The promise and peril of smart contracts

Joseph Bonneau

Department of Computer Science
New York University
Understanding smart contracts
The Bitcoin ledger track (mostly) payments

Create 25 coins and credit to Alice

Transfer 17 coins from Alice to Bob

Transfer 8 coins from Bob to Carol

Transfer 5 coins from Carol to Alice

Transfer 10 coins from Alice to David

Alice: 3
Bob: 9
Carol: 3
David: 10
Hardcode app-specific ledgers (e.g. Namecoin)

NAME_NEW y

NAME_FIRST_UPDATE jbonneau,r; 68...

NAME_UPDATE jbonneau, 2001:...

google → 172.217.18.110 [owner: K_g]
reddit → 151.101.65.140 [owner: K_r]
jbonneau → 2001:... [owner: K_j]
Universal ledgers allow uploading, running code

Create 25 coins and credit to Alice

Transfer 15 coins from Alice to Bob

Create a contract with code C, balance 5

Call C.foo("bar") with value 3

Call C.foo("baz") with value 4

Transfer 4 coins from Alice to Carol

Create a contract with code D, balance 2

Call D.fum("snork") with value 3

<table>
<thead>
<tr>
<th>Account</th>
<th>Code</th>
<th>Balance</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>$K_{Alice}$</td>
<td>10</td>
<td>∅</td>
</tr>
<tr>
<td>Bob</td>
<td>$K_{Bob}$</td>
<td>5</td>
<td>∅</td>
</tr>
<tr>
<td>Carol</td>
<td>$K_{Carol}$</td>
<td>1</td>
<td>∅</td>
</tr>
<tr>
<td>C</td>
<td>int main() { string...}</td>
<td>8</td>
<td>{“a”: 5, ...}</td>
</tr>
<tr>
<td>D</td>
<td>int main() { int i...}</td>
<td>1</td>
<td>{“b”: 42 ...}</td>
</tr>
</tbody>
</table>
A simple example:
Rock-paper-scissors
Rock Paper Scissors API

1. function add_player() payable;
   Takes player’s deposit of 1 ETH.

2. function input(uint choice);
   Records player’s choice (0 or 1 or 2)

3. function check_winner();
   Decides who wins, pays the winner
struct Player {
    uint choice;
    uint addr;
}

function add_player() payable {
    assert(num_players < 2);
    assert(msg.value >= 2000 szabo);
    reward += msg.value;
    player[num_players].addr = msg.sender;
    num_players++;
}

function input(uint choice, uint idx) {
    assert(msg.sender == player[idx].addr);
    player[idx].choice = choice;
}

uint num_players = 0;
uint reward = 0;
mapping (uint => Player) player;

function check_winner() returns(int) {
    var p0_choice = player[0].choice;
    var p1_choice = player[1].choice;
    if (p0_choice - p1_choice % 3 == 1) {
        // Player 0 wins
        player[0].addr.send(reward);
    } else if (p0_choice - p1_choice % 3 == 2) {
        // Player 1 wins
        player[1].addr.send(reward);
    } else {
        player[0].addr.send(reward/2);
        player[1].addr.send(reward/2);
    }
}
Namecoin in Ethereum
contract Namespace {

    struct NameEntry {
        address owner;
        bytes32 value;
    }

    uint32 constant REGISTRATION_COST = 100;
    uint32 constant UPDATE_COST = 10;
    mapping(bytes32 => NameEntry) data;

    function nameNew(bytes32 hash) {
        if (msg.value >= REGISTRATION_COST) {
            data[hash].owner = msg.sender;
        }
    }

    function nameUpdate(bytes32 name, bytes32 newValue, address newOwner) {
        bytes32 hash = sha3(name);
        if (data[hash].owner == msg.sender && msg.value >= UPDATE_COST) {
            data[hash].value = newValue;
            if (newOwner != 0) {
                data[hash].owner = newOwner;
            }
        }
    }

    function nameLookup(bytes32 name) {
        return data[sha3(name)];
    }
}
Ethereum project
Ethereum code written in Solidity, compiled to EVM

- **Viper**: Untyped, looks like python
- **Solidity**: Typed, looks like JS
- **Bamboo**: Typed, looks like Go
- **Future?**

**Lower-Level Language**

**Ethereum VM Bytecode Stack Language**

Looks like Forth. Defined in Yellowpaper
Solidity, EVM are not limited to Ethereum

- Enterprise Ethereum Alliance, still in infancy (Announced Feb 28)
- Goal: support EVM, Solidity and tools for private blockchains
  - maintain compatibility with Ethereum network
- Many law firms members: Cooley, Debevoise & Plimpton, Goodwin, Hogan Lovells, Holland & Knight, Jones Day, Latham & Watkins, Morrison & Foerster, Perkins Coie and Shearman & Sterling
Ethereum is “run” by the Ethereum Foundation

Compatible “alt-clients” exist (e.g. Parity, Consensys)
Explore the blockchain: https://etherscan.io
Explore Dapps: [https://dapps.ethercasts.com/](https://dapps.ethercasts.com/)

<table>
<thead>
<tr>
<th>Dapp Name</th>
<th>Creator(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClimateCoin</td>
<td>Dennis Peterson</td>
<td>Coins for those who offset carbon</td>
</tr>
<tr>
<td>Token Creation Ser.</td>
<td>Minereum Team</td>
<td>Create your own Ethereum Token with just a Minereum Transaction</td>
</tr>
<tr>
<td>HitFin</td>
<td>Patrick Salami</td>
<td>CTC Derivatives Settlement</td>
</tr>
<tr>
<td>Seglos</td>
<td>Maytham Naei</td>
<td>Spend your Ethereum without losing out on the future grow</td>
</tr>
<tr>
<td>Ether.Camp</td>
<td>Roman Mandelei</td>
<td>Blockchain explorer</td>
</tr>
<tr>
<td>TimeBank</td>
<td>Isaac Ibiapina</td>
<td>Store Ether enforceably with a time lock</td>
</tr>
<tr>
<td>Vevue</td>
<td>Thomas Olson</td>
<td>Bringing Google Street View to life</td>
</tr>
<tr>
<td>PowerBall</td>
<td>Peter Borah</td>
<td>&quot;PowerBall&quot;-style lottery</td>
</tr>
<tr>
<td>cyber•Fund</td>
<td>Dima Starodubcev</td>
<td>Make digital investments comprehensible, accessible, easy and safe</td>
</tr>
<tr>
<td>WeiLend</td>
<td>Massi Terzi</td>
<td>P2P Lending</td>
</tr>
<tr>
<td>Truffle</td>
<td>Tim Coulter</td>
<td>Development framework for Ethereum</td>
</tr>
<tr>
<td>TrustlessPrivacy</td>
<td><a href="mailto:sam@trustlessprivacy.com">sam@trustlessprivacy.com</a></td>
<td>Interoperable electronic health records</td>
</tr>
</tbody>
</table>
Ethereum may soon overtake Bitcoin in market cap

Source: Investopedia
Price remains volatile

https://ethereumprice.org/

http://eth.rollercoaster.one/
The promise of smart contracts
“Smart contracts” conceptualized by Szabo in 1994

A smart contract is a computerized transaction protocol that executes the terms of a contract. The general objectives are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs.

-Nick Szabo “The Idea of Smart Contracts”
A “dumb contract” example: pay for a hash pre-image

Alice will reveal to Bob a value $x$ such that $\text{SHA-256}(x) = 0x2a...$

In exchange, Bob will pay US$10.

If Alice does not reveal by July 1, 2017, then she will pay a penalty of US$1 per day that she is late, up to US$100.

Signed: Alice

Bob
## Traditional contracts vs. smart contracts

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Smart</th>
</tr>
</thead>
<tbody>
<tr>
<td>specification</td>
<td>Natural language + “legalese”</td>
<td>Code</td>
</tr>
<tr>
<td>assent</td>
<td>Signatures</td>
<td>Digital signatures</td>
</tr>
<tr>
<td>dispute resolution</td>
<td>Judges, arbitrators</td>
<td>Decentralized platform</td>
</tr>
<tr>
<td>nullification</td>
<td>By judges</td>
<td>?????</td>
</tr>
<tr>
<td>payment</td>
<td>As specified</td>
<td>built-in</td>
</tr>
<tr>
<td>escrow</td>
<td>Trusted third party</td>
<td>built-in</td>
</tr>
</tbody>
</table>
(technical)
Limits to smart contracts
Smart contracts only know what you tell them

**Easy application:** Alice and Bob play chess
- Rules, moves explicitly told to contract

**Harder application:** Alice and Bob bet on the Super Bowl
- Contract doesn’t know who won
- Alice and Bob may disagree!
Smart contracts can only send money

**Easy application:** Alice pays Bob $10 if she loses in chess
- Sending money is built-in & atomic

**Harder application:** Alice sells Bob her car for $10
- Can’t atomically send a car
“Smart property” looks to the ledger for instructions

Create a contract with code C, balance 0

Call C.offer() with value 1000

Call C.transfer(“Alice”) with value 0

Call C.accept(“Bob”) with value 0

Bob is my owner now!

Alice is my owner now!
Legal questions
- Smart contract vs. Smart legal contract
- Does contacting a smart contract indicate “acceptance”
- Jurisdiction and enforcement
- Legal status of smart-contract issued “tokens”
How it can all go wrong: Uncaught errors
Forcing an error via callstack exhaustion

**Attack Contract:**
```solidity
function recurse(int i) {
    if (i == 1022)
        Etherpot.cash(r, idx)
    else recurse(i+1);
    return OK;
}
```

```solidity
function cash(uint roundIndex, uint subpotIndex){
    var winner = calculateWinner(roundIndex, subpotIndex);
    var subpot = getSubpot(roundIndex);

    winner.send(subpot);

    rounds[roundIndex].isCashed[subpotIndex] = true;
    //Mark the round as cashed
}
```
How it can all go wrong:
Re-entrancy bugs
Re-entrancy hazards in Ethereum

**Contract A:**

```solidity
mapping (address => int64) balances;

function withdraw(uint x) {
    if (balances[msg.sender] >= x)
        callee.recv.value(balance)();
    balance -= x;
}
```

**Attacker Contract**

```solidity
function startAttack() {
    A.withdraw(100);
}

function recv() {
    if (counter == 2) return;
    Counter += 1;
    A.withdraw(100);
}
```
The DAO
slock.it  a Blockchain + IoT company

Example use case:

1. AirBnB user submits payment to the Ethereum blockchain

2. Slock Home Server (Ethereum client) receives the transaction

3. Power switch connected to Home Server receives “unlock” command, unlocks the door
DAO

Service provider

**Tasks:**
- Fund the development
- Vote on major decisions
- Control the funds
- Profitable

**Reward**
- 1% fee of every transaction
- One-time deployment fee

**Tasks:**
- Produce Slocks
- Marketing
- Partnerships

---

**Slock Home Server**
- Supports:
  - Z-Wave
  - Zigbee
  - Bluetooth LE

**Slock Power Switch**

**In Progress (with partners):**
- Slock Door Lock
- Slock Bike Lock
- Slock Pad Lock
- Slock Car Lock
THE DAO IS AUTONOMOUS.

1071.36 M
DAO TOKENS CREATED

10.73 M
TOTAL ETH

116.81 M
USD EQUIVALENT

1.10
CURRENT RATE
ETH / 100 DAO TOKENS

15 hours
NEXT PRICE PHASE

11 days
LEFT
ENDS 28 MAY 09:00 GMT

Raised ~150 million dollars in ~ 1 month
An attacker used re-entrancy to drain the DAO

```javascript
function () {
  // To be called by a vulnerable contract with a withdraw function.
  // This will double withdraw.

  vulnerableContract v;
  uint times;
  if (times == 0 && attackModeIsOn) {
    times = 1;
    v.withdraw();
  } else { times = 0; }
}
```

Attacker contract calls “withdraw” again before returning
Timeline and Aftermath of The DAO

- June 12: slock.it developers announce that the bug is found, but no funds at risk
- June 17 (Morning): attacker drains ⅓ of the DAO’s Ether ($50M) over 24 hrs
  Attacker’s funds were trapped in a subcontract for 40 days (July 27)
- June 17 (Evening): Eth Foundation proposes a “Soft Fork” to freeze the funds
- June 28: Cornell freshmen identify a flaw in the Soft Fork Proposal
- July 15 (Morning): Eth Foundation proposes a “Hard Fork” to recover funds
- July 15 (Evening): “Ethereum Classic” manifesto published on github
- July 19: “Hard Fork” moves funds from attacker’s contract to recovery contract
  Ethereum Classic blockchain survives and is traded on exchanges
Both Ethereum and Ethereum Classic are both around, reached new peaks
Reentrancy was known before the DAO

2014: Forum post on re-entrancy hazards
  - Suggested mitigations at the language level

2015: ETH-commissioned report on EVM security
  - Official ETH examples (crowdfund.se) also exhibit this flaw
    (they happen not to be exploitable, but without showing why)

  “the refund callback could make a new donation, triggering another refund cycle, potentially
double-refunding the earlier contributions, or failing to refund later ones”

2016: The DAO happens anyway
The anti hard-fork group has the following arguments:

- Code is law - the original statement of The DAO terms and conditions should stand under any circumstances
- Things that happen on the blockchain are immutable and they should never change regardless of what the outcome is
- There is a slippery slope and once you modify / censor for one course/reason there is not a lot to keep you from doing it for other contracts
- The decision to return the money is short sighted and you might reduce the value of ETH down the line based on your decision to act now
- This is a bailout

Users that supported the hard fork argued that:

- Code is law is too drastic of a statement at the current time and humans should have the final say through social consensus
- The Hacker could not be allowed to profit from the exploit as it is ethically wrong and the community should intervene
- The slippery slope argument is not valid as the community is not beholden to past decisions, people can act rationally and fairly in each situation
- It would be problematic to leave such a big piece of the Ether supply in the hands of a malicious actor and it might harm the value of Ether down the line
- This is not a bailout as you are not taking money from the community, it is just a return of funds to the original investors
- It would stop an ongoing war between the white-hat hackers and the hacker that would demoralize the community and possible continue for many years
- The exploit was big enough to take action and reverse it
- If the community acts now it will make people that are unethical think twice before using Ethereum as their platform of choice
- A hard-fork to return the funds would keep regulators and the legal system out of the debate: our mess, we fixed it.
Research challenges
Verifying consistency of Ethereum implementations

● at least 7 EVM implementations
  ○ C++, Go, Haskell, Java, Python, Ruby, Rust

● Inconsistency can be exploited to cause a hard fork!

Summary: Erroneous implementation of BLOCKHASH can trigger a chain reorganisation leading to consensus problems

Affected configurations: All geth versions up to 1.1.3 and 1.2.2. All eth versions prior to 1.0.0.

Likelihood: Low

Severity: Medium

Impact: Medium

Details: Both C++ (eth) and Go (geth) clients have an erroneous implementation of an edge case in the Ethereum virtual machine, specifically which chain the BLOCKHASH instruction uses for retrieving a block hash. This edge case is very unlikely to happen on a live network as it would only be triggered in certain types of chain reorganisations (a contract executing BLOCKHASH(N – 1) where N is the head of a non-canonical subchain that is not-yet reorganised to become the canonical (best/longest) chain but will be after the block is processed).
function splitDAO(
    uint _proposalID,
    address _newCurator
) noEther onlyTokenholders returns (bool _success) {

    ...

    uint fundsToBeMoved =
        (balances[msg.sender] * p.splitData[0].splitBalance) /
        p.splitData[0].totalSupply;

    if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved) (msg.sender) == false)
        throw;

    ...

    // Burn DAO Tokens

Can you spot the bug?
Ethereum makes all data public

- Proposals:
  - Project Alchemy-exchange Eth for Zcash
  - SNARKs for token-issuing contracts
    - Acceleration within EVM?
  - Hawk: The blockchain model of cryptography and privacy-preserving smart contracts [Khosba et al. 2016]
Ethereum scaling limited as nodes verify all contracts

- Can’t always determine which state a tx will change

- Goal is to support sharding
  - Most nodes track only a random subset of contracts
  - Super nodes process cross-shard communication
  - Details get complicated... great research topic!

Ethereum has long held plans to adopt proof-of-stake

Vote on neither
EV = 0

Vote on A
EV = 0.9

Vote on B
EV = 0.1 - 0.9 * 5 = -4.4

Vote on both
EV = 0.1 + 0.9 - 5 = -4

https://medium.com/@VitalikButerin/safety-under-dynamic-validator-sets-ef0c3bbdf9f6
Explore more!
# Hard Forks are planned in Ethereum

<table>
<thead>
<tr>
<th>release</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontier</td>
<td>July 2015</td>
</tr>
<tr>
<td>Homestead</td>
<td>March 2016</td>
</tr>
<tr>
<td>DAO hard fork</td>
<td>July 2016</td>
</tr>
<tr>
<td>Metropolis</td>
<td>2017?</td>
</tr>
<tr>
<td>Serenity</td>
<td>??</td>
</tr>
</tbody>
</table>
## Ethereum blockchain is different than Bitcoins

<table>
<thead>
<tr>
<th></th>
<th>Ethereum</th>
<th>Bitcoin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target time between blocks</td>
<td>14.5 seconds</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Proof of work</td>
<td>Equihash</td>
<td>SHA-256²</td>
</tr>
<tr>
<td>Stale block rewards</td>
<td>Uncle rewards</td>
<td>none</td>
</tr>
</tbody>
</table>
A decentralized app: NameCoin in Ethereum
slock.it built The DAO as a custom fundraising tool

“DAO”: Decentralized Autonomous Organization (coined by Vitalik in 2013)

Built by slock.it to raise funds for their company

Main idea: A decentralized hedge fund

Investors contribute funds, receive ownership “tokens”

Investors jointly decide how to spend funds, by voting in proportion to tokens

Many additional mechanisms:

“Splitting” to prevent hostile takeover

Reward disbursing