Verifiable Computation using Smart Contracts

Mahmudun Nabi
University of Calgary, Canada

June 26, 2020

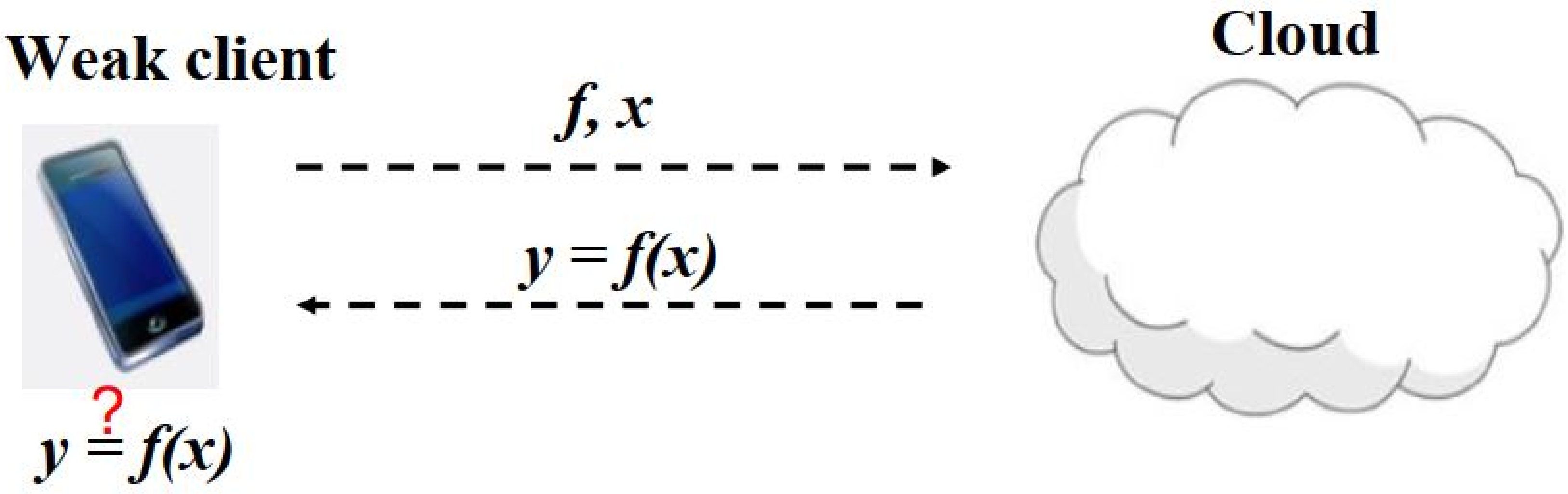
Outline

UNIVERSITY OF CALGARY

- Verifiable computation
- Backgrounds
  - CRR Protocol
  - Ethereum and smart contract
  - Merkle Hash Tree
- Our Work
  - Verifiable Computation using Smart Contracts
- Conclusion

Motivation

Outsourcing Computation



Verifiable outsourcing: Efficiently verify the correctness of a computation result that is provided by the cloud.

Verifiable Outsourcing

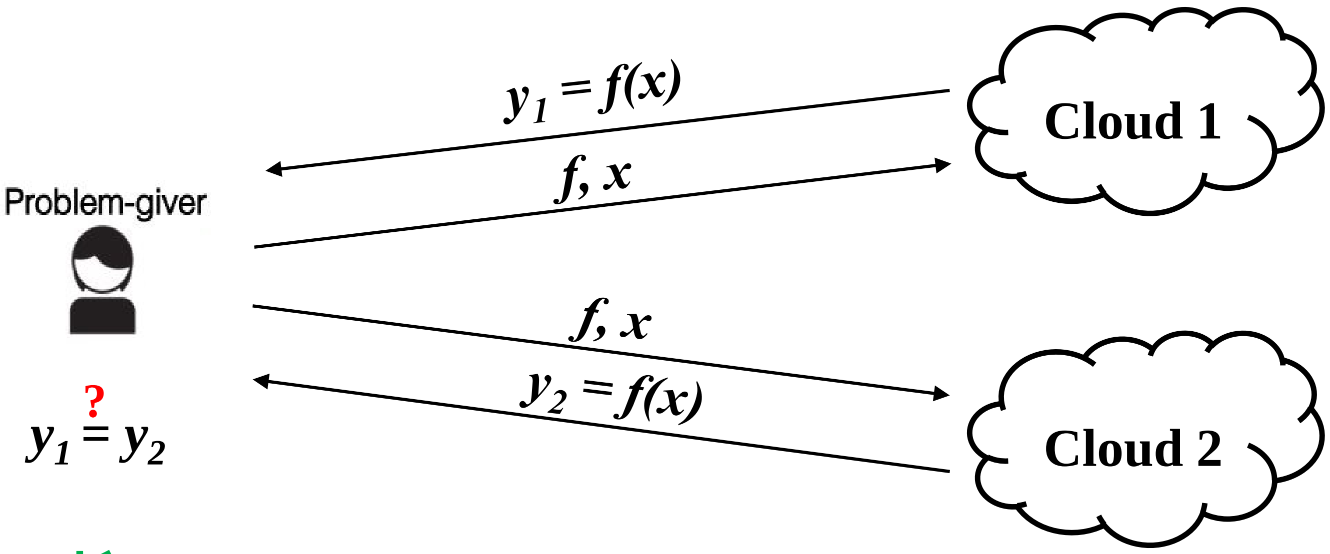
(Existing approaches)

- **Using cryptography:**
  - →Probabilistic checkable proofs [Kil92, Mic00]
  - →Homomorphic Encryption [GGP10, CKV10, AIK10]
    - →Expensive computation, inflexible
- **Outsourcing by replication**:
  - → Outsource the computation to a number of clouds.
  - → Select a solution that is generated by the majority of the clouds as the correct solution.
  - **Verifiable outsourcing using two clouds (Canetti, Rothblum and Riva [CRR11])**

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). STOC, 92
[Mic00] Silvio Micali. Computationally sound proofs. SIAM Journal on Computing, 2000.
[GGP10] Gennaro, R., Gentry C., and Parno B. Non-interactive verifiable computing: outsourcing computation to untrusted workers, CRYPTO'10.
[CKV10] Chung K.M., Kalai Y., and Vadhan S. Improved delegation of computation using fully homomorphic encryption, CRYPT'10
[AIK10] Applebaum B., Ishai Y., and Kushilevitz E.: From secrecy to soundness: efficient verification via secure computation. ICALP'10
[CRR11] Canetti, R., Riva, B., & Rothblum, G. N.: Practical delegation of computation using multiple servers, CCS'11

CRR Protocol

- ➢ Refereed Delegation of Computation (RDoC)



$y_1 \overset{\checkmark}{=} y_2$ Correct answer- Accept

$y_1 \overset{\times}{=} y_2$ Play refereed game - Identify malicious cloud
- binary-search
- verify-reduced-step

Strength:
- **Provable security**

Weakness:
- **Client is trusted**

[CRR11] Canetti, R., Riva, B., & Rothblum, G. N.: "*Practical delegation of computation using multiple servers*", CCS'11

Blockchain

- Key Components of Blockchain:

Node

- Full node
- Mining node (aka miner)
- Lightweight node

Transaction

- A cryptographically signed piece of instruction that is generated by a node and submitted to the blockchain.

Block

- Transaction data is permanently recorded in files called blocks.

Consensus

- To add a new block to the blockchain, all participating nodes must come to a common agreement (also called consensus).

- Forming blockchain: by chaining blocks

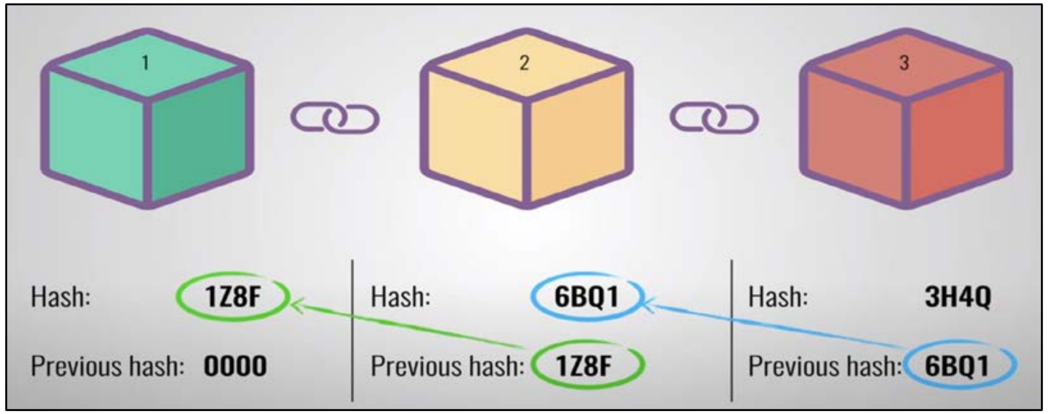


Figure: Example of forming blockchain

- Key Characteristics:
  - Decentralization
  - Anonymity
  - Transparency
  - Immutability

Types of Blockchain

- From Academic point of view
  - Public
  - Private
- From administrative point of view
  - Permissionless
  - Permissioned
- Example:
  - Bitcoin, Ethereum, Zerocash: Public
  - Hyperledger fabric, Ripple, Corda: Private

Ethereum

UNIVERSITY OF CALGARY

- Ethereum: An open source, decentralized computing platform
- Enables users to develop ***smart contracts*** and decentralized applications (DApps).
- Key terms
  - Peer-to-peer network of computers
  - Accounts
    - externally owned accounts (EOA)
    - contract accounts
  - Consensus algorithm
  - Ethereum Virtual Machine (EVM)
  - Smart contract
  - Gas
- Digital currency: *Ether*

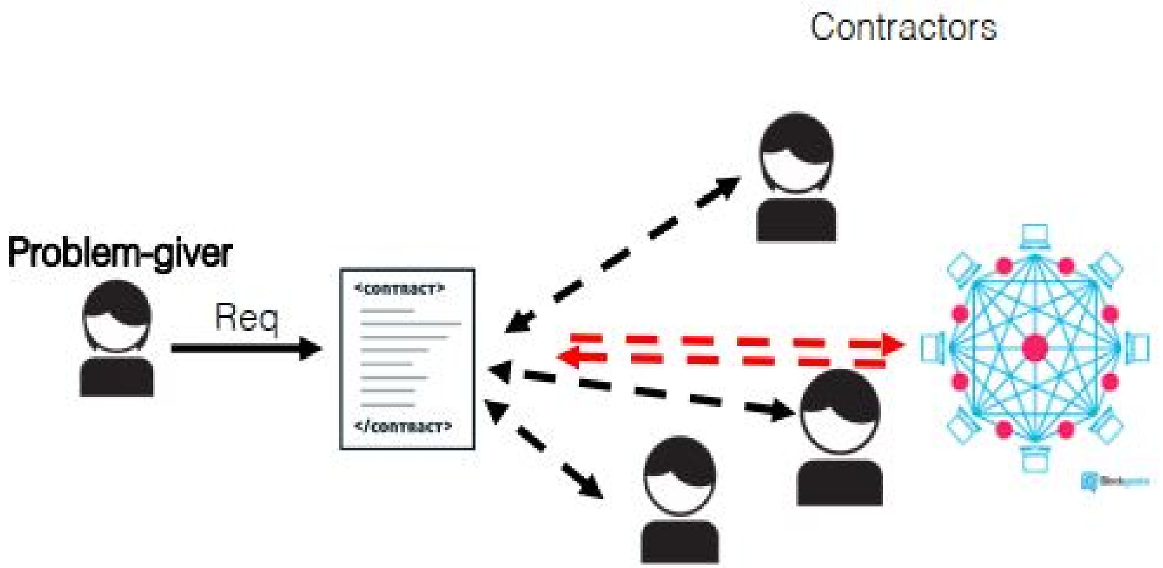
Smart contracts

- A smart contract is a computer program that is stored on the blockchain.
- A *contract creation transaction* deploys the contract code in the blockchain.
- The execution of the code is triggered by the transactions added to the blockchain
- Execution fees are defined in terms of ***gas*** and smart contract execution in Ethereum is bounded by ***gas limit***.

→**Advantages**:

- Guarantee correctness
- Manage interaction between parties
- Manage payments
- Immutable

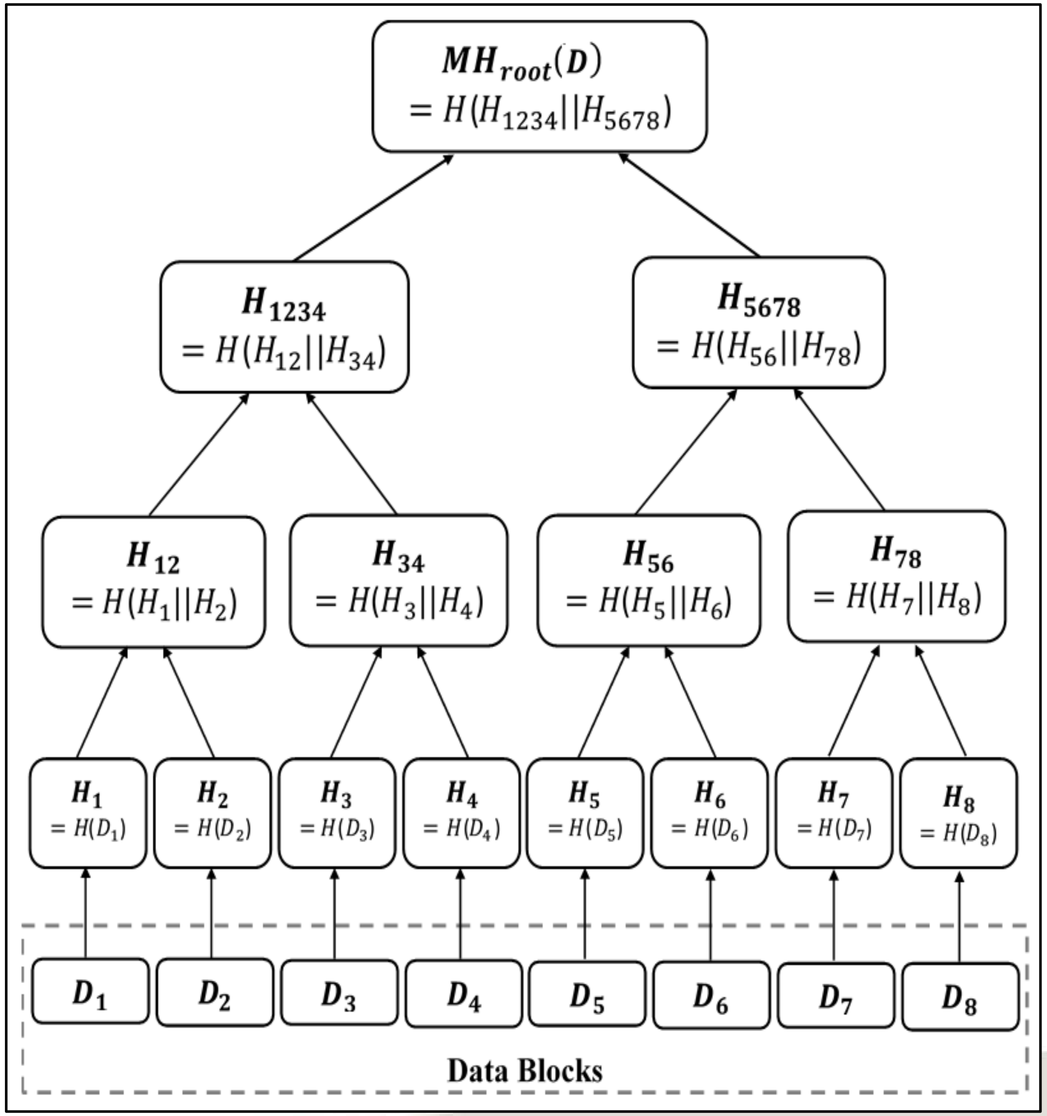
Goal: Smart contracts as a TTP for outsourcing



Merkle Hash Tree

- Binary tree constructed using collision-resistant hash function where,
  - each ***leaf node*** is the hash of data element D_i of set *D* of *n* elements,
  - every ***internal node*** is the hash of the concatenation of its two child nodes, and
  - the ***root*** is the hash for the full data set, denoted as $\boldsymbol{MH_{root}(D)}$, where $D = \{D_1, \dots, D_n\}$

- **Merkle Proof,** $(\boldsymbol{\rho_i})$
  - A path consisting of hash values along the path from the *i*th leaf to the root.
  - Used to efficiently prove that an element is included in the Merkle tree.
- **VerifyMHProof**
  - Function that verifies whether the *i*th leaf element corresponds to a Merkle tree with root $\boldsymbol{MH_{root}(D)}$ using proof $\boldsymbol{\rho_i}$.

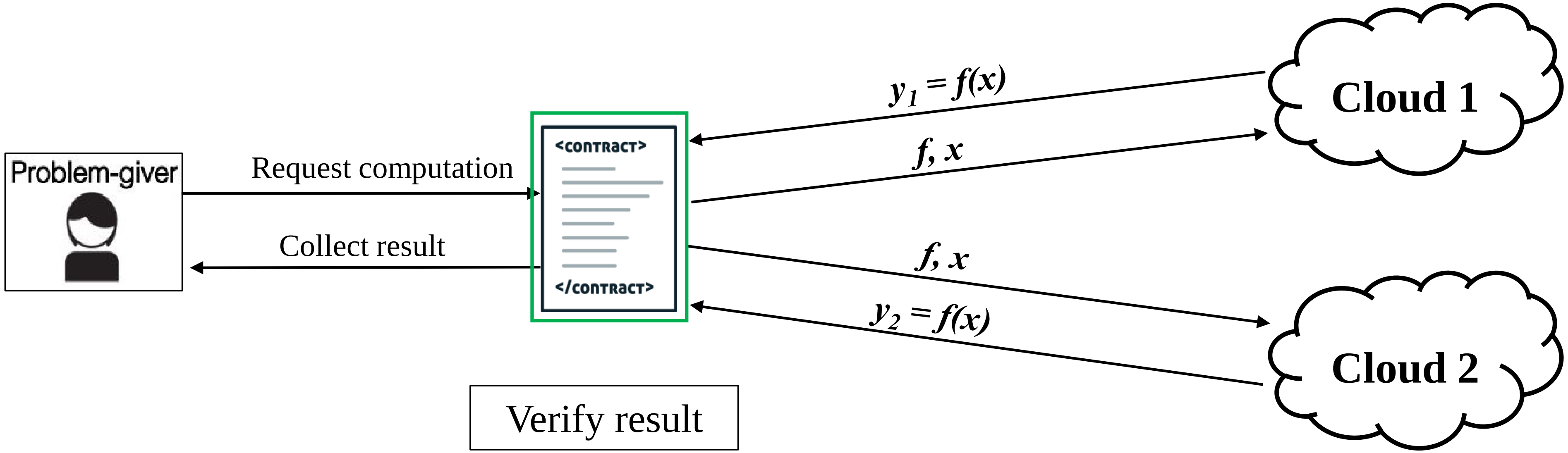


Our Contribution

- Verifiable Outsourcing
  - by using a smart contract
  - by using the CRR protocol for verifiable computation using two clouds
- Copy Attack
- Protection Mechanism
  - *Result Confirmation (RC)* protocol
- Implementation idea
- Delay analysis

Our proposal

Verifiable Computation using Smart Contracts

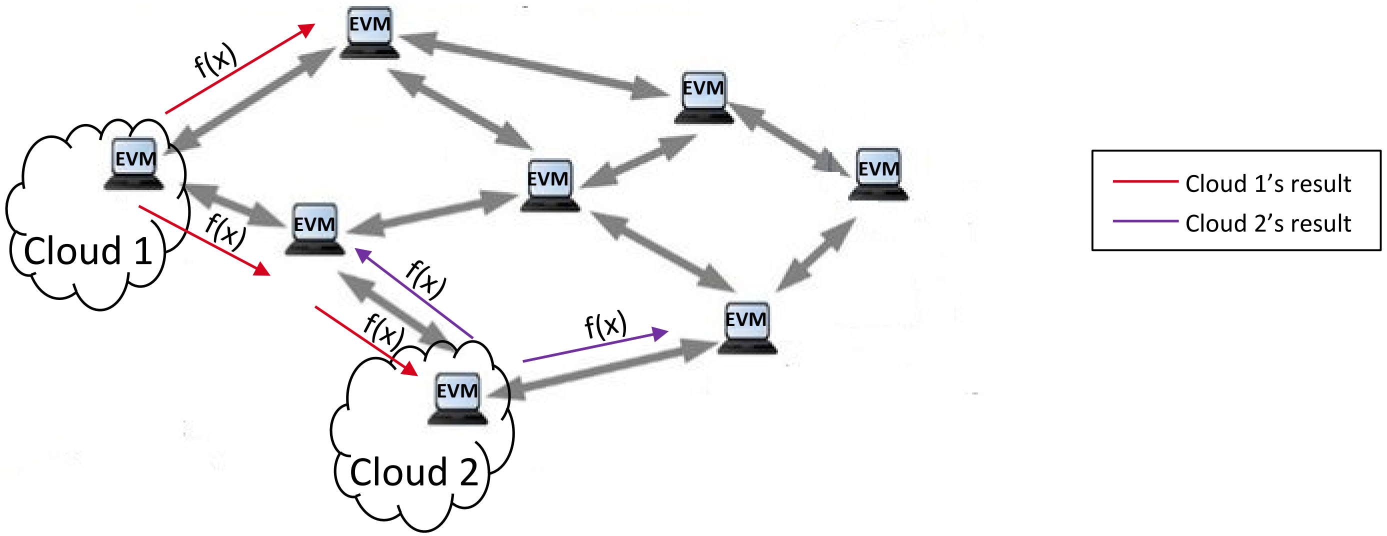


Assumptions:
- Client is untrusted.
- One of the clouds is malicious and the other is rational.

Problem:
Copy Attack

Copy Attack

1. Cloud 1 sends f(x) to smart contract.

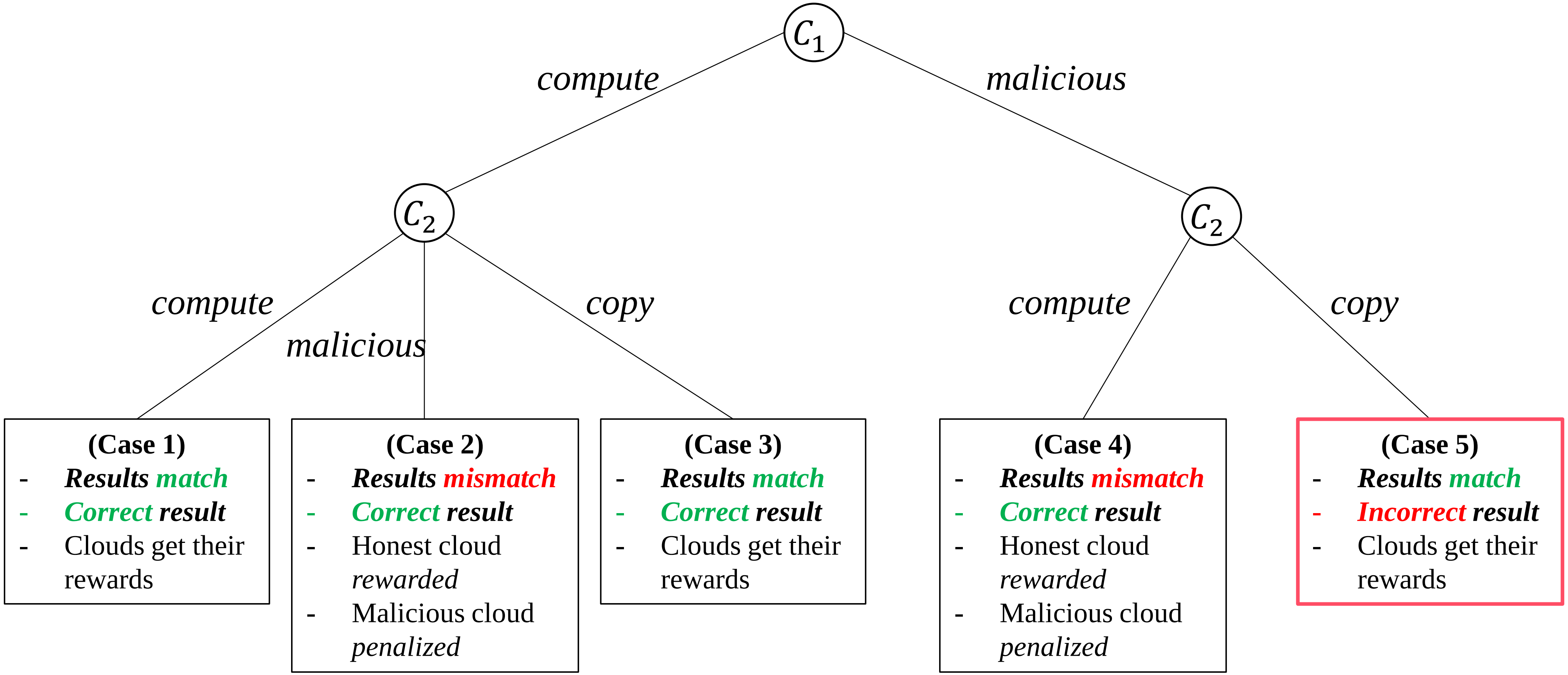


2. Cloud 2 sees f(x); copies f(x) and sends as its result it to the network.

3. All Ethereum nodes see two identical values from two clouds.
The result is accepted as correct.

Copy attack

- An attractive strategy for rational (uncorrupted) cloud.



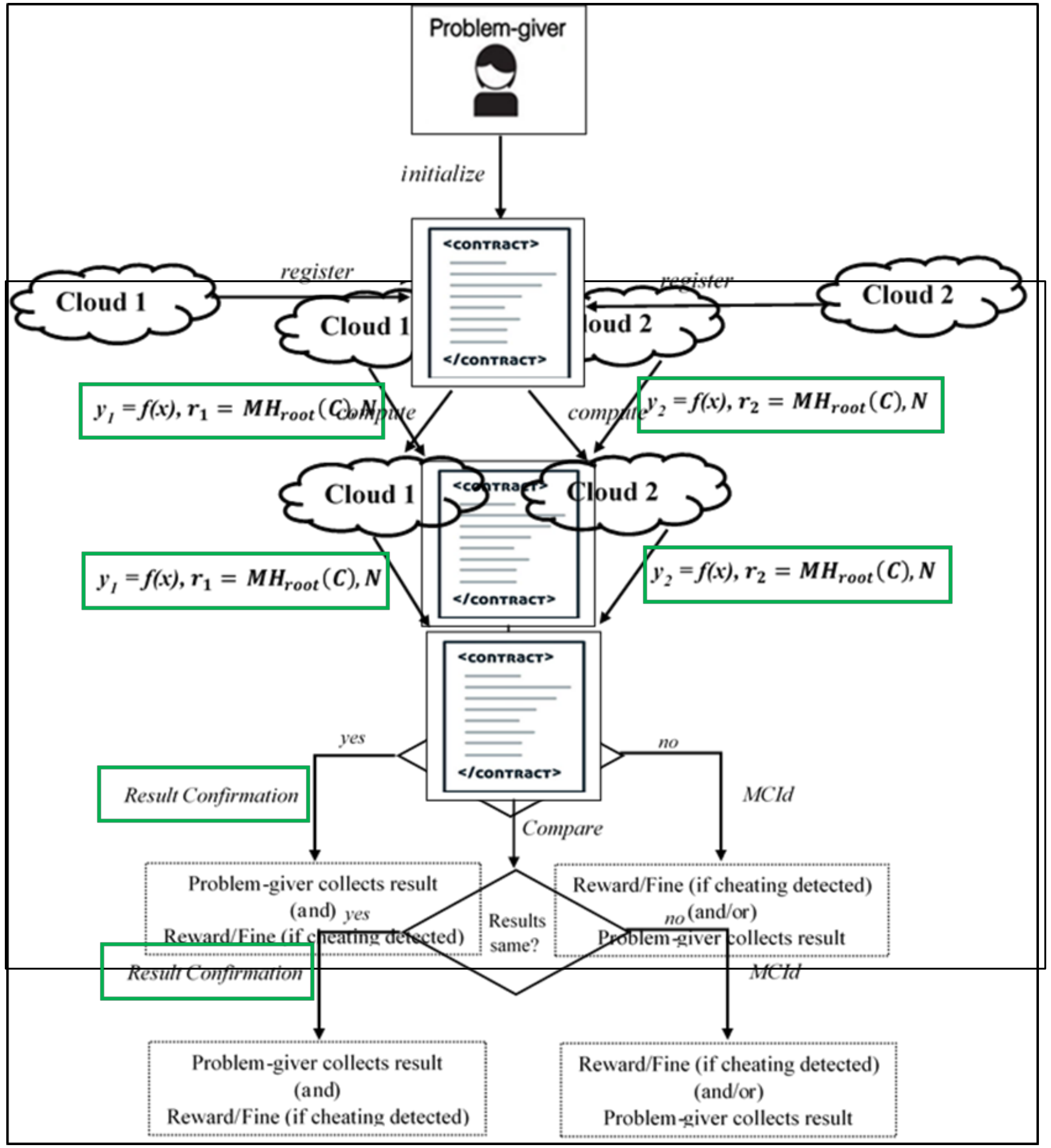
Protection against copy attack

scCRR (smart contract using CRR) **Protocol**:

- Each cloud i sends its result:

  $$(y_i,\ \boldsymbol{MH_{root}(C)},\ \boldsymbol{N})$$

- If the results match,
  - ***Result Confirmation*** (RC) protocol is used.
- If the results do not match,
  - ***Malicious Cloud Identification (MCId)*** protocol (of CRR) is used.



Notations:
C = *array of reduced configuartions*
$r_i = MH_{root}(C)$ = *Merkle Hash root constructed on C by Cloud i*
N = *length of the array C*

Computation Model

Reduced Turing Machine configuration:
$(state,\ head,\ tape[head], MH_{root}(tape))$

t: tape of configuration rc_1
$rc_1 = (s_1, h_1, v_1, root_1)$

Array of reduced configuartions:

C:

rc_1	rc_2	…	…	…	…	…	rc_N

Result Confirmation (RC)

- $SC \rightarrow Cloud_i\colon q_i = (i,\ x_i) \quad \forall\, i \in \{1,2\}$
- $Cloud_i \rightarrow SC\colon p_{x_i}$
- For each cloud *i*
  - $SC\colon VerifyMHProof(r_i\,, p_{x_i})$
    - If *True* => ***valid***
      Else ***invalid***

Theorem: *Let H be a collision resistant hash function that is used to construct the Merkle hash tree on the array of reduced configurations, C. Then RC protocol provide protection against copy attack.*

RC (an example)

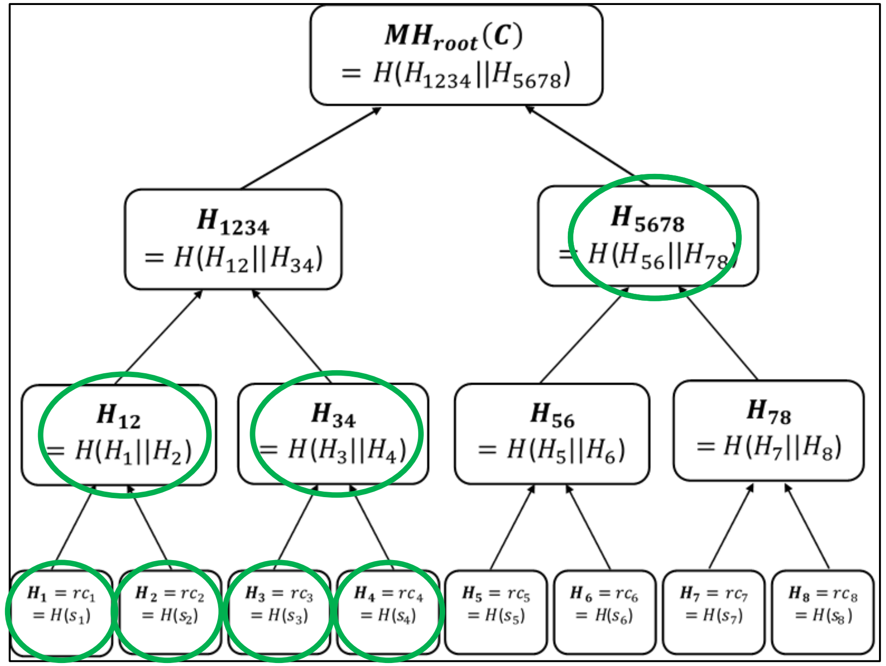
- $SC \rightarrow Cloud_1$: $q_1 = (1,\ 1)$
- $SC \rightarrow Cloud_2$: $q_2 = (2,\ 3)$

- $Cloud_1 \rightarrow SC$: $p_1 = (H_1, H_2, H_{34}, H_{5678})$
- $Cloud_2 \rightarrow SC$: $p_3 = (H_3, H_4, H_{12}, H_{5678})$

- Smart contract verifies:

$$r_1 \overset{?}{=} H(H(H(H_1||H_2)||H_{34})||H_{5678})$$

$$r_2 \overset{?}{=} H(H(H(H_3||H_4)||H_{12})||H_{5678})$$

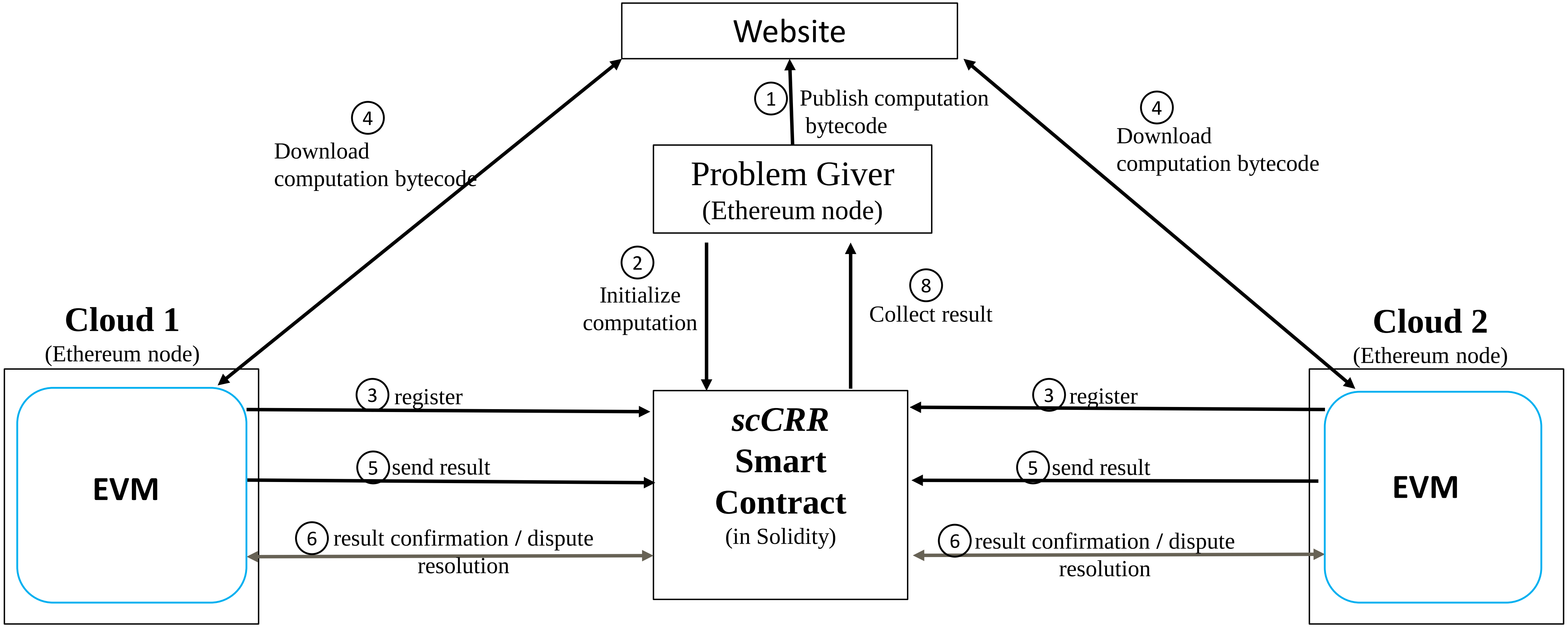


Abstract scCRR smart contract

UNIVERSITY OF CALGARY

```
pragma solidity >=0.4.0 <0.6.0;
contract scCRR {
constructor () public;
function Initialize (uint256 _task_url, uint256 _web_hash,
uint256 _comp_hash, uint _reward, uint min_deposit) public on-
lyOwner;
function Register (address _sender, uint _amount) public
payable;
function receiveResults (uint256 _result, uint256 _root, uint
tape_length) public;
function Compare (uint256 _result1, uint256 root1, uint256 _re-
sult2, uint256 root2) internal;
function resultConfirmation () internal returns (bool, bool);
function QueryGen (uint256 _k, uint256 d, uint256 N, uint256
idx) internal returns (uint, uint);
function binary-search (uint min_steps) internal returns (uint);
function verify-reduced-step (uint256 rc_ng, uint256 rc_nb,
uint256 p_ng) internal returns (bool);
function Pay (uint _case) internal;
function shutDown() internal;
}
```

Sketch of the implementation



Delay analysis

UNIVERSITY OF CALGARY

- The number of transactions that will be sent and received between the clouds and the smart contract for a given computation.

Phase	Register	receiveResult	RC	MCId
# **Transactions**	2	2	4	$4log(N) + 3$

Table: Number of transactions required in different phases of the smart contract execution.

Conclusion

UNIVERSITY OF CALGARY

- Verifiable Computation system based on CRR protocol using Smart Contracts.
  - Smart contract as a TTP
  - Copy attack and protection mechanism

- Future works

Thanks