Proof concurrency in **Aardvark**

Derek Leung  
MIT CSAIL

Yossi Gilad  
Hebrew University of Jerusalem

Sergey Gorbunov  
University of Waterloo

Leonid Reyzin  
Boston University

Nickolai Zeldovich  
MIT CSAIL

**Aardvark: An Asynchronous Authenticated Dictionary with Applications to Account-based Cryptocurrencies**

Part of work done at Algorand, Inc.
Motivation
Cryptocurrency

<table>
<thead>
<tr>
<th></th>
<th>PKLeo</th>
<th>$100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKDerek</td>
<td></td>
<td>$75</td>
</tr>
<tr>
<td>PKFlorist</td>
<td></td>
<td>$0</td>
</tr>
</tbody>
</table>

Validators

Derek
Leo
Yossi
Motivation
Cryptocurrency

<table>
<thead>
<tr>
<th></th>
<th>PK</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leo</td>
<td>PK_{Leo}</td>
<td>$100</td>
</tr>
<tr>
<td>Derek</td>
<td>PK_{Derek}</td>
<td>$75</td>
</tr>
<tr>
<td>Florist</td>
<td>PK_{Florist}</td>
<td>$0</td>
</tr>
</tbody>
</table>

Validator

Leo
Derek
Yossi
Motivation

Cryptocurrency

 Validators

<table>
<thead>
<tr>
<th>PKLeo</th>
<th>$100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKDerek</td>
<td>$72</td>
</tr>
<tr>
<td>PKFlorist</td>
<td>$3</td>
</tr>
</tbody>
</table>

txn

Derek
Leo
Yossi
Motivation

Large state

| PK_{Leo} | $100 |
| PK_{Derek} | $72 |
| PK_{Florist} | $3 |

... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...
... ... ...

Validators

txn

txn

txn

txn

Derek

Leo

Yossi
Motivation
Offload storage

Validators

Archives
CDN
Motivation
Offload storage

Derek

(PK_{Derek}, $75)
Motivation
Offload storage

Derek (PK_{Derek}, $75)

txn (PK_{Derek}, $75)

Validators

Archives
CDN
Motivation
Authenticated Dictionary

Derek

txn \( (PK_{Derek}, \$75) \)

 Validators

 Archives
CDN (tampering)
Motivation
Authenticated Dictionary

\[ \text{tx} \pi(C, (PK_{\text{Derek}}, \$75)) \]

\[ \pi(C, (PK_{\text{Derek}}, \$75)) \]

Archives
CDN (tampering)
Motivation
Authenticated Dictionary

\[ \text{txn} \pi(C, (PK_{\text{Derek}}, \$75)) \]

\[ \pi(C, (PK_{\text{Derek}}, \$75)) \]

CDN (tampering)
Why:
low end-to-end latency
at high throughput

\[
\pi(C, (PK_{Derek}, $75))
\]

Observations
System

\[
\pi(C, (PK_{Derek}, $75))
\]

\[
\pi(C, (PK_{Derek}, $75))
\]

• Want:
  low end-to-end latency at high throughput
Observations

System

Want:
low end-to-end latency at high throughput

Bandwidth is bottleneck

π(C, (PK_{Derek}, $75))

π(C, (PK_{Derek}, $75))

txn

CDN (tampering)

 Archives

Commitment

C’
Requirements

Authenticated Dictionary

Proof overhead

Compute (vs. signature) | Transmit (bandwidth)
## Requirements

**Authenticated Dictionary**

<table>
<thead>
<tr>
<th></th>
<th>Proof overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compute (vs. signature)</td>
</tr>
<tr>
<td>Hash-based (e.g., Merkle Tree)</td>
<td>Almost free</td>
</tr>
</tbody>
</table>
## Requirements

### Authenticated Dictionary

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Proof overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compute (vs. signature)</td>
</tr>
<tr>
<td>Hash-based (e.g., Merkle Tree)</td>
<td>Almost free</td>
</tr>
<tr>
<td>Pairing-based (e.g., Vector Commitment)</td>
<td>Substantial</td>
</tr>
</tbody>
</table>
## Requirements

**Authenticated Dictionary**

<table>
<thead>
<tr>
<th></th>
<th>Compute (vs. signature)</th>
<th>Transmit (bandwidth)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hash-based</strong></td>
<td><strong>Almost free</strong></td>
<td><strong>10x overhead</strong></td>
</tr>
<tr>
<td>(e.g., Merkle Tree)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Pairing-based</strong></td>
<td><strong>Substantial</strong></td>
<td><strong>~1x overhead</strong></td>
</tr>
<tr>
<td>(e.g., Vector Commitment)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>??</strong></td>
<td><strong>Almost free?</strong></td>
<td><strong>~1x overhead?</strong></td>
</tr>
</tbody>
</table>
## Requirements

### Authenticated Dictionary

<table>
<thead>
<tr>
<th></th>
<th>Compute (vs. signature)</th>
<th>Transmit (bandwidth)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash-based (e.g., Merkle Tree)</td>
<td>Almost free</td>
<td>10x overhead</td>
</tr>
<tr>
<td>Pairing-based (e.g., Vector Commitment)</td>
<td>Substantial</td>
<td>~1x overhead</td>
</tr>
<tr>
<td>Pairing-based + Versioning</td>
<td>Almost Not quite free: Enable parallelism</td>
<td>1—2x overhead</td>
</tr>
</tbody>
</table>


Aardvark contributions

An authenticated dictionary with
• Short proofs + commitments
   Built from vector commitments, via short proofs of nonmembership
Aardvark contributions

An authenticated dictionary with

- Short proofs + commitments
  
  Built from vector commitments, via short proofs of nonmembership

- Low latency at high throughput
  
  Using transactional concurrency control, via versioning state
This talk

An authenticated dictionary with

- Short proofs + commitments
  
  Built from vector commitments, via short proofs of nonmembership

- Low latency at high throughput
  
  Using transactional concurrency control, via versioning state
Background: Vector commitments

\[ V \]  
\[ \longrightarrow \text{Commit} \]  
\[ \rightarrow c \]
Background: Vector commitments

\[ V \xrightarrow{\text{Open}} i \xrightarrow{} v_i \xrightarrow{\text{Commit}} c \]

Verify \((c, i, v_i, \pi) \rightarrow \text{ok/}\perp\)
Background: Vector commitments

$$V$$

- **Open**
  - $$v_i$$, $$\pi$$

- **ProofUpdate**
  - $$\pi'$$

- **Commit**
  - $$c$$

- **CommitUpdate**
  - $$c'$$

Verify $$(c, i, v_i, \pi) \rightarrow \text{ok/\bot}$$
Background: Vector commitments

vs. Merkle Trees: Proof compactness

Commit \((V) \rightarrow c\)

Open \((V, i) \rightarrow v_i, \pi\)

Verify \((c, i, v_i, \pi) \rightarrow \text{ok/\bot}\)

CommitUpdate \((c, (i, v_i, v'_i)) \rightarrow c'\)

ProofUpdate \((\pi, j, (i, v_i, v'_i)) \rightarrow \pi'\)

Only 48B!
Background: Vector commitments

vs. Merkle Trees: Slowdown

30x
Commit \( (V) \rightarrow c \)

>40,000x
Open \( (V, i) \rightarrow v_i, \pi \)

400x
Verify \( (c, i, v_i, \pi) \rightarrow \text{ok/\perp} \)

7x
CommitUpdate \( (c, (i, v_i, v'_i)) \rightarrow c' \)

>60x
ProofUpdate \( (\pi, j, (i, v_i, v'_i)) \rightarrow \pi' \)

\(|V| = 1000\)

Only 48B!
Problem: Concurrent Updates

\[
\pi(C, (k1, v1)) \quad \pi(C, (k2, v2))
\]
Problem: Concurrent Updates

"run txn with \( \pi(C, (k1, v1)) \)"
Problem: Concurrent Updates

Have:
\[\pi(C, (k2, v2))\]

Have:
\[\pi(C, (k1, v1))\]

“run txn with \(\pi(C, (k1, v1))\)”

\(C'\)

CommitUpdate

\(k1: v1'\)

\(k2: v2\)
Updating $\pi$: Quadratic Scaling

Throughput bottleneck

Have: $\pi(C, (k2, v2))$

Need: $\pi(C', (k2, v2))$
Updating $\pi$: Quadratic Scaling

Throughput bottleneck

Options:

- Archive re-runs Open
- Client runs ProofUpdate
- Validator runs ProofUpdate

Have: $\pi(C, (k2, v2))$

Need: $\pi(C', (k2, v2))$
Updating $\pi$: Quadratic Scaling

Throughput bottleneck

Options:

• Archive re-runs open Many round trips
• Client runs ProofUpdate
• Validator runs ProofUpdate

Have: 
$\pi(C, (k2, v2))$

Need: 
$\pi(C', (k2, v2))$

$C'$

$k1: v1'$

$k2: v2'$
Updating $\pi$: Quadratic Scaling

Throughput bottleneck

Options:

- Archive re-runs Open Many round trips
- Client runs ProofUpdate Expensive
- Validator runs ProofUpdate

Have: $\pi(C, (k_2, v_2))$

Need: $\pi(C', (k_2, v_2))$
Updating \( \pi \): Quadratic Scaling

Throughput bottleneck

Options:

- Archive re-runs Open **Many round trips**
- Client runs ProofUpdate **Expensive**
- Validator runs ProofUpdate **Congestion collapse**

Have: \( \pi(C, (k2, v2)) \)

Need: \( \pi(C', (k2, v2)) \)
Stale proofs force serialization: Cache old state for parallelism
Solution: Dictionary versioning
Solution: Dictionary versioning

```
"run txn with π(C, (k1, v1) @1)"

π(C, (k1, v1) @1)
π(C, (k2, v2) @1)
```

CommitUpdate

```
k1: v1'
k2: v2
```
Two transactions, one key?

"run txn with $\pi(C, (k1, v1) \@1)$"
Solution: Dictionary versioning + data caching

“run txn with π(C, (k1, v1) @1)”

π(C, (k1, v1) @1)

π(C, (k1, v1) @1)

k1: v1’ @2

CommitUpdate

C’ @2

C @1

k2: v2

k1: v1’
Solution: Dictionary versioning + data caching

Details

Nontrivial interaction with nonmembership proofs
• How to handle key insertion/deletion?

See paper for more details…
Evaluation

Integrated into implementation of Algorand cryptocurrency
Evaluation

Integrated into implementation of Algorand cryptocurrency
✓ Storage costs: reduced by >800x
✓ Proof size: 100–200B
Evaluation

Integrated into implementation of Algorand cryptocurrency

✓ Storage costs: reduced by >800x

✓ Proof size: 100–200B

• Effect on throughput and latency?
  • Focus: processing at validators
Evaluation: validator slowdown

AWS c5.metal, 100,000 {put, delete} ops / 10 blocks, 1M keys
Evaluation: validator slowdown

AWS c5.metal, 100,000 {put, delete} ops / 10 blocks, 1M keys

Processing overhead / 1MB block:
34–68s
Evaluation: parallelism helps

AWS c5.metal, 100,000 {put, delete} ops / 10 blocks, 1M keys

Processing overhead / 1MB block:

34–68s: 1 core
3.4–6.7s: 32 cores

8–10x speedup
Related work

- Merkle Trees
- EDRAx (Chepurnoy et al.) (+ other VC-based schemes)
  - Aardvark versioning can help manage high compute costs
Related work

- Merkle Trees
- EDRA\textsuperscript{X} \textit{(Chepurnoy et al.)} (+ other VC-based schemes)
  - Aardvark versioning can help manage high compute costs
- Rollups \textit{(Ethereum community)} (+ zk-SNARKs)
  - Verification: Near constant processing and transmission costs
  - Proof creation: >10x slower/txn than VCs; forced batching
Conclusion

An authenticated dictionary with

- Short proofs (100–200B) + commitments (0.1% storage cost)
  
  Built from vector commitments, via short proofs of nonmembership

- Low latency at high throughput
  
  Using transactional concurrency control, via versioning state

dtl@mit.edu  https://github.com/derbear/aardvark-prototype
Backup slides
Transaction detail

```python
a = txn.Get(alice)
b = txn.Get(bob)
assert a >= p
if a-p == 0:
   txn.Delete(alice)
else:
    txn.Put(alice, a-p)
txn.Put(bob, b+p)
```
Transaction restriction: static keys

with Transaction(alice, bob) as txn:
    a = txn.Get(alice)
    b = txn.Get(bob)
    assert a >= p
    if a-p == 0:
        txn.Delete(alice)
    else:
        txn.Put(alice, a-p)
    txn.Put(bob, b+p)
Transaction restriction: static keys

\[
\begin{align*}
\pi & \quad \pi \\
\end{align*}
\]

with Transaction((alice, bob)) as txn:

\[
\begin{align*}
\text{a} &= \text{txn.} \text{Get(a)} \\
\text{b} &= \text{txn.} \text{Get(b)} \\
\text{assert a} & \geq \text{p} \\
\text{if a-p} & = 0: \\
& \quad \text{txn.} \text{Delete(a)} \\
\text{else:} \\
& \quad \text{txn.} \text{Put(a, a-p)} \\
& \quad \text{txn.} \text{Put(b, b+p)}
\end{align*}
\]
Table 1: VC Operation Latency (mean ± SD µs)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Aardvark</th>
<th>Merkle Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>40262 ± 129</td>
<td>1317 ± 4</td>
</tr>
<tr>
<td>Open</td>
<td>40277 ± 444</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>Verify</td>
<td>3707 ± 10</td>
<td>9 ± 0</td>
</tr>
<tr>
<td>CommitUpdate</td>
<td>62 ± 1</td>
<td>9 ± 0</td>
</tr>
<tr>
<td>ProofUpdate</td>
<td>62 ± 1</td>
<td>&lt; 1</td>
</tr>
</tbody>
</table>

|N| = 1000
<table>
<thead>
<tr>
<th>i.e., about</th>
<th>Operation</th>
<th>Aardvark</th>
<th>Merkle Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>40 ms</td>
<td>Commit</td>
<td>$40262 \pm 129$</td>
<td>$1317 \pm 4$</td>
</tr>
<tr>
<td>40 ms</td>
<td>Open</td>
<td>$40277 \pm 444$</td>
<td>$&lt; 1$</td>
</tr>
<tr>
<td>4 ms</td>
<td>Verify</td>
<td>$3707 \pm 10$</td>
<td>$9 \pm 0$</td>
</tr>
<tr>
<td>60 μs</td>
<td>CommitUpdate</td>
<td>$62 \pm 1$</td>
<td>$9 \pm 0$</td>
</tr>
<tr>
<td>60 μs</td>
<td>ProofUpdate</td>
<td>$62 \pm 1$</td>
<td>$&lt; 1$</td>
</tr>
</tbody>
</table>

$|N| = 1000$
<table>
<thead>
<tr>
<th>Operation</th>
<th>Aardvark</th>
<th>Merkle Tree</th>
<th>Edrax w/o SNARK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>40,262 ± 129</td>
<td>1,317 ± 4</td>
<td>—</td>
</tr>
<tr>
<td>Open</td>
<td>40,277 ± 444</td>
<td>&lt; 1</td>
<td>—</td>
</tr>
<tr>
<td>Verify</td>
<td>3,707 ± 10</td>
<td>9 ± 0</td>
<td>3,131 ± 9</td>
</tr>
<tr>
<td>CommitUpdate</td>
<td>62 ± 1</td>
<td>9 ± 0</td>
<td>13 ± 1</td>
</tr>
<tr>
<td>ProofUpdate</td>
<td>62 ± 1</td>
<td>&lt; 1</td>
<td>27 ± 19</td>
</tr>
</tbody>
</table>