Vault: Fast Bootstrapping for the Algorand Cryptocurrency

Derek Leung, Adam Suhl, Yossi Gilad, Nickolai Zeldovich

Algorand / MIT CSAIL
<table>
<thead>
<tr>
<th>Name</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>$10,000</td>
</tr>
<tr>
<td>Derek</td>
<td>$5,000</td>
</tr>
<tr>
<td>Nickolai</td>
<td>$100,000</td>
</tr>
<tr>
<td>Yossi</td>
<td>$40,000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

State 0
Derek → Landlord $1,000
Nickolai → Derek $3
...
...

Block 1

State 0

Adam $10,000
Derek $5,000
Nickolai $100,000
Yossi $40,000
...
...
Block 1

Derek → Landlord $1,000
Nickolai → Derek $3
...
...

State 0

Adam $10,000
Derek $5,000
Nickolai $100,000
Yossi $40,000
...
...

State 1

Adam $10,000
Derek $4,003
Nickolai $99,997
Yossi $40,000
...
...
Blocks

States

\[
\begin{array}{ccc}
0 & 1 & 2 \\
\end{array}
\]

... 

\[
1,000,000
\]
Blocks

States

0

1

2

...
How does a new user join the system?
Bootstrapping

How does a new user join the system?
Bootstrapping

How does a new user join the system?
Bootstrapping

How does a new user join the system?
Bootstrapping

How does a new user join the system?
Bootstrapping

Goal: Securely and efficiently enable a new user to join, given initial state

- Minimize state transmitted
- Minimize proof that state is valid
Contribution

- Design of Vault, a system with secure and efficient bootstrapping
- 3 techniques for reducing sizes of state and proof of state
- 477MB data transfer cost for 500M transactions (Bitcoin: 143GB)
<table>
<thead>
<tr>
<th>Approach</th>
<th>Challenge</th>
<th>Vault’s Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce state transmitted:</td>
<td>Transaction replay</td>
<td>Force transactions to expire</td>
</tr>
<tr>
<td>Garbage collection</td>
<td>attacks</td>
<td></td>
</tr>
</tbody>
</table>

Vault Techniques
## Vault Techniques

<table>
<thead>
<tr>
<th>Approach</th>
<th>Challenge</th>
<th>Vault’s Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce state transmitted: Garbage collection</td>
<td>Transaction replay attacks</td>
<td>Force transactions to expire</td>
</tr>
<tr>
<td>Reduce state transmitted: Shard state</td>
<td>Small shards lose security</td>
<td>Adaptive Merkle Tree sharding</td>
</tr>
</tbody>
</table>

---

**Vault’s Solution**

1. **Force transactions to expire**
2. **Adaptive Merkle Tree sharding**
# Vault Techniques

<table>
<thead>
<tr>
<th>Approach</th>
<th>Challenge</th>
<th>Vault’s Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce state transmitted: Garbage collection</td>
<td>Transaction replay attacks</td>
<td>Force transactions to expire</td>
</tr>
<tr>
<td>Reduce state transmitted: Shard state</td>
<td>Small shards lose security</td>
<td>Adaptive Merkle Tree sharding</td>
</tr>
<tr>
<td>Reduce size of state proof: Compress history</td>
<td>Attacker tampers with history</td>
<td>Succinct certificates</td>
</tr>
</tbody>
</table>
## Vault Techniques

<table>
<thead>
<tr>
<th>Approach</th>
<th>Challenge</th>
<th>Vault’s Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce state transmitted: Garbage collection</td>
<td>Transaction replay attacks</td>
<td>Force transactions to expire</td>
</tr>
<tr>
<td>Reduce state transmitted: Shard state</td>
<td>Small shards lose security</td>
<td>Adaptive Merkle Tree sharding</td>
</tr>
<tr>
<td>Reduce size of state proof: Compress history</td>
<td>Attacker tampers with history</td>
<td>Succinct certificates</td>
</tr>
</tbody>
</table>
Background
Algorand Background

- Permissionless
- Proof-of-stake
- Cryptographic proof that state is correct
Security Model

• Standard cryptographic assumptions
• New user knows state 0 (i.e., the “genesis”)
• $f \leq 20\%$ of stake is malicious
Proving a Block Correct
Proving a Block Correct

S1

sign B2

B2
Proving a Block Correct

S1

sign B2

>T signatures
(67% of users)

B2
Proving a Block Correct

Millions of users

S1

sign B2

B2

>T signatures (67% of users)
Proving a Block Correct

>\( T \) signatures (75% of sample)

Millions of users
Stake-weighted sample
Proving a Block Correct

Millions of users
Stake-weighted sample

S1

> T signatures
(75% of sample)

B4

delay
Base Bootstrapping
Base Bootstrapping

Cert.

Block

State

1 2 3 4
Base Bootstrapping
Base Bootstrapping
Base Bootstrapping

Cert.  Block  State

1  2  3  4  5
Base Bootstrapping
Base Bootstrapping

<table>
<thead>
<tr>
<th>Cert.</th>
<th>0.7MB</th>
<th>0.7MB</th>
<th>0.7MB</th>
<th>0.7MB</th>
<th>0.7MB</th>
<th>0.7MB</th>
<th>0.7MB</th>
<th>0.7MB</th>
<th>0.7MB</th>
<th>0.7MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td>1MB</td>
<td>1MB</td>
<td>1MB</td>
<td>1MB</td>
<td>1MB</td>
<td>1MB</td>
<td>1MB</td>
<td>1MB</td>
<td>1MB</td>
<td>1MB</td>
</tr>
<tr>
<td>State</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>
# Base Bootstrapping

Must download every block and certificate
Vault: Compress History

1. Vault skips blocks

2. Vault shrinks certificates
1. Skipping Blocks
Base Bootstrapping

delay

Cert.  
Block  
State

1  2  3  4  5  6  7  8  9  10
Base Bootstrapping
Skipping Blocks

Cert.  
Block  
State  

\textit{delay} \hspace{1cm} \textit{delay}
Since we skipped blocks, we don’t know what this is
Since we skipped blocks, we don’t know what this is.
Skipping Blocks

Only need headers, certificates, and final state
Skipping Blocks

Certificates are 10x bigger due to Merkle proofs

Only need headers, certificates, and final state
2. Shrinking Certificates
Proving a Block Correct

S1

sign B4

>T signatures (75% of sample)

Millions of users
Stake-weighted sample
Smaller Proofs

Millions of users
Stake-weighted sample

> T signatures (90% of sample)
Smaller Proofs

- Smaller sample: higher variance
- Must raise certificate threshold to compensate (e.g., 75% to 90%)
- May fail to meet threshold!

Stake-weighted sample

Millions of users

S1

\text{sign B4}

\text{B4}

> T signatures (90\% of sample)
During bootstrapping, trade off availability for size
Trading Away Availability

Certificate Size (# stake-weighted signatures)

Availability (%)

74x improvement
Trading Away Availability

Shrink certificates by 74x (tunable)

On failure, use block header hashes to bypass failure

(0.1MB to skip < delay blocks)
Block Hash Vaulting

Cert.  

Header  

1  2  3  4  5
Block Hash Vaulting
Block Hash Vaulting

Cert.

Block headers contain hash of previous block headers
Block Hash Vaulting

Block headers contains hash of previous block headers
Block Hash Vaulting

Block headers contains hash of previous block headers
Block Hash Vaulting

Cert.

0.1MB

1KB

0.1MB

0.1MB

1KB

0.1MB

Header

1

2

3

4

5

6

7

8

9

10

Block headers contains hash of previous block headers
Block Hash Vaulting

Block headers contain hash of previous block headers

Fall back to large certificates (7MB to skip delay blocks)
Evaluation
• Simulate trace of 500 million simple transactions

• Prototype data structures in Bitcoin, Ethereum, Algorand, and Vault
  
  • All 3 Vault techniques: transaction expiration, adaptive sharding, and succinct certificates
  
  • \( delay = 1000, \#shards = 1000 \)

• Measure bootstrapping data transfer cost
Data Transfer Cost

- Algorand
- Bitcoin
- Ethereum
- Vault

New User Data Transfer Cost

- 99.7%
- 90%
Related Work

- Minimize State:
  - Lightning Network (Poon and Dryja)
  - Edrax (Chepurnoy et al.)
  - OmniLedger (Kokoris-Kogias et al.)

- Minimize Proof of State:
  - MimbleWimble (Poelstra)
  - Chainiac (Nikitin et al.)
Conclusion

- Bootstrapping costs prevent scaling
- Vault techniques address these costs
  - Succinct certificates securely compress history, trading off availability
- Reduction in bootstrapping costs to 477MB: lower by 90.5% (Ethereum) or 99.7% (Bitcoin)
Backup slides
Different settings of $f$
Bootstrapping Proof Size

![Graph showing proof size vs. number of transactions for Algorand and Vault.]
Bootstrapping with Randomization
Skipping Blocks with Randomization

```plaintext
1 2 3 4 5 6 7
```

```plaintext
delay
delay
```
Vaulting with Randomization

Block headers contains hash of previous block headers
Vaulting with Randomization

Block headers contains hash of previous block headers
Vaulting with Randomization

Block headers contain hash of previous block headers
Fallback with Algorand

Large Certificate

![Diagram showing a sequence of certificates with one marked as incorrect.](image-url)
Fallback with Algorand

Large Certificate