Obsidian: Safely Programming Contracts on the Blockchain

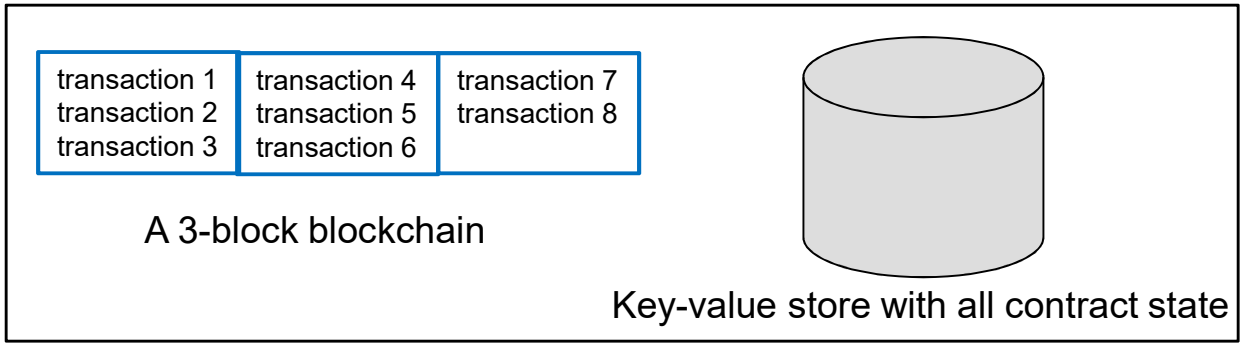
Jonathan Aldrich
aldrich@cs.cmu.edu
http://www.cs.cmu.edu/~aldrich/

CyLab Partners Conference
September 2017

contributions from:
Michael Coblenz
Celeste Barnaby
Tyler Etzel
Jenna Wise
Eliezer Kanal
Joshua Sunshine
Brad Myers

Carnegie Mellon University
School of Computer Science

What is blockchain programming?



- **Blockchains** allow mutually untrusting parties to agree on state
  - State: a growing chain of blocks
  - Parties incentivized to extend the longest chain

- **Smart contracts** *codify* and *implement* agreements between parties
  - Contracts transfer virtual currency and record transactions on the blockchain
  - Supported by Ethereum, Hyperledger blockchains

A Simple Blockchain Contract

```
contract Bond {
    Account seller, buyer;

    Bond(Account s) { seller = s; }

    transaction buy(Money m, Account b) {
        seller.pay(m)
        buyer = b;
    }

    transaction makePayment(Money m) {
        buyer.pay(m)
    }
}
```

Money is treated specially. Passing it around moves virtual currency between contracts and accounts.

Misusing the Contract

```
contract Bond {
    Bond(Account s) { … }
    transaction buy(Money m, Account b) { … }
    transaction makePayment(Money m) { … }
}

contract ErroneousClient {
    Bond b = new Bond(…)
    b.buy(…);
    b.buy(…);
}
```

This isn't sensible! Can't buy a bond that has already been purchased.

A more complex, re-entrant misuse of the DAO contract allowed attackers to steal $70 million worth of ether in June 2016

Specifying and Enforcing State

```
contract Bond {
    Account seller;
    Bond(Account s) {
        seller = s;
        this -> Offered{}; }
    state Offered {
        transaction buy(Money m, Account b) {
            seller.pay(m)
            this -> Sold{ buyer = b; }
    } }
    state Sold {
        Account buyer;
        transaction makePayment(Money m) {
            buyer.pay(m)
} } }
```

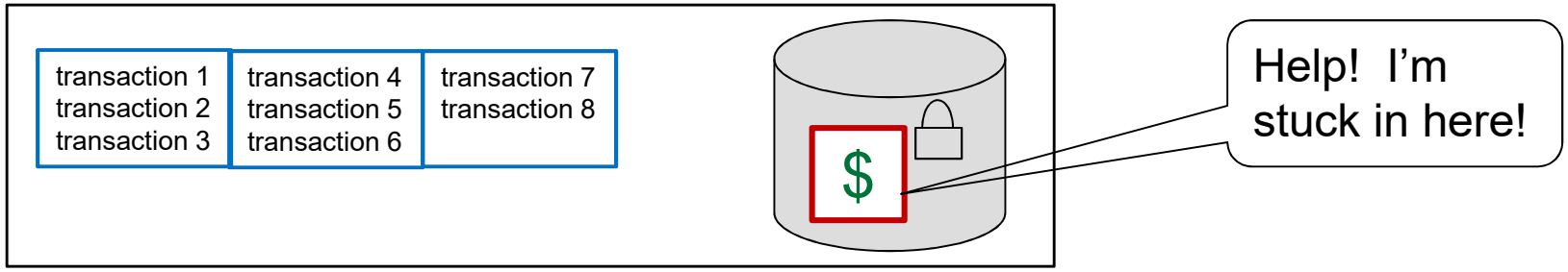
We start in the Offered state

Specify each conceptual state in the contract

Transition to Sold and specify the buyer

makePayment() is the only legal transaction in the Sold state

Tracking Money and Other Resources



- Contracts manipulate money, which can become "stuck" in a poorly-written contract.
  - Other resources may also be represented in the blockchain – e.g. representations of commodities being traded
- We are investigating **linear types**, which can help ensure that resources are not duplicated or lost
  - Linear types also help find state-related errors at compile time, before a contract is instantiated
- We plan to use **model checking** to ensure that there is always a way to get money out of a contract

Obsidian: Safer Blockchain Programming

- Obsidian is a new smart contract programming language
  - **Explicit states** help programmers avoid reentrancy bugs
  - **Linear types** and **model checking** help avoid loss of resources

- Project status
  - Initial compiler for the dynamic core of Obsidian
  - Ongoing work on linear types, model checking
  - Formative evaluation underway: Can programmers use Obsidian?
    - Later: does Obsidian help in practice?