT, TXEXECGAS, REVERT, INVALID, SELFDESTRUCT
```

Instructions - Stack

STOP, ADD, MUL, SUB, DIV, SDIV, MOD, SMOD, ADDMOD, MULMOD,
EXP, SIGNEXTEND, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR,
NOT, BYTE, SHA3, ADDRESS, BALANCE, ORIGIN, CALLER,
CALLVALUE, CALLDATALOAD, CALLDATASIZE, CALLDATACOPY,
CODESIZE, CODECOPY, GASPRICE, EXTCODESIZE, EXTCODECOPY,
BLOCKHASH, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY,
GASLIMIT, POP, MLOAD, MSTORE, MSTORE8, SLOAD, SSTORE, JUMP,
JUMPI, GETPC, MSIZE, GAS, JUMPDEST, PUSH, DUP, SWAP, LOG,
CREATE, CALL, CALLCODE, RETURN, DELEGATECALL, REVERT,
INVALID, SELFDESTRUCT

Instructions - Memory

```
STOP, ADD, MUL, SUB, DIV, SDIV, MOD, SMOD, ADDMOD, MULMOD,
EXP, SIGNEXTEND, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR,
NOT, BYTE, SHA3, ADDRESS, BALANCE, ORIGIN, CALLER,
CALLVALUE, CALLDATALOAD, CALLDATASIZE, CALLDATACOPY,
CODESIZE, CODECOPY, GASPRICE, EXTCODESIZE, EXTCODECOPY,
BLOCKHASH, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY,
GASLIMIT, POP, MLOAD, MSTORE, MSTORE8, SLOAD, SSTORE, JUMP,
JUMPI, GETPC, MSIZE, GAS, JUMPDEST, PUSH, DUP, SWAP, LOG,
CREATE, CALL, CALLCODE, RETURN, DELEGATECALL, REVERT,
INVALID, SELFDESTRUCT
```

Instructions - Flow control

```
STOP, ADD, MUL, SUB, DIV, SDIV, MOD, SMOD, ADDMOD, MULMOD,
EXP, SIGNEXTEND, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR,
NOT, BYTE, SHA3, ADDRESS, BALANCE, ORIGIN, CALLER,
CALLVALUE, CALLDATALOAD, CALLDATASIZE, CALLDATACOPY,
CODESIZE, CODECOPY, GASPRICE, EXTCODESIZE, EXTCODECOPY,
BLOCKHASH, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY,
GASLIMIT, POP, MLOAD, MSTORE, MSTORE8, SLOAD, SSTORE, JUMP,
JUMPI, GETPC, MSIZE, GAS, JUMPDEST, PUSH, DUP, SWAP, LOG,
CREATE, CALL, CALLCODE, RETURN, DELEGATECALL, REVERT,
INVALID, SELFDESTRUCT
```

Instructions - Arithmetic

STOP, ADD, MUL, SUB, DIV, SDIV, MOD, SMOD, ADDMOD, MULMOD, EXP, SIGNEXTEND, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR, NOT, BYTE, SHA3, ADDRESS, BALANCE, ORIGIN, CALLER, CALLVALUE, CALLDATALOAD, CALLDATASIZE, CALLDATACOPY, CODESIZE, CODECOPY, GASPRICE, EXTCODESIZE, EXTCODECOPY, BLOCKHASH, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY, GASLIMIT, POP, MLOAD, MSTORE, MSTORE8, SLOAD, SSTORE, JUMP, JUMPI, GETPC, MSIZE, GAS, JUMPDEST, PUSH, DUP, SWAP, LOG, CREATE, CALL, CALLCODE, RETURN, DELEGATECALL, REVERT, INVALID, SELFDESTRUCT

Instructions - SHA3

STOP, ADD, MUL, SUB, DIV, SDIV, MOD, SMOD, ADDMOD, MULMOD, EXP, SIGNEXTEND, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR, NOT, BYTE, SHA3, ADDRESS, BALANCE, ORIGIN, CALLER, CALLVALUE, CALLDATALOAD, CALLDATASIZE, CALLDATACOPY, CODESIZE, CODECOPY, GASPRICE, EXTCODESIZE, EXTCODECOPY, BLOCKHASH, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY, GASLIMIT, POP, MLOAD, MSTORE, MSTORE8, SLOAD, SSTORE, JUMP, JUMPI, GETPC, MSIZE, GAS, JUMPDEST, PUSH, DUP, SWAP, LOG, CREATE, CALL, CALLCODE, RETURN, DELEGATECALL, REVERT, INVALID, SELFDESTRUCT

Instructions - Control transactions

STOP, ADD, MUL, SUB, DIV, SDIV, MOD, SMOD, ADDMOD, MULMOD,
EXP, SIGNEXTEND, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR,
NOT, BYTE, SHA3, ADDRESS, BALANCE, ORIGIN, CALLER,
CALLVALUE, CALLDATALOAD, CALLDATASIZE, CALLDATACOPY,
CODESIZE, CODECOPY, GASPRICE, EXTCODESIZE, EXTCODECOPY,
BLOCKHASH, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY,
GASLIMIT, POP, MLOAD, MSTORE, MSTORE8, SLOAD, SSTORE, JUMP,
JUMPI, GETPC, MSIZE, GAS, JUMPDEST, PUSH, DUP, SWAP, LOG,
CREATE, CALL, CALLCODE, RETURN, DELEGATECALL, REVERT,
INVALID, SELFDESTRUCT

Ethereum Virtual Machine

PC	`0x0004: MSTORE  Save word to memory.`
Stack	`0x0000000000000000000000000000000000000000000000000000000000000010`
	`0xa0a1a2a3a4a5a6a7a8a9aaabacadaeafb0b1b2b3b4b5b6b7b8b9babbbcbdbebf`
	...

Mem	`0000:`
	`0010:`
	`0020:`
	`0030:`
	`0040:`
	`0050:`

1024

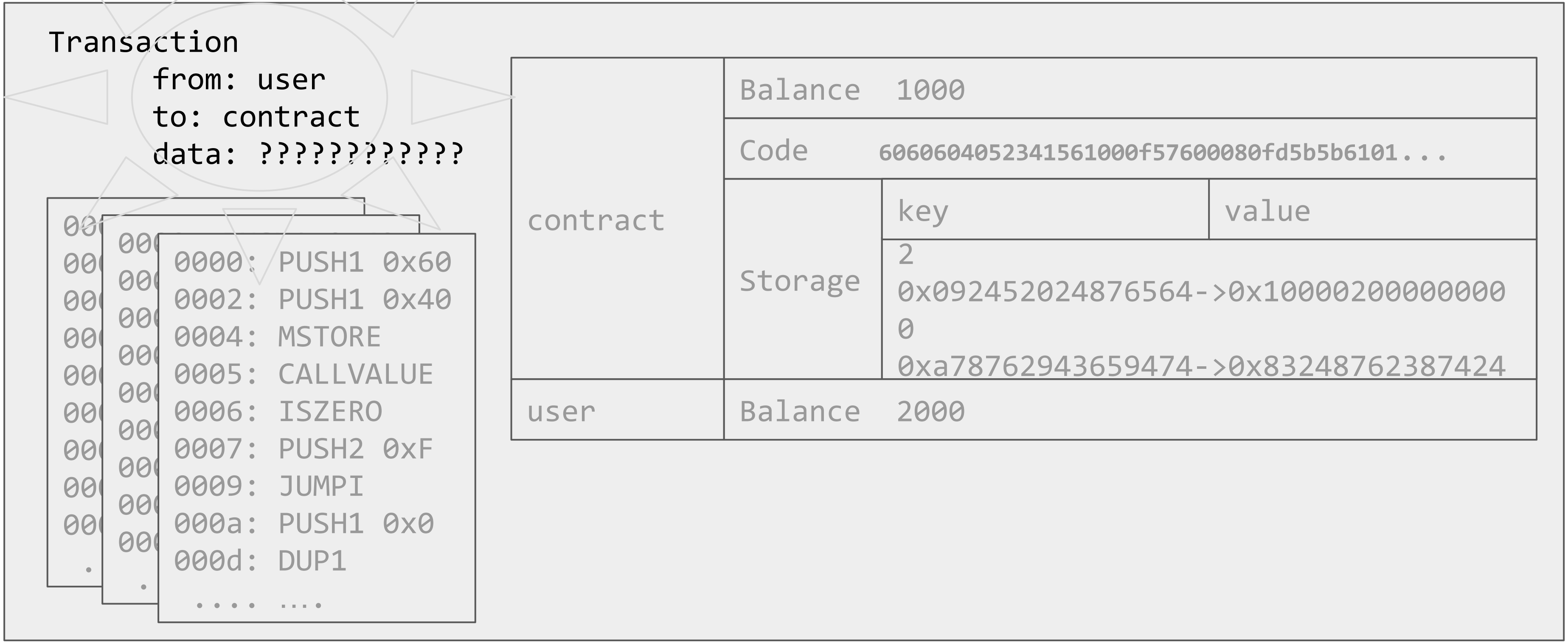
Ethereum Virtual Machine

PC	`0x0004: MSTORE  Save word to memory.`
Stack	
	...

Mem	`0000:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`
	`0010:   a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af`
	`0020:   b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf`
	`0030:`
	`0040:`
	`0050:`

0x30 allocated

Ethereum World - A transaction

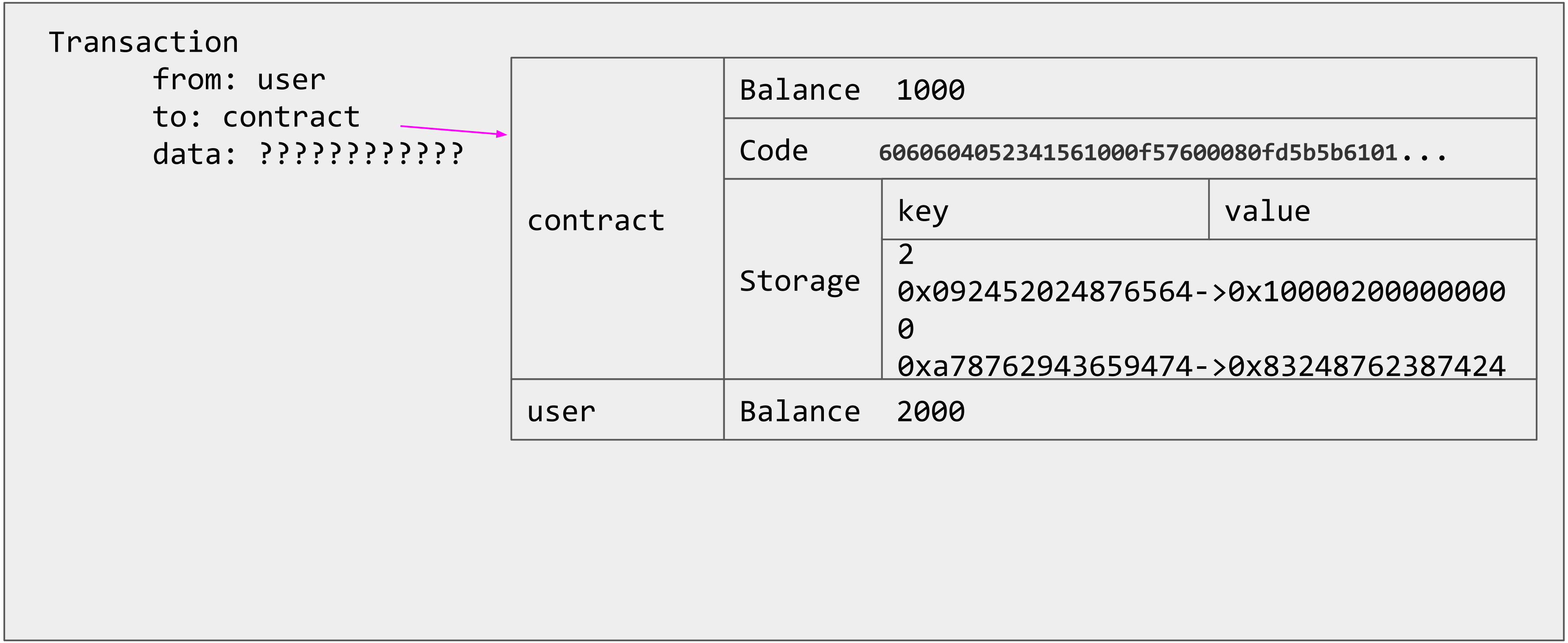


```
Transaction
      from: user
      to: contract
      data: ????????????
```

```
0000: PUSH1 0x60
0002: PUSH1 0x40
0004: MSTORE
0005: CALLVALUE
0006: ISZERO
0007: PUSH2 0xF
0009: JUMPI
000a: PUSH1 0x0
000d: DUP1
 ....  ....
```

contract	Balance 1000		
	Code 6060604052341561000f57600080fd5b5b6101...		
	Storage	key	value
		2 0x092452024876564->0x10000200000000 0 0xa78762943659474->0x83248762387424	
user	Balance 2000		

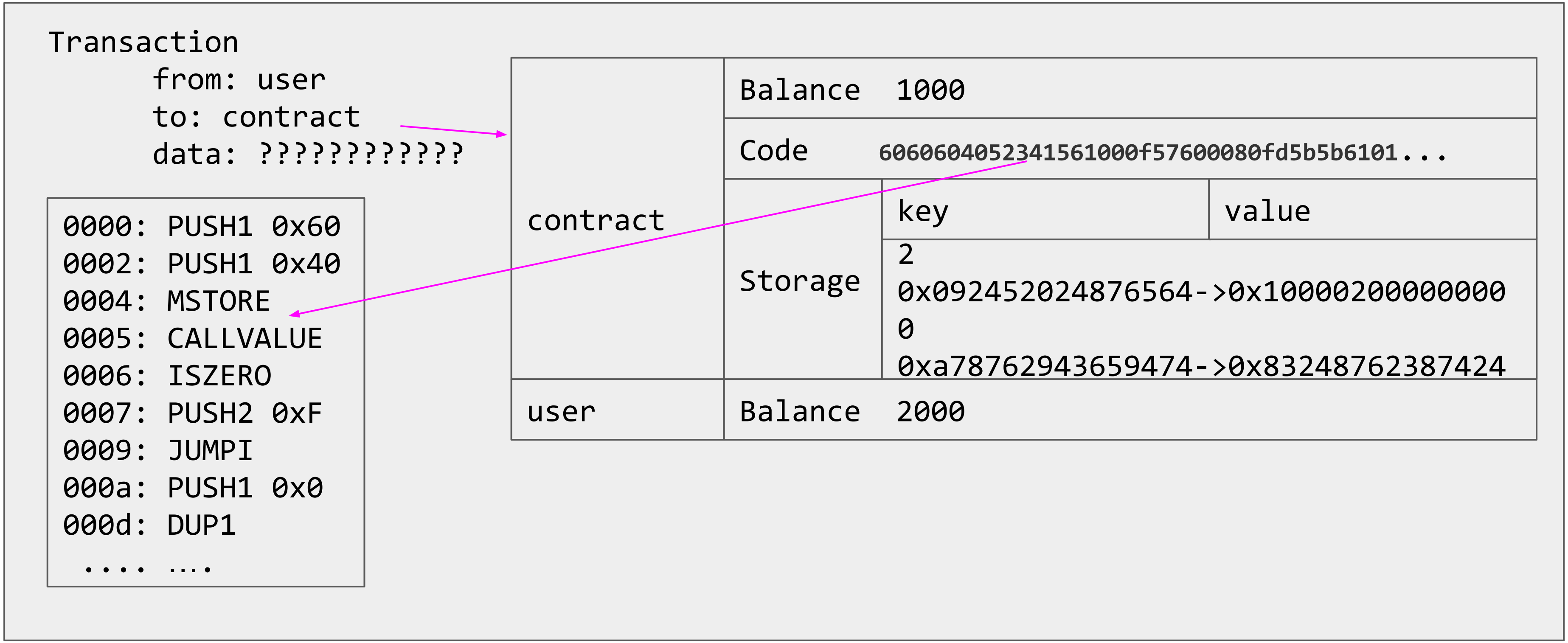
Ethereum World - A transaction



```
Transaction
      from: user
      to: contract
      data: ????????????
```

contract	Balance	1000	
	Code	6060604052341561000f57600080fd5b5b6101...	
	Storage	key	value
		2 0x092452024876564->0x10000200000000 0 0xa78762943659474->0x83248762387424	
user	Balance	2000	

Ethereum World - A transaction

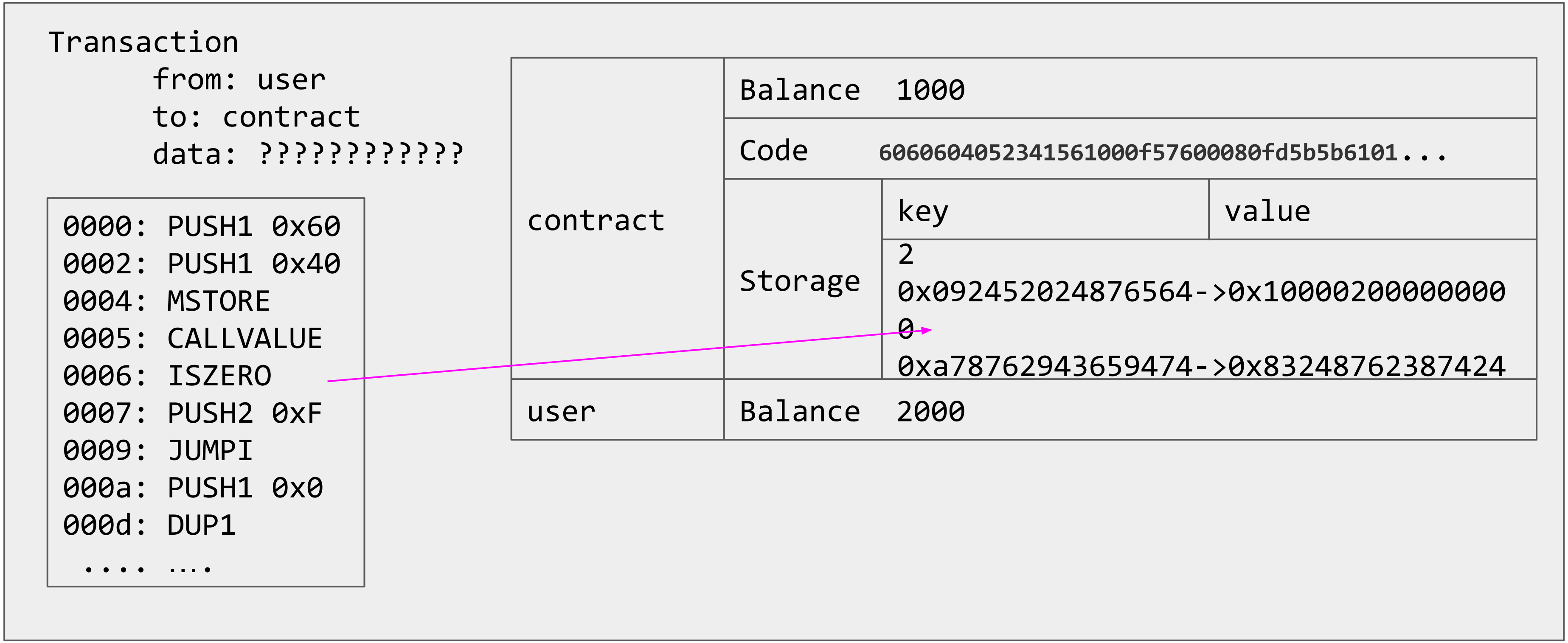


```
Transaction
      from: user
      to: contract
      data: ????????????
```

```
0000: PUSH1 0x60
0002: PUSH1 0x40
0004: MSTORE
0005: CALLVALUE
0006: ISZERO
0007: PUSH2 0xF
0009: JUMPI
000a: PUSH1 0x0
000d: DUP1
 .... ....
```

contract	Balance	1000	
	Code	6060604052341561000f57600080fd5b5b6101...	
	Storage	key	value
		2 0x092452024876564->0x10000200000000 0 0xa78762943659474->0x83248762387424	
user	Balance	2000	

Ethereum World - A transaction

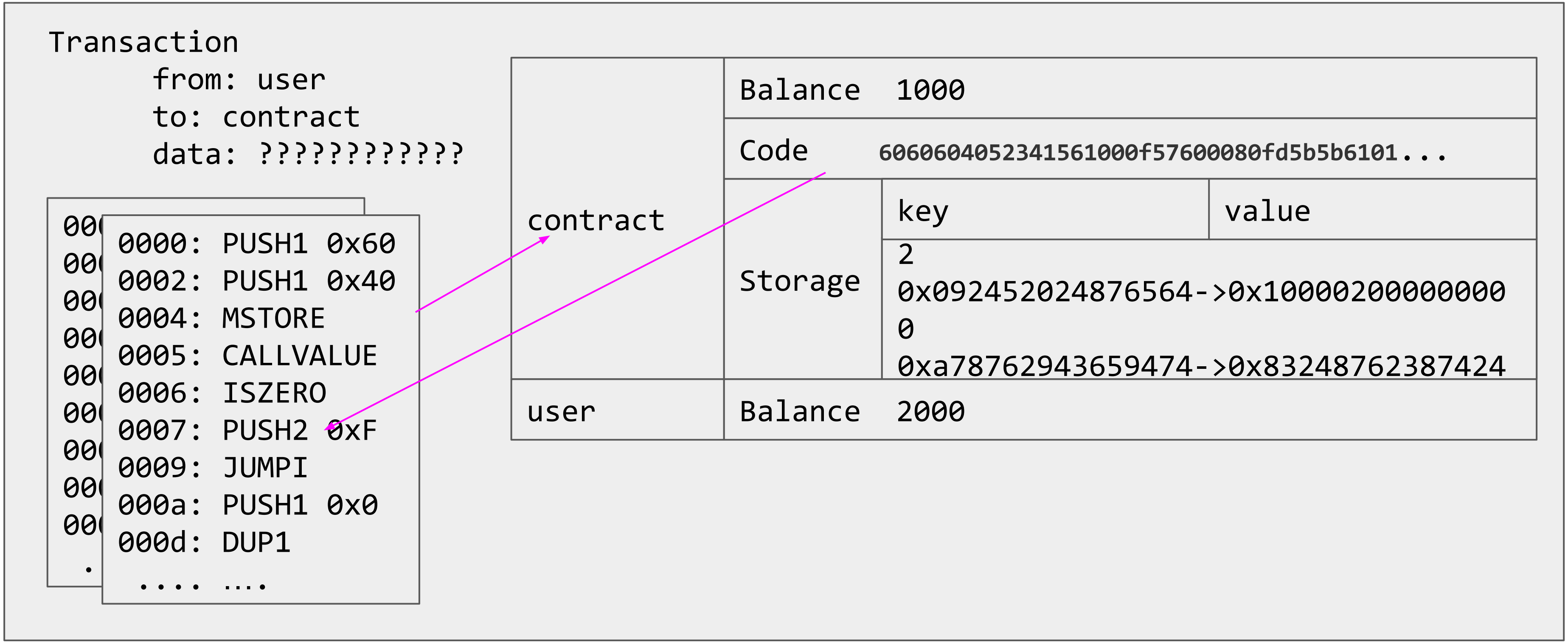


```
Transaction
      from: user
      to: contract
      data: ????????????
```

```
0000: PUSH1 0x60
0002: PUSH1 0x40
0004: MSTORE
0005: CALLVALUE
0006: ISZERO
0007: PUSH2 0xF
0009: JUMPI
000a: PUSH1 0x0
000d: DUP1
 .... ....
```

contract	Balance 1000		
	Code 6060604052341561000f57600080fd5b5b6101...		
	Storage	key	value
		2 0x092452024876564->0x10000200000000 0 0xa78762943659474->0x83248762387424	
user	Balance 2000		

Ethereum World - A transaction

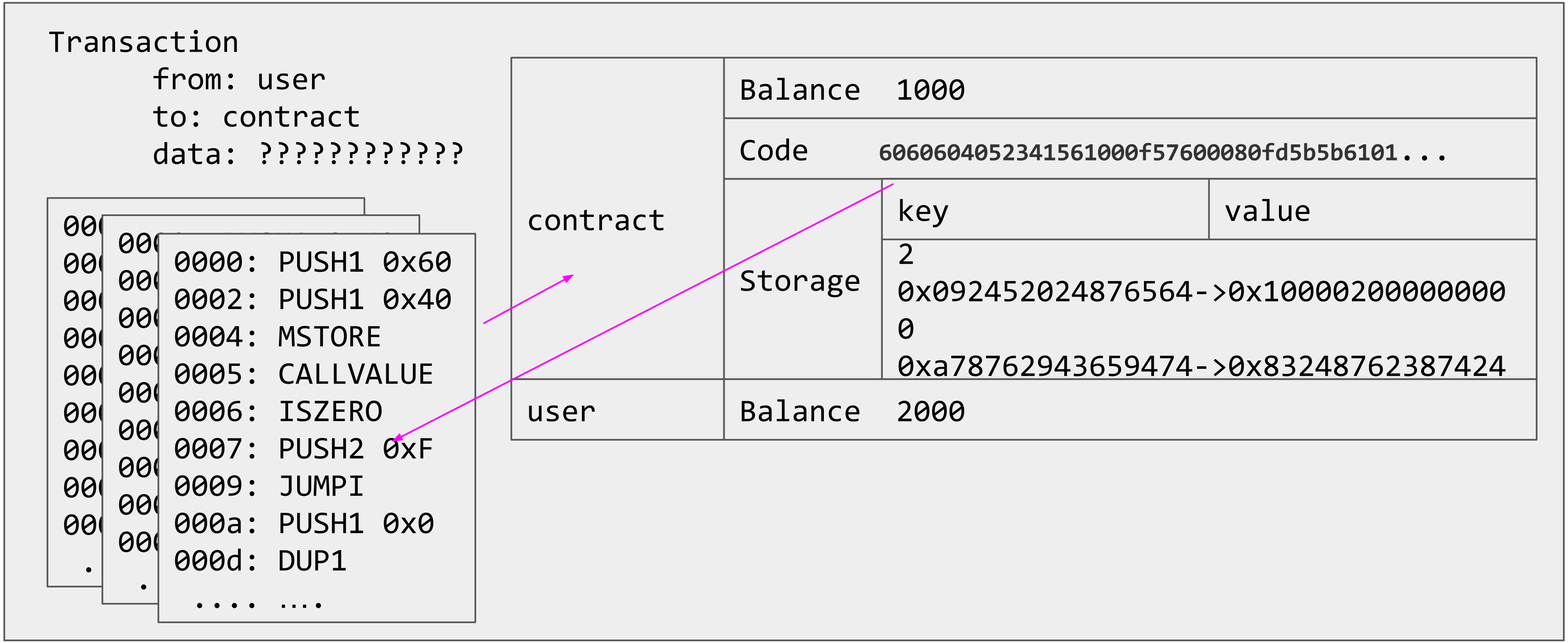


```
Transaction
      from: user
      to: contract
      data: ????????????
```

```
0000: PUSH1 0x60
0002: PUSH1 0x40
0004: MSTORE
0005: CALLVALUE
0006: ISZERO
0007: PUSH2 0xF
0009: JUMPI
000a: PUSH1 0x0
000d: DUP1
 .... ....
```

contract	Balance	1000	
	Code	6060604052341561000f57600080fd5b5b6101...	
	Storage	key	value
		2 0x092452024876564->0x10000200000000 0 0xa78762943659474->0x83248762387424	
user	Balance	2000	

Ethereum World - A transaction

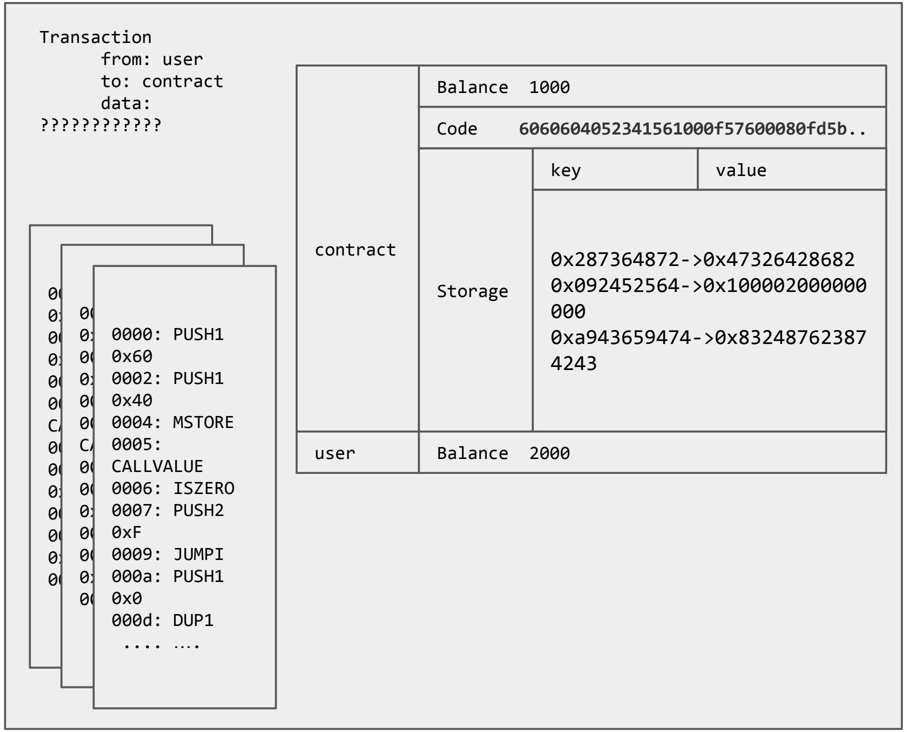


```
Transaction
      from: user
      to: contract
      data: ????????????
```

```
0000: PUSH1 0x60
0002: PUSH1 0x40
0004: MSTORE
0005: CALLVALUE
0006: ISZERO
0007: PUSH2 0xF
0009: JUMPI
000a: PUSH1 0x0
000d: DUP1
 .... ....
```

contract	Balance 1000		
	Code 6060604052341561000f57600080fd5b5b6101...		
	Storage	key	value
		2 0x092452024876564->0x10000200000000 0 0xa78762943659474->0x83248762387424	
user	Balance 2000		

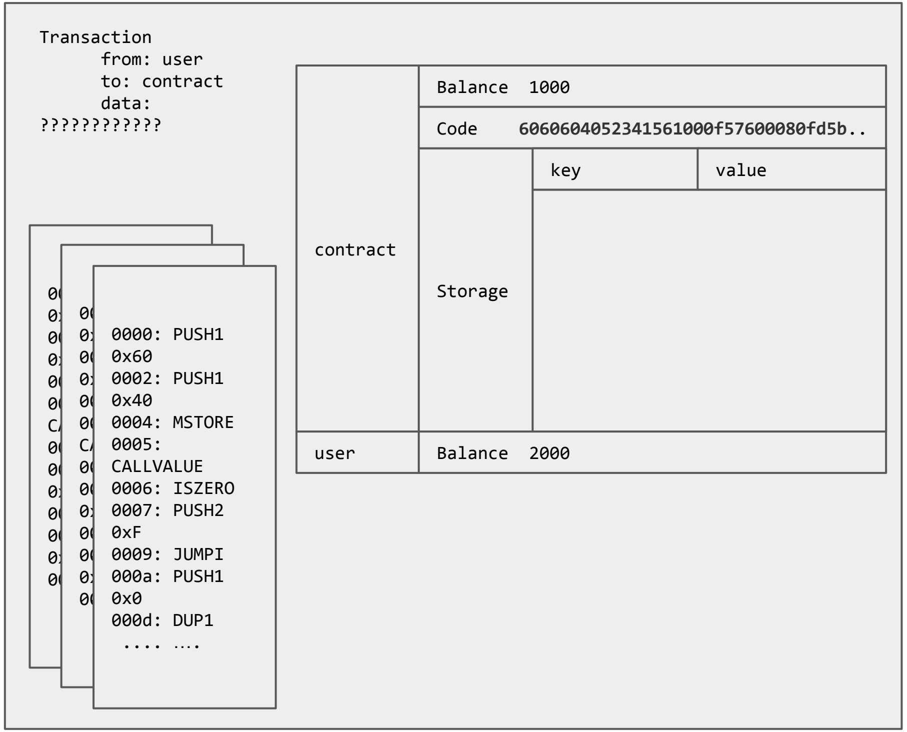
Ethereum World - Fork



```
Transaction
      from: user
      to: contract
      data:
????????????
```

contract	Balance	1000	
	Code	6060604052341561000f57600080fd5b..	
	Storage	key	value
		0x287364872->0x47326428682 0x092452564->0x100002000000000 0xa943659474->0x832487623874243	
user	Balance	2000	

```
0000: PUSH1
0x60
0002: PUSH1
0x40
0004: MSTORE
0005:
CALLVALUE
0006: ISZERO
0007: PUSH2
0xF
0009: JUMPI
000a: PUSH1
0x0
000d: DUP1
 .... ....
```



```
Transaction
      from: user
      to: contract
      data:
????????????
```

contract	Balance	1000	
	Code	6060604052341561000f57600080fd5b..	
	Storage	key	value
user	Balance	2000	

```
0000: PUSH1
0x60
0002: PUSH1
0x40
0004: MSTORE
0005:
CALLVALUE
0006: ISZERO
0007: PUSH2
0xF
0009: JUMPI
000a: PUSH1
0x0
000d: DUP1
 .... ....
```

Create an Ethereum contract

- Any account can create a contract ($)
- Solidity source code -> initialization bytecode
- The initialization bytecode sets up the storage and returns the runtime bytecode
- Constructor parameters are appended to the init bytecode

Toy Contract vs. Manticore

```
contract Simple {
    event Log(string);

    function() payable {
        if (msg.data[0] == 'A') {
            Log("Got an A");
        }else{
            Log("Got something else");
        }
    }
}
```

Toy Contract - Initialization

```
from seth import ManticoreEVM
seth = ManticoreEVM()

print "[+] Creating a user account"
user_account = seth.create_account(balance=1000)

print "[+] Creating a contract account"
bytecode = seth.compile(source_code)

print "[+] Creating a contract account"
contract_account = seth.create_contract(owner=user_account,
                                        init=bytecode)
```

Toy Contract - Initialization

```
from seth import ManticoreEVM
seth = ManticoreEVM()

print "[+] Creating a user account"
user_account = seth.create_account(balance=1000)

print "[+] Creating a contract account"
bytecode = seth.compile(source_code)

print "[+] Creating a contract account"
contract_account = seth.create_contract(owner=user_account,
                                        init=bytecode)
```

Toy Contract - Initialization

```
from seth import ManticoreEVM
seth = ManticoreEVM()

print "[+] Creating a user account"
user_account = seth.create_account(balance=1000)

print "[+] Creating a contract account"
bytecode = seth.compile(source_code)

print "[+] Creating a contract account"
contract_account = seth.create_contract(owner=user_account,
                                        init=bytecode)
```

Toy Contract - Initialization

```
from seth import ManticoreEVM
seth = ManticoreEVM()

print "[+] Creating a user account"
user_account = seth.create_account(balance=1000)

print "[+] Creating a contract account"
bytecode = seth.compile(source_code)

print "[+] Creating a contract account"
contract_account = seth.create_contract(owner=user_account,
                                        init=bytecode)
```

Toy Contract - Initialization

```
from seth import ManticoreEVM
seth = ManticoreEVM()

print "[+] Creating a user account"
user_account = seth.create_account(balance=1000)

print "[+] Creating a contract account"
bytecode = seth.compile(source_code)

print "[+] Creating a contract account"
contract_account = seth.create_contract(owner=user_account,
                                        init=bytecode)
```

Toy Contract - Initialization

```
from seth import ManticoreEVM
seth = ManticoreEVM()

print "[+] Creating a user account"
user_account = seth.create_account(balance=1000)

print "[+] Creating a contract account"
bytecode = seth.compile(source_code)

print "[+] Creating a contract account"
contract_account = seth.create_contract(owner=user_account,
                                        init=bytecode)
```

2 accounts

Toy Contract - Transaction

```
seth.transaction(caller=user_account,
                 address=contract_account,
                 data=seth.SByte(16),    #Symbolic buffer
                 value=seth.SValue       #Symbolic value
                 )
print "[+] There are %d reverted states now"% len(seth.final_state_ids)
for state_id in seth.final_state_ids:
    seth.report(state_id)

print "[+] There are %d alive states now"% len(seth.running_state_ids)
for state_id in seth.running_state_ids:
    seth.report(state_id)

print "[+] Global coverage:"
print seth.coverage(contract_account)
```

Toy Contract - Transaction

```
seth.transaction(caller=user_account,
                 address=contract_account,
                 data=seth.SByte(16),    #Symbolic buffer
                 value=seth.SValue       #Symbolic value
                 )
print "[+] There are %d reverted states now"% len(seth.final_state_ids)
for state_id in seth.final_state_ids:
    seth.report(state_id)

print "[+] There are %d alive states now"% len(seth.running_state_ids)
for state_id in seth.running_state_ids:
    seth.report(state_id)

print "[+] Global coverage:"
print seth.coverage(contract_account)    #Print covered instructions
```

Reading Manticore results

```
====================
REPORT: STOP
LOG: 0xa9eb72624f93de30ccd1118fbccfb637cd367b35L "Got something else"
buffer_1: 01 040000000000000000000000000000

====================
REPORT: STOP
LOG: 0xa9eb72624f93de30ccd1118fbccfb637cd367b35L "Got an A"
buffer_1: 41 010000000000000000000000000000

Total assembler lines: 131
Total assembler lines visited: 111
Coverage: 84.73 %
```

Example - Integer overflow

```
pragma solidity ^0.4.15;

contract Overflow {
    uint private sellerBalance=0;

    function add(uint value) returns (bool){
        sellerBalance += value; // complicated math with possible overflow

        // possible auditor assert
        assert(sellerBalance >= value);
    }
}
```

Example - Integer overflow

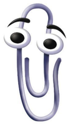
```
pragma solidity ^0.4.15;

contract Overflow {
    uint private sellerBalance=0;

    function add(uint value) returns (bool){
        sellerBalance += value; // complicated math with possible overflow

        // possible auditor assert
        assert(sellerBalance >= value);
    }
}
```

Needs two transactions



Example - Integer overflow - Initialization

```
from seth import *
seth = ManticoreEVM()

#Initialize user and contracts
user_account = seth.create_account(balance=1000)
bytecode = seth.compile(source_code)

contract_account = seth.create_contract(owner=user_account,
                                        balance=0,
                                        init=bytecode)
```

2 accounts

Example - Integer overflow - 2 Transactions

```
#First add will not overflow uint256 representation
symbolic_data = seth.make_function_call('add(uint256)', seth.Svalue)
seth.transaction(  caller=user_account,
                              address=contract_account,
                              value=0,
                              data=symbolic_data,
                           )
```

tx1

```
#Potential overflow
symbolic_data = seth.make_function_call('add(uint256)', seth.Svalue)
seth.transaction(  caller=user_account,
                              address=contract_account,
                              value=0,
                              data=symbolic_data
                           )
```

tx2

Example - Integer overflow - Reporting

```
print "[+] There are %d reverted states now"% len(seth.final_state_ids)
for state_id in seth.final_state_ids:
    seth.report(state_id)

print "[+] There are %d alive states now"% len(seth.running_state_ids)
for state_id in seth.running_state_ids:
    seth.report(state_id)

print "[+] Global coverage: %x"% contract_account
print seth.coverage(contract_account)
```

Reading Manticore results

```
REPORT: THROW
     data_1: 1003e2d2
8000000000000000000000000000000000000000000000000000000000000000
     data_3: 1003e2d2
7000000000000000000000000000000000000000000000000000000000000000
BALANCES
     0xd30a286ec6737b8b2a6a7b5fbb5d75b895f62956L 1000
     0x1bfa530d5d685155e98cd7d9dd23f7b6a801cfefL 0

REPORT: RETURN
     data_1: 1003e2d2
0200000000000000000000000000000000000000000000000000000000000000
     data_3: 1003e2d2
0100000000000000000000000000000000000000000000000000000000000000
BALANCES
     0xd30a286ec6737b8b2a6a7b5fbb5d75b895f62956L 1000
     0x1bfa530d5d685155e98cd7d9dd23f7b6a801cfefL 0

[+] Global coverage: 76.47%
```

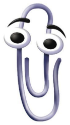
Symbolic hash

The SHA3 problem

```
buffer = msg.data; // symbolic free data

if (sha3(buffer) == 0x11223344){
    do_something();
}
else{
    do_something_else();
}
```

Solidity `mappings` are implemented with sha3()



The SHA3 problem - Solutions?

- Return a free symbolic hash -> False positives

```
sha3(symbolic_buffer) -> free_256bitvector
```

- Concretization and fork over known hashes
- Symbolic SHA3 over known solutions ⇐

Symbolic SHA3 over known solutions

```
hash("Nunca escapa el cimarron") -> 0xbfbf649c
hash("Que dispara por la loma") -> 0x04844965

sbuffer = input()
hash(sbuffer) ????

ITE(sbuffer == "Nunca escapa el cimarron",  0xbfbf649c
     ITE(sbuffer == "Si dispara por la loma" , 0x04844965,
                                                        ...)
```

Handling mappings

```
contract Test {
    event Log(string);
    mapping(address => uint) private balances;
    function Test(){
        balances[0x11111111111111111111111111111111] = 10;
        balances[0x22222222222222222222222222222222] = 20;
        balances[0x33333333333333333333333333333333] = 30;
        balances[0x44444444444444444444444444444444] = 40;
        balances[0x55555555555555555555555555555555] = 50;
    }
    function target(address key) returns (bool){
        if (balances[key] > 20)
            Log("Balance greater than 20");
        else
            Log("Balance less or equal than 20");
    }
}
```

Handling mappings

```
contract Test {
    event Log(string);
    mapping(address => uint) private balances;
    function Test(){
        balances[0x11111111111111111111111111111111] = 10;
        balances[0x22222222222222222222222222222222] = 20;
        balances[0x33333333333333333333333333333333] = 30;
        balances[0x44444444444444444444444444444444] = 40;
        balances[0x55555555555555555555555555555555] = 50;
    }
    function target(address key) returns (bool){
        if (balances[key] > 20)
            Log("Balance greater than 20");
        else
            Log("Balance less or equal than 20");
    }
}
```

Known hashes

Handling mappings

```
contract Test {
    event Log(string);
    mapping(address => uint) private balances;
    function Test(){
        balances[0x11111111111111111111111111111111] = 10;
        balances[0x22222222222222222222222222222222] = 20;
        balances[0x33333333333333333333333333333333] = 30;
        balances[0x44444444444444444444444444444444] = 40;
        balances[0x55555555555555555555555555555555] = 50;
    }
    function target(address key) returns (bool){
        if (balances[key] > 20)
            Log("Balance greater than 20");
        else
            Log("Balance less or equal than 20");
    }
}
```

Known hashes

Make special expression

Reading Manticore results

```
====================
REPORT: REVERT
msg.data_1: 0000002000000000000000000000000000000000000000000000000000000000
            0000000000000000000000000000000000000000000000000000000000000000
====================
REPORT: RETURN
LOGS: ”Balance greater than 20”
msg.data_1: dad9da89000000000000000000000000000000005555555555555555555555555
            5555555555555555555555555555555555555555555555555555555555555555
====================
REPORT: RETURN
LOGS: “Balance less or equal than”
msg.data_1: dad9da89000000000000000000000000000101010000000000000000000000000
            0000000000000000000000000000000000000000000000000000000000000000
```

many solutions

Conclusions & Future Work

Conclusions

- Smart contracts on the blockchain is a new technology
  - Already a lot of money = good target for attackers
  - Developers are not always aware of the security best practices
  - We need more usable tools to perform audits

- We will probably see other large hacks in a near future
  - There is a need for contract verification/analysis

- EVM is a good fit for Symbolic Execution
  - Gas limitation, not many paths
- No memory safety heuristics
  - Need a human to provide require() and assert()

Manticore - Further work

- Add gas support
  - Calculate real max gas spent on functions
- Bindiff between 2 versions of the same contract:
  - contractA(symbolic_input_a) == contractB(symbolic_input_a)
- Add ABI helpers for building input
- Add vulnerabilities detection heuristics
- Special instruction for meta-assert

Ok, entonces todavía falta para poder vaciar la blockchain. 😕

Other Tools for Audits

- https://github.com/hrishioa/Oyente (symbolic executor)
    - Paper: Making Smart Contracts Smarter
    - Detects: call stack / concurrency / time dependency / reentrancy
- https://github.com/pirapira/dry-analyzer (symbolic executor)
    - “Dr. Y's Ethereum Contract Analyzer”
- https://ethereum.github.io/browser-solidity (static analysis)
    - Detects: similar variables names, re-entracy
- http://securify.ch/ (static analysis-based verification)
    - Still in development

Manticore Github

https://github.com/trailofbits/manticore/tree/dev-evm-eko

> trailofbits/manticore
>
> manticore - Dynamic binary analysis tool

Thanks!