Phase 1: Blockchain Research

Blockchain Senior Design: Caleb, Megan, Savannah, Timothy, Virginia

Implementing a Simple Blockchain using Python

- SHA-256 hashing algorithm to create the block’s hash for fingerprinting.
- Each block contains its hash and the previous block’s hash.
- Uses proof of work to mine for the next block.
- After several blocks have been added it checks the blockchain’s validity.
- The web application is made using Flask and can be deployed locally or publicly as needed by the user.

Simulation Code Output:


```
{
"index":2,
"message":"A block is MINED",
"previous_hash":"2d83a826f87415edb31b7e12b35949b9dbf702aee7e383cbab119456847b9
57c",
"proof":533,
"timestamp":"2020-06-01 22:47:59.309000"
}
```

```
{
"chain":[{"index":1,
"previous_hash":"0",
"proof":1,
"timestamp":"2020-06-01 22:47:05.915000"},{"index":2,
"previous_hash":"2d83a826f87415edb31b7e12b35949b9dbf702aee7e383cbab119456847b9
57c",
"proof":533,
"timestamp":"2020-06-01 22:47:59.309000"}],
"length":2
}
```

```
{"message":"The Blockchain is valid."}
```

Proof of Authority

- Used in a permissioned blockchain and requires all nodes to be pre-authenticated.
- Nodes can create a new block if they have proven their authority to do so.
- Needs less computational power, rate at which blocks are created is predictable, high transaction rate, and requires 51% of nodes to be compromised in order to go down.

Implementing it into our simulation:

- Calculate the leading node: leader = ((time - first) / step) % nodes
- New block is generated by the leader node and the leader role is passed to the next “validator” node from the list of authorized nodes.
- Block is validated
- Fork - longest blockchain rule

Proof of Authentication

- Used in a permissioned network with resource constrained distributed systems.
- Uses minimal resources, requires less time, reduces unstable network connectivity, and only trusted peers can authenticate.

Implementing it into our simulation:

- To start all nodes follow SHA-256 hash and each node has private and public keys.
- The nodes combine the various transactions to form a block.
- The nodes then sign the block with their private key and broadcast the block to the network.
- The trusted nodes will then verify the signature with the source public key.
- If the block is authenticated, the trusted node will broadcast the block for the other nodes to add to their blockchain.
- If the block is not authenticated, it will be dropped.

Practical Byzantine Fault Tolerance

- There is a primary node and other secondary nodes, and any node can become the primary node.
- No more than ⅓ of the nodes in the system can be malicious and with more nodes the system becomes more secure.
- It is energy efficient, does not require multiple transaction confirmations, and all nodes take part in handling client requests.

Implementing it into our simulation:

- A proposer is elected and it collects transactions from the transaction pool, creates a block proposal and broadcasts it to the network as “pre-prepared”.
- Validators receive the proposal and verify the proposal and broadcast it to other validator nodes as “prepared” and broadcast commit messages.
- Validators wait for commit messages and then enter into the committed state where the messages are received and add the block into the blockchain.
- After the block is added, the validators enter the final state where it is finished and will begin the new round.

Simulation Demo & Results

Timothy’s Discussion on Blocks

Goal: To build a trusted blockchain without using permissioned blockchain. By applying cybersecurity standards and known malicious behavior, we will integrate security into the blockchain and remove untrustworthy nodes.

- Basically, if node is untrustworthy:
  - Don’t add its data to the blockchain
  - Remove all traces from newest block
  - Mark as untrustworthy
- Uses Smart Contracts
  - Before running, check if contract is revoked
  - Used to detect, deter, and defy malicious devices

Smart Contracts

Smart contract vulnerabilities: Praitheeshan, Purathani, et al. "Security analysis methods on ethereum smart contract vulnerabilities: a survey." *arXiv preprint arXiv:1908.08605* (2019).

- For Ethereum, programmed in Solidity
  - Use newest version
  - Use tools discussed in article to design and test
- Most smart contracts not built for all possible scenarios of exploitation
  - Like needing mutexes and sending money out before checking the balance.
- Smart contracts will search transactions for indicators of malicious activity
- Smart contracts will inform devices to not trust devices it finds malicious
  - Would need to be aware of false positives?
  - Would this need an appeal system to fix issue?

Block Security

Read paper on public key cryptography in blockchain (PoAuth)

- Can be used for security and authenticity
- Options for distributing public key:
  - When node joins, public key broadcasted to nodes in network
  - Next block has the new node's public key and all updated keys in newest recent block
- Options to verify:
  - Digitally sign current transaction
  - Redundant transaction that is digitally signed
    - Saves time decrypting signature
  - Signed transaction input → verified transaction in block:
    - If digital signature legitimate, add
    - Validators take slightly more time to process
- When device leaves network, remove key from list of public keys:
  - Option: Either key manager on each device, store public keys in xml file, or just pull from blockchain.

Block Size

There are trade-offs between scalability and decentralisation. For example, a larger block size would allow the network to support a higher transaction rate, at the cost of placing more work on validators — a centralisation risk. –Enabling Blockchain Innovations with Pegged Sidechains, 2014

- Tradeoff between larger and smaller blocks leading to differing capacity
- Unfeasible to collect all network traffic, so store portion of traffic in blockchain:
  - Important messages: ICMP, ARP, TELNET, etc.
  - Any additional messages/protocols to always record, like ssh connections/teardowns?
  - Tcpdump plugin or continually running packet capture from persistent device
  - Goal is to save messages that will identify patterns similar to cyberattacks
- Need to size collection window correctly to optimize for speed and bandwidth
- Either window of time for transactions or fixed number/size of transactions per block
- Any suggestions on size or should we set an initial time and modify based on performance?

What goes in a Block?

A Block consists of:

- Index
- Timestamp
- Block hash
- Previous block hash
- Transactions
    - Are transactions made from a packet dump of network traffic?
    - Selective packet transmissions - ARP messages, dissociation frames, & ICMP msg
    - Invocations of smart contracts - results added to most convenient immediate block
    - Any new smart contracts
    - All smart contracts for defense located in first/second block for easy access. Only original node can create removal contracts unless updates made
    - Addition of members - maybe updated list of members?
    - Updated list of public keys

Proposed Block Contents:

- Standard elements:
    - Index
    - Timestamp
    - Current block’s hash
    - Previous block’s hash
- Data:
    - Smart Contracts:
        - Revocation of old contract - requires digital signature of all parties
        - All instances of smart contract invocation and target
            - For defense: when device was removed from list of approved devices
            - Prevent changing of IP by blacklisting MAC address.
            - Further deter by comparing new device’s behavior to that of previous device
        - New contracts
    - Active Members list
        - Includes new members
    - Malicious members list
        - MAC address and IP
    - Updated list of member’s public keys
    - Data transmissions:
        - What do we want for transactions - ARP messages, dissociation frames, and ICMP messages
        - Other relevant packets i.e. ssh or connection to persistent devices

Networking

Goal: “Instead of relying on traditional cloud data centres, blockchain interconnects computer nodes, including virtual machines on cloud and external computers, to build a fully decentralized storage system without requiring a central authority.”

Source: “Integration of Blockchain and Cloud of Things: Architecture, Applications and Challenges” by Nguyen, Pathirana, Ding, & Seneviratne

Network Rules

Qualifications:

- “eliminating the requirement for a trusted third party
- …peer-to-peer architecture …
  - “**most** important feature is **decentralization**… not rely on a central point of control to manage transactions”
- allows all network participants to verify …
- equal validation rights.”
- “capability to create,
- authenticate and
- validate the new transaction to be recorded in the blockchain”

Network Questions

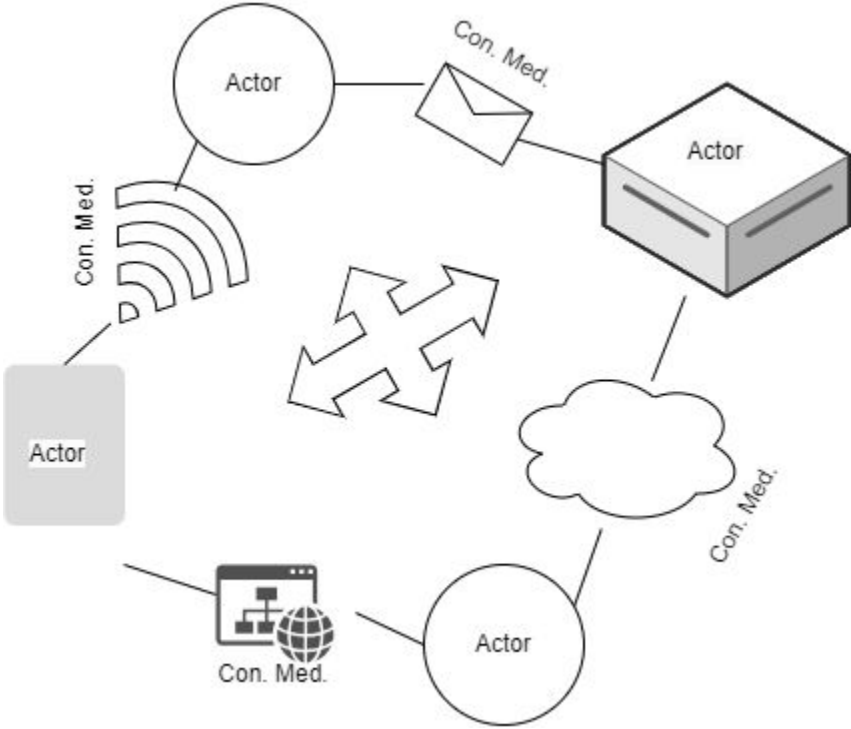
- What quantity of data will we be handling?
  - “In largescale blockchain applications, the **number of transactions**… can be enormous… necessary to provide powerful data processing services… the **cloud** can offer on-demand computing resources for blockchain operations thanks to its **elasticity and scalability**”
  - Concern - “Many current blockchain systems suffer from high block generation time which in turn reduces the overall blockchain throughput. Further, if all transactions are stored in a chain, the blockchain size will become very large”
- “public (or permission-less) and private (or permissioned) blockchain”?
  - “private blockchain is an invitation-only network managed by a central entity and all participations in blockchain for submitting or writing transactions have to be permissioned by a validation mechanism”

Network Actors (simulated)

- Phone, tablet:
  - “holds a blockchain account (like a wallet in Bitcoin)...join the blockchain network to perform transactions …resource-limited …lightweight node that can participate in the validation process of a transaction. All interactions… with blockchain… are **performed by the gateway**”
- Computer:
  - “serve other lightweight IoT sensors and maintain the full blockchain. …interact … through IoT gateways to achieve corporative communication (e.g., device to device (D2D) communication in collaborative networks)...
  - **hybrid communication concept**”
- Cloud
  - Data aggregated by IoT gateways… received… and kept in the cloud blockchain storage
  - Shared ledger, consensus, smart contract, cryptography
  - Transforms idea to ‘blockchain as a service”

Network structure

- Actors
- Connection medium - “network of computing servers which are interconnected robustly by collaborative clouds”
  - (“the inter-cloud ecosystem can enable the blockchain system to operate continuously”)
- Mechanisms: “data block, distributed ledger, consensus, and smart contracts.”
  - Some mechanisms are also items
  - Sent, stored, and/or executed



Using the cloud for all CMs
=
“joint cloud collaboration environment where multiple clouds are interconnected securely by a peer-to-peer ledge network”

More on Network mechanisms

Distributed ledger (database) - “same copy of records of blockchain spreads across … all blockchain users can fully access, verify and track transaction activities over the network with equal rights”

consensus (verification)

smart contract - build logic & trust, self executing, access authentication, data sharing confirmation

cryptography (public key)

Novel concepts

- distributed side blockchains
- fog nodes
  - Mediary before cloud
- Multi-blockchain
  - applied to integrity evaluation
- two-layer blockchain - integrity focus
  - data validation layer
  - PoW task layer
  - integrated with IaaS cloud
- Micro-clouds
  - Ubuntu Linux microcloud

Smart Contracts

Pieces of code, automation

Store, verify, execute rules

“Vending Machine” (conditional)

Immutable, distributed, deterministic

Resides in the blockchain, triggered by call to its address

ethereum/solidity

Security Analysis Methods on Ethereum Smart Contract Vulnerabilities - A Survey

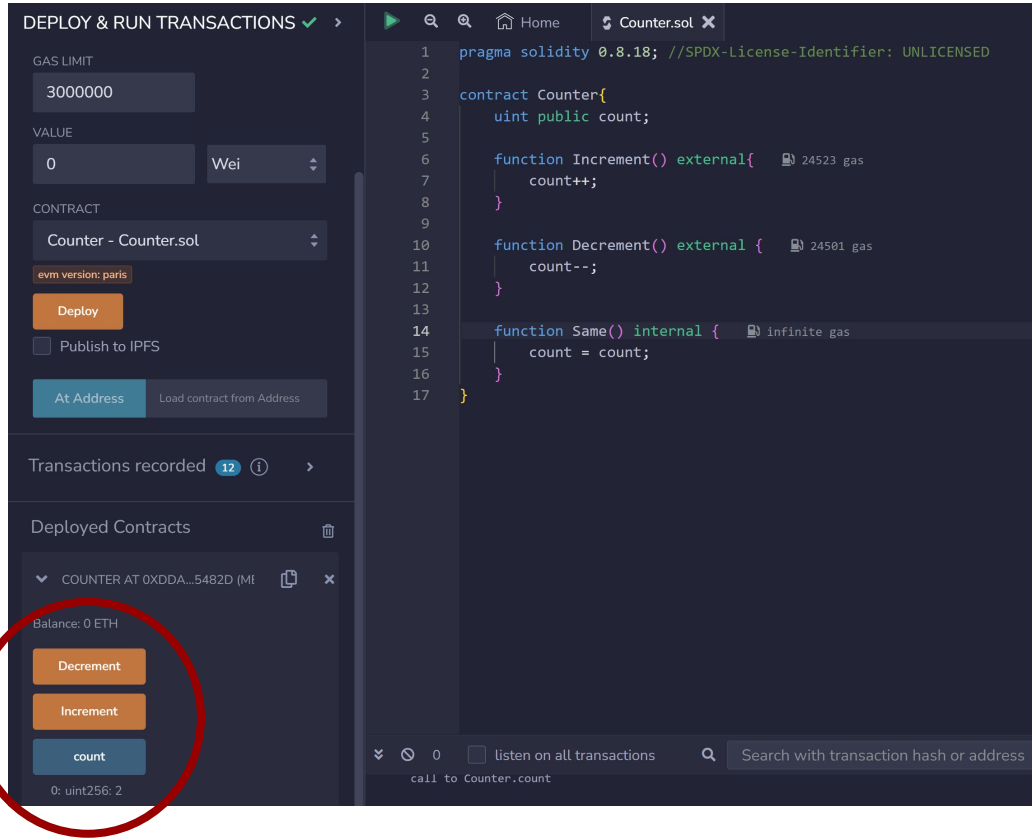
“The combination of vulnerabilities in Ethereum blockchain and Solidity programming language makes the security checks more challenging in smart contracts development”

Many different potential vulnerabilities

Tools to assist

Mitigate through testing and good practices

remix.ethereum.org



Online IDE to create and test Smart Contracts on a simulated network

internal/external

Uint256

public/private