Ethereum and smart contracts

Arvind Narayanan
Goals

• Understand smart contract platforms (without getting bogged down in Ethereum details)

• Appreciate why smart contracts are powerful

• Learn their current limitations and open problems
Pay value \(<v>\) from addr \(<A>\) to addr \(<B>\)

signed by Alice

Users broadcast transactions to the network

Miners assemble them into blocks, achieve consensus
Users broadcast arbitrary messages to the network.

Miners assemble them into blocks, achieve consensus.
Users broadcast **programs** to the network.

Miners: consensus + program execution

Global state
Extension 3: smart contract platform

Access control for writes to global state
  – Not for reads: everything is public

Designate some variables as tokens/money
  – Native instructions to send/receive money
  – Virtual machine enforces usual rules of money

Programs are long-lived, pass messages to other programs

Consequence: programs are agents!
A smart contract is an algorithmic agent

Agent’s actions algorithmically specified, fixed

Decentralized — no one controls it

No private memory, communication channels

Register alice.bit
Pay 10 tokens
OK
contract NameRegistry {
    mapping(bytes32 => address) public registryTable;
    function claimName(bytes32 name) {
        if (msg.value < 10) {
            throw;
        }
        if (registryTable[name] == 0) {
            registryTable[name] = msg.sender;
        }
    }
}
Puzzle: how to look up a domain name?

Can’t download entire blockchain – too inefficient

Can’t simply ask a miner – no one is trusted

Hint: in Bitcoin, how to confirm that you’ve received a payment w/o downloading blockchain?

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>alice.bit</td>
<td>xxx</td>
</tr>
</tbody>
</table>

Register alice.bit
Pay 10 tokens
OK
Solution: store a succinct snapshot of smart contract execution in the blockchain

Blockchain contains hash tree of all current key-value pairs in the system

User stores root hash

User queries key

Miner returns value, hash chain to root
Vision: markets/commerce without gatekeepers
Honest Ponzi scheme
Limitations of today’s smart contract platforms

1. Verifier’s dilemma (see reading)
2. Data feeds
3. Scaling & sharding (see reading)
4. Endpoint security
5. Contract security (next slide)
Ethereum: poor design choices w.r.t. security

1. No handling of race conditions
2. No random number generator
3. Poor exception handling
4. Reentrancy is unsafe

Paper: *Making smart contracts smarter*
A note of caution

Many smart contract applications are attempts to solve social problems using technology

Example:
   a land registry smart contract won’t help against a corrupt gov’t: they have all the guns and can come take your land anyway

Example:
   healthcare smart contracts “solve” the “problem” of patients not trusting their doctors (!!)