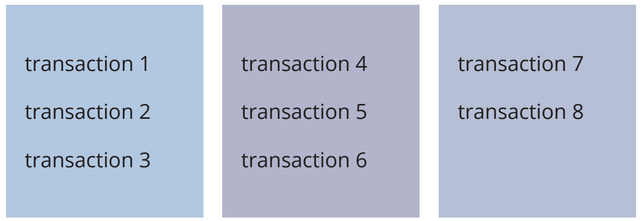
Obsidian

A Safer Blockchain Programming Language

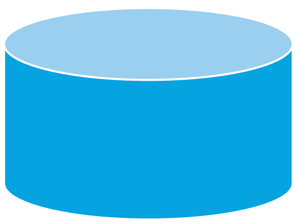
What is a blockchain?

- Blockchain programming environments provide *shared, global* state on *untrusted, distributed computing nodes.*
- Global state consists of *smart contracts,* which include both data structures and *transactions* that manipulate them.
- Transactions can *deploy* smart contracts to the blockchain (initializing their state), or *invoke* code implemented in specific deployed smart contracts.
- "Code is law": a principle that suggests that a contract's code specifies an agreement between parties. This principle implies that contracts are *immutable*; bugs in them cannot be fixed after deployment.

One blockchain network node



A 3-block blockchain



Key-value store with state of all contracts

The nodes execute a *consensus protocol* by which they agree on which transactions have been processed and in what order. After consensus is reached, all nodes agree on the transactions and their ordering.

Blockchain facts	Consequences	Design approach
Smart contracts can hold **resources**.	Bugs can transfer resources to the wrong party or lose them entirely. A bug was recently exploited to steal over US$40M from a contract.	Obsidian models resources with **linear types**, which statically restrict lifecycles, so that **resources** cannot be accidentally lost.
Modifying a contract's code could change an agreement that parties have settled on.	Smart contracts are **immutable** once deployed, but this means that bugs are not fixable.	Support developer-authored **specifications** and use **verification** tools to prove that the code satisfies the specifications. Use a strong, static type system to detect as many bugs as possible at compile time.
Proposed application domains, such as finance and medical records, cannot tolerate serious bugs.	**Correct software** is paramount.	
Many proposed applications are inherently *stateful*.	Blockchain applications commonly implement **state machines**. Behavior and available transactions depend on the current state.	Obsidian is a **typestate-oriented language**, representing *state* in *types* and statically preventing some invalid invocations. For example, a Bond that has been bought is in the Bought state and cannot be bought again.

Annotated example code

```
interface account { transaction pay (money m); }
resource contract money {…};   ← Instances of resource contracts, unlike other contracts, always have exactly one owning reference.
contract Bond {
  account seller;
  Bond(account s) {
    seller = s;
    -> Offered();   ← The constructor transitions to Offered state at the end.
  }
  state Offered {
    transaction buy(money m, account b) {
      seller.pay(m);   ← After this line, the transaction no longer owns m.
      -> Sold({buyer = b});   ← Transition to Sold state with buyer field set to b.
    }
  }
  state Sold {
    account buyer;
    transaction makePayment(money m) {
      buyer.pay(m);
    }
  }
}
contract ErroneousClient {
  transaction badTransaction() {
    Bond.Offered b = new Bond(…);
    b.buy(…);   ← Type of b after this line is Bond.Sold due to buy() invocation.
    b.buy(…);   ← Compile error: b is of type Bond.Sold, which has no buy() transaction.
  }
}
```

Status and Upcoming Work

- An initial compiler translates Obsidian code to Java (for execution on a mock blockchain) and to Dafny (for verification).
- The compiler includes a typechecker based on a draft typesystem.
- We plan to port Obsidian to Hyperledger Fabric and complete the compiler.
- Our evaluation will ask users to write programs in Obsidian and either Hyperledger Composer or Solidity (existing blockchain development tools) and compare task completion times and rates and kinds of bugs.
- We are iteratively evaluating portions of the language design in the context of Java so that we can isolate those parts of the design for study.

Carnegie Mellon University
Software Engineering Institute

Michael Coblenz mcoblenz@cs.cmu.edu
Eliezer Kanal ekanal@cert.org

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM17-0741
Obsidian: A Safer Blockchain Programming Language