Encrypted Blockchain Databases

Daniel Adkins, Archita Agarwal, Seny Kamara, Tarik Moataz
Blockchain DBs

Decentralized
Tamper-proof
Auditability

low-query latency
query expressiveness

Blockchain DBs
Blockchain DBs
Blockchain DBs
Blockchain DBs

- electronic healthcare records
- financial records
- legal documents
- customer data
Blockchain DBs

CONFIDENTIALITY?

- electronic healthcare records
- financial records
- legal documents
- customer data
Blockchain DBs

CONFIDENTIALITY?

- electronic healthcare records
- financial records
- legal documents
- customer data
In this work …

How to store Encrypted Multi-Maps on BCs
In this work ...

How to store Encrypted Multi-Maps on BCs

a kind of NoSQL Databases, like Key-Value Stores
What are Multimaps?

Set of label/value tuples

<table>
<thead>
<tr>
<th>ℓ</th>
<th>(v_{1}, v_{2})</th>
<th>(v_{1}, v_{2}, v_{3})</th>
<th>(v_{3})</th>
<th>(v_{4}, v_{5})</th>
<th>(v_{2}, v_{5}, v_{7}, v_{8})</th>
</tr>
</thead>
<tbody>
<tr>
<td>ℓ_{1}</td>
<td>(v_{1}, v_{2})</td>
<td>(v_{1}, v_{2}, v_{3})</td>
<td>(v_{3})</td>
<td>(v_{4}, v_{5})</td>
<td>(v_{2}, v_{5}, v_{7}, v_{8})</td>
</tr>
<tr>
<td>ℓ_{2}</td>
<td>(v_{1}, v_{2}, v_{3})</td>
<td>(v_{3})</td>
<td>(v_{4}, v_{5})</td>
<td>(v_{2}, v_{5}, v_{7}, v_{8})</td>
<td></td>
</tr>
<tr>
<td>ℓ_{3}</td>
<td>(v_{3})</td>
<td>(v_{4}, v_{5})</td>
<td>(v_{2}, v_{5}, v_{7}, v_{8})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ℓ_{4}</td>
<td>(v_{4}, v_{5})</td>
<td>(v_{2}, v_{5}, v_{7}, v_{8})</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ℓ_{5}</td>
<td>(v_{2}, v_{5}, v_{7}, v_{8})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Varying lengths
### What are Multimaps?

Multimaps are a set of label/value tuples. Each tuple contains multiple values associated with a label. Here is an example of a Multimap:

<table>
<thead>
<tr>
<th>ℓ</th>
<th>(v, v', v'')</th>
</tr>
</thead>
<tbody>
<tr>
<td>ℓ₁</td>
<td>(v₁, v₂)</td>
</tr>
<tr>
<td>ℓ₂</td>
<td>(v₁, v₂, v₃)</td>
</tr>
<tr>
<td>ℓ₃</td>
<td>(v₃)</td>
</tr>
<tr>
<td>ℓ₄</td>
<td>(v₄, v₅)</td>
</tr>
<tr>
<td>ℓ₅</td>
<td>(v₂, v₅, v₇, v₈)</td>
</tr>
</tbody>
</table>
What are Multimaps?

Set of label/value tuples

<table>
<thead>
<tr>
<th>$\ell_1$</th>
<th>$(v_1, v_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell_2$</td>
<td>$(v_1, v_2, v_3)$</td>
</tr>
<tr>
<td>$\ell_3$</td>
<td>$(v_3)$</td>
</tr>
<tr>
<td>$\ell_4$</td>
<td>$(v_4, v_5)$</td>
</tr>
<tr>
<td>$\ell_5$</td>
<td>$(v_2, v_5, v_7, v_8)$</td>
</tr>
</tbody>
</table>
What are Multimaps?

Set of label/value tuples

<table>
<thead>
<tr>
<th>( \ell )</th>
<th>((v_1, v_2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ell_1 )</td>
<td></td>
</tr>
<tr>
<td>( \ell_2 )</td>
<td>((v_1, v_2, v_3))</td>
</tr>
<tr>
<td>( \ell_3 )</td>
<td>((v_3))</td>
</tr>
<tr>
<td>( \ell_4 )</td>
<td>((v_4, v_5))</td>
</tr>
<tr>
<td>( \ell_5 )</td>
<td>((v_2, v_5, v_7, v_8))</td>
</tr>
</tbody>
</table>
What are Multimaps?

Set of label/value tuples

| $\ell_1$ | $(v_1, v_2)$ |
| $\ell_2$ | $(v_1, v_2, v_3)$ |
| $\ell_3$ | $(v_3)$ |
| $\ell_4$ | $(v_4, v_5)$ |
| $\ell_5$ | $(v_2, v_5, v_7, v_8)$ |

$t \leftarrow \text{Query}(\ell)$

$\text{EMM'} \leftarrow \text{Add}(\ell, t')$
What are Multimaps?

Set of label/value tuples

<table>
<thead>
<tr>
<th>( \ell )</th>
<th>(( v_1, v_2 ))</th>
<th>(( v_1, v_2, v_3 ))</th>
<th>(( v_3, v_4 ))</th>
<th>(( v_4, v_5 ))</th>
<th>(( v_2, v_5, v_7, v_8 ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ell_1 )</td>
<td>( (v_1, v_2) )</td>
<td>( (v_1, v_2, v_3) )</td>
<td>( (v_3, v_4) )</td>
<td>( (v_4, v_5) )</td>
<td>( (v_2, v_5, v_7, v_8) )</td>
</tr>
</tbody>
</table>

\( t \leftarrow \text{Query}(\ell) \)

\( \text{EMM'} \leftarrow \text{Add}(\ell, t') \)
What are Multimaps?

Set of label/value tuples

<table>
<thead>
<tr>
<th>( \ell )</th>
<th>(( v_1, v_2 ))</th>
<th>(( v_1, v_2, v_3 ))</th>
<th>(( v_3, v_4 ))</th>
<th>(( v_4, v_5 ))</th>
<th>(( v_2, v_5, v_7, v_8 ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ell_1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \ell_2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \ell_3 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \ell_4 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \ell_5 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In our work

LSX  List-Based
TRX  Tree-Based
PAX  Patch-Based
In our work

LSX  List-Based
TRX  Tree-Based
PAX  Patch-Based
In our work

LSX
List-Based

TRX
Tree-Based

PAX
Patch-Based
In our work

LSX
List-Based

TRX
Tree-Based

PAX
Patch-Based
In our work

LSX  List-Based

TRX  Tree-Based

PAX  Patch-Based
Storing data structures on BCs
Storing data structures on BCs
Storing data structures on BCs
Storing linked-list on BCs

$v_1 \leftarrow v_2 \leftarrow v_3 \leftarrow v_4$
Storing linked-list on BCs
Storing linked-list on BCs

$v_1 \circ r_1 \rightarrow v_2 \circ r_1 \leftarrow v_3 \leftarrow v_4$

head

$\rightarrow$
Storing linked-list on BCs
Storing linked-list on BCs
Storing linked-list on BCs
Storing linked-list on BCs

$v_1 \circ \perp$

$r_1$

$v_2 \circ r_1$

$r_2$

$v_3 \circ r_2$

$r_3$

head
Storing linked-list on BCs

$v_1 \leftarrow v_2 \leftarrow v_3 \leftarrow v_4$

…

$r_1 \leftarrow v_1 \cdot \perp \leftarrow \ldots \leftarrow v_4 \cdot r_3 \leftarrow \ldots \leftarrow r_4$
Storing linked-list on BCs

\[ v_1 \circ r_1 \]  
\[ v_2 \circ r_1 \]  
\[ v_3 \circ r_2 \]  
\[ v_4 \circ r_3 \]
Storing linked-list on BCs
Stabilization Complexity  
(aka number of write rounds)

number of values in linked-list

$\nu_2$ can’t be stored until address of $\nu_2$ is known

$\nu_3$ can’t be stored until address of $\nu_3$ is known

$\nu_4$ can’t be stored until address of $\nu_4$ is known

head

$r_1 \nu_1 \perp$

$r_2 \nu_2 \circ r_1$

$r_3 \nu_3 \circ r_2$

$r_4 \nu_4 \circ r_3$
Storing data structures on BCs
\[ v_6 \circ \bot \circ \bot \]

\[ r_6 \]

\[ v_5 \circ \bot \circ \bot \]

\[ r_5 \]

\[ v_4 \circ \bot \circ \bot \]

\[ r_4 \]
\( v_2 \) just dependent on \( v_6 \)
Just dependent on \( v_2 \) and \( v_6 \).
Written in parallel
Written in parallel

\[ v_6 \circ \perp \circ \perp \circ \perp r_6 \]
\[ v_5 \circ \perp \circ \perp \circ \perp r_5 \]
\[ v_4 \circ \perp \circ \perp \circ \perp r_4 \]
\[ v_3 \circ \perp \circ \perp \circ \perp r_3 \]
\[ v_2 \circ \perp \circ \perp \circ \perp r_2 \]
\[ v_1 \circ \perp \circ \perp \circ \perp r_1 \]

root

just dependent on \( v_2 \) \( v_6 \)
Stabilization Complexity

logarithmic in number of values

Written in parallel
• What are Multi-Maps (MMs)
• What are Encrypted Multi-Maps (EMMs)
• How to store EMMs on Blockchains
  • LSX
  • TRX
  • PAX
• Real World Deployment
LSX

A list-based EMM
\[ \ell_1 \quad (v_1, v_2) \]

\[ \ell_2 \quad (v_1, v_2, v_3) \]
• Represent each label as separate list on BC

<table>
<thead>
<tr>
<th>$\ell_1$</th>
<th>$(v_1, v_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell_2$</td>
<td>$(v_1, v_2, v_3)$</td>
</tr>
</tbody>
</table>
• Represent each label as separate list on BC

\[
\begin{array}{c|c}
\ell_1 & (v_1, v_2) \\
\ell_2 & (v_1, v_2, v_3) \\
\end{array}
\]
• Represent each label as separate list on BC
• Encrypt each value before storing on BC
• Represent each label as separate list on BC
• Encrypt each value before storing on BC
Add($\ell_2, v_4$)
- Append $\text{Enc}(v_4, r_3^2)$ at head of $\ell_2$
- Update the head

Delete($\ell_1, v_1$)

Query($\ell_1$)
Add($\ell_2, v_4$)

- Append $\text{Enc}(v_4, r_3^2)$ at head of $\ell_2$
- Update the head

Delete($\ell_1, v_1$)

Query($\ell_1$)

• $v_1 \cdot \perp$
• $v_2 \cdot r_1^2$
• $v_3 \cdot r_2^2$
• $v_4 \cdot r_3^2$
Add($\ell_2, v_4$)

- Append $\text{Enc}(v_4, r_3^2)$ at head of $\ell_2$
- Update the head

Delete($\ell_1, v_1$)

Query($\ell_1$)
Add($\ell_2, v_4$)

- Append Enc($v_4, r_3^2$) at head of $\ell_2$
- Update the head

Delete($\ell_1, v_1$)

Query($\ell_1$)

• Append Enc($v_4, r_3^2$) at head of $\ell_2$
• Update the head
Add(ℓ₂, v₄)

- Append Enc(v₄, r₃²) at head of ℓ₂
- Update the head

Delete(ℓ₁, v₁)

- Same as Add
- Append a DEL flag to the value

Query(ℓ₁)

- Append Enc(v₄, r₃²) at head of ℓ₂
- Update the head

- Append a DEL flag to the value

Add($\ell_2, v_4$)

- Append Enc($v_4, r^2_3$) at head of $\ell_2$
- Update the head

Delete($\ell_1, v_1$)

- Same as Add
- Append a DEL flag to the value

Query($\ell_1$)

- Append Enc($v_1, r^1_2$) at head of $\ell_1$
- Update the head

- Same as Add
- Append a DEL flag to the value
Add($\ell_2, v_4$)

- Append $\text{Enc}(v_4, r_3^2)$ at head of $\ell_2$
- Update the head

Delete($\ell_1, v_1$)

- Same as Add
- Append a DEL flag to the value

Query($\ell_1$)

Add($\ell_2, v_4$)
- Append Enc($v_4, r_3^2$) at head of $\ell_2$
- Update the head

Delete($\ell_1, v_1$)
- Same as Add
- Append a DEL flag to the value

Query($\ell_1$)
Add($\ell_2, v_4$)
- Append $\text{Enc}(v_4, r_2^3)$ at head of $\ell_2$
- Update the head

Delete($\ell_1, v_1$)
- Same as Add
- Append a DEL flag to the value

Query($\ell_1$)
- Read all values from the list
- Output un-deleted values
Add($\ell_2, v_4$)

- Append $\text{Enc}(v_4, r_3^2)$ at head of $\ell_2$
- Update the head

Optimal: $O(|t|)$

delete($\ell_1, v_1$)

- Same as Add
- Append a DEL flag to the value

Optimal: $O(|t|)$

query($\ell_1$)

- Read all values from the list
- Output un-deleted values

Optimal: $O(|t|)$
Add(\(\ell_2, v_4\))
- Append Enc(\(v_4, r_3^2\)) at head of \(\ell_2\)
- Update the head

Optimal: \(O(|t|)\)

Delete(\(\ell_1, v_1\))
- Same as Add
- Append a DEL flag to the value

Optimal: \(O(|t|)\)

Query(\(\ell_1\))
- Read all values from the list
- Output un-deleted values

# values ever added/ deleted
**Add($\ell_2, v_4$)**
- Append $\text{Enc}(v_4, r_3^2)$ at head of $\ell_2$
- Update the head

**Delete($\ell_1, v_1$)**
- Same as Add
- Append a DEL flag to the value

**Query($\ell_1$)**
- Read all values from the list
- Output un-deleted values

# values ever added/deleted

**Optimal:** $O(|t|)$

---

**Add($\ell_2, v_4$)**
- Append $v_4 \circ r_3^2$

**Delete($\ell_1, v_1$)**
- Same as Add
- Append a DEL flag to the value

**Query($\ell_1$)**
- Read all values from the list
- Output un-deleted values

**Optimal:** $O(|t|)$

---

**Add($\ell_2, v_4$)**
- Append $v_4 \circ r_3^2$

**Delete($\ell_1, v_1$)**
- Same as Add
- Append a DEL flag to the value

**Query($\ell_1$)**
- Read all values from the list
- Output un-deleted values

**Optimal:** $O(|t|)$

---

**Add($\ell_2, v_4$)**
- Append $v_4 \circ r_3^2$

**Delete($\ell_1, v_1$)**
- Same as Add
- Append a DEL flag to the value

**Query($\ell_1$)**
- Read all values from the list
- Output un-deleted values

**Optimal:** $O(|t|)$
**Add**($\ell_2, v_4$)
- Append $\text{Enc}(v_4, r^2_3)$ at head of $\ell_2$
- Update the head

**Delete**($\ell_1, v_1$)
- Same as Add
- Append a DEL flag to the value

**Query**($\ell_1$)
- Read all values from the list
- Output un-deleted values

# values ever added/deleted

**Optimal:** $O(|t|)$

---

**Diagram:**
- **Add($\ell_2, v_4$):**
  - $v_1 \cdot \perp$
  - $r^2_1$
- **Delete($\ell_1, v_1$):**
  - $v_1 \cdot \perp$
  - $r^1_1$
- **Query($\ell_1$):**
  - $v_4 \cdot r^2_3$
  - $r^3_3$
- **Head $\ell_2$:**
  - $v_2 \cdot r^2_1$
  - $r^2_2$
- **Head $\ell_1$:**
  - $v_1 \cdot r^1_2 \cdot \text{DEL}$
TRX

A tree-based EMM

Optimal: $O(|t|)$

$O(|t|)$

$O(\log |t|)$

$O(\log |t|)$

$O(|t|)$

$O(|t|)$

$O(|t|)$

$O(|t|)$

# values ever added/deleted
TRX

A tree-based EMM

Optimal: $O(|t|)$

$O(\log |t|)$

$O(|t|)$

$O(\log |t|)$

# values ever added/deleted
PAX

A patched EMM

Add($\ell_2, v_4$)
Optimal: $O(|t|)$
$O(|t|)$

Delete($\ell_1, v_1$)
Optimal: $O(|t|)$
$O(|t|)$

Query($\ell_1$)
# values ever added/deleted
Query($\ell_1$)

# values ever added/
deleted

$\mathbf{v}_1 \leftarrow \mathbf{v}_2 \leftarrow \mathbf{v}_3 \leftarrow \mathbf{v}_4 \leftarrow \mathbf{v}_5$

head
Deletion by addition

Query($\ell_1$)

# values ever added/deleted
$\text{Query}(\ell_1)$

# values ever added/
deprecated

$v_1 \leftarrow v_2 \leftarrow v_3 \leftarrow v_4 \leftarrow v_5$

head
Query($\ell_1$)

# values ever added/deleted

Patch

head

$v_1$ ← $v_2$ ← $v_3$ ← $v_4$ ← $v_5$
Query($\ell_1$)

# values ever added/deleted

\[ v_1 \xrightarrow{\text{time}} v_2 \xleftarrow{\text{Patch}} v_3 \xleftarrow{\text{Patch}} v_4 \xleftarrow{\text{Patch}} v_5 \]
Query($\mathcal{E}_1$)

# values ever added/deleted

Patch

$v_1 \leftarrow v_2 \leftarrow v_3 \leftarrow v_4 \leftarrow v_5$

$v_5 \rightarrow v_3 \leftarrow v_3 \rightarrow v_1$

head PL

head
Let’s delete $v_3$
Let’s delete $v_3$
Let’s delete $v_3$
Let’s delete $v_3$
Let's delete $v_3$

1. Two patches from $v_5$

Bad for correctness
Let’s delete $v_3$

Two patches from $v_5$

Bad for correctness
Let’s delete $v_3$

1. Two patches from $v_5$
   - Bad for correctness
2. Size of PL = # values deleted
   - Bad for efficiency

Let's delete $v_3$.
Let’s delete $v_3$

1. Two patches from $v_5$
   - Bad for correctness

2. Size of PL = # values deleted
   - Bad for efficiency

Need to delete these two
Let’s delete $v_3$

1. Two patches from $v_5$
   - Bad for correctness

2. Size of PL = # values deleted
   - Bad for efficiency

Need to delete these two
Let’s delete $v_3$

Two patches from $v_5$

Size of PL = # values deleted

Bad for correctness

Bad for efficiency

Need to delete these two
Let's delete $v_3$

Copy On Write

1. Two patches from $v_3$
   - Bad for correctness
2. Size of PL = # values deleted
   - Bad for efficiency

Need to delete these two
Expensive!
Expensive! Organize patches in a BST
Expensive! Organize patches in a BST
Expensive!  Organize patches in a BST
Expensive!  Organize patches in a BST

Show

# nodes in BST
≤ # values in multi-map
<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th>stabilization</th>
<th>time LSX</th>
<th>stabilization LSX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>$O(</td>
<td>t</td>
<td>)$</td>
<td>$O(</td>
</tr>
<tr>
<td>Delete</td>
<td>$O(</td>
<td>t</td>
<td>)$</td>
<td>$O(</td>
</tr>
<tr>
<td>Query</td>
<td>$O(# \text{ values ever added/deleted})$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>Time</td>
<td>Stabilization</td>
<td>Time LSX</td>
<td>Stabilization LSX</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>---------------</td>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Add</td>
<td>$O(</td>
<td>t</td>
<td>)$</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td></td>
<td></td>
<td>$O(</td>
<td>t</td>
</tr>
<tr>
<td>Query</td>
<td></td>
<td></td>
<td>$O(\text{# values ever added/deleted})$</td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>Time</td>
<td>Stabilization</td>
<td>Time LSX</td>
<td>Stabilization LSX</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>---------------</td>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td>Add</td>
<td>$O(</td>
<td>t</td>
<td>)$</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>$O(</td>
<td>t</td>
<td>\log \text{(size of MM)})$</td>
<td>$O(</td>
</tr>
<tr>
<td>Query</td>
<td></td>
<td>$O(# \text{values ever added/deleted})$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>Time</td>
<td>Stabilization</td>
<td>Time LSX</td>
<td>Stabilization LSX</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>---------------</td>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Add</td>
<td>$O(</td>
<td>t</td>
<td>)$</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>$O(</td>
<td>t</td>
<td>\log \text{size of MM})$</td>
<td></td>
</tr>
<tr>
<td>Query</td>
<td>$O(\text{size of MM})$</td>
<td></td>
<td>$O(# \text{values ever added/deleted})$</td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>Time</td>
<td>Stabilization</td>
<td>Time LSX</td>
<td>Stabilization LSX</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>---------------</td>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Add</td>
<td>$O(</td>
<td>t</td>
<td>)$</td>
<td>$O(</td>
</tr>
<tr>
<td>Delete</td>
<td>$O(</td>
<td>t</td>
<td>\log \text{(size of MM)})$</td>
<td>$O(</td>
</tr>
<tr>
<td>Query</td>
<td>$O(\text{size of MM})$</td>
<td>$O(\text{# values ever added/deleted})$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• What are Multi-Maps (MMs)
• What are Encrypted Multi-Maps (EMMs)
• How to store EMMs on Blockchains
  • LSX
  • TRX
  • PAX
• Real World Deployment
Real World Deployments

Algorand

Ethereum
Real World Deployments

Algorand

Ethereum

Some things to think about ...
What should we use as addresses?
1. What should we use as addresses?

TX hashes

Used for look-ups
What should we use as addresses?

TX hashes

- All writes can be sent in parallel
  - LSX = TRX
  - Stabilization complexity = $O(1)$
What should we use as addresses?

TX hashes

- All writes can be sent in parallel
  - LSX = TRX
  - Stabilization complexity = $O(1)$

- Might require database indexing
- Many BCs do not support this
1. What should we use as addresses?

**TX hashes**

- All writes can be sent in parallel
  - LSX = TRX
  - Stabilization complexity = $O(1)$

**Block numbers + TX hashes**

- Might require database indexing
- Many BCs do not support this

Used for look-ups
What should we use as addresses?

- TX hashes
  - All writes can be sent in parallel
    - LSX = TRX
    - Stabilization complexity = $O(1)$
  - Might require database indexing
  - Many BCs do not support this

- Block numbers + TX hashes
  - BCs support fast lookup by block numbers

Used for look-ups
1. What should we use as addresses?

**TX hashes**
- All writes can be sent in parallel
  - LSX = TRX
  - Stabilization complexity = $O(1)$
- Might require database indexing
- Many BCs do not support this

**Block numbers + TX hashes**
- BCs support fast lookup by block numbers
- Higher bandwidth required

Used for look-ups
What should we use as addresses?

- All writes can be sent in parallel
- LSX = TRX
- Stabilization complexity = $O(1)$
- Might require database indexing
- Many BCs do not support this

Add on Algorand

Used for look-ups

Block numbers + TX hashes

Higher bandwidth required

Add on Algorand

LSX (block)  LSX (tx)
What should we use as addresses?

- All writes can be sent in parallel
- $\text{LSX} = \text{TRX}$
- Stabilization complexity: $O(1)$
- Might require database indexing
- Many BCs do not support this

Add on Algorand

- Used for look-ups
- Block numbers + TX hashes
- BCs support fast lookup by block numbers
- Higher bandwidth required
2 Packing

Saves time & money

\[ v_1 \circ v_2 \circ v_3 \perp \]

Store multiple values in a tx
Packing

PAX doesn’t support packing

\[ v_1 \circ r_1 \]

\[ v_2 \circ r_1 \]

\[ v_3 \circ r_2 \]

\[ v_1 \circ v_2 \circ v_3 \]

Store multiple values in a tx
2) Packing

PAX doesn’t support packing

Query on Algorand

- time PAX: $O(\text{size of MM})$
- time TRX: $O(\# \text{ values ever added/deleted})$

Graph showing performance comparison between PAX, TRX, and others.

- LSX block
- LSX block (ex)
- LSX tx
- PAX block
- PAX block (ex)
- PAX tx
- TRX block
- TRX block (ex)
- TRX tx
2. Packing

PAX doesn’t support packing

Query on Algorand

time PAX $O(\text{size of MM})$

time TRX $O(\text{# values ever added/deleted})$

![Graph showing comparison between PAX and TRX]
2. Packing

PAX doesn’t support packing

Query on Algorand
(after most values have been deleted)

time PAX
$O(\text{size of MM})$

time TRX
$O(\text{# values ever added/deleted})$