ITU – NBTC Training
On

“Building Distributed Ledger Technologies (Blockchain) Projects”

5 – 8 November 2019,
Bangkok, Thailand

Session 6: Working Group

“Introduction to Ethereum”

Jamie Cerexhe

Contents

- Landscape of popular DLT platforms
- Why Ethereum?
- Purpose of Ethereum
- History
- How does Ethereum work?
- Criticisms of Ethereum
- Solidity basics
- Dapp Development

Landscape of popular DLT platforms

- Hyperledger Fabric
- EOS
- Corda
- NEO
- Waves
- Nem
- Ethereum
- Ardor

HYPERLEDGER

NEO
smart economy

Why Ethereum?

Not the only protocol in this space. We will focus on Ethereum this week because:

- It has good examples of successes and failures
- It is the most well known in the DLT community
- Its concepts transfer well to other protocols
- It has helped lead many of the DLT standards

It is not the best or worst protocol. Always do your research to find which solution is best for your use-case.

Purpose of Ethereum

Ethereum is an open-source public platform, distributed and based on blockchain technology, to run applications without censorship or third-party interference.

It has a native currency, *ether*.

It can run programs on a *distributed virtual machine*. These are also known as Smart Contracts.

ethereum

History

9 May, 2015 Olympic: Public testing of the Ethereum protocol before release

20 July, 2015 Frontier: Ethereum genesis block mined

14 March, 2016 Homestead: First planned hard fork, new version of Solidity, introduced Mist wallet

20 July, 2016 (DAO Fork): The DAO was hacked and a hard fork occurred to reverse the hack. The original chain became Ethereum Classic

16 October, 2017 Metropolis (Byzantium): Changes to mining. Introduced changes for transition to Proof of Stake

28 February, 2018 Metropolis (Constantinople): Implemented a number of EIPs, reduced block reward, delayed Difficulty Bomb (intended to help transition towards PoS but PoS not ready yet)

? Istanbul & Serenity: Ethereum 2.0 (Proof of Stake)
https://docs.ethhub.io/ethereum-roadmap/ethereum-2.0/eth-2.0-phases/

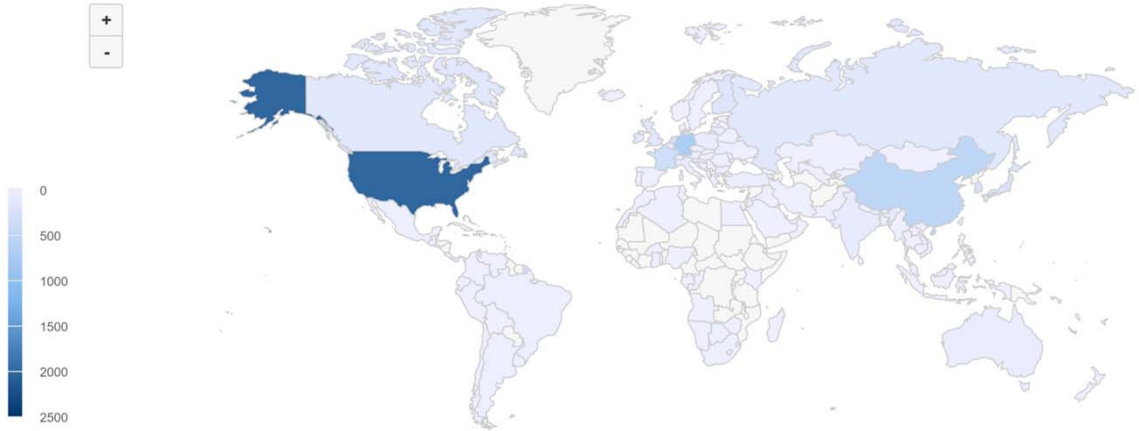
Key Components

ethereum

The Ethereum Network

Made up of over 8000 nodes, distributed across the world.

Every node holds the *state* of the Ethereum network.



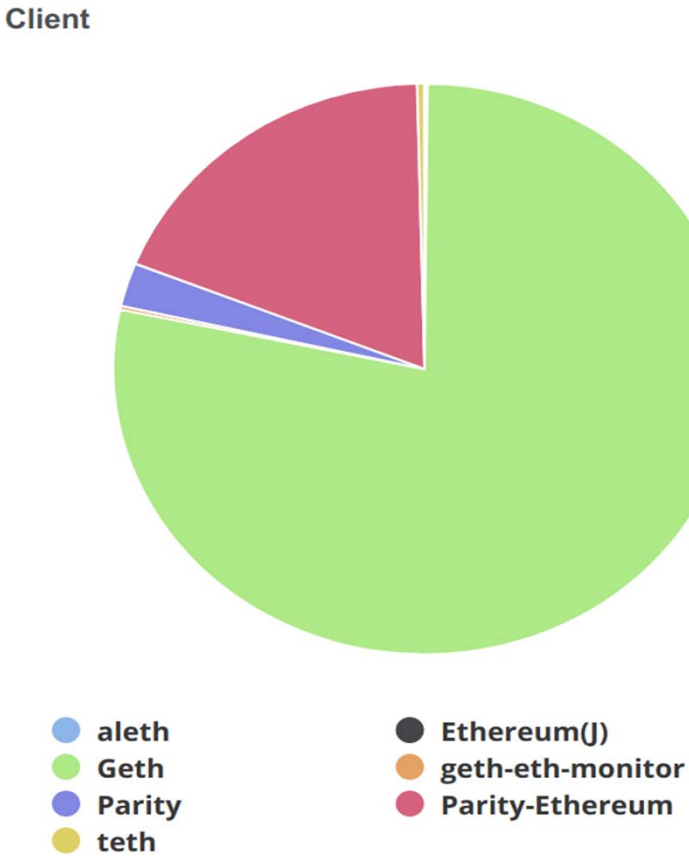
Ethereum Nodes

Ethereum is an open standard and supports multiple nodes:

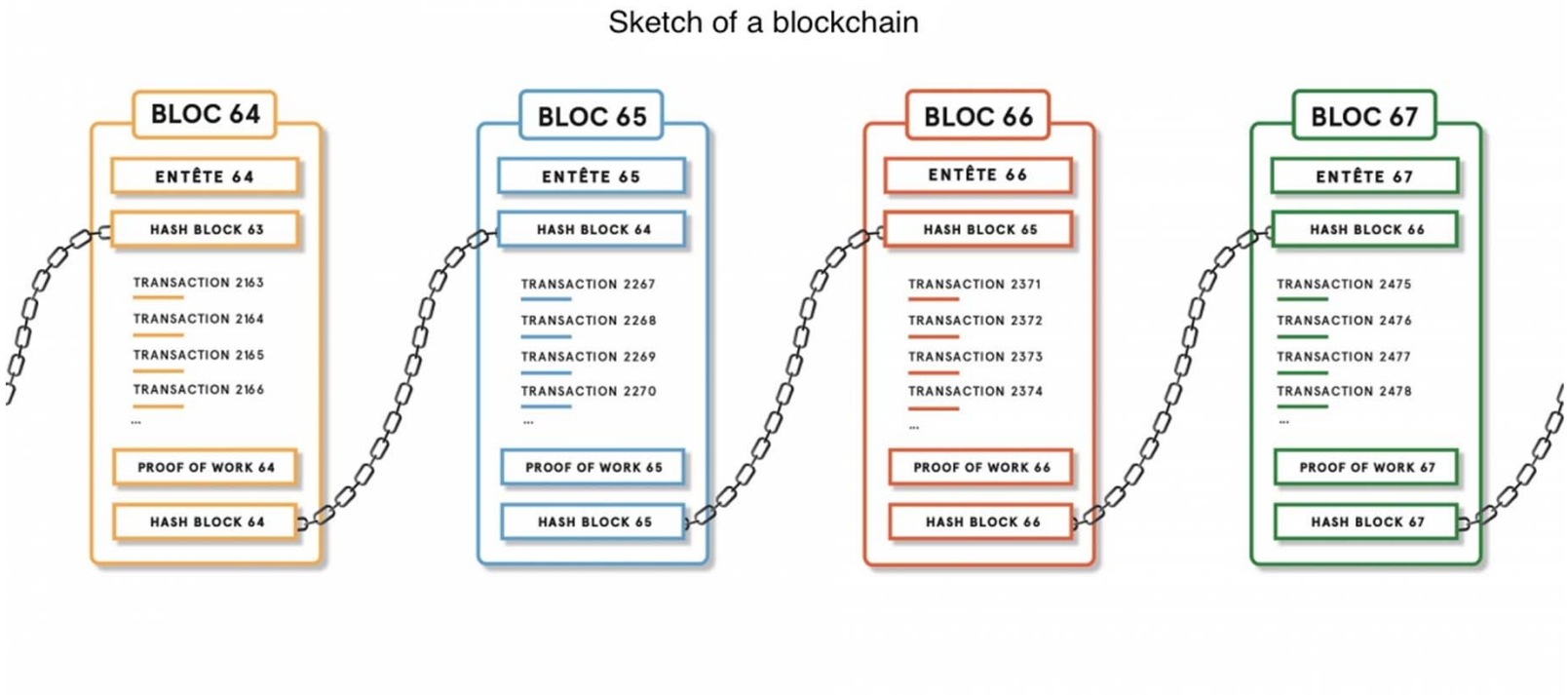
- Geth (Ethereum Foundation)
- Parity (Parity Technologies)
- Quorum (J.P. Morgan)

Eth-2.0 (in progress) nodes:

- Pantheon
- Lighthouse
- Prysm
- Nimbus



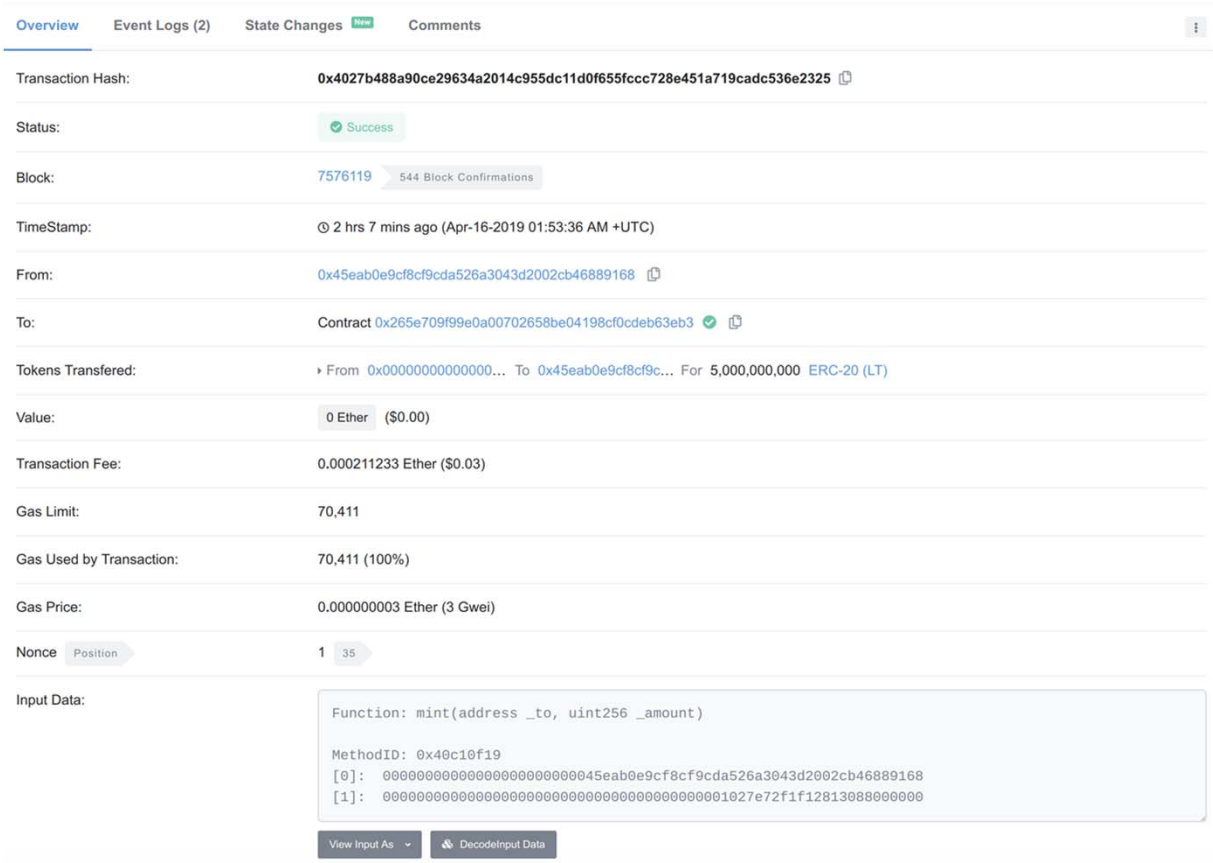
Blocks


Sketch of a blockchain
BLOC 64
ENTÊTE 64
HASH BLOCK 63
TRANSACTION 2163
TRANSACTION 2164
TRANSACTION 2165
TRANSACTION 2166
...
PROOF OF WORK 64
HASH BLOCK 64
BLOC 65
ENTÊTE 65
HASH BLOCK 64
TRANSACTION 2267
TRANSACTION 2268
TRANSACTION 2269
TRANSACTION 2270
...
PROOF OF WORK 65
HASH BLOCK 65
BLOC 66
ENTÊTE 66
HASH BLOCK 65
TRANSACTION 2371
TRANSACTION 2372
TRANSACTION 2373
TRANSACTION 2374
...
PROOF OF WORK 66
HASH BLOCK 66
BLOC 67
ENTÊTE 67
HASH BLOCK 66
TRANSACTION 2475
TRANSACTION 2476
TRANSACTION 2477
TRANSACTION 2478
...
PROOF OF WORK 67
HASH BLOCK 67

Transactions - ether

Transaction Hash:	**0x11bc2bb02cadd4a0e936547bcd4f322ee4e088d4a62b2f7b1b358b4cf6273064**
Status:	Success
Block:	7576579 7 Block Confirmations
TimeStamp:	1 min ago (Apr-16-2019 03:39:42 AM +UTC)
From:	0x9334f0aa74d2744b97b0b1be6896788ee46f4aaa
To:	0xae1858e7927596912ad3cbe3db035e764331ddc1
Value:	0.00011566 Ether ($0.02)
Transaction Fee:	0.000044584 Ether ($0.007191)

Transactions – smart contracts



Overview | Event Logs (2) | State Changes New | Comments

Transaction Hash:	0x4027b488a90ce29634a2014c955dc11d0f655fccc728e451a719cadc536e2325
Status:	Success
Block:	7576119 544 Block Confirmations
TimeStamp:	2 hrs 7 mins ago (Apr-16-2019 01:53:36 AM +UTC)
From:	0x45eab0e9cf8cf9cda526a3043d2002cb46889168
To:	Contract 0x265e709f99e0a00702658be04198cf0cdeb63eb3
Tokens Transfered:	From 0x00000000000000... To 0x45eab0e9cf8cf9c... For 5,000,000,000 ERC-20 (LT)
Value:	0 Ether ($0.00)
Transaction Fee:	0.000211233 Ether ($0.03)
Gas Limit:	70,411
Gas Used by Transaction:	70,411 (100%)
Gas Price:	0.000000003 Ether (3 Gwei)
Nonce Position	1 35
Input Data:	Function: mint(address _to, uint256 _amount) MethodID: 0x40c10f19 [0]: 00000000000000000000000045eab0e9cf8cf9cda526a3043d2002cb46889168 [1]: 0000000000000000000000000000000000000000001027e72f1f12813088000000

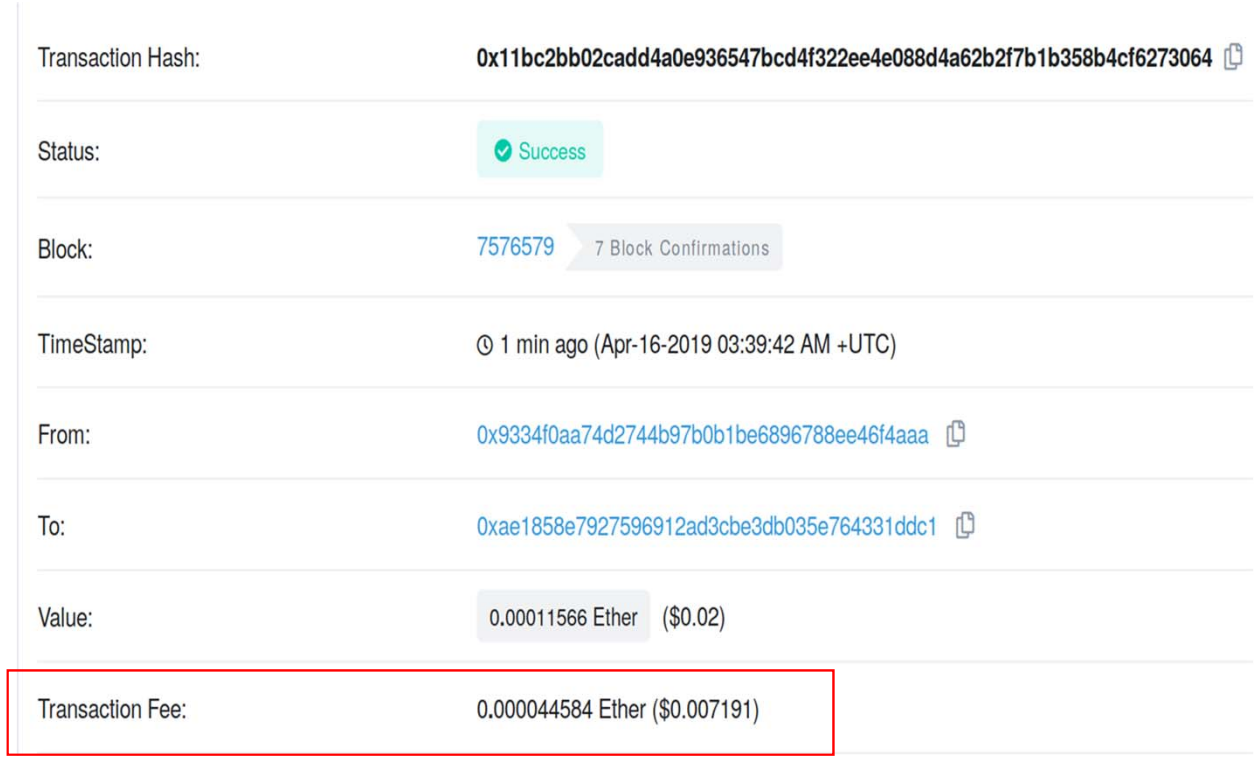
View Input As | DecodeInput Data

Transaction fees

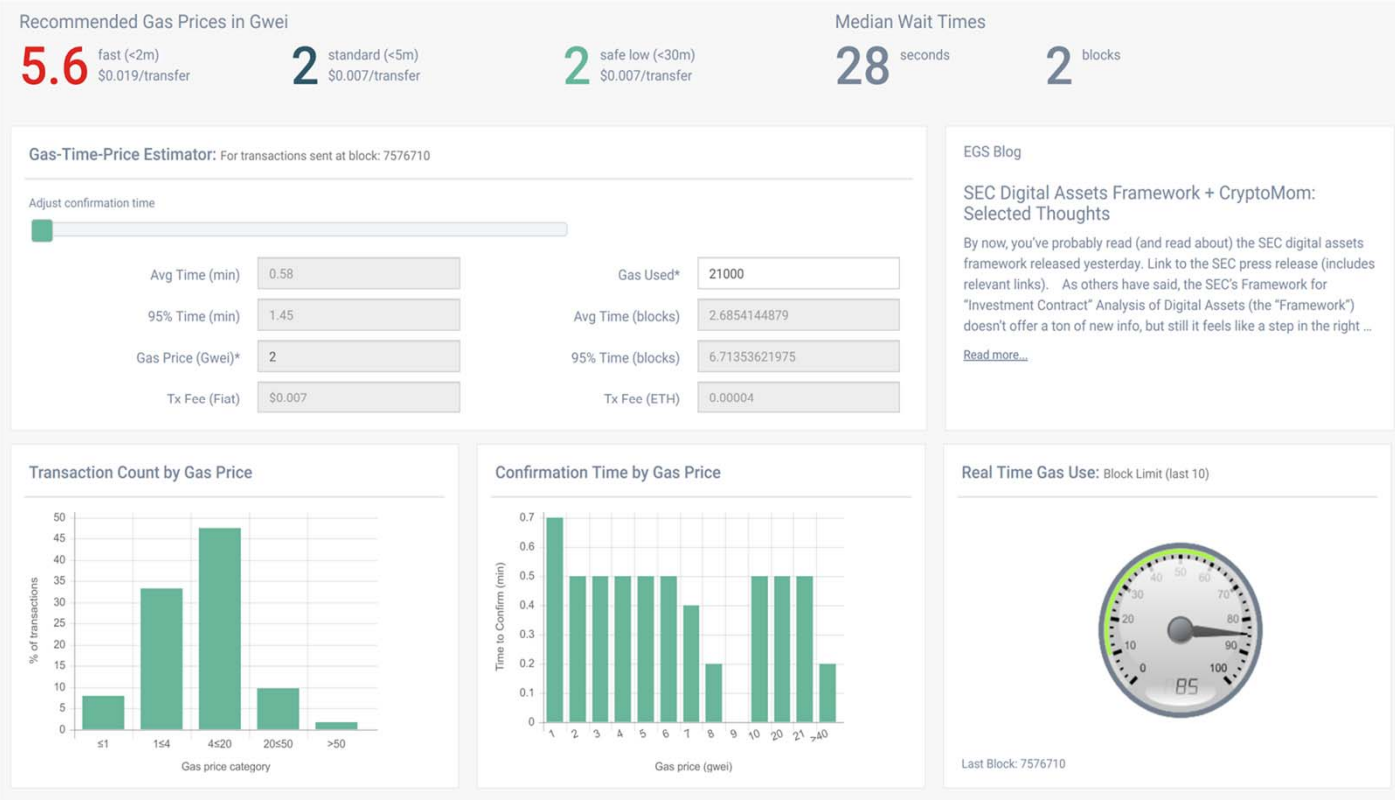
Transaction fees are proportional to the size of the transaction.

Transaction fees are paid according to *gas*, a unit of transaction complexity.

Transaction fee = gas price * gas used.



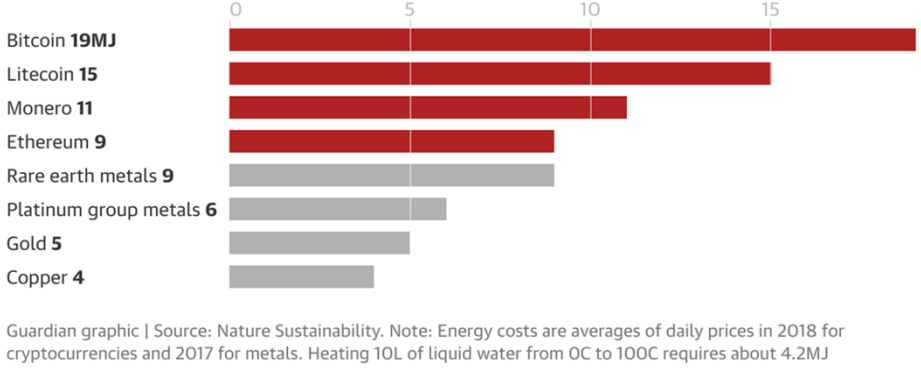
Fees – you get what you pay for

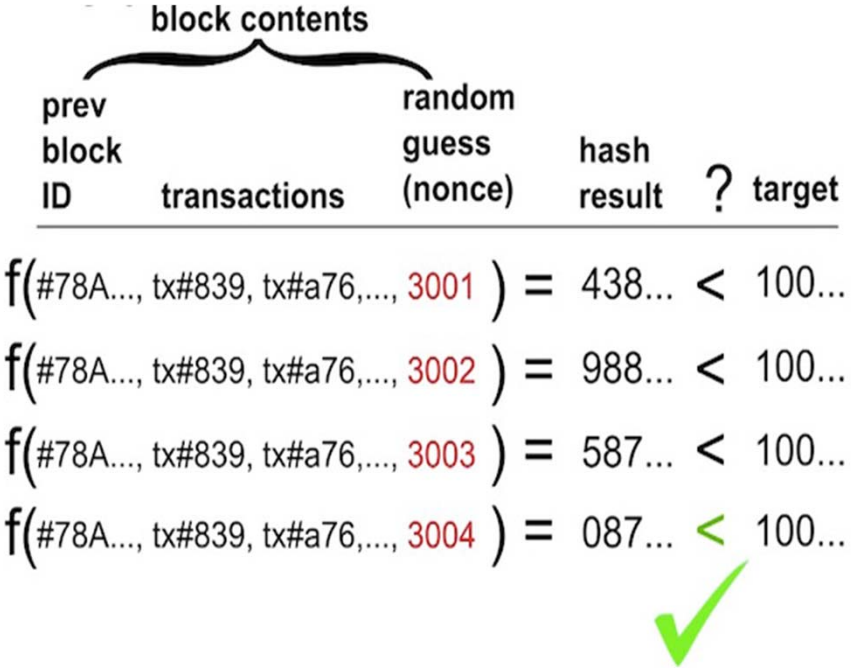


Mining

Ethereum uses a *proof-of-work* mechanism for security.

Nodes must solve time-consuming calculations to make it prohibitively expensive to attack the network.





Accounts

An account is generated from a private key.

Accounts can hold ether and perform actions on smart contracts.

An account has an *address*, derived from its private key.

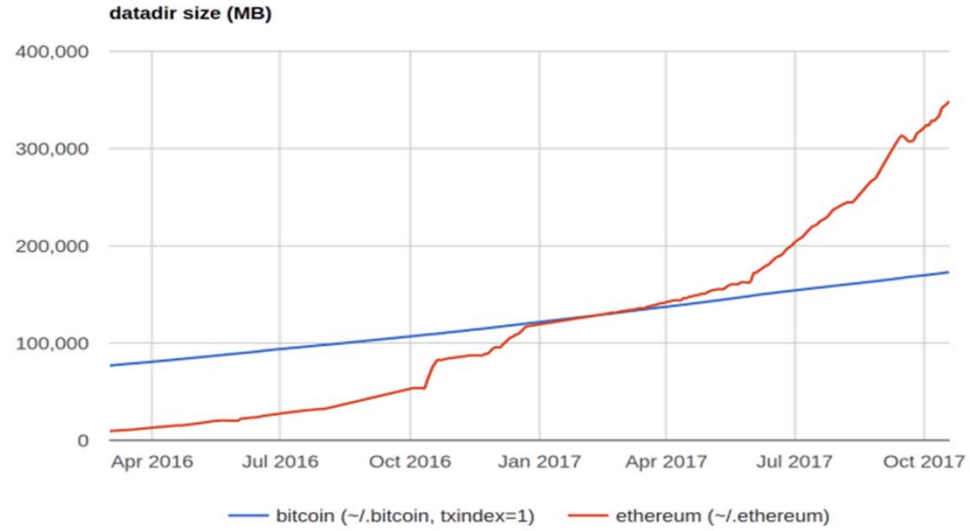
Storage

The uncompressed blockchain is currently 231.99 GB.

The compressed / pruned blockchain is about 50GB.

Light clients can help reduce the size stored in a node to 100MB.

(Numbers are higher now)



Dealing with failure

Bugs in smart contracts can lead to loss of funds

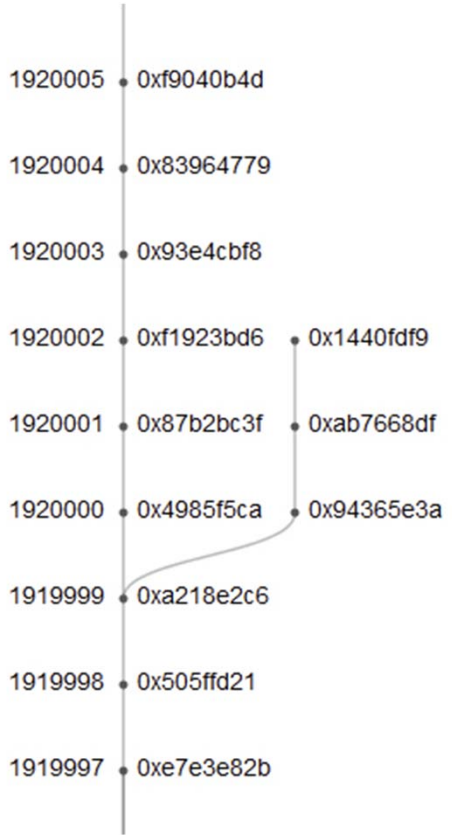
DAO hack: 3.6 million ETH lost (678 million USD)

Parity wallet hack: 150,000 ETH lost (28 million USD)

Dealing with failure: hard fork

A *hard fork* can fix a bug or attack that has already occurred.

The first hard fork fix recovered the funds stolen from the DAO attack.



Dealing with failure: upgradable contracts

Upgradeable contracts can fix bugs before they are exploited.

But they are difficult to write and error-prone.

Ethereum Governance

Ethereum is a public protocol with no central governance.

The Ethereum Foundation funds research and development.

Governance decisions are made by consensus.

BTFP

ERCs

A community process for standardising and improving Ethereum.

Final

Number	Title	Author
20	ERC-20 Token Standard	Fabian Vogelsteller, Vitalik Buterin
55	Mixed-case checksum address encoding	Vitalik Buterin, Alex Van de Sande
137	Ethereum Domain Name Service - Specification	Nick Johnson
162	Initial ENS Hash Registrar	Maurelian, Nick Johnson, Alex Van de Sande
165	ERC-165 Standard Interface Detection	Christian Reitwießner, Nick Johnson, Fabian Vogelsteller, Jordi Baylina, Konrad Feldmeier, William Entriken
181	ENS support for reverse resolution of Ethereum addresses	Nick Johnson

Criticisms of Ethereum

- Lack of scalability
- Lack of progression of roadmap
- Migration to PoS may cause huge issues for existing platforms
- Cult-like following of Ethereum's founder, Vitalik Buterin. Single point of failure with huge influence
- Speculative investment in ether
- Solidity not very approachable
- Storage size needed to host a node

BTFP

Solidity

Solidity is a contract-oriented high-level language, with similar syntax to JavaScript. It is statically typed, supports inheritance, libraries and complex user-defined types. It compiles to EVM assembly, which is run by the nodes.

Benefits of using Solidity:

- Most popular Ethereum smart contract language
- Syntax similar to C and JavaScript
- Supported by almost Ethereum tools

Example: declare version

The compiler pragma locks in the compiler version to use.

```
pragma solidity ^0.5.0;
```

BTFP

Example: contract keyword

A *contract* is similar to a class in Python or Java. It encapsulates state and methods.

```
contract NewSmartContract {
    ...
}
```

BTFP

Example: variables

Contracts can have public and private variables, which are stored on the blockchain. Like C, variables are explicitly typed.

```
contract NewSmartContract {
    bytes32[] private varNames;
    mapping(bytes32 => string) private varValues;
    ...
}
```

BTFP

Example: arrays

Arrays can be fixed-size (like C) or dynamically-sized (like JavaScript).

```
contract NewSmartContract {
    bytes32[] private arrayName;
    ...
}
```

Solidity

```
class NewSmartContract {
  let arrayName = [];
}
```

JavaScript

Example: mappings

Mappings are similar to a Python dictionary, but can only map from one type to another.

```
contract NewSmartContract {
    mapping(bytes32 => string) private newDict;
    ...
}
```

Solidity

```
class NewSmartContract :
  newDict = {}
```

Python

BTFP

Example: constructor

The constructor function initialises the initial contract data.

```
constructor(bytes32[] memory _ varNames) public {
    varNames = _ varNames;
}
```

BTFP

Example: storage vs memory

Variables in *memory* are discarded after a transaction completes.
Variables in *storage* are saved to the blockchain.

```
constructor(bytes32[] memory _reservedKeys) public {
    reservedKeys = _reservedKeys;
}
```

storag
e

memory

Example: public functions

Public functions can be called by any account.

```
function setValue(bytes32 _key, string memory _value) public {
    require(!isReservedKey(_key));

    values[_key] = _value;
}
```

Example: require

require statements enforce rules. If they fail, the whole transaction will fail.
They are equivalent to asserts in C.

```
require(!isReservedKey(_key));
```

Solidity

```
assert(!isReservedKey(_key));
```

C

Example: function returns

Functions must explicitly state their return type.

```
function getValue(bytes32 _key) public view returns (string memory) {
    return values[_key];
}
```

Example: private functions

Private functions can only be called from within the contract. *View* functions can’t change state.

```
function isReservedKey(bytes32 _key) private view returns (bool) {
    for (uint256 i = 0; i < reservedKeys.length; i++) {
        if (reservedKeys[i] == _key) {
            return true;
        }
    }

    return false;
}
```

BTFP

Example: loops

For loops use the same syntax as JavaScript.

```
for (
    uint256 i = 0;
    i < myArray.length;
    i++) {
    ...
}
```

Solidity

```
for (
    let i = 0;
    i < myArray.length;
    i++) {
    ...
}
```

JavaScript

BTFP

Example: if statements

If statements use the same syntax as JavaScript.

```
if (reservedKeys[i] == _key) {
    ...
}
```

Solidity

```
if (reservedKeys[i] === _key) {
    ...
}
```

JavaScript

Dapp Development

Development of dapps involves a few differences to normal software.

1. Local environment
    a. Nodes
    b. Network
2. Interacting with the blockchain
    a. Metamask
    b. Seth
    c. Remix
3. Frameworks
4. Testing
5. Security tools

Local environment - nodes

When developing Dapps we need a local node to test against.

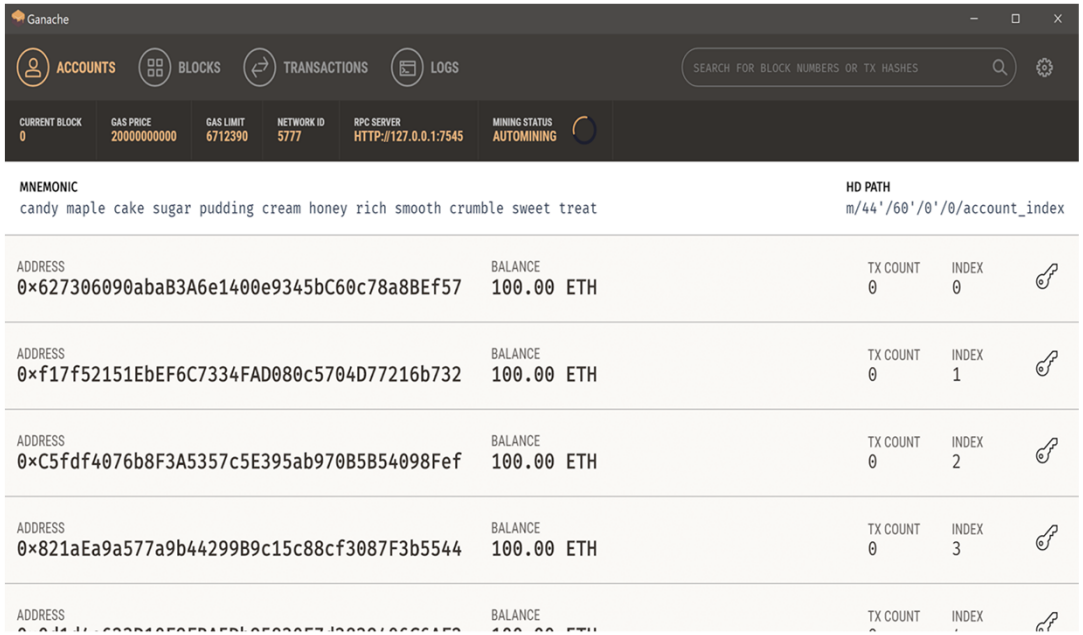
Could use:

- Geth/Parity (production nodes)
- Ganache (test node)

Ganache

Test environment with support for:

- Creating accounts with free ether
- Viewing transactions and blocks
- Connecting external tools to simulate an Ethereum network
- Decoding smart contract events



Local environment - networks

It’s a good idea to test using a network of multiple nodes.

Advantages over using a single node:

- More realistic transaction speed
- Simulate failures

Create local networks using production nodes like Geth or Quorum.

Interacting with the blockchain

Once a node is running, there are many tools that can interact with it:

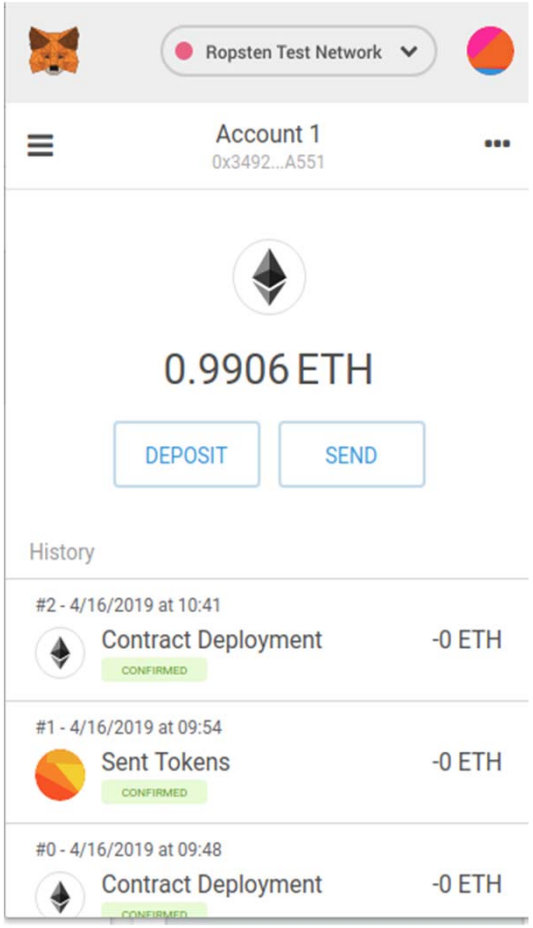
- Metamask
- Seth
- Remix

Metamask

Browser extension for interacting with Dapps.

Supports accounts, transactions, and interacting with contracts.

Can connect to main network, testnets, and local development nodes.



Seth

Low-level command-line utility for interacting with blockchain nodes.

```
# Call a function on a contract
> seth call 0xa4beda0e8e94a9750c02e8f75278416f7d541f61 "getVotes()(uint256)"
567

# View information about a transaction
> seth tx 0xb3c2f71b4ad33799c1a9ed5a37d45424bb7c028c66c0965d58401b1ffb270d0a
blockHash               0x403a6c8c2932aef797f16825dfc9507e3c8ff59e80b8282d6cf06fd310eddd16
blockNumber             253
from                    0x37620ab71430d186e70ffbc77189e7385de51858
gas                     6721975
gasPrice                0
hash                    0xb3c2f71b4ad33799c1a9ed5a37d45424bb7c028c66c0965d58401b1ffb270d0a
...
```

Remix

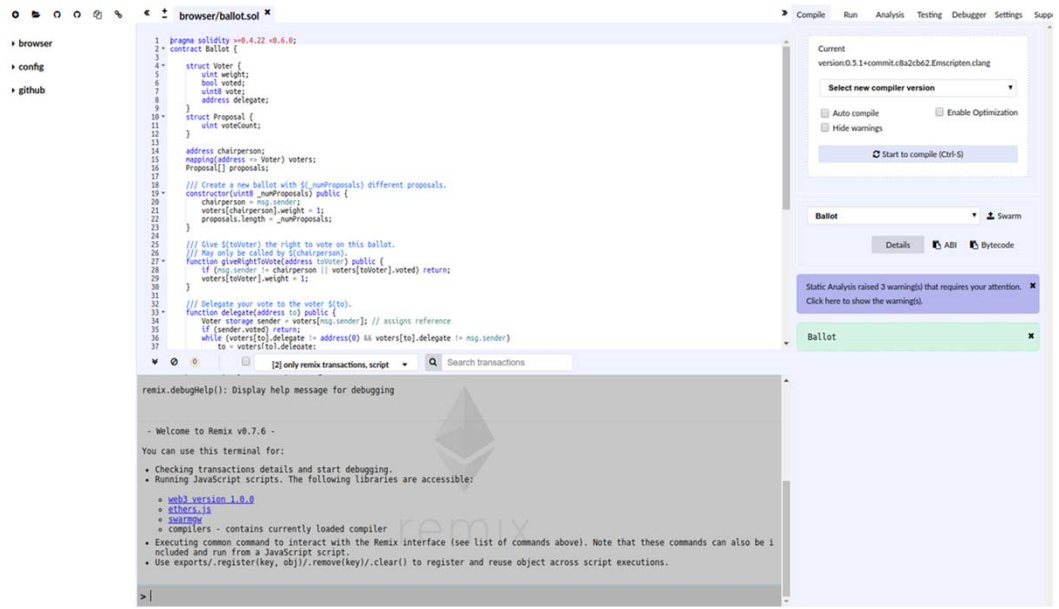
Integrated development environment (IDE) for developing smart contracts.

Support for compiling, deploying, interacting with, and debugging smart contracts.

Interacting with contracts

Call and send transactions to contracts.

Use accounts from Metamask, or run on an in-browser development blockchain.

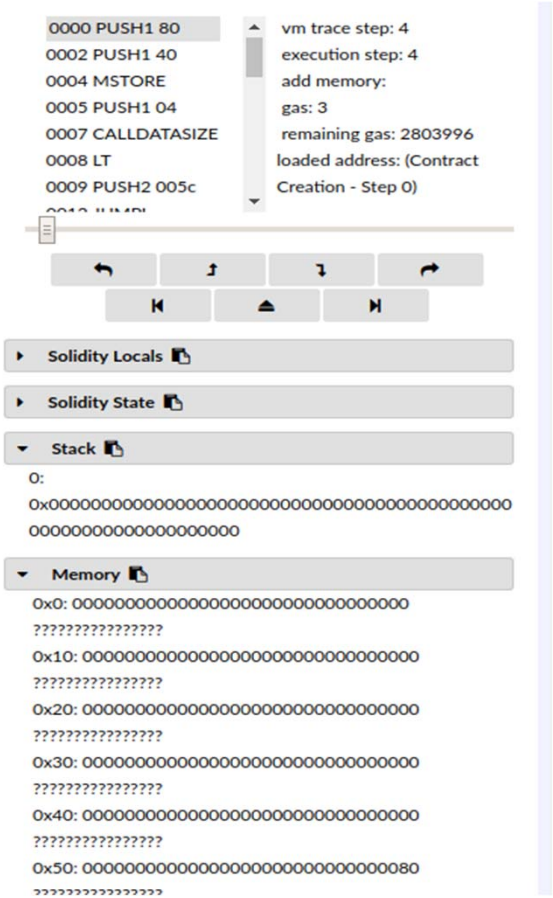


Remix debugging

Step-by-step debugging.

Show bytecode next to source code functions.

Inspect memory layout, state, and stack.

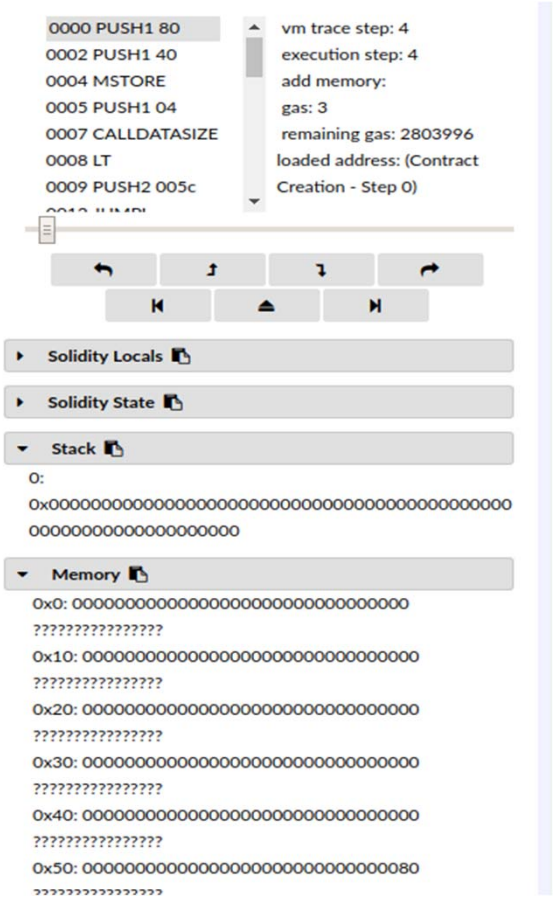


Remix debugging

Step-by-step debugging.

Show bytecode next to source code functions.

Inspect memory layout, state, and stack.



Frameworks

Solutions for combining front-end, back-end, and smart contracts.

Notable frameworks:

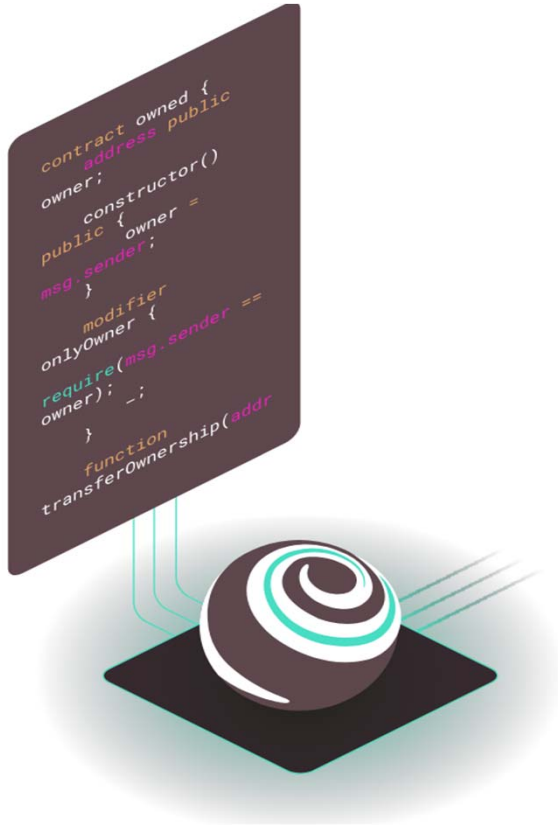
- Truffle
- Embark
- DappTools

Frameworks – Truffle

JavaScript-based framework for Dapp development.

Supports:

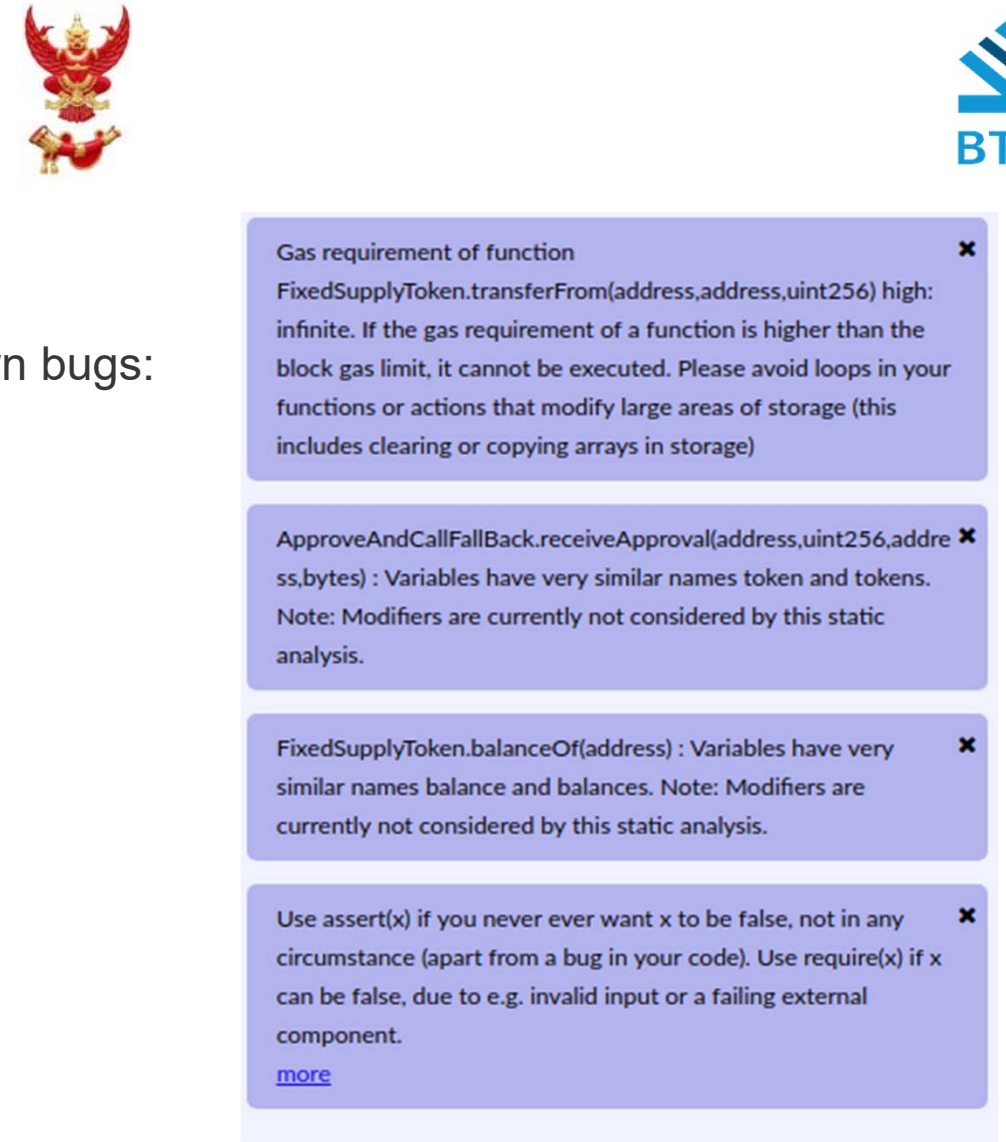
- Automated tests and deployment of smart contracts
- Interacting with smart contracts from front- and back-end
- React, Vue, and other front-end frameworks
- Plugins for extra functionality
- Migration of contracts



Security tools

Static analysis tools check code for known bugs:

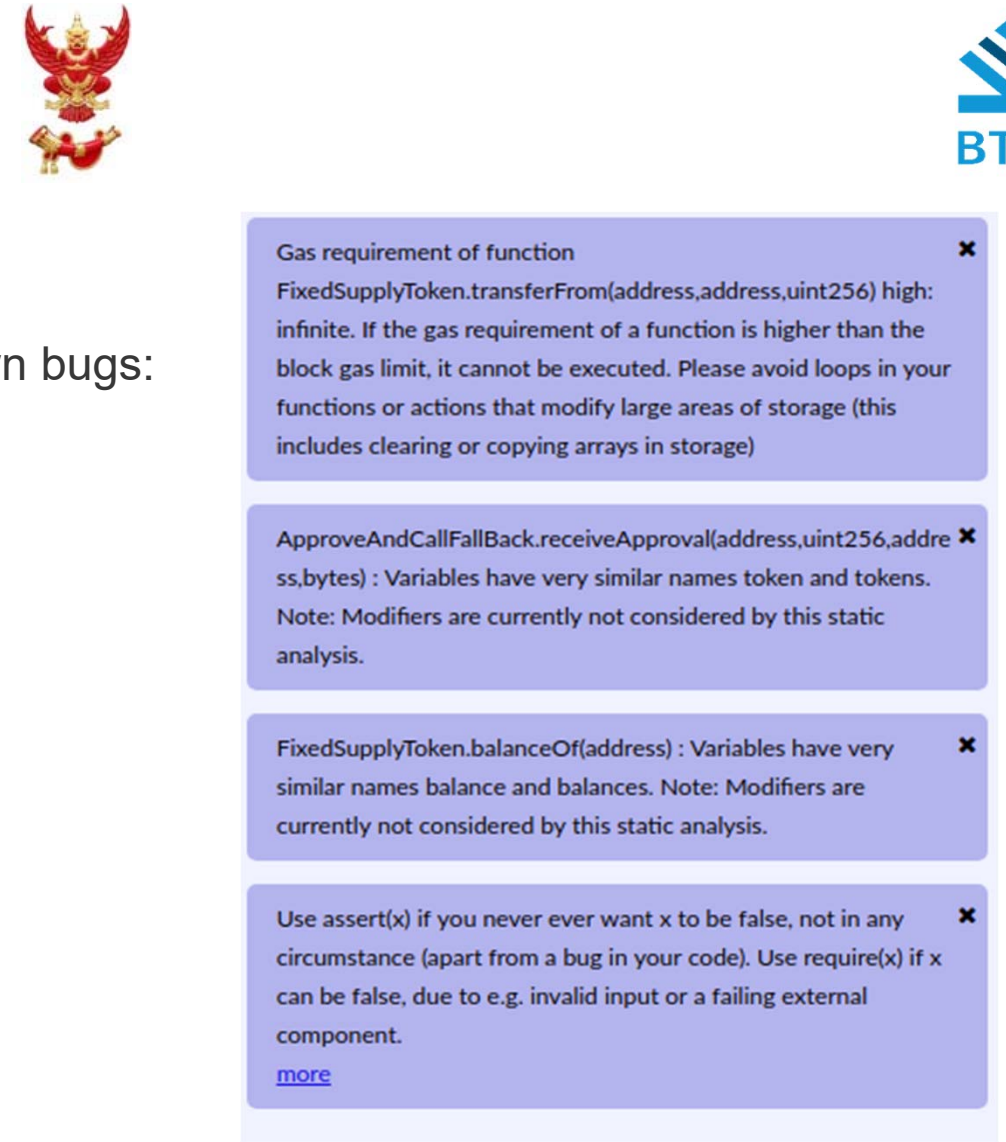
- Remix (built-in)
- Slither
- Oyente



Security tools

Static analysis tools check code for known bugs:

- Remix (built-in)
- Slither
- Oyente



Thank You