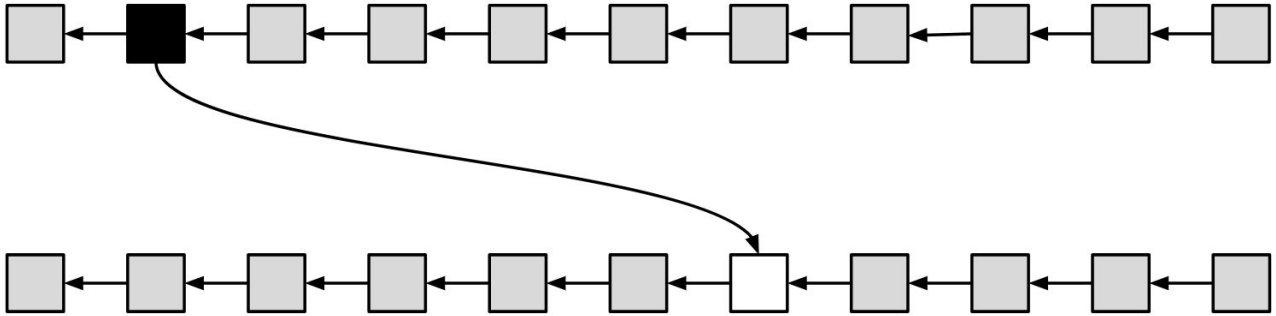
A Sidechain Protocol
for Proof-of-Work Blockchains



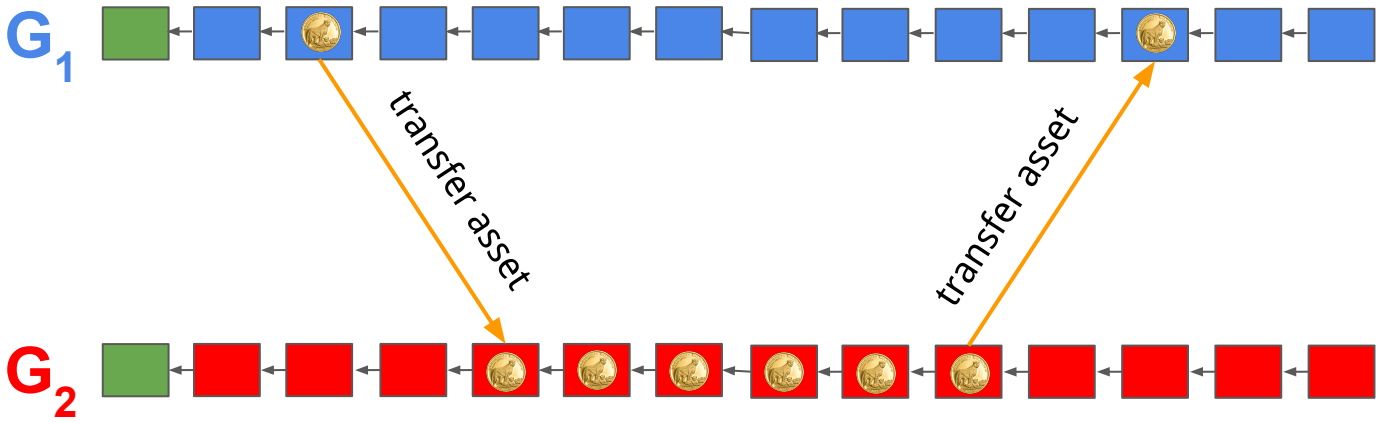
Aggelos Kiayias, Dionysis Zindros

National and Kapodistrian
UNIVERSITY OF ATHENS

INPUT | OUTPUT

Financial Crypto 2019

The two-way peg: A cross-chain asset



Two-way pegging

- Given two blockchains G_1, G_2

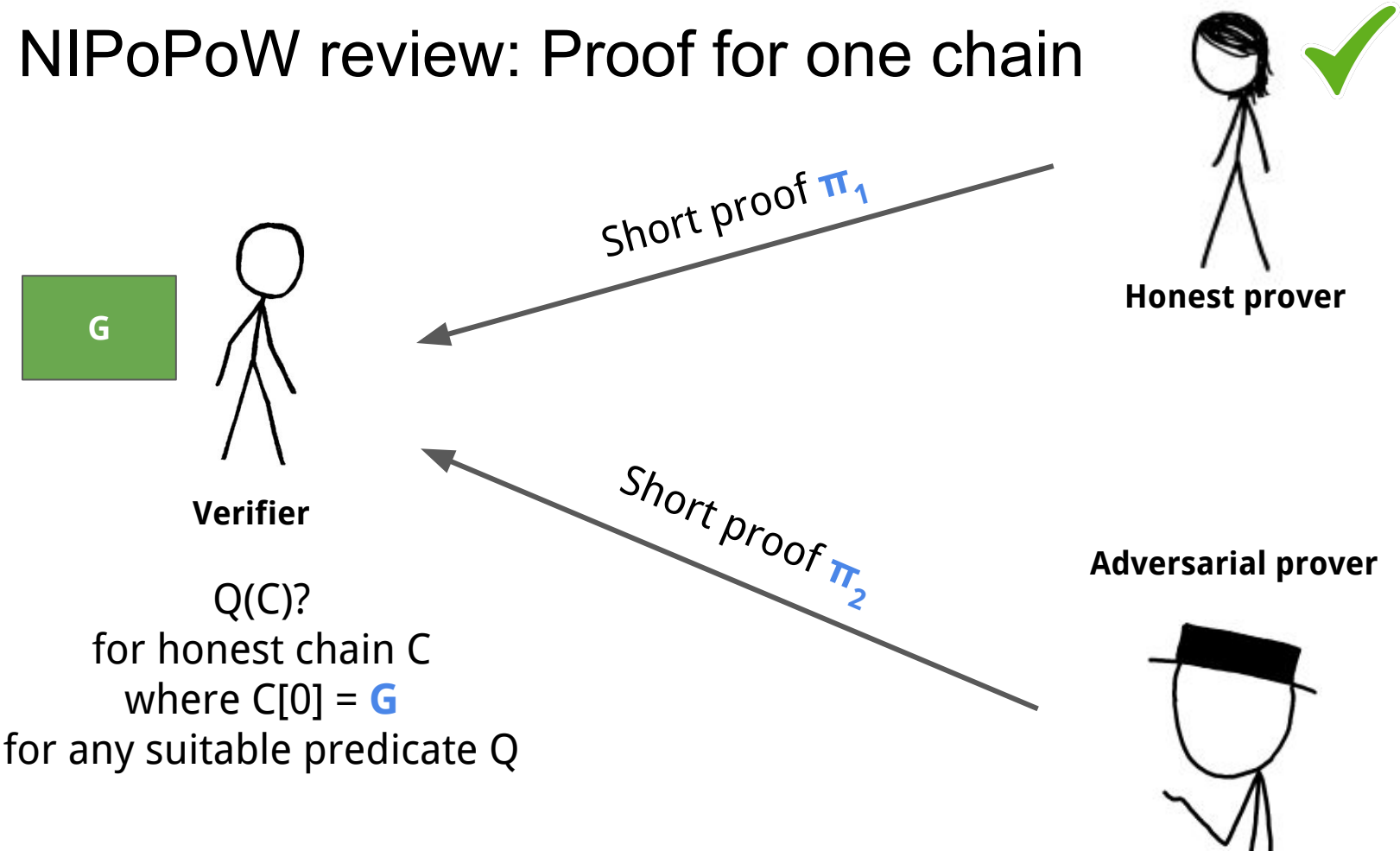
Assumption: Both chains:

- Are **Turing-complete** (Solidity)
- Have **Proof-of-Work** consensus
- Have (velvet) support for **NIPoPoWs** back to their Geneses
- Are individually secure (2 honest majority assumptions)

Construction must be trustless and miner-isolated!

We will work with the **native** asset of G_1 and mirror it as ERC-20 on G_2

NIPoPoW review: Proof for one chain



Q(C)?
for honest chain C
where C[0] = **G**
for any suitable predicate Q

Predicate for two-way peg

Predicate *Q* for G_1 (similarly for G_2)

- There was a money-locking transaction *tx*
- *tx* called a particular smart contract function
- *tx* was paid for by *author*
- *tx* has amount *amount*
- *tx* belongs to a block *B*
- *B* is *k*-confirmed in a chain *C*
- *C* is an honestly adopted chain
- *C*[0] = G_1

Predicate will be represented by **Solidity event**:
It has a **name** and **parameters**, and the solidity code can **fire** it

Smart contract strategy

- Create a **base** smart contract **crosschain**
- Make two smart contracts, one for each chain

sidechain$_1$ contract inherits from **crosschain**, runs on **G_1**, and can:

- be **paid** in **G_1** native currency
- **pay back** in **G_1** native currency

} fires **Deposited$_1$** event when paid

sidechain$_2$ contract inherits from **crosschain**, runs on **G_2**, and can:

- be **paid** in **G_2** ERC-20 currency
- **pay back** in **G_2** ERC-20 currency

} fires **Deposited$_2$** event when paid

```
contract sidechain_1 extends crosschain_{k,m,z}
    payable function deposit(target)
        ▷ Emit an event to be picked up by the remote contract
        ctr ← ctr + 1
        emit Deposited_1(target, msg.value, ctr)
    end function
end contract
contract sidechain_2 extends crosschain_{k,m,z}; ERC20
    mapping(address ⇒ int) balances
    function deposit(target, amount)
        ▷ Charge account of sender
        if balances[msg.sender] < amount then
            return ⊥
        end if
        balances[msg.sender] ← balances[msg.sender] − amount
        ▷ Emit an event to be picked up by the remote contract
        ctr ← ctr + 1
        emit Deposited_2(target, amount, ctr)
    end function
end contract
```

payment event emission

standard ERC-20 behavior

payment event emission

Initialization timeline

1. **sidechain$_1$** smart contract constructed
2. **sidechain$_1$** instantiated in G_1 address **sidechain$_1^{addr}$**
3. **sidechain$_2$** smart contract constructed
4. **sidechain$_2$** instantiated in G_2 address **sidechain$_2^{addr}$**
5. Owner calls **sidechain$_1$** initializer with parameters: H(**G_2**), **sidechain$_2^{addr}$**
6. Owner calls **sidechain$_2$** initializer with parameters: H(**G_1**), **sidechain$_1^{addr}$**
7. Both owners discarded so that the process is now trustless
8. People wishing to use smart contract verify correct initialization

```
contract crosschain_{k,m,z}
    internal function initialize(G)
        this.G ← G
    end function
    ...
end contract
contract sidechain_1 extends crosschain_{k,m,z}
    initialized ← false
    function initialize(G_2, sidechain_2)
        if ¬initialized then
            ctr ← 0
            crosschain.initialize(G_2)   ▷ Initialize with the remote chain genesis block
            initialized ← true
            this.sidechain_2 ← sidechain_2
        end if
    end function
    ...
end contract
```

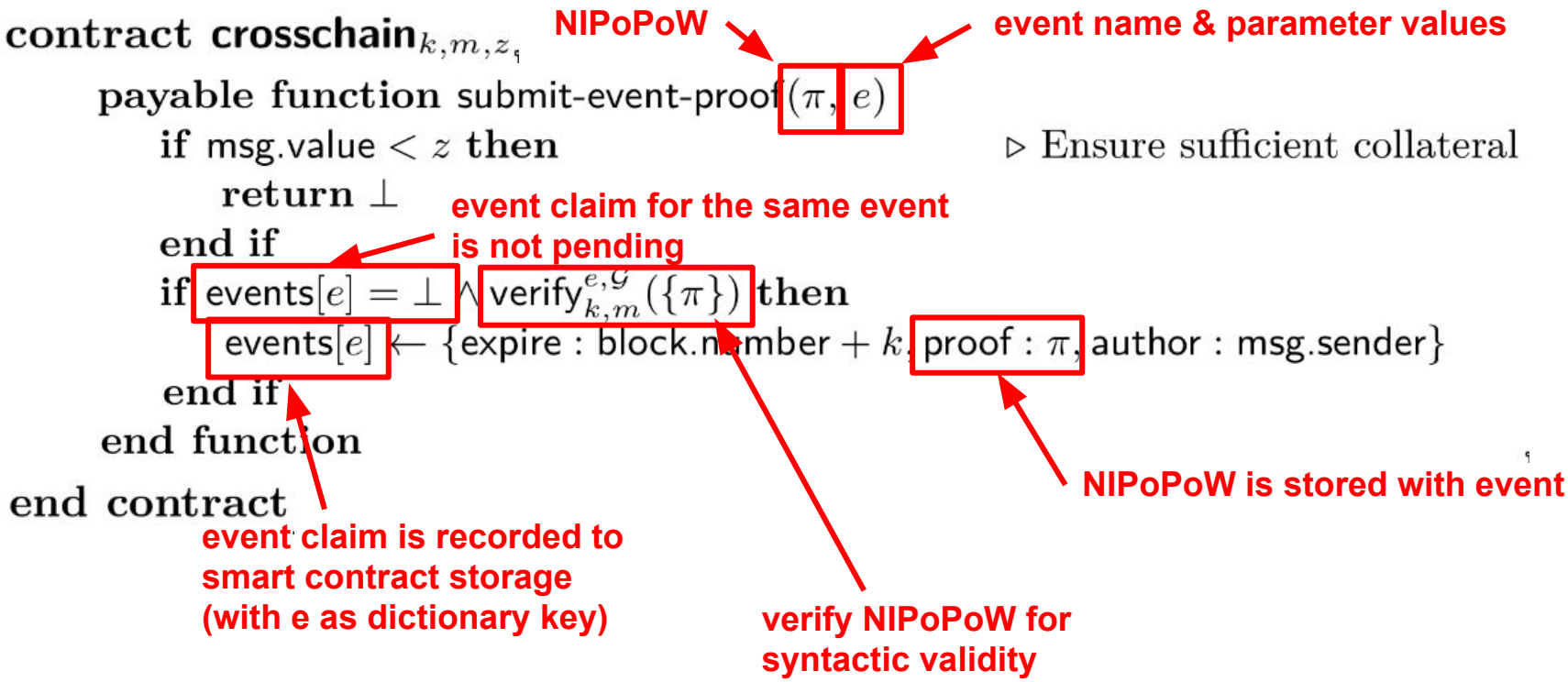
record remote genesis hash

make contract owner powerless

record remote sibling contract address

Recording cross-chain events

- **crosschain** contract allow recording remote events
- **Claim** that event took place is submitted to contract with call to **submit-event-proof** smart contract function
- Event submission takes parameters NIPoPoW **π** and event details **e**
- Event claim is recorded to **events** storage for now


contract crosschain$_{k,m,z,}$
payable function submit-event-proof(π, e)
if msg.value $< z$ then
return $\perp$
end if
if events$[e] = \perp \wedge$ verify$^{e,\mathcal{G}}_{k,m}(\{\pi\})$ then
events$[e] \leftarrow$ {expire : block.number $+ k$, proof : π, author : msg.sender}
end if
end function
end contract
▷ Ensure sufficient collateral
NIPoPoW
event name & parameter values
event claim for the same event is not pending
NIPoPoW is stored with event
event claim is recorded to smart contract storage (with e as dictionary key)
verify NIPoPoW for syntactic validity

Disproving event claims

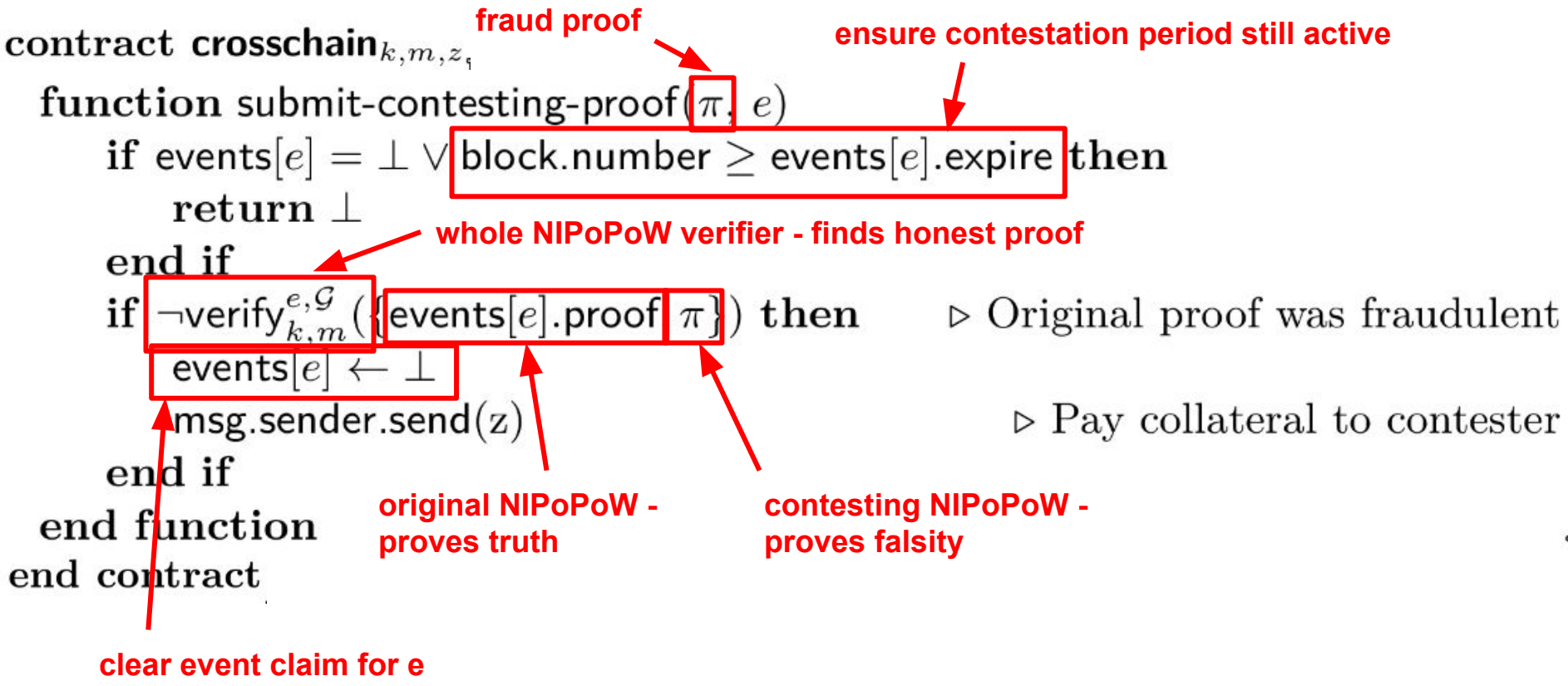
- An event claim can be **fraudulent**
- Allow an **expiration period** of **k** blocks after claim submission for fraud to be reported
- **k** here corresponds to target blockchain liveness parameter
- Party making claim must **pay collateral *z*** during claim
- If fraud is proved, collateral is **paid to fraud prover**
- If fraud is not proved, collateral is **reclaimed by claimer**
- Fraud proof can be provided by calling **submit-contesting-proof** function
- Parties are “cryptoeconomically” incentivized to submit fraud proofs
- collateral z ≥ gas of fraud proof submission + cost of chain monitoring

```
contract crosschain_{k,m,z}
    payable function submit-event-proof(π, e)
        if msg.value < z then                          ▷ Ensure sufficient collateral
            return ⊥
        end if
        if events[e] = ⊥ ∧ verify^{e,G}_{k,m}({π}) then
            events[e] ← {expire : block.number + k, proof : π, author : msg.sender}
        end if
    end function
end contract
```

collateral is hard-coded

block after which claim will be finalized and no fraud proofs will be possible

author is recorded with event claim to return collateral if honest


contract crosschain$_{k,m,z_1}$
function submit-contesting-proof(π, e)
if events[e] $= \perp \vee$ block.number $\geq$ events[e].expire then
return $\perp$
end if
if $\neg$verify$^{e,\mathcal{G}}_{k,m}$({events[e].proof, π}) then ▷ Original proof was fraudulent
events[e] $\leftarrow \perp$
msg.sender.send(z) ▷ Pay collateral to contester
end if
end function
end contract
fraud proof
ensure contestation period still active
whole NIPoPoW verifier - finds honest proof
original NIPoPoW - proves truth
contesting NIPoPoW - proves falsity
clear event claim for e

Finalizing an event

- If no fraud claim is made during contestation period, then event proof was honest
- We know event happened
- Record it in **finalized-events** permanently
- Allow child contracts to check if event has been finalized by invoking **event-exists**

contract crosschain$_{k,m,z}$

function finalize-event(e)
if events[e] $= \perp \vee$ block.number $<$ events[e].expire **then**
return $\perp$
end if
finalized-events $\leftarrow$ finalized-events $\cup \{e\}$
author $\leftarrow$ events[e].author
events[e] $\leftarrow \perp$
author.send(z) ▷ Return collateral
end function

function event-exists(e)
return $e \in$ finalized-events
end function

end contract

event no longer a claim,
but a fact

Withdrawing money

- Simply check if remote payment event took place
- If so, release funds
- Record event as “used” to avoid double spending (not shown in code)

```
contract sidechain_1 extends crosschain_{k,m,z}
    function withdraw(amount, target, ctr)
        ▷ Validate that event took place on remote chain
        if ¬event-exists((sidechain_2, Deposited_2, (amount, target, ctr))) then
            return ⊥
        end if
        msg.sender.send(amount)
    end function
end contract
contract sidechain_2 extends crosschain_{k,m,z}; ERC20
    function withdraw(amount, target, ctr)
        ▷ Validate that event took place on remote chain
        if ¬event-exists((sidechain_1, Deposited_1, (amount, target, ctr))) then
            return ⊥
        end if
        ▷ Credit target account
        balances[target] ← balances[target] + amount
    end function
end contract
```

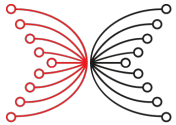
References

- Aggelos Kiayias, Nikolaos Lamprou, and Aikaterini-Panagiota Stouka
  "**Proofs of Proofs of Work with Sublinear Complexity**", FC 2016
- Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos
  "**The Bitcoin Backbone Protocol: Analysis and Applications**", EUROCRYPT 2015
- Aggelos Kiayias, Andrew Miller, Dionysis Zindros
  "**Non-interactive proofs of proof-of-work**" (ePrint 2017)
- Peter Gaži, Aggelos Kiayias, Dionysis Zindros
  "**Proof-of-Stake Sidechains**" (IEEE S&P, 2019)
- Aggelos Kiayias, Dionysis Zindros. "**Proof-of-Work Sidechains**", FC 2019
- Kiayias, Aggelos, Alexander Russell, Bernardo David, Roman Oliynykov
  "**Ouroboros: A provably secure proof-of-stake blockchain protocol**", CRYPTO 2017
- Back, Adam, et al. "**Enabling blockchain innovations with pegged sidechains**" (2014)

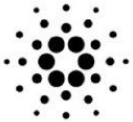
45DC 00AE FDDF 5D5C B988 EC86 2DA4 50F3 AFB0 46C7

@dionyziz
dionyziz@di.uoa.gr

Thanks! Questions?



INPUT | OUTPUT



CARDANO

National and Kapodistrian
UNIVERSITY OF ATHENS

```
contract crosschain_{k,m,z}
    internal function initialize(G)
        this.G ← G
    end function
    payable function submit-event-proof(π, e)
        if msg.value < z then                          ▷ Ensure sufficient collateral
            return ⊥
        end if
        if events[e] = ⊥ ∧ verify^{e,G}_{k,m}({π}) then
            events[e] ← {expire : block.number + k, proof : π, author : msg.sender}
        end if
    end function
    function finalize-event(e)
        if events[e] = ⊥ ∨ block.number < events[e].expire then
            return ⊥
        end if
        finalized-events ← finalized-events ∪ {e}
        author ← events[e].author
        events[e] ← ⊥
        author.send(z)                                 ▷ Return collateral
    end function
    function submit-contesting-proof(π, e)
        if events[e] = ⊥ ∨ block.number ≥ events[e].expire then
            return ⊥
        end if
        if ¬verify^{e,G}_{k,m}({events[e].proof, π}) then   ▷ Original proof was fraudulent
            events[e] ← ⊥
            msg.sender.send(z)                         ▷ Pay collateral to contester
        end if
    end function
    function event-exists(e)
        return e ∈ finalized-events
    end function
end contract
```

```
1: contract sidechain_1 extends crosschain_{k,m,z}
2:     initialized ← false
3:     function initialize(G_2, sidechain_2)
4:         if ¬initialized then
5:             ctr ← 0
6:             crosschain.initialize(G_2)   ▷ Initialize with the remote chain genesis block
7:             initialized ← true
8:             this.sidechain_2 ← sidechain_2
9:         end if
10:    end function
11:    payable function deposit(target)
12:        ▷ Emit an event to be picked up by the remote contract
13:        ctr ← ctr + 1
14:        emit Deposited_1(target, msg.value, ctr)
15:    end function
16:    function withdraw(amount, target, ctr)
17:        ▷ Validate that event took place on remote chain
18:        if ¬event-exists((sidechain_2, Deposited_2, (amount, target, ctr))) then
19:            return ⊥
20:        end if
21:        msg.sender.send(amount)
22:    end function
23: end contract
```

```
1: contract sidechain_2 extends crosschain_{k,m,z}; ERC20
2:     mapping(address ⇒ int) balances
3:     initialized ← false
4:     function initialize(G_1, sidechain_1)
5:         if ¬initialized then
6:             ctr ← 0
7:             crosschain.initialize(G_1)   ▷ Initialize with the remote chain genesis block
8:             initialized ← true
9:             this.sidechain_1 ← sidechain_1
10:        end if
11:    end function
12:    function deposit(target, amount)
13:        ▷ Charge account of sender
14:        if balances[msg.sender] < amount then
15:            return ⊥
16:        end if
17:        balances[msg.sender] ← balances[msg.sender] − amount
18:        ▷ Emit an event to be picked up by the remote contract
19:        ctr ← ctr + 1
20:        emit Deposited_2(target, amount, ctr)
21:    end function
22:    function withdraw(amount, target, ctr)
23:        ▷ Validate that event took place on remote chain
24:        if ¬event-exists((sidechain_1, Deposited_1, (amount, target, ctr))) then
25:            return ⊥
26:        end if
27:        ▷ Credit target account
28:        balances[target] ← balances[target] + amount
29:    end function
30: end contract
```