Blockchain and Cloud
Chain-of-Custody

8 June 2017

Val A. Red
Cyber Integration Program Manager
Associate of (ISC)²
Information Directorate
Air Force Research Laboratory
The views I am about to express are my own and do not necessarily represent the views of the Air Force Research Laboratory, the United States Air Force, the Department of Defense, or the United States Government.
Agenda

- Why blockchain?
- Why not centralized databases?
- What and which blockchain?
- What should I ask before considering blockchain?
- Blockchain and chain-of-custody via data provenance
- AFRL Research on blockchain chain of custody

Slides based on:

Why blockchain?

• Securely and traceably share data among multiple parties without intermediaries even as no single party is completely trusted
  – Jurisdiction differences
  – Compromised, untrusted peer
  – Insider threat

• Aforementioned problems translate nicely to IoT applicability.

• Not to mention economic benefits of disintermediation.

• Consensus built-in enables sensor data to be immediately actionable.

Blockchain at a minimum provides immutable and traceable records of ownership, custody, and state of data in environments of distrust.
Why not blockchain?

Traditional databases are still superior in two categories: (1) Efficiency of performance in cases where mass signature verification, consensus, and redundancy are unneeded. (2) Confidentiality: When database current state & state changes from transactions must be private. ^ Open, permissionless blockchain: **pseudonymous but NOT private**

<table>
<thead>
<tr>
<th>Number Of Transactions</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>0 (Mainchain)</td>
</tr>
<tr>
<td>Block Reward</td>
<td>50 BTC</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Jan 3, 2009 1:15:05 PM</td>
</tr>
<tr>
<td>Mined by</td>
<td></td>
</tr>
<tr>
<td>Merkle Root</td>
<td>4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b</td>
</tr>
<tr>
<td>Difficulty</td>
<td>1</td>
</tr>
<tr>
<td>Bits</td>
<td>1d00ffff</td>
</tr>
<tr>
<td>Size (bytes)</td>
<td>285</td>
</tr>
<tr>
<td>Version</td>
<td>1</td>
</tr>
<tr>
<td>Nonce</td>
<td>2083236893</td>
</tr>
<tr>
<td>Next Block</td>
<td>1</td>
</tr>
</tbody>
</table>

The first Bitcoin block (2009 January 3)
Why not a centralized database?

Traditional databases require expensive engineering or configuration to overcome the following:
- Variable trust among peers (no intermediaries; server-client risky)
- Robust decentralization – Traditional databases require heavy configuration required for replication and failover to tolerate byzantine failures

The Byzantine generals’ problem illustrates how in centralized circumstances, variable trust could induce failure in a whole system.
What blockchain?

Disregarding every other important feature built on top of it (consensus, smart contracts, etc.), blockchain is a **distributed** and **cryptographically secure** log of hashes stored in blocks that traceably links to the previous block of hashes.

Distributed

Cryptographically Secure

Hashes with SHA256;
Digital Signatures with ECDSA
   (Elliptic Curve).

In other, other words: Bitcoin miners are crunching numbers at the equivalent rate of solving a SHA1 collision every second.
Log of hashes stored in blocks

Merkle Tree - Storage-efficient verification of large data hashes
• Directed Acyclic Graphs – instead of a singular block-to-block traceback, one block can link back to multiple preceding blocks.
Modules or features that distinguish among distributed ledgers:

- Consensus
- Smart Contracts (chaincode)

Combinations of these & their quality make or break respective implementations (proof-of-work consensus for bitcoin, for example)
What should I ask before even considering blockchain?

From: “Avoiding the pointless blockchain project” – Gideon Greenspan, CEO/founder of Coin Sciences article:

1) Is a traditional relational database not enough?
2) Are there multiple parties modifying the database?
3) Are the multiple parties in different trust domains?
4) Are intermediaries unacceptable?
5) Are database records interdependent?
6) Do we have rules to control input of records?
7) Are validators in the form of consensus necessary?

ProvChain: A Blockchain-based Data Provenance Architecture in Cloud Environment

Sachin Shetty
Associate Professor
Virginia Modeling, Analysis and Simulation Center
Department of Modeling, Simulation and Visualization Engineering
Center for Cyber Security Education and Research
Old Dominion University

Collaborators:
Xueping Liang (Tennessee State University)
Deepak Tosh (Norfolk State University)
Charles Kamhoua (AFRL/RIGA)
Kevin Kwiat (AFRL/RIGA)
Laurent Njilla (AFRL/RIGA)
Lt. Val Red (AFRL/RIGD)
Roadmap

• Blockchain Overview

• Provchain- Data Provenance in Blockcloud
  – Motivation
  – ProvChain Architecture
  – ProvChain Operation Phases
  – Performance Evaluation
  – Conclusion
Problem we are trying to solve: How do multiple entities agree on history?

“He who controls the past, controls the future.” (Orwell)
• Alice sends Bob message indicating first move
• Bob responds back to Alice with the message indicating second move
• .... So on and so forth
• *Is there a necessary condition that needs to be satisfied in order for them to be able to play the game?*
Blockchain Overview
Need agreement on the state of the board

Alice and Bob cannot play the game if they don’t agree on the board state

1. Aware of the sequence of messages exchanged.
2. Also aware of the starting positions of the board.
3. We have necessary conditions to reconstruct the board state

If multiple entities have consensus on history, we agree on the current system state!
Blockchain Overview
Agreement on history among multiple untrusted entities

- **Necessary conditions to agree on the current system state**
  - Starting state + history $\rightarrow$ current state
  - Money in account
  - Property ownership
  - What’s the current state of my program?

- **Current Solution** - Centralized trusted arbiter (ex. database) to report on current system state.
  - Need for decentralized control and eliminate single point of failure or compromise
Blockchain Overview
Blockchain Data Structure

• Distributed database maintains list of records (blocks)
  – Secured from tampering and revision by distributed storage and continuous verification.

• Each record is maintained in the blockchain which ensures a temporal listing of transactions
  – Linear order using a persistent, immutable, yet append-only data structure that is globally viewed
Blockchain Overview
Blockchain Security and Hash Functions

• Each block within the blockchain is identified by a hash, generated using the SHA256 cryptographic hash algorithm on the header of the block.
  – Each block also references a previous block, known as the *parent* block, through the "previous block hash”

• Sequence of hashes linking each block to its parent creates a chain going back all the way to the first block ever created

• Child’s own identity changes if the parent’s identity changes
  – Cascade effect ensures that once a block has many generations following it, it cannot be changed without forcing a recalculation of all subsequent blocks.

• Recalculation requires enormous computation which guarantees the immutable property of blockchain
Blockchain Overview

Merkle Tree

• Each block in the bitcoin blockchain contains a summary of all the transactions in the block, using a merkle tree.

• Merkle trees are binary trees containing cryptographic hashes.

• A merkle tree, also known as a binary hash tree is constructed by hashing paired data (the leaves), then pairing and hashing the results until a single hash remains, the merkle root.

• If any single leaf transaction is changed, all hashes along the branch would be changed and ultimately the merkle root as well.
  – Key property ensuring security of the blockchain
Blockchain Overview

Merkle Tree

Source: Bitcoin and Blockchain, OReilly
Blockchain Overview
Blockchain Infrastructures

**Public blockchain:**
- Permission less – anyone can use it
- “Proof” consensus – “Proof of Work” for Bitcoin
- Public nodes
- Cryptocurrency token
- Open wallet access/internet

**Semi-private & private blockchain:**
- Permissioned – defined group can participate
- Custom consensus engine – rules set by participators
- Private nodes – closed group
- Optional token
- Closed wallet access/VPN
Blockchain Overview
Summarizing

• No need to trust each other or have a trusted third party
• Distributed system
• Agreement on history translates to agreeing on system state
• Nth record in the hash chain commits to all previous records.
• Any change in previous record invalidates hash chain
• A blockchain is a hash chain with procedures for validity and resolve disagreements
  – Permissionless vs Permissioned infrastructure
  – Proof of Work vs Proof of Stake vs Proof of Storage, etc
• Blockchain facilitates secure transfer of information through a sequence of cryptographically-secure keys.
  – No need for a central authority
  – System of distributed ledgers which records all actions
  – Transactions in the public ledger verified by a consensus of majority of participating entities.
  – Data provenance in the distributed public ledger catalogs all operations on data related to an asset.
  – Owner of an asset can authenticate a transaction and facilitate transfer to another owner without the need for an arbitrator.
Data Provenance and Blockchain

• Users will use cloud storage system to store and share data.
  – User can control and audit file operations
• Malicious behavior detection and notification
• Store all file operations on the blockchain.
  – Every node on the blockchain can verify the operation by mining the block so that data provenance is authentic.
  – Cloud service provider assign read and write
  – A detection method can be adopted to instantly identify malicious access and an alert system can be adopted to send messages to all stakeholders.
ProvChain architecture

<table>
<thead>
<tr>
<th>FileID</th>
<th>Time</th>
<th>User</th>
<th>Operation</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileX</td>
<td>20160824</td>
<td>Alice</td>
<td>Create</td>
<td>CloudA/A</td>
</tr>
<tr>
<td>FileY</td>
<td>20160825</td>
<td>Bob</td>
<td>ModifiedFromX</td>
<td>CloudA/B</td>
</tr>
<tr>
<td>FileZ</td>
<td>20160825</td>
<td>Eve</td>
<td>SharedFromX</td>
<td>CloudA/C</td>
</tr>
</tbody>
</table>

Layer 1: Data Storage
Layer 2: Trace Blockchain
Layer 3: Log Database
Block Structure

![Block Structure Diagram](image-url)
ProvChain architecture

- Data storage layer
  - File as the data unit

- Blockchain layer
  - File operation is recorded on the block
  - Each block has the previous block’s hash

- Database layer
  - Complete file operation record
  - Backup for future query
## ProvChain architecture

### File Operation Definition

<table>
<thead>
<tr>
<th>RecordID</th>
<th>Date and Time</th>
<th>UserID</th>
<th>FileID</th>
<th>FileHash</th>
<th>Operation</th>
<th>Description</th>
<th>hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>2016-9-7 10:00:00</td>
<td>A</td>
<td>X</td>
<td>n-bits</td>
<td>Create</td>
<td>Create at CloudA/A</td>
<td>m-bits</td>
</tr>
<tr>
<td>002</td>
<td>2016-9-7 11:00:00</td>
<td>A</td>
<td>X</td>
<td>n-bits</td>
<td>Copy</td>
<td>Copy to CloudB/A/backup</td>
<td>m-bits</td>
</tr>
<tr>
<td>003</td>
<td>2016-9-7 12:00:00</td>
<td>A</td>
<td>X</td>
<td>n-bits</td>
<td>Read</td>
<td>Read at CloudA/A</td>
<td>m-bits</td>
</tr>
<tr>
<td>004</td>
<td>2016-9-7 13:00:00</td>
<td>A</td>
<td>Y</td>
<td>n-bits</td>
<td>Write</td>
<td>Write at CloudA/A and save as File Y</td>
<td>m-bits</td>
</tr>
<tr>
<td>005</td>
<td>2016-9-7 14:00:00</td>
<td>A</td>
<td>X</td>
<td>n-bits</td>
<td>Share</td>
<td>Share to User B</td>
<td>m-bits</td>
</tr>
<tr>
<td>006</td>
<td>2016-9-8 09:00:00</td>
<td>B</td>
<td>Z</td>
<td>n-bits</td>
<td>Create</td>
<td>Create at CloudB/B from shared file in CloudA/A</td>
<td>m-bits</td>
</tr>
<tr>
<td>007</td>
<td>2016-9-8 09:00:00</td>
<td>B</td>
<td>Z</td>
<td>n-bits</td>
<td>Read</td>
<td>Read at CloudB/B</td>
<td>m-bits</td>
</tr>
<tr>
<td>008</td>
<td>2016-9-8 10:00:00</td>
<td>B</td>
<td>Z</td>
<td>n-bits</td>
<td>Read</td>
<td>Read at CloudB/B</td>
<td>m-bits</td>
</tr>
<tr>
<td>009</td>
<td>2016-9-8 11:00:00</td>
<td>B</td>
<td>Z</td>
<td>n-bits</td>
<td>Delete</td>
<td>Delete from CloudB/B</td>
<td>m-bits</td>
</tr>
<tr>
<td>010</td>
<td>2016-9-9 14:00:00</td>
<td>A</td>
<td>X</td>
<td>n-bits</td>
<td>Share</td>
<td>Share to public</td>
<td>m-bits</td>
</tr>
</tbody>
</table>
ProvChain architecture

1. User registration using registration key $K_{UR}$

2(a). Request data access using data encryption key $K_{DE}$

2(b). Share data using data sharing key $(PK_{DS}, PR_{DS})$

2(c). Request data

3. Publish provenance data using $K_{PV}$

4. Store provenance data locally

5. Request data provenance validation

6. Validate user provenance data using provenance validation key $K_{PV}$

7(a). Return validation result

7(b). Block # Transactions

7(c). Blockchain receipt

8. Update provenance data validation status

Blockchain Network

Provenance Auditor (PA)

Provenance Database

User

Cloud Service Provider (CSP)
ProvChain Entities

- **User**
  - User has ownership over their data and can audit their own data based on the provenance data hashed and stored on the blockchain.
  - Users are part of the public ledger.
- **Cloud Service Provider**
  - provides a cloud storage service
  - responsible for user registration for provenance and blockchain setup
- **Provenance Auditor (PA)**
  - retrieve all the provenance data from the blockchain into a log database
  - detect malicious behaviors based on the provenance data
- **Log database**
  - Append data provenance entry
- **Blockchain network**
  - Block miming to achieving consensus
ProvChain Activities

- Monitor user activities in real time using hooks and listeners for generating provenance data.
- Store all hashed data operations on blockchain network.
- Every node on the blockchain can verify the operation by mining the block
- Publish provenance data and hash user identification to preserve anonymity
- Provenance auditor validates provenance data by retrieving transactions from the blockchain network
Key Establishment

• Blockchain node enrollment key
  – Cloud service provider generates the enrollment key
  – assigned read and write permission to that node for blockchain access.
  – User authentication key

• Provenance auditor enrollment key
  – assigned read permission

• Data encryption key
  – user has the option to encrypt the file

• Data sharing key
  – Indicate data ownership

• Provenance verification key
  – Assign a provenance auditor to audit the provenance data for investigation purposes
ProvChain Phases

• Provenance Data Collection
  – Once user performs actions on the data files stored in the cloud, the data operation will be monitored and recorded.

• Provenance Data Storage
  – When an operation is monitored, the record is generated, which will be uploaded to the blockchain network and stored in the provenance database.

• Provenance Data Validation
  – The blockchain receipt contains information of the blockchain transaction and the Merkle proof used to validate the transaction.
ProvChain Phases

- Provenance Data Collection and Storage Phase
• Original Provenance Data in JSON format

```json
{
  "app": "files",
  "type": "file_changed",
  "affecteduser": "test",
  "user": "test",
  "timestamp": "1475679929",
  "subject": "changed_self",
  "message": "",
  "messageparams": "[]",
  "priority": "30",
  "object_type": "files",
  "object_id": "142",
  "object_name": "66.txt",
  "link": "/apps/files/"
}
```
ProvChain Phases

• Provenance Data Validation Phase

```json
{
  "@context": "https://w3id.org/chainpoint/v2",
  "type": "ChainpointSHA256v2",
  "targetHash": "82e46fffd212d680b3e1a169e6a8b59472985ac55398b8740832fe94fd5e5fd63",
  "merkleRoot": "9f010005a430539796817ce626d84cbb5485453e4d558cf3353e4d4a7e59031",
  "proof": [
    {
      "right": "0f6117e8bddd7f6c713aa5365e74aafe34f5cc31fd654ed84ea37976d873c087",
    },
    {
      "left": "f860e7697ba57d944d925f311cce786e6d20833071d1c16e6e5fef3fc4749c96",
    },
    {
      "right": "de4b5b29183d193b95905ae9741a928ab056cbbbefb9a537ac9282fe180c78bd",
    },
    {
      "right": "e75da94bc44a3c9778b2ec7a5fffd58e4a622d4ce4c20676215eb88a4764bb335"
    }
  ],
  "anchors": [
    {
      "type": "BTCOpReturn",
      "sourceId": "0b956b057330591cd63c90e5572ba364c6f9f08299c3e8ee0c893411db1c30a6"
    }
  ]
}
```
• Provenance Data Validation Phase

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>H4 = Hash(H3 + D)</td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>H3 = Hash(H2 + C)</td>
<td>H2 = Hash(B + H1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H1 = Hash(targetHash + A)</td>
<td>C</td>
</tr>
<tr>
<td>targetHash</td>
<td></td>
<td>A</td>
</tr>
</tbody>
</table>
```
ProvChain Phases

• Provenance Data Validation Phase

Algorithm 1 Blockchain Receipt Validation Algorithm

1: procedure VALIDATE(proof, merkleRoot, targetHash)
2:    nodeNum ← number of Merkle tree nodes in proof
3:    h ← targetHash
4:    i ← 0
5:    while i < nodeNum do
6:        if proof(i).key = right then
7:            h ← hash(h + proof(i).value).
8:        else
9:            h ← hash(proof(i).value + h).
10:       end if
11:       i ← i + 1
12:    end while
13:    if h = merkleRoot then return true
14:       return false
15: end procedure
ProvChain Phases

- Transaction and Block Information

<table>
<thead>
<tr>
<th>Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Hash: 000000000000000002cd3c7cbe44e4d95dc1db4511f632b79e35889a8feda1f1</td>
</tr>
<tr>
<td>Prev Block: 00000000000000000000014c5703fc65a02a7be1f3c3b0af041456c4380bf552a6d712</td>
</tr>
<tr>
<td>Next Block: 000000000000000000000169ca10626ef3f0308e8bf5d56989bd73d51b870025d9c47</td>
</tr>
<tr>
<td>Merkle Root: b1b8647c3f12ccf3505b60cd3e4a4bbcc442838bc69dc4df1bc0710fb990ec33cc6</td>
</tr>
</tbody>
</table>

| Height: 432992 |
| Confirmations: 1755 |
| Size: 998,130 Bytes |
| Tx Count: 2,634 |
| Time: 2016-10-05 15:23:38 |
| Version: 0x20000000 |
| Difficulty: 392.44 G / 241.23 G |
| Bits: 0x18048ed4 |
| Nonce: 0xf9526b4d |
| Relayed: BW.COM |

<table>
<thead>
<tr>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Txid: 0b956b057330591cd63e90e5572ba364e6f9f6299c3e8ee0c893411db1c30a6</td>
</tr>
<tr>
<td>Size: 234 Bytes</td>
</tr>
<tr>
<td>Height: 432992</td>
</tr>
<tr>
<td>Confirmations: 1754</td>
</tr>
<tr>
<td>Timestamp: 2016-10-05 15:23:38</td>
</tr>
<tr>
<td>Output Scripts:</td>
</tr>
</tbody>
</table>
  - OP_RETURN 9f0100055e430539796817ce626d84ccb5485453e4d558cf3353e4d4a7e59031 (decoded) |
  - OP_DUP OP_HASH160 7003cc5915f6c23fd512b38daeeecfddfe7a587e9 OP_EQUALVERIFY OP_CHECKSIG |

| Transaction |
Evaluation

• Summary of ProvChain’s capabilities
  – Real-time auditing
  – Unalterable fingerprint of file operations
  – Secure and permanent record keeping and tamper-proof timestamp
  – Decentralized and trustless architecture
  – Privacy-preserving and anonymity
• Performance and Overhead
  – For provenance collection, we use Apache Jmeter to assess the performance of the provenance enabled ownCloud application.

  – Cloud server is set up in Ubuntu 14.04.4LTS, with the Intel(R) Xeon(R) CPU and ownCloud 9.1.

  – We perform file create with random file names and file contents for 500 repetitions in Jmeter. The file size ranges from 1KB to 2MB.
Blockchain Development Platform and Applications

- **Ethereum**
  - Generalized blockchain platform
- **Multichain**
  - Permissioned blockchain network
- **Hyperledger Fabric**
  - Open standard for blockchain for business
- **Tierion**
  - Supports integration of applications within blockchain network
- **Guardtime**
  - Industrial scale blockchain service with keyless signature infrastructure and secure one way function
Evaluation

- Average response time
Evaluation

• Number of Transactions per Second

![Graph showing number of transactions per second over time]
Evaluation

• Throughput over Time

(a) Owncloud without ProvChain

(b) Owncloud with ProvChain
Evaluation

• Number of Transactions per Second

![Graph showing the number of transactions per second over time for Owncloud with and without ProvChain.](image)
Evaluation

- Overall transaction time distribution

(a) Owncloud without ProvChain  
(b) Owncloud with ProvChain
Evaluation

• For provenance retrieval, we focus on the efficiency of requesting blockchain receipt for each of the provenance data entry.
• In our experiment, we query 10 records each time with a total size of 1.004KB, which uses an average time of 221ms.
• Overhead for provenance data retrieval

<table>
<thead>
<tr>
<th>Record Type</th>
<th>Size of Data Transferred</th>
<th>Average Time Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Create</td>
<td>1.07KB</td>
<td>0.838s</td>
</tr>
<tr>
<td>File Change</td>
<td>1.06KB</td>
<td>0.676s</td>
</tr>
<tr>
<td>File Delete</td>
<td>1.07KB</td>
<td>0.675s</td>
</tr>
<tr>
<td>File Share</td>
<td>1.07KB</td>
<td>0.790s</td>
</tr>
</tbody>
</table>
Summary

• Presented the design and implementation of ProvChain, a blockchain based data provenance system for cloud auditing, with preserved user privacy and increased availability.

• Using blockchain technology, we make the record with unalterable timestamp and generate blockchain receipt for each of the data records for validation.

• Evaluation results demonstrate that provenance enabled ownCloud imposes a low overhead.


Sachin Shetty, Ph.D.
Associate Professor
Virginia Modeling, Analysis and Simulation Center
Center for Cybersecurity Education and Research
Dept. of Modeling, Simulation and Visualization Engineering
Old Dominion University
1030 University Blvd
Suffolk, VA 23435
Office: 1231 VMASC | Lab: 2108 VMASC
Phone: 757.638.6233
Email: sshetty@odu.edu

Web: https://www.odu.edu/~sshetty