WHY IS SCILLA BETTER THAN SOLIDITY?

INCIDENTS WITH SMART CONTRACTS

$60 MIL STOLEN

$300 MIL FROZEN

UNDERLYING CAUSES

COMPLEXITY

EXPECTED VS
UNEXPECTED
BEHAVIOR

NO FORMAL
VERIFICATION

SCILLA

SMART CONTRACTS ON ZILLIQA

NON-TURING COMPLETE

DECIDABLE CONTRACTS

AMENABLE TO FORMAL VERIFICATION

CLEAN SEPARATION: COMMUNICATION VS COMPUTATION

KICKSTARTER IN SCILLA

IMMUTABLE PARAMS

```
contract Crowdfunding
         (owner    : ByStr20,
          deadline : BNum,
          goal     : Uint128)
```

MUTABLE STATE

```
field backers : Map Address ⇒ Int = Emp
field success : Bool = False
```

STATE TRANSITIONS

```
transition Donate ()
```

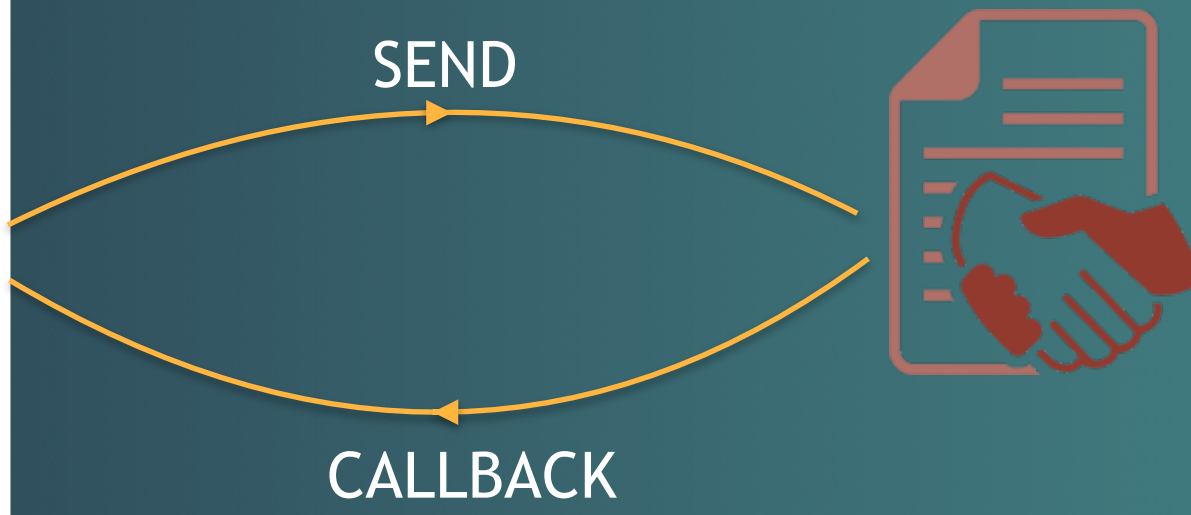
```
transition GetFunds ()
```

```
transition ClaimBack ()
```

DAO INCIDENT

SOLIDITY

```
function reclaim
{
    uint amount = contributors[msg.sender]
    if(msg.sender.call.value(amount) == false)
        throw
    // reset the amount for sender
    contributors[msg.sender] = 0;
}
```



INSTRUCTION NEVER EXECUTED

PREVENTING DAO INCIDENT

SECURITY RECOMMENDATION

```
// THIS CONTRACT HAS A BUG, DO NOT USE
function reclaim
{
    uint amount = contributors[msg.sender];
    if(msg.sender.call.value(amount) == false)
        throw
    // reset the amount for sender
    contributors[msg.sender] = 0;
}
```

```
// SAFE TO USE

function reclaim
{
    uint amount = contributors[msg.sender];
    contributors[msg.sender] = 0;
    msg.sender.transfer(amount);
}
```

CHECK-EFFECTS-INTERACTIONS

PREVENTING DAO INCIDENT

AT THE LANGUAGE LEVEL

SOLIDITY

```
// SAFE TO USE

function reclai
{
    uint amount
    contributor
    msg.sender.transfer(amount);
}
```

SCILLA

```
transition Reclaim
        // Check if the sender is eligible to
        reclaim
    if ( … )
                                        unt → 0,
                                        → "failure">)
        // remove sender from the list
        let v = get(contributors, sender) in
        contributors := remove(contributors,
        sender);
        send (<to → sender, amount → v,
                tag → “main", msg → "refunded">)
```

EXTERNAL CALLS ALWAYS HAPPEN AT THE END

REENTRANCY ATTACKS ARE INEFFECTIVE

HACKS ON PARITY MULTISIG CONTRACT

```
contract UnsafeLibraryContract2{
    address owner;

    function initowner(address _owner) { owner = _owner; }

    function transferTo(uint _amount, address _recipient) {
        if (msg.sender == owner)
            _recipient.transfer(_amount);
    }
}
```

Anyone can set `_owner`

Transfer once `_owner` is set

Solidity does not have a clean separation between libraries (a contract with only mathematical functions) and a non-library contract (which can handle states such as `_owner`)

INTEGER OVERFLOW ATTACKS

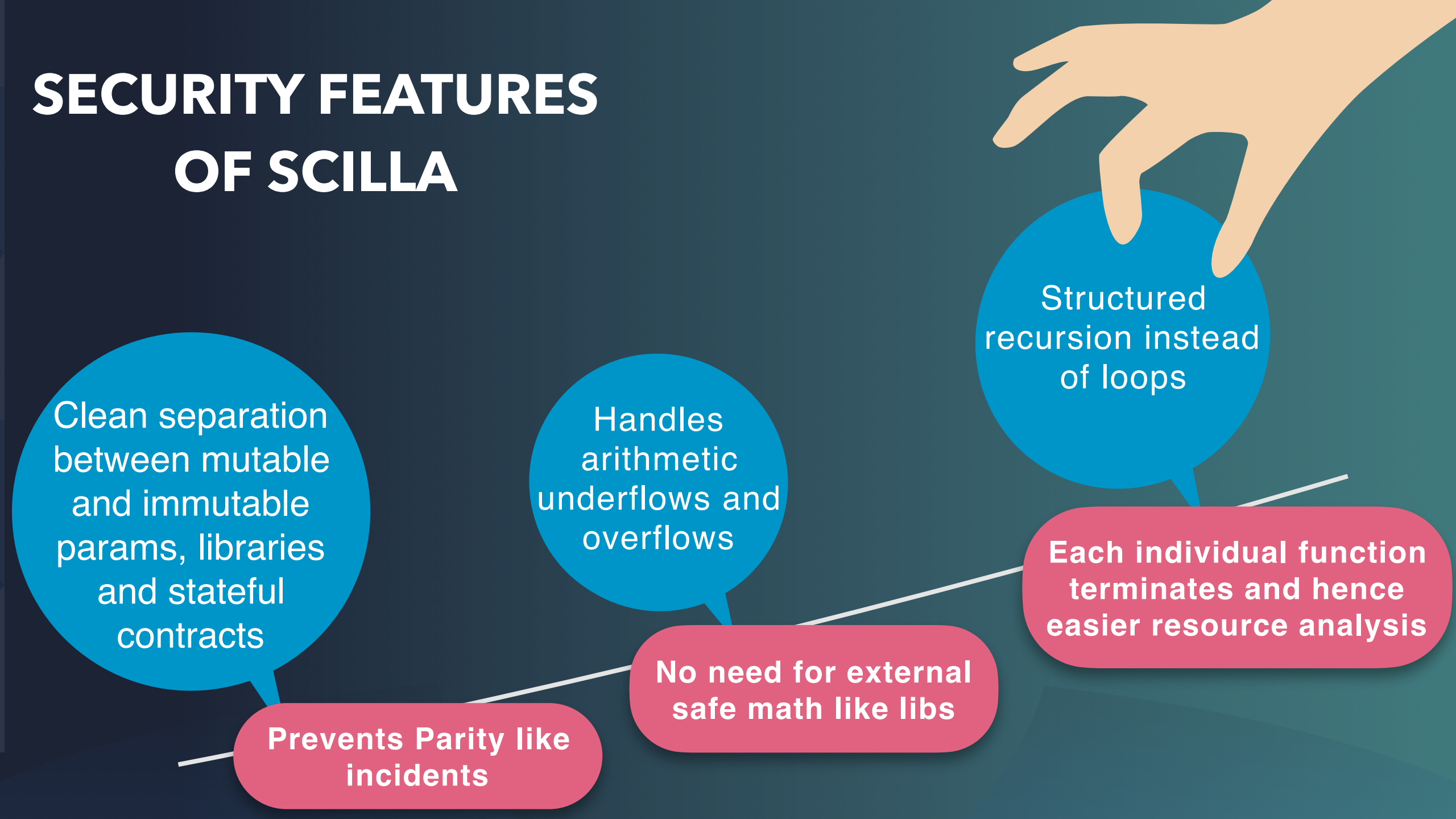
```
function batchTransfer(address[]receivers, uint256
(bool) {
  uint cnt = _receivers.length;
  uint256 amount = uint256(cnt) * _value;
  require(_value > 0 && balances[msg.sender] >= amount);

  balances[msg.sender] = balances[msg.sender].sub(amount
   for (uint i = 0; i < cnt; i++) {
        balances[_receivers[i]] = balances[_receivers[i
        Transfer(msg.sender, _receivers[i], _value);
   }
   return true;
}
```

What if:
_value = 2^255
cnt = 2

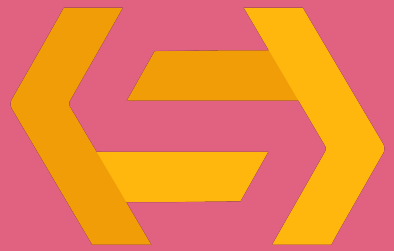
amount = 0

Becomes possible to transfer a large number of tokens

SECURITY FEATURES OF SCILLA



FORMAL VERIFICATION



+

Coq

LEMMA 1: CONTRACT WILL HAVE ENOUGH FUNDS TO REFUND.

LEMMA 2: CONTRACT WILL NOT ALTER ITS CONTRIBUTION RECORDS.

LEMMA 3: EACH CONTRIBUTOR IS REFUNDED THE RIGHT AMOUNT.

THANK YOU!

ZILLIQA

/ˈZILIKə/

ENQUIRY@ZILLIQA.COM @ZILLIQA ZILLIQA.COM