PONZI SCHEME

How did you make the money?
Commercial Secret!
Research Question

<table>
<thead>
<tr>
<th>Transactions</th>
<th>Portfolio</th>
<th>Public Info.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cash Stock1</td>
<td>Units Total</td>
</tr>
<tr>
<td></td>
<td>Stock2</td>
<td>NAV</td>
</tr>
<tr>
<td>0</td>
<td>$0</td>
<td>$50</td>
</tr>
<tr>
<td>10</td>
<td>$500</td>
<td>$50</td>
</tr>
<tr>
<td>30</td>
<td>$1500</td>
<td>$50</td>
</tr>
<tr>
<td>45</td>
<td>$2250</td>
<td>$50</td>
</tr>
<tr>
<td>45</td>
<td>$2250</td>
<td>$50</td>
</tr>
<tr>
<td>45</td>
<td>$2700</td>
<td>$60</td>
</tr>
<tr>
<td>35</td>
<td>$2100</td>
<td>$60</td>
</tr>
</tbody>
</table>

Audit Transactions But Knowing Nothing?
Our Contributions

ZeroAudit Protocol

- Auditor Maintain Fund Portfolio – Work on Commitments
- Inspect Each Transaction at Fair Market Price
- Certify Net Asset Value Any Time
- No Trusted Setup!
  - Eq. Proof of Commits. over 2 Groups of Different Prime Orders
  - Using Range Proof

Zk-Membership Proof for Merkle Tree

Given $C(r)$ and $C(n)$:
Prove $n \in Tree(r)$

Implementation & Optimization

- 25K GO on Fabric
- Small Test Exponent Test
- Chained Schnorr Groups
- Pippenger’s Multi-Exponent.
- Bullet Range Proof
- …

Zk-Membership Proof for Merkle Tree

Given $C(r)$ and $C(n)$:
Prove $n \in Tree(r)$
ZeroAUDIT Architecture

Buy 10 Units of Fund A for $1000
Trading Operation

Fund A

1. 2 Spend Keys, 2 New Cert Roots
2. Encrypt(Buy 10 Shares of Stock 1)
3. Zk-Proof

ZeroAUDIT

1. Fund A has enough cash (range proof)
2. Spending key is valid
3. Quantities in 1 same as 2

Brokerage
Verify Fair Market Price

- Given $\text{Commit}(\text{SID})$ and $\text{Commit}(\text{price})$
- Prove: $(\text{SID}, \text{price}) \in \text{Tree}$
Chained Schnorr Groups

\[ Commit_p(P) \in G_p \]
\[ Commit_q(L) \in G_q \]
\[ Commit_q(R) \in G_q \]

\[ g_q^L g_q g_q^r \mod p \]
\[ g_q^L g_q^r' \mod p \]

**Pedersen Commitment:**

2 Primes \( q < p \): \( G_q \) subgroup of \( Z_p \)

2\(^{nd} \) Larger Group

\( p < r \): \( G_p \) subgroup of \( Z_r \)

\[ P = \text{hash}(L, R) \]
\[ P = Commit_q(L, R) \]
\[ P = g_q^L g_q^R g_q^r \mod p \]
**DDLP in Child Proof**

\[ \text{Commit}_p(P) \quad P = g^L_{q1} g^R_{q2} g^r_{q3} \mod p \]

\[ \text{Commit}_q(L) \]

\[ \text{Commit}_q(R) \]

\[ P = \text{Commit}_q(L) \text{Commit}_q(R) g^D_{q3} \mod p \text{ for some } D \]

Prover should not release D or \( g^D_{q3} \)

Prover can disclose \( \text{Commit}_p(g^D_{q3}) \)

**Key Observation:**

\[ \text{Commit}_p(g^D_{q3}) \text{Commit}_q(L) \text{Commit}_q(R) \]

Prover knows the D

Double Discrete Logarithm Proof (DDLP) [Stadler’97]
Complete Merkle Tree Proof

- Disjunctive Zk-Proof
- Zk-Equality Proof Over Groups of **Different** Order
  - [Camenisch CRYPT’99] Chain Commit. on **Unknown** Order
  - Our Approach **[No Trusted Setup]**
    - **Range Proof** to Confine Values [Special Chained Schnorr Setting]
    - Challenge: Huge Range \([0, 2^{1800}]\)
    - Incorporate **Inaccurate** Range Proof
System Implementation

- 25K Golang
- Over Hyperledger Fabric

<table>
<thead>
<tr>
<th>Proof</th>
<th>Reps</th>
<th>Build</th>
<th>Verif</th>
<th>Size</th>
<th>SmallExp</th>
<th>LargeExp</th>
</tr>
</thead>
<tbody>
<tr>
<td>πDLP</td>
<td>200</td>
<td>1.08</td>
<td>0.78</td>
<td>0.29</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>πDLP</td>
<td>200</td>
<td>561.98</td>
<td>48.23</td>
<td>76.22</td>
<td>189</td>
<td>1</td>
</tr>
<tr>
<td>πNAME</td>
<td>200</td>
<td>57.43</td>
<td>5.75</td>
<td>2.08</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>πOR</td>
<td>100</td>
<td>3614.53</td>
<td>59.13</td>
<td>494.67</td>
<td>91</td>
<td>2</td>
</tr>
<tr>
<td>πEDGE</td>
<td>100</td>
<td>4241.48</td>
<td>111.20</td>
<td>557.44</td>
<td>259</td>
<td>4</td>
</tr>
<tr>
<td>πTREE</td>
<td>1</td>
<td>42778.61</td>
<td>1597.28</td>
<td>5732.72</td>
<td>3457</td>
<td>87</td>
</tr>
<tr>
<td>πRANGE</td>
<td>100</td>
<td>7995.79</td>
<td>69.65</td>
<td>6.24</td>
<td>21</td>
<td>18</td>
</tr>
<tr>
<td>πREPLACE</td>
<td>1</td>
<td>75163.24</td>
<td>2640.24</td>
<td>103243.36</td>
<td>6370</td>
<td>111</td>
</tr>
<tr>
<td>πORDER</td>
<td>1</td>
<td>216961.74</td>
<td>8382.76</td>
<td>28726.65</td>
<td>18722</td>
<td>238</td>
</tr>
<tr>
<td>πHOLDING</td>
<td>1</td>
<td>116534.11</td>
<td>3811.67</td>
<td>16091.07</td>
<td>8342</td>
<td>151</td>
</tr>
</tbody>
</table>

Build & Verif: Proof construction and verification time in milliseconds. SmallExp: number of small exponentiation (exp < 2^1). LargeExp: large exponentiation. Size: proof size in KB Tree height

settings for πREPLACE, πORDER, and πHOLDING: TπR (11) and Tπ (16).

Table 3: Zero Knowledge Proof Cost

<table>
<thead>
<tr>
<th>Operation</th>
<th>Verif Time (ms)</th>
<th>Chaincode Request (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewFund (per cert)</td>
<td>571</td>
<td>3648</td>
</tr>
<tr>
<td>Order</td>
<td>15443</td>
<td>36742</td>
</tr>
<tr>
<td>Invest</td>
<td>7062</td>
<td>16051</td>
</tr>
<tr>
<td>SetNAV (per cert)</td>
<td>8076</td>
<td>20425</td>
</tr>
</tbody>
</table>

Table 4: Hyperledger Fabric Client Operation Cost
### Performance Data

<table>
<thead>
<tr>
<th></th>
<th>Trusted Setup</th>
<th>Assumptions</th>
<th>Approach</th>
<th>Security Strength</th>
<th>Anonymity Set Size</th>
<th>Proof Size</th>
<th>Proof Time (ms)</th>
<th>Verify Time (ms)</th>
<th>Verify Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZeroCash [21]</td>
<td>Yes</td>
<td>DH, KEP</td>
<td>zkSnark</td>
<td>128-bit</td>
<td>$2^{24}$</td>
<td>0.28KB</td>
<td>175000</td>
<td>8.5</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>zkLedger [35]</td>
<td>No</td>
<td>DH</td>
<td>Columnar Update</td>
<td>128-bit</td>
<td>20</td>
<td>96KB</td>
<td>160</td>
<td>140</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Monero [53]</td>
<td>No</td>
<td>DLP, DH</td>
<td>Ring Signature LogSize Set Member. Prf</td>
<td>128-bit</td>
<td>$2^{10}$</td>
<td>1.3KB</td>
<td>5000</td>
<td>3000</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>ZeroAUDIT</td>
<td>No</td>
<td>DLP, RO</td>
<td>Anony. Merkle Tree Member. Prf</td>
<td>112-bit</td>
<td>$2^{15}$</td>
<td>28MB</td>
<td>216961</td>
<td>8382</td>
<td>$O(\log_2 n)$</td>
</tr>
</tbody>
</table>

**Assumptions.** DLP: Discrete Logarithm; DH: Diffie-Hellman Variants; KEH: Knowledge of Exponent; RO: Random Oracle; SRSA: Strong RSA. **Verify Complexity.** the verification run-time on the anonymity set size $n$.  --- not reported.  $^a$ Estimate is based on 160-bit serial numbers.  $^b$ Data is from Figure 3 (a), (b).  $^c$ Data is from Figure 3.  $^d$ Data is estimated from Table 1. Using k=20.  $^e$ Data is estimated from Figure 2(a)(b) with input size 2.  $^f$ Data is from the cost of $\pi_{ORDER}$ in Table 3.

**Table 5: Performance Comparison**

- **Extra Cost Paid for No Trusted Setup**
- **Proof Size Huge (Needs Improvement)**
Conclusion

- **ZeroAUDIT**
  - Inspect Trading Transactions Without Knowing Details
- **Anonymous Merkle Tree Membership Zk-Proof**
  - No Trusted Setup
  - Simple Assumptions: DLP + Random Oracle
  - Log(n) Proof Size and Verification Time (n: anonymity set size)
  - Equality Proof (on Different Order Groups) Using Range Proof
- **Varieties of Optimization Techniques**
Questions