A Distributed Ledger Technology Design using Hyperledger Fabric and a Clinical Trial Use Case

Rick Kuhn, Joshua Roberts, David Ferraiolo and Joanna DeFranco*

US National Institute of Standards and Technology

kuhn@nist.gov, joshua.roberts@nist.gov, david.ferraiolo@nist.gov

*Penn State University  jfd104@psu.edu
Summary of talk

• Blockchain has valuable properties, but conflicts with privacy and exception management – deletion impossible

• Sometimes don’t need blockchain, just some blockchain features

• Data structure called *blockmatrix* provides integrity protection of blockchain, but allows controlled edits for privacy, corrections

• Blockmatrix is a component for distributed database solutions; it is one design option, blockchain is another, choice depends on application needs; implemented in Hyperledger

• Drop-in compatibility for Hyperledger Fabric applications
Blockchain/distributed ledger could use a different approach for many applications


Structure of a Traditional Blockchain

Blockchain has been defined as "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way".
Why is this a problem for applications?

• The permanence/immutability property that makes blockchain technology useful also leads to difficulty supporting privacy requirements

• Privacy rules such as those of European Union General Data Protection Regulation (GDPR) requires that all information related to a particular person can be deleted at that person's request
  • *personal* data, defined as "any information concerning an identified or identifiable natural person" - data for which blockchains are designed
  • "Personal data which have undergone pseudonymisation, which could be attributed to a natural person by the use of additional information should be considered to be information on an identifiable natural person."

• US states adopting similar privacy rules – California, Virginia, Colorado
How well do blockchain properties apply to traditional distributed data management applications?

<table>
<thead>
<tr>
<th>Cryptocurrency</th>
<th>Finance, supply chain, e-commerce, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Partial anonymity</td>
<td>ID required for contracts or government regulation</td>
</tr>
<tr>
<td>2. Public access/transparency</td>
<td>Controlled access</td>
</tr>
<tr>
<td>3. Small transaction size</td>
<td>Range of message sizes up to large documents, images</td>
</tr>
<tr>
<td>4. Immutable records</td>
<td>Changes and deletions, often required by law</td>
</tr>
<tr>
<td>5. Proof of work</td>
<td>Flexible consensus models</td>
</tr>
<tr>
<td>6. Block ordering guarantees</td>
<td>Timestamps often required</td>
</tr>
<tr>
<td>7. Decentralization</td>
<td>Same in many applications</td>
</tr>
<tr>
<td>8. Replication</td>
<td>Same in many applications</td>
</tr>
<tr>
<td>9. Data integrity guarantees</td>
<td>Same in many applications</td>
</tr>
</tbody>
</table>
What’s been tried to solve blockchain/privacy conflict?

• Don’t put personal data on blockchain
  • Pseudo-anonymized data are still considered personal
  • Even if not directly tied to a person – dynamic IP address can be considered personal if it can be indirectly tied
  • Financial transactions are obviously personal data

• Encrypt data and destroy key to delete
  • Data must be secure for decades
  • Advancements in cryptography usually compromise old crypto – e.g., quantum computing puts current public key systems at risk
Many blockchain applications don’t need blockchain, just some blockchain features

Can we try something else?

• **Datablock matrix** – uses two hash values per block instead of a linked chain
  • Java or Go components available as open source
  • Incorporated into Next Gen Access Control – practical demo
  • Hyperledger component implementation nearing completion

• Verified time – high resolution time stamp instead of ordering guarantee
What are blockmatrix constraints and assumptions?

• Hash integrity protection must not be disrupted for blocks not deleted

• Must ensure auditability and accountability – distributed trust

• Designed for permissioned/private distributed ledger systems – such as supply chain, medical records management, electronic funds transfer

• Provide distributed consensus and guaranteed shared view
# Changing data in blockchain vs. datablock matrix

<table>
<thead>
<tr>
<th><strong>Blockchain</strong></th>
<th><strong>Datablock matrix</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Initial data entry -&gt; transaction in a block</td>
<td>• Initial data entry -&gt; transaction in a block</td>
</tr>
<tr>
<td>• Modification -&gt; new transaction keyed to previous</td>
<td>• Modification -&gt; delete/replace transaction by owner</td>
</tr>
<tr>
<td>• Use key to new value, not allow use of previous, obsolete, value</td>
<td>• Use previous key, new value found in block</td>
</tr>
<tr>
<td>• Dependent on proof of work to ensure sequence</td>
<td>• Sequence not needed since only one value exists</td>
</tr>
</tbody>
</table>
Datablock matrix data structure

- A data structure that provides integrity assurance using hash-linked records while also allowing the deletion of records

  - Stores hashes of each row and column
  - => each block within the matrix is protected by two hashes
  - Suggested use for private/permissioned distributed ledger systems

![Figure 1. Block matrix](image-url)
How does this work?

- Suppose we want to delete block 12

- disrupts the hash values of $H_{3,-}$ for row 3 and $H_{-,2}$ and column 2

- blocks of row 3 are included in the hashes for columns 0, 1, 3, and 4

- blocks of column 2 are included in the hashes for rows 0, 1, 2, and 4
Datablock Matrix Population Algorithm

• Algorithm

```java
while (new blocks) { // i, j = row, column indices
    if (i == j) { add null block; i = 0; j++; }
    else if (i < j) { add block(i,j); swap(i,j); }
    else if (i > j) { add block(i,j); j++; swap(i,j); }
}
```

• Basic algorithm is simple, many variations possible
• Implemented as Java code
• Github project

• Block ordering provides desirable properties

Figure 2. Block matrix with numbered cells
Data Structure Properties

• **Balance**: upper half (above diagonal) contains at most one additional cell more than the lower half.

• **Hash sequence length**: number of blocks in a row or column hash proportional to $\sqrt{N}$ for a matrix with $N$ blocks, by the balance property.

• **Number of blocks**: The total number of data blocks in the matrix is $k^2 - k$ for $k$ rows/columns since the diagonal is null.

• **Block dispersal**: No consecutive blocks in same row or column, in sector 0 (below diagonal) or sector 1(above) for $b \text{ mod } 2$ for block $b$
Structure can be extended to multiple dimensions

• Block dispersal for 3 dimensions
• Location in sectors 0..5 according to $b \mod 6$ for block $b$
## Comparison Summary

<table>
<thead>
<tr>
<th>Blockchain</th>
<th>New approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Integrity protection</td>
<td>• Integrity protection</td>
</tr>
<tr>
<td>• Transparency – global</td>
<td>• Transparency – global</td>
</tr>
<tr>
<td>• Permanence, proof of work</td>
<td>• Editable, timestamps</td>
</tr>
</tbody>
</table>
1. Database access down to field level
2. Attribute-based access
3. Non-intrusive overlay on existing DBMS
4. Uses datablock matrix to provide distributed, integrity protected, and revocable access permissions
So what? Why use this data structure?

Again, many blockchain applications don’t need blockchain, just some features

Enlarge the market for blockchain
• Solve the conflict between blockchain and privacy regulations
• Allow for exception management

Replace network communication with local data
• You can obviously do this with conventional database functions
• New data structure adds integrity checks as in blockchain

Our goal is to make this a basic easy-to-use component for distributed database design
Do you need a shared, consistent data store?

YES

Does more than one entity need to contribute data?

NO

Data records, once written, are never updated or deleted?

NO

Sensitive identifiers WILL NOT be written to the data store?

NO

Are the entities with write access having a hard time deciding who should be in control of the data store?

NO

Do you want a tamperproof log of all writes to the data store?

NO

You may have a useful blockchain use case

YES

NIST blockchain decision flowchart

Uses handled by blockmatrix that cannot be done in blockchain

Yes

NO

NO

NO

NO

NO

NO

NO

NO

NO

YES

YES

YES

YES

YES

YES

YES

YES

YES

YES

YES

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO

NO


What about tech transfer?

• Won NIST Technology Maturation Acceleration Program funding – for technology transfer and commercialization

• Integrating with Next Generation Database Access Control

• Patent approved – assures availability of technology

• Hyperledger component nearing completion
Hyperledger blockmatrix implementation

• Hyperledger is widely-used open source project started by IBM, Intel, and SAP

• Hyperledger Fabric - intended for large distributed systems

• Blockmatrix to be dynamic, increasing capacity as more blocks are added

• Designed to use existing API as closely as possible – add blocks in same manner as adding to blockchain
Integration with Hyperledger Fabric

• Minimal code changes

• Changes primarily in blkstorage package, reducing potential for errors and easing future updates and maintenance

• Use of the blockmatrix is configurable at the channel level
  • User can configure to use conventional blockchain or blockmatrix
  • If a deployment uses two channels, one can be a blockchain and the other can be a blockmatrix
Compatible with current Hyperledger applications

Application (e-commerce, supply chain, etc.)

No change

Application (e-commerce, supply chain, etc.)

Hyperledger Fabric

Hyperledger Fabric

blkstorage module (current)

blkstorage module (blockmatrix added)
Hyperledger Integration Summary

• Blockmatrix implemented in Hyperledger Fabric, widely used for DLT functions
• Uses existing API for ease of application coding
• Minimal changes to Hyperledger code
• Potential applications include current uses of Hyperledger Fabric – e.g., supply chain and logistics, e-commerce, digital currency – adding privacy support

2. Patient consents to participation and allows access to orthopedist data only related to this injury and limited access (denies access to previous sensitive diagnoses) at primary physician. Patient also restricts access to 6 months.

3. Researcher is onboarded to primary physician and is able to access data specified by patient for 6 months.
Where are we now?

• Integrated with Next Gen Database Access Control

• Implemented blockmatrix as plug-and-play component in Hyperledger Fabric

• Demonstrate – clinical trials, logistics/supply chain, other - also new European Central Bank report says Hyperledger Fabric fits needs of ‘digital euro’ – can blockmatrix help?
More information:


Project site with links to source code and publications
• [https://csrc.nist.gov/Projects/enhanced-distributed-ledger-technology](https://csrc.nist.gov/Projects/enhanced-distributed-ledger-technology)

Acknowledgements

▪ Jeff Voas, Dylan Yaga, NIST
▪ Temur Saidkhodjaev, University of Maryland College Park
▪ Arsen Klyuev, Johns Hopkins University
▪ Gokhan Kocak, Asena, Inc.
EXTRA
Key points – blockchain properties

• Blockchain was designed to solve the problem of double-spending in digital currency

• Blockchain’s desirable properties have made it attractive for distributed system applications other than cryptocurrency

• But many of its features make it very unattractive for distributed applications

• Consequently much current research in blockchain is devoted to getting around its built-in properties

• We can provide integrity guarantees and sequencing like blockchain but with low resource consumption and allow revising blocks
What is the rationale for blockchain properties?

• Blockchain and proof-of-work protocol were designed to solve the problem of double spending in cryptocurrencies.

• As with all design choices, blockchain properties have tradeoffs

  • Proof of work provides an ordering guarantee, => at the expense of enormous processing time and expense

  • Linked hash records provide trust and integrity guarantee, => at the expense of losing modification or erasure mechanisms required for privacy
Why is deletion a problem for blockchains?

- Because it is supposed to be – change to one block changes hashes of all; provides integrity protection
- Hashes provide assurance that information in every other block is unchanged if one block is modified
- If we have to delete a block, hash values for others are no longer valid; requires entire new chain
- Don’t want to create a new chain
message BlockchainInfo {
  uint64 height = 1;
  bytes currentBlockHash = 2;
  bytes previousBlockHash = 3;
}

message BlockMatrixInfo {
  uint64 size = 1;
  uint64 blockCount = 2;
  repeated bytes rowHashes = 3;
  repeated bytes columnHashes = 4;
}
Hyperledger
Example

<table>
<thead>
<tr>
<th>Block Header</th>
<th></th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number: 1</td>
<td>Hash: hash(1)</td>
<td></td>
</tr>
<tr>
<td>Block Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• key1=value1</td>
<td>• key2=value2</td>
<td>• key3=value3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>x</th>
<th>5</th>
</tr>
</thead>
</table>

| 4 | 6 | x |
Hyperledger Example

<table>
<thead>
<tr>
<th>Block Header</th>
<th>Block Data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number: 1</td>
<td>key1=value1</td>
<td>3</td>
</tr>
<tr>
<td>Hash: hash(1)</td>
<td>key2=value2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>key3=value3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block Header</th>
<th>Block Data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number: 2</td>
<td>key1=newValue</td>
<td>5</td>
</tr>
<tr>
<td>Hash: hash(2)</td>
<td>key4=value4</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>31</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Block 38**

```
{
    "namespace": "dsnoc",
    "rwset": {
        "reads": [],
        "range_queries_info": [],
        "writes": {
            "key": "test",
            "is_delete": false,
            "value": "hello world"
        },
        "metadata_write": []
    },
    "collection_hased_RWset": []
}
```
Block 38
{
    "namespace": "dunce",
    "rsset": {
        "reads": [],
        "range_query_info": [],
        "writes": [],
        "metadata_writes": []
    },
    "collection_hashed_rsset": []
}
Hyperledger Example

<table>
<thead>
<tr>
<th>Block Header</th>
<th>Block Data</th>
<th>Block Header</th>
<th>Block Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number: 1</td>
<td>key1=value1</td>
<td>Number: 2</td>
<td>key1=newValue</td>
</tr>
<tr>
<td>Hash: hash(1)</td>
<td>key2=value2</td>
<td>Hash: hash(2)</td>
<td>key4=value4</td>
</tr>
<tr>
<td></td>
<td>key3=value3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block Data</td>
<td>key1=nil</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>key5=value5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

| 4            | 6          | x            |
Hyperledger Example

<table>
<thead>
<tr>
<th>Block Header</th>
<th>Block Data</th>
<th>Block Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number: 1</td>
<td>key1=value1</td>
<td>key1=nil</td>
</tr>
<tr>
<td>Hash: hash(1)</td>
<td>key2=value2</td>
<td>key5=value5</td>
</tr>
<tr>
<td></td>
<td>key3=value3</td>
<td></td>
</tr>
<tr>
<td>Number: 2</td>
<td>key1=newValue</td>
<td></td>
</tr>
<tr>
<td>Hash: hash(2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>5</td>
</tr>
</tbody>
</table>

| 4 | 6 | x |