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 <u>design option</u>, blockchain is another, <u>choice depends on application needs;</u> implemented in Hyperledger

- Drop-in compatibility for Hyperledger Fabric applications

Blockchain/distributed ledger could use a different approach for many applications

Kuhn, R., Yaga, D., & Voas, J. (2019). Rethinking distributed ledger technology. *IEEE Computer*, *52*(2), 68-72.

Kuhn, R. (2018). *A Data Structure for Integrity Protection with Erasure Capability*. National Institute of Standards and Technology.

Stavrou, A., & Voas, J. (2017). Verified time. *IEEE Computer*, *50*(3), 78-82.

CYBERTRUST

Rethinking Distributed Ledger Technology

Rick Kuhn and Dylan Yaga, National Institute of Standards and Technology
Jeffrey Voas, IEEE Fellow

Distributed ledger technology (DLT) offers new and unique advantages for information systems, but some of its features are not a good fit for many applications. We review the properties of DLT and show how two recently developed ideas can be used to retain its advantages while simplifying design.

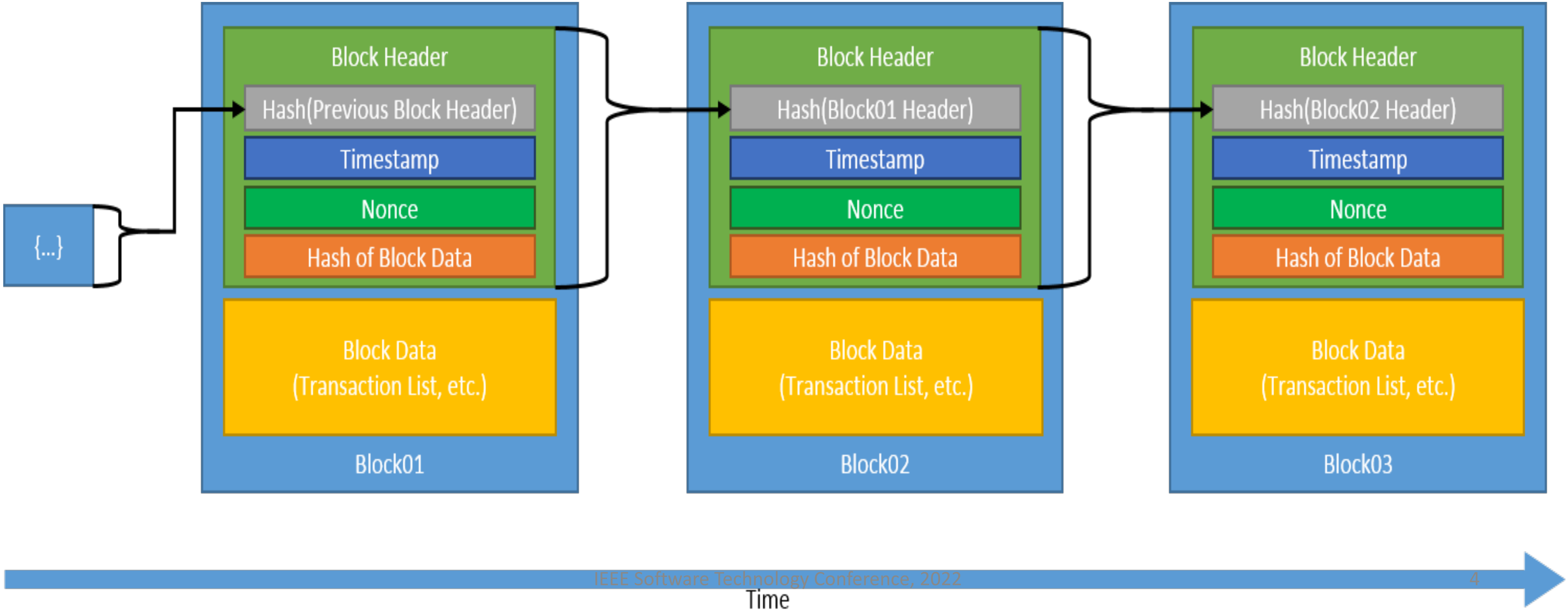
While most of the excitement around blockchain stems from its use in cryptocurrencies, designers are beginning to find interesting ways to solve system problems using it and other forms of DLT. The most commonly used data structure for distributed ledgers is the blockchain. A key feature of a blockchain-based system is the decentralized, replicated data synchronized among separate network nodes, which may be geographically dispersed. There is substantial discussion around some terms in DLT, *public* versus *private* in particular. We believe it is better to distinguish blockchain systems based on their permission model—permissioned or permissionless—because that is directly tied to the technology, whereas private or public may apply to the visibility of the network or ledger itself.

With its features providing distributed, trusted data using no central server, DLT seems to be a natural tool for many complex distributed systems, and a number of implementations have been proposed. However, some environments and applications are not well suited to using an append-only ledger. For example, an analysis of DLT for the international banking consortium the Society for Worldwide Interbank Financial Telecommunication (SWIFT) found that the permissionless model used by Bitcoin and other cryptocurrencies "does not provide the level of trust, transparency, and accountability required by the

Digital Object Identifier 10.1109/MC.2019.2898162
Date of publication: 22 March 2019

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?

Cryptocurrency	Finance, supply chain, e-commerce, etc.
1. Partial anonymity	ID required for contracts or government regulation
2. Public access/transparency	Controlled access
3. Small transaction size	Range of message sizes up to large documents, images
4. Immutable records	Changes and deletions, often required by law
5. Proof of work	Flexible consensus models
6. Block ordering guarantees	Timestamps often required
7. Decentralization	Same in many applications
8. Replication	Same in many applications
9. Data integrity guarantees	Same in many applications

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, <u>just some blockchain features</u>

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

Blockchain

- Initial data entry -> transaction in a block
- Modification -> new transaction keyed to previous
- Use key to new value, not allow use of previous, obsolete, value
- Dependent on proof of work to ensure sequence

Datablock matrix

- Initial data entry -> transaction in a block
- Modification -> delete/replace transaction by owner
- Use previous key, new value found in block
- Sequence not needed since only one value exists

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 <u>private/permissioned </u>distributed ledger systems

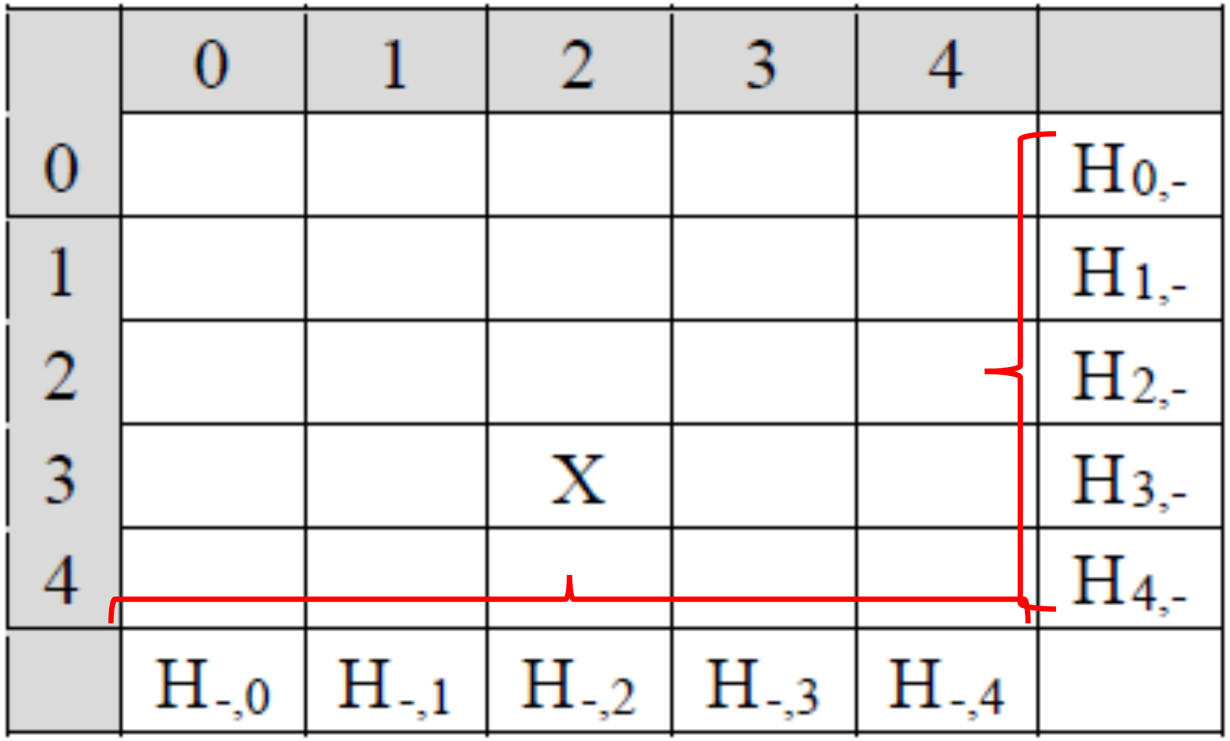


Figure 1. Block matrix

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

	0	1	2	3	4	
0	•	1	3	7	13	$H_{0,-}$
1	2	•	5	9	15	$H_{1,-}$
2	4	6	•	11	17	$H_{2,-}$
3	8	10	12	•	19	$H_{3,-}$
4	14	16	18	20	•	$H_{4,-}$
	$H_{-,0}$	$H_{-,1}$	$H_{-,2}$	$H_{-,3}$	$H_{-,4}$	etc.

Datablock Matrix Population Algorithm

- Algorithm

```
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

	0	1	2	3	4	
0	•	1	3	7	13	$H_{0,-}$
1	2	•	5	9	15	$H_{1,-}$
2	4	6	•	11	17	$H_{2,-}$
3	8	10	12	•	19	$H_{3,-}$
4	14	16	18	20	•	$H_{4,-}$
	$H_{-,0}$	$H_{-,1}$	$H_{-,2}$	$H_{-,3}$	$H_{-,4}$	etc.

Figure 2. Block matrix with numbered cells

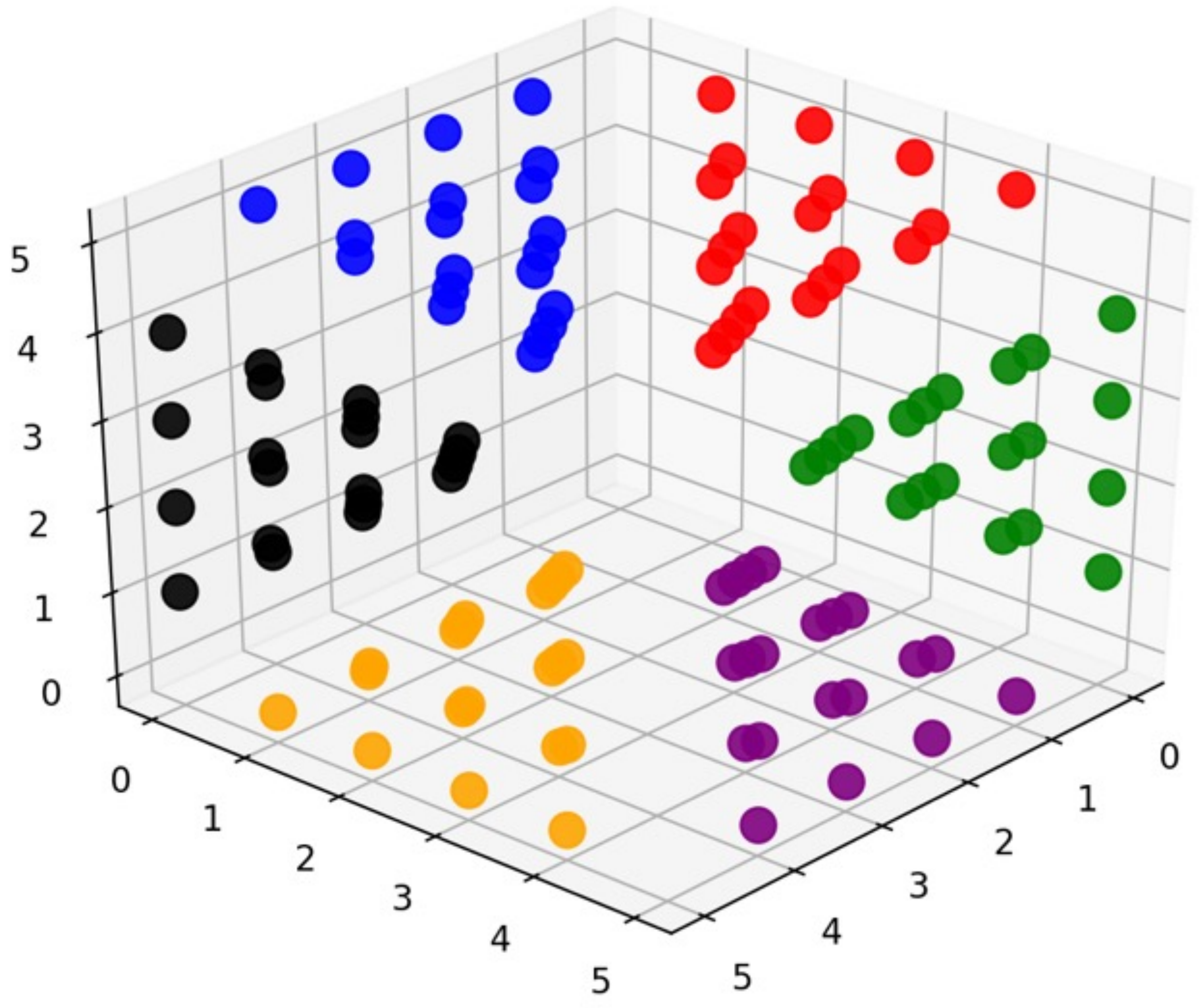
Data Structure Properties

- *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* mod 2 for block *b*

	0	1	2	3	4	
0	•	1	3	7	13	$H_{0,-}$
1	2	•	5	9	15	$H_{1,-}$
2	4	6	•	11	17	$H_{2,-}$
3	8	10	12	•	19	$H_{3,-}$
4	14	16	18	20	•	$H_{4,-}$
	$H_{-,0}$	$H_{-,1}$	$H_{-,2}$	$H_{-,3}$	$H_{-,4}$	etc.

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

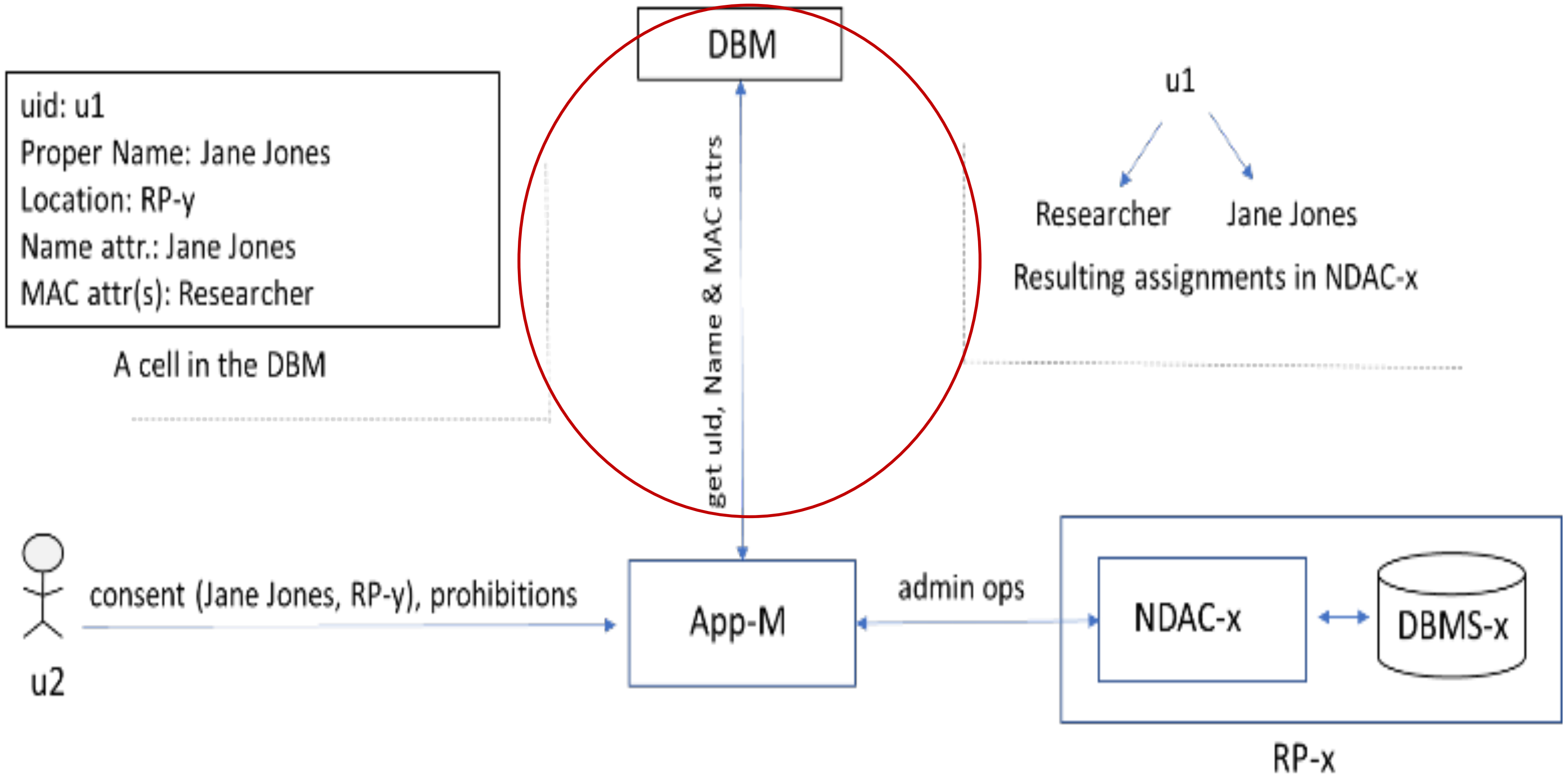
Blockchain

- Integrity protection
- Transparency – global
- Permanence, proof of work

New approach

- Integrity protection
- Transparency – global
- Editable, timestamps

Integrated with Next Gen Database Access Control



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 <u>don't need blockchain</u>, <u>just some features</u>

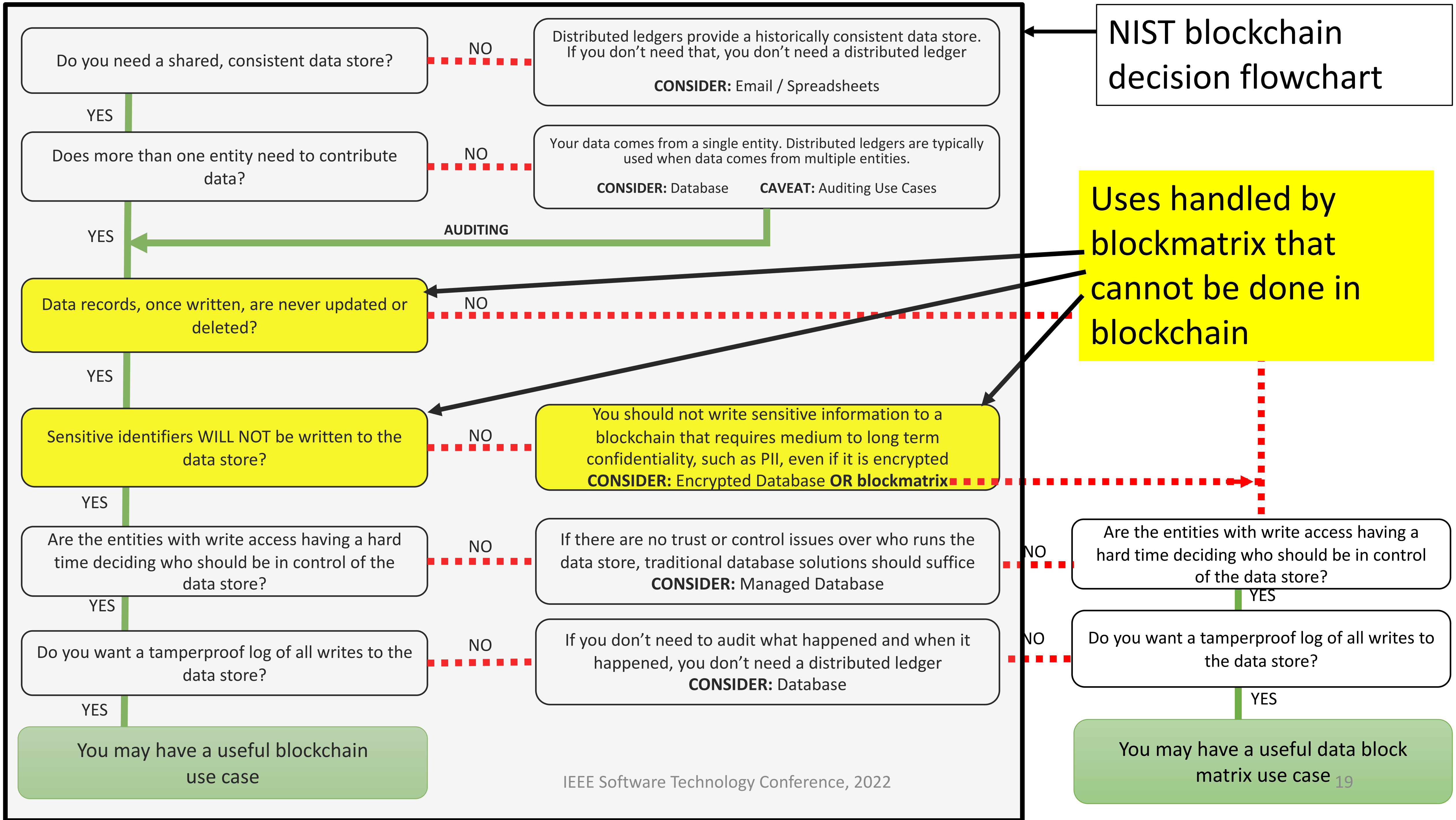
Enlarge the market for blockchain

- 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


NIST blockchain decision flowchart
Do you need a shared, consistent data store?
NO
Distributed ledgers provide a historically consistent data store. If you don't need that, you don't need a distributed ledger
CONSIDER: Email / Spreadsheets
YES
Does more than one entity need to contribute data?
NO
Your data comes from a single entity. Distributed ledgers are typically used when data comes from multiple entities.
CONSIDER: Database
CAVEAT: Auditing Use Cases
AUDITING
YES
Data records, once written, are never updated or deleted?
NO
Uses handled by blockmatrix that cannot be done in blockchain
YES
Sensitive identifiers WILL NOT be written to the data store?
NO
You should not write sensitive information to a blockchain that requires medium to long term confidentiality, such as PII, even if it is encrypted
CONSIDER: Encrypted Database OR blockmatrix
YES
Are the entities with write access having a hard time deciding who should be in control of the data store?
NO
If there are no trust or control issues over who runs the data store, traditional database solutions should suffice
CONSIDER: Managed Database
YES
Do you want a tamperproof log of all writes to the data store?
NO
If you don't need to audit what happened and when it happened, you don't need a distributed ledger
CONSIDER: Database
YES
You may have a useful blockchain use case
NO
Are the entities with write access having a hard time deciding who should be in control of the data store?
YES
NO
Do you want a tamperproof log of all writes to the data store?
YES
You may have a useful data block matrix use case

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
- <u>Hyperledger component nearing completion</u>

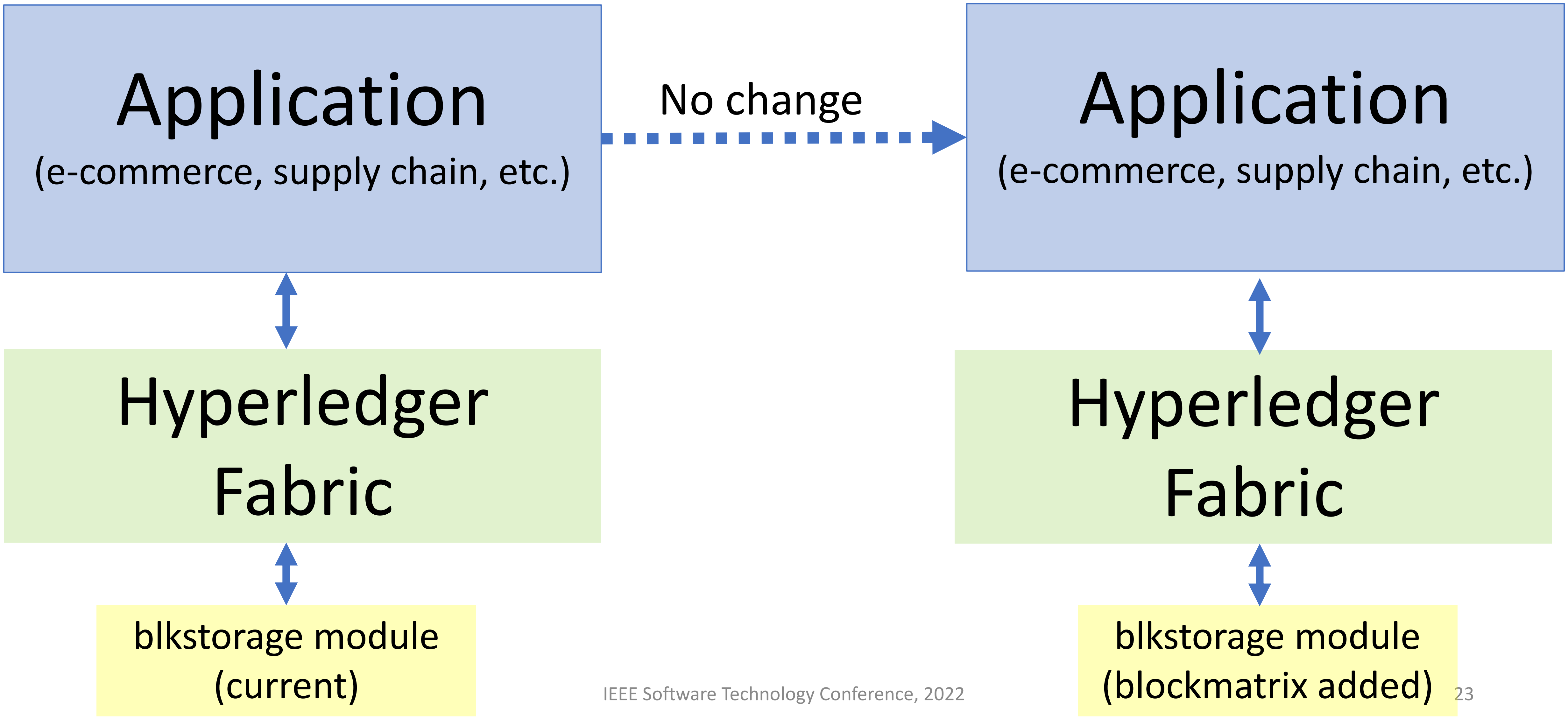
Hyperledger blockmatrix implementation

- 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

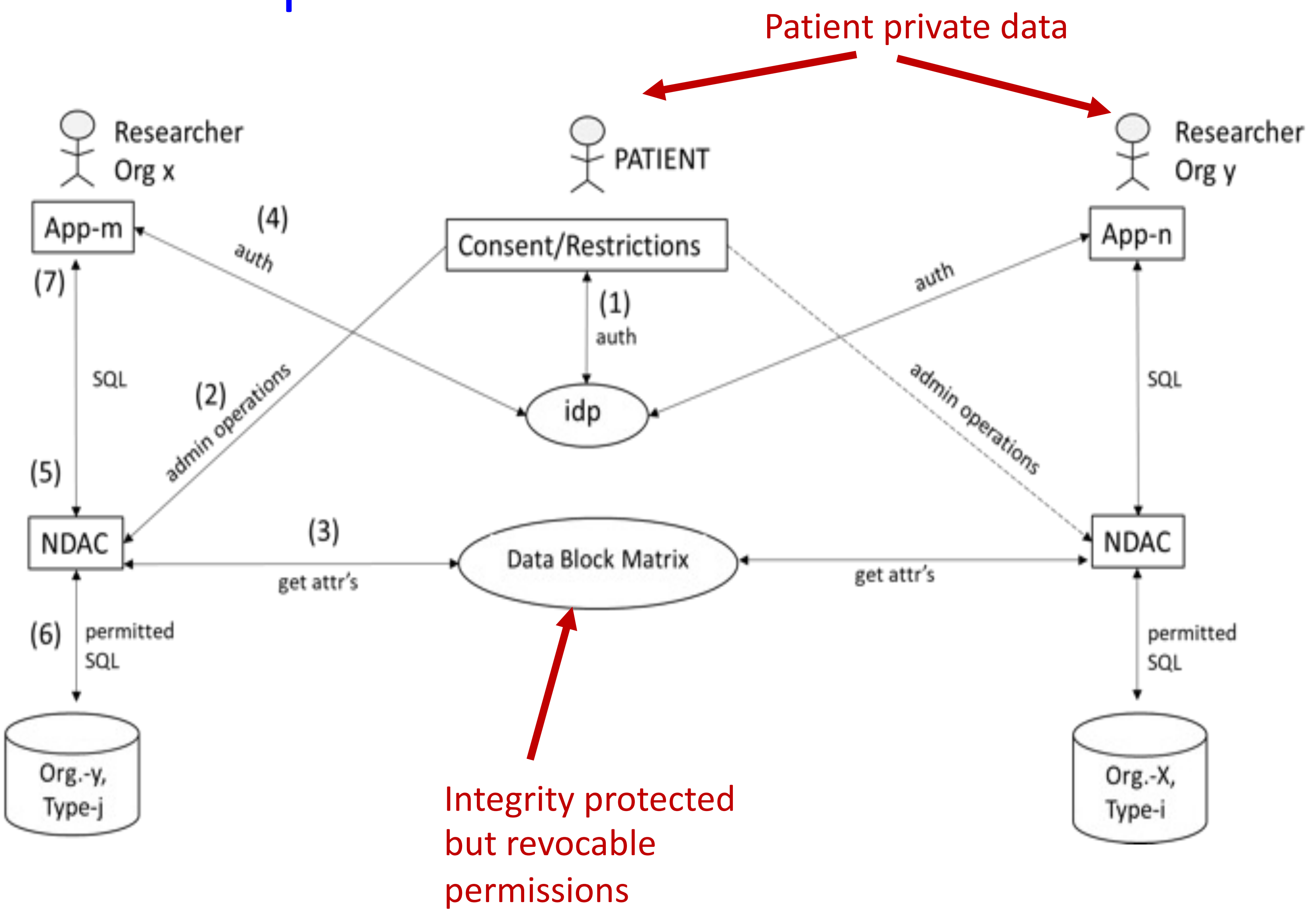
Compatible with current Hyperledger applications



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

Example use case



1. Patient in a hospital. Researcher asks for participation consent. Researcher requests data from patient's physicians.

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:

- Kuhn, R., Yaga, D. and Voas, J., 2019. Rethinking Distributed Ledger Technology. *Computer*, *52*(2), pp.68-72.
- Kuhn, D. R. (2018). A Data Structure for Integrity Protection with Erasure Capability. https://csrc.nist.gov/publications/detail/white-paper/2018/05/31/data-structure-for-integrity-protection-with-erasure-capability/draft

Project site with links to source code and publications

- 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.

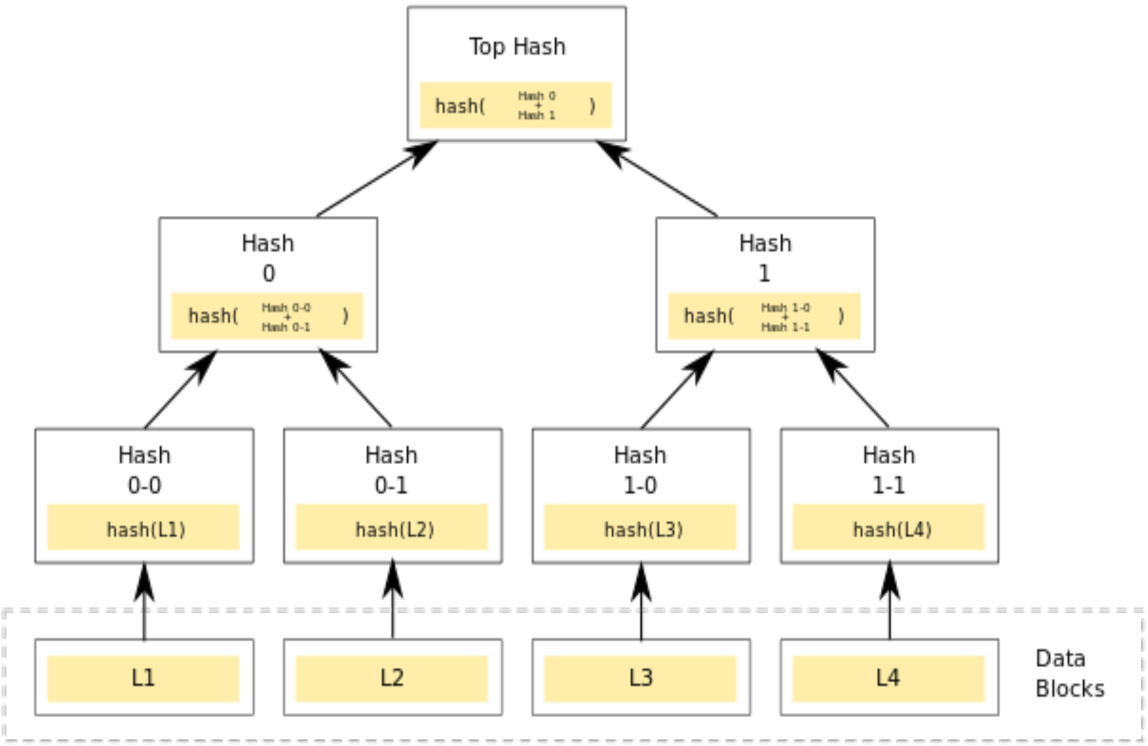
- As with all design choices, blockchain properties have <u>tradeoffs</u>

  - Proof of work provides an <u>ordering guarantee</u>,
    => at the expense of enormous processing time and expense

  - Linked hash records provide <u>trust and integrity guarantee</u>,
    => 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
- <u>Don't want to create a new chain</u>



BlockchainInfo vs BlockMatrixInfo

```
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;
}
```

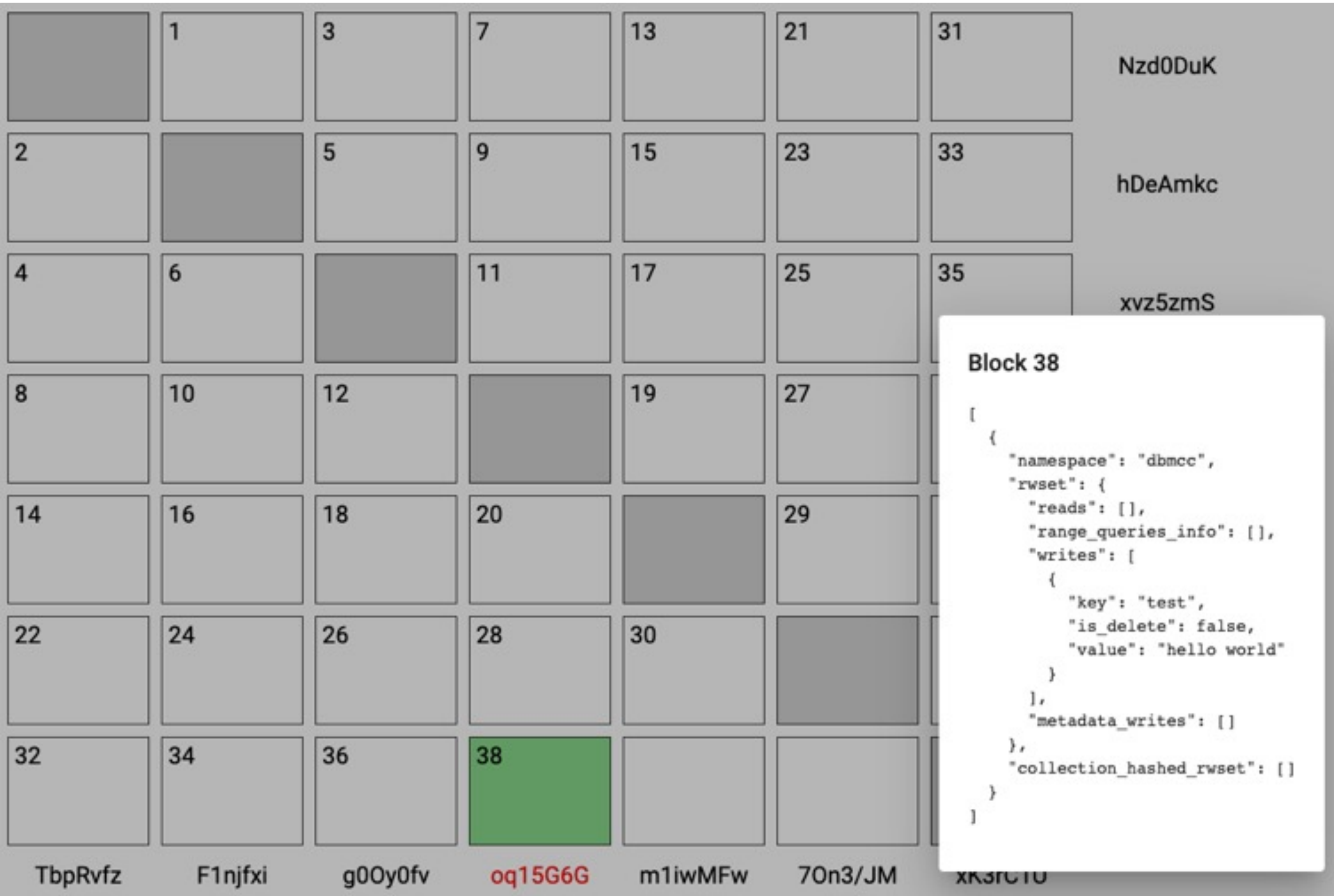
Hyperledger Example

x	**Block Header** Number: 1 Hash: hash(1) **Block Data** • key1=value1 • key2=value2 • key3=value3	3
2	x	5
4	6	x

Hyperledger Example

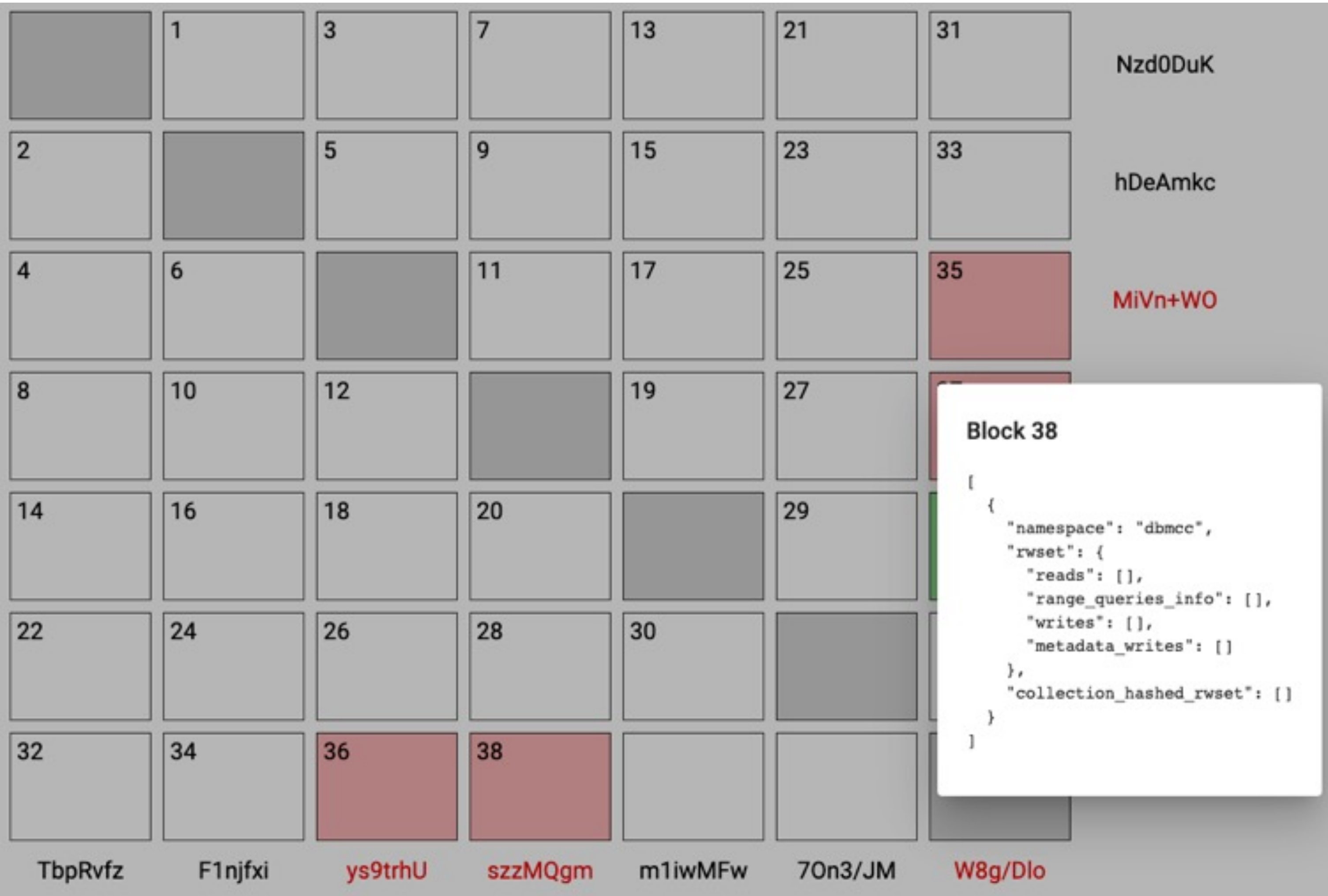
x	**Block Header** Number: 1 Hash: hash(1) **Block Data** • key1=value1 • key2=value2 • key3=value3	3
Block Header Number: 2 Hash: hash(2) **Block Data** • key1=newValue • key4=value4	x	5
4	6	x

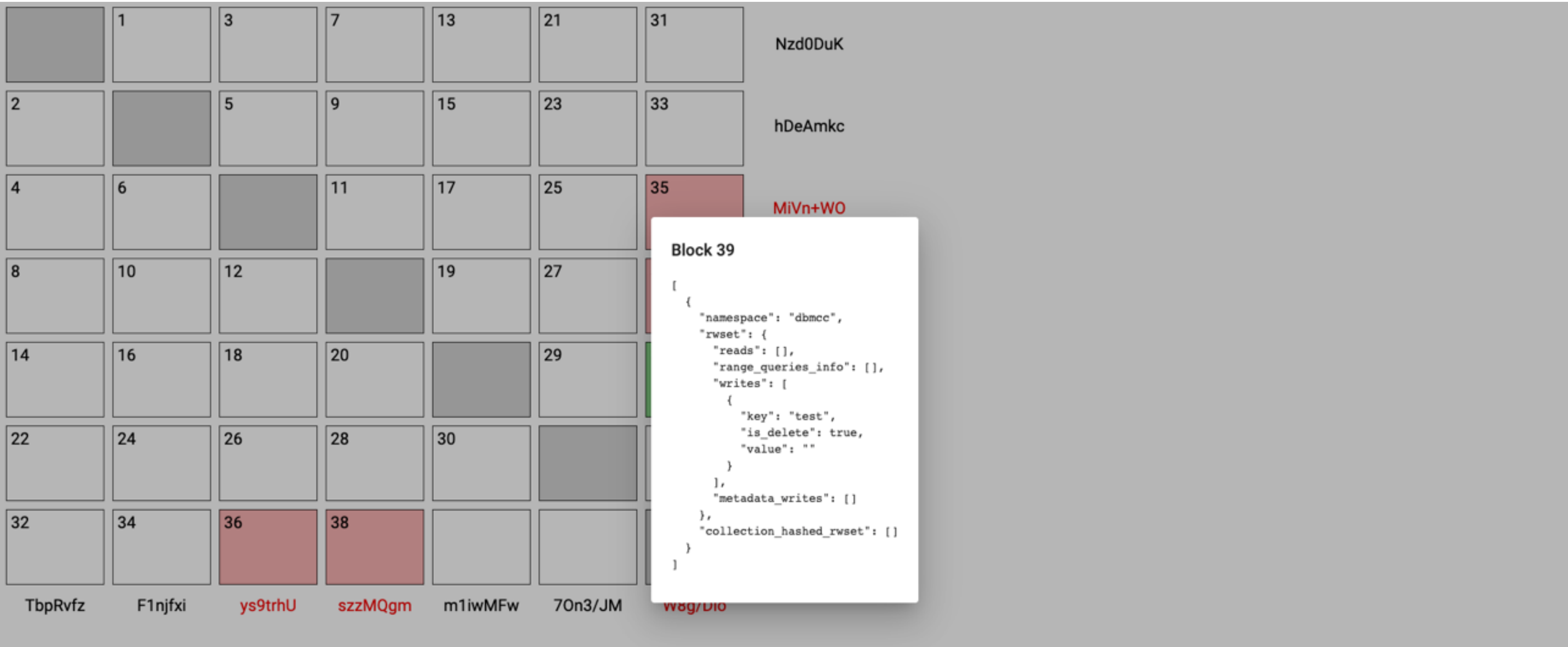
	1	3	7	13	21	31	Nzd0DuK
2		5	9	15	23	33	hDeAmkc
4	6		11	17	25	35	xvz5zmS
8	10	12		19	27	37	FolkVHk
14	16	18	20		29		ZK1Puoo
22	24	26	28	30			Ybdqqho
32	34	36	38				A7Ig2BM
TbpRvfz	F1njfxi	g0Oy0fv	oq15G6G	m1iwMFw	7On3/JM	xK3rC1U	



1
3
7
13
21
31
Nzd0DuK
2
5
9
15
23
33
hDeAmkc
4
6
11
17
25
35
xvz5zmS
8
10
12
19
27
14
16
18
20
29
22
24
26
28
30
32
34
36
38
TbpRvfz
F1njfxi
g0Oy0fv
oq15G6G
m1iwMFw
7On3/JM
Block 38
[
{
"namespace": "dbmcc",
"rwset": {
"reads": [],
"range_queries_info": [],
"writes": [
{
"key": "test",
"is_delete": false,
"value": "hello world"
}
],
"metadata_writes": []
},
"collection_hashed_rwset": []
}
]

	1	3	7	13	21	31	Nzd0DuK
2		5	9	15	23	33	hDeAmkc
4	6		11	17	25	35	MiVn+WO
8	10	12		19	27	37	aMICbp5
14	16	18	20		29	39	QFI84Ea
22	24	26	28	30			Ybdqqho
32	34	36	38				7CMzJBi
TbpRvfz	F1njfxi	ys9trhU	szzMQgm	m1iwMFw	7On3/JM	W8g/Dlo	


1
3
7
13
21
31
Nzd0DuK
2
5
9
15
23
33
hDeAmkc
4
6
11
17
25
35
MiVn+WO
8
10
12
19
27
14
16
18
20
29
22
24
26
28
30
32
34
36
38
TbpRvfz
F1njfxi
ys9trhU
szzMQgm
m1iwMFw
7On3/JM
W8g/Dlo
Block 38
[
{
"namespace": "dbmcc",
"rwset": {
"reads": [],
"range_queries_info": [],
"writes": [],
"metadata_writes": []
},
"collection_hashed_rwset": []
}
]


1
3
7
13
21
31
Nzd0DuK
2
5
9
15
23
33
hDeAmkc
4
6
11
17
25
35
MiVn+WO
8
10
12
19
27
14
16
18
20
29
22
24
26
28
30
32
34
36
38
TbpRvfz
F1njfxi
ys9trhU
szzMQgm
m1iwMFw
7On3/JM
Block 39
[
{
"namespace": "dbmcc",
"rwset": {
"reads": [],
"range_queries_info": [],
"writes": [
{
"key": "test",
"is_delete": true,
"value": ""
}
],
"metadata_writes": []
},
"collection_hashed_rwset": []
}
]

Hyperledger Example

x	**Block Header** Number: 1 Hash: hash(1) **Block Data** • key1=value1 • key2=value2 • key3=value3	**Block Header** Number: 3 Hash: hash(3) **Block Data** • key1=nil • key5=value5
Block Header Number: 2 Hash: hash(2) **Block Data** • key1=newValue • key4=value4	x	5
4	6	x

Hyperledger Example

x	**Block Header** Number: 1 Hash: hash(1) **Block Data** • ~~key1=value1~~ • key2=value2 • key3=value3	**Block Header** Number: 3 Hash: hash(3) **Block Data** • key1=nil • key5=value5
Block Header Number: 2 Hash: hash(2) **Block Data** • ~~key1=newValue~~ • key4=value4	x	5
4	6	x