Ethereum:
A Next-Generation Smart Contract and Decentralized Application Platform

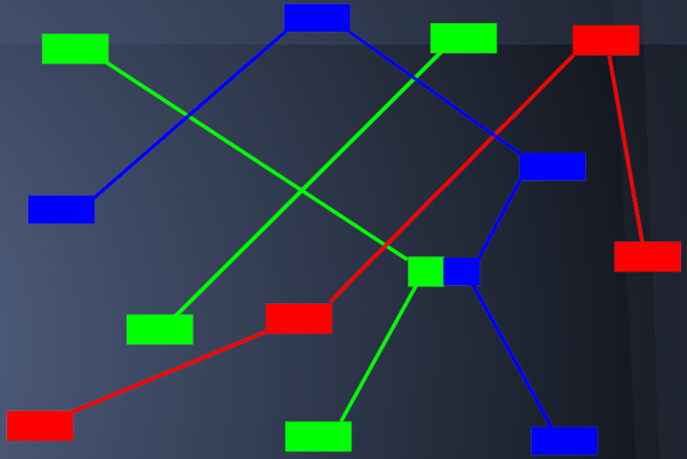
by Vitalik Buterin

A little bit of history...

- Namecoin (2010)
- Escrow transactions (2011)
- Colored coins (2012)
- Decentralized exchange (2013)
- Smart property
- Smart contracts
- Decentralized autonomous organizations

Case study: Colored Coins

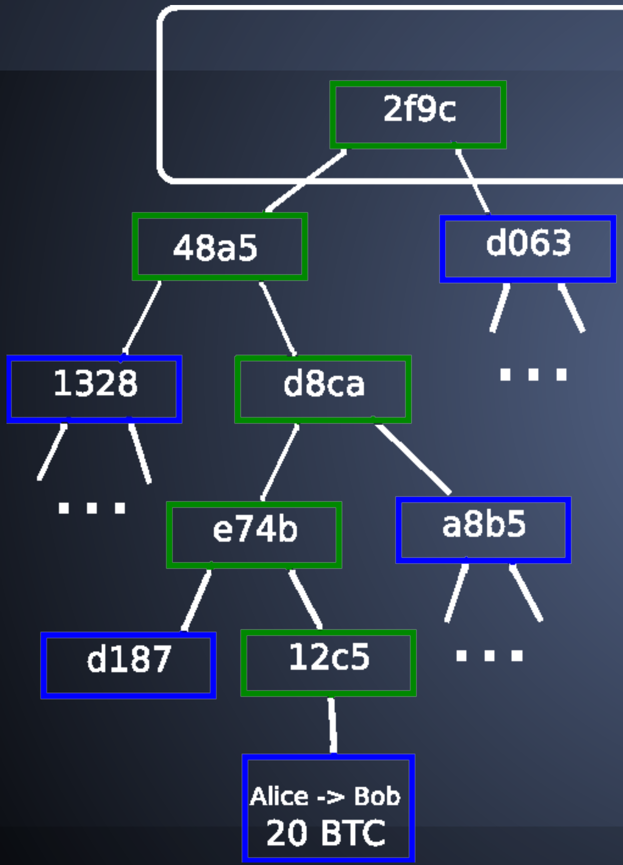
1. Issuer declares that some TX output has color X
2. TX inputs have color X -> TX outputs have color X
3. TX inputs mixed -> special rules apply
4. Trace transactions back through the blockchain to determine their color

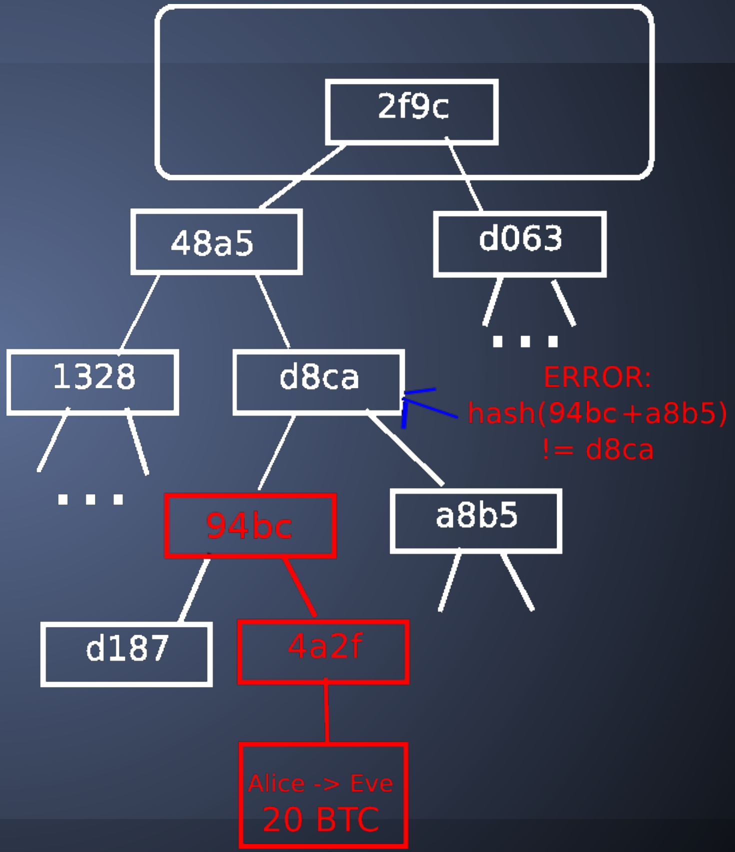


Case study: Metacoins

1. Encode metacoin data in Bitcoin TX outputs
2. Metacoin clients scan all Bitcoin TXs, filter and process metacoin TXs, and calculate the present balance sheet

Simplified Payment Verification





Scalability

- Bitcoin: Easy (use SPV)
- Colored coins: Hard (backtrace and SPV hundreds of transactions)
- BTC metacoins: Hardest (all nodes must process everything)

So What is Ethereum?

- Next-generation foundational layer
- On its own blockchain
- State tree and TX list separation
- Built-in Turing-complete programming language

What is a contract?

- Automated agent
- Lives inside the blockchain
- Has an Ethereum address and balance
- “Activated” when it receives a transaction

Ethereum Script

- Stack-based language with 60 opcodes
- Non-persistent stack
- 2^{256} memory entries (also non-persistent)
- 2^{256} storage entries, which constitute the contract’s permanent state

Users will not need to code in this language

☺

Compilers for other languages will be available

A few examples...

Namecoin

```
if tx.value < block.basefee * 200:
  stop
if contract.storage[tx.data[0]] or tx.data[0] < 100:
  stop
contract.storage[tx.data[0]] = tx.data[1]
```

Sub-currencies

```
if tx.value < 100 * block.basefee:
  stop
elif contract.storage[1000]:
  from = tx.sender
  to = tx.data[0]
  value = tx.data[1]
  if to <= 1000:
    stop
  if contract.storage[from] < value:
    stop
  contract.storage[from] = contract.storage[from] - value
  contract.storage[to] = contract.storage[to] + value
else:
  contract.storage[mycreator] = 10^18
  contract.storage[1000] = 1
```

Hedging Contracts

```
if tx.value < 200 * block.basefee:
  stop
state = contract.storage[1000]
if state == 0:
  if tx.value < 1000 * 10^18:
    stop
  contract.storage[1001] = 998 * block.contract_storage(D)[I]
  contract.storage[1002] = block.timestamp + 30 * 86400
  contract.storage[1003] = tx.sender
else:
  ethervalue = contract.storage[1000] / block.contract_storage(D)[I]
  if ethervalue >= 5000 * 10^18:
    mktx(contract.storage[1003],5000 * 10^18,0,0)
  else if block.timestamp > contract.storage[1002]:
    mktx(contract.storage[1003],ethervalue,0,0)
  mktx(A,5000 - ethervalue,0,0)
```

Decentralized Autonomous Orgs

```
if tx.value < tx.basefee * 200:
  stop
if contract.storage[tx.sender] == 0:
  stop
k = sha3(32,tx.data[1])
if tx.data[0] == 0:
  if contract.storage[k + tx.sender] == 0:
     contract.storage[k + tx.sender] = 1
     contract.storage[k] += 1
else if tx.data[0] == 1:
   if tx.value <= tx.datan * block.basefee * 200:
     stop
   if contract.storage[k]:
     stop
   i = 2
   while i < tx.datan:
     contract.storage[k + i] = tx.data[i]
     i = i + 1
    contract.storage[k] = 1
    contract.storage[k+1] = tx.datan
else if tx.data[0] == 2:

    a = contract.storage[2 ^ 255] * 2 / 3

    if contract.storage[k] >= a:
      if tx.value <= tx.datan * block.basefee * 200:
        stop
      i = 3
      L = contract.storage[k+1]
      loc = contract.storage[k+2]
      while i < L:
        contract.storage[loc+i-3] = tx.data[i]
        i = i + 1
if contract.storage[2 ^ 255 + 1] == 0:
  contract.storage[2 ^ 255 + 1] = 1
  contract.storage[C] = 1
```

...And More

- Savings Wallets
- Crop Insurance
- Financial contracts
- P2P Gambling
- Decentralized Exchange
- Data storage
- Mesh networking
- Reputation systems
- Decentralized social networks
- Skynet?

Mining

- Dagger (memory-hard PoW)
- Slasher (next-gen PoS)
- ASIC-resistant PoW contest
- PoB, PoA, PoMP, PoE?

Issuance model

- Initial fundraiser: 1000-2000 ether/BTC
- 0.225X to fiduciary members / early contractors
- 0.05X allocated to expenses/rewards fund
- 0.225X in long-term reserve pool
- 0.4X mine per year forever, after that point

Development process

- Now: C++ client 90% done, Go/Python 70%
- Feb-Apr: Agile Development
  - Use real world testing data to identify inefficiencies in software and ES language design
  - Make modifications accordingly
  - Repeat
- Release public testnet
- Hoping Q2 release of mainnet

Thanks!

http://ethereum.org

vitalik@ethereum.org

blog.ethereum.org - forum.ethereum.org
ethereum.org/ethereum.html - reddit.com/r/ethereum