Dubhe: Succinct Zero-Knowledge Proofs for Standard AES and Related Applications

Changchang Ding and Yan Huang

USENIX Security 2023
Zero-Knowledge Proofs of Knowledge

Allow a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that $\mathcal{P}$ holds secret witnesses $w$ s.t. $C(x, w) = 1$, without revealing $w$

**Succinct:** proof size / verifier time sublinear in $\left\{\begin{array}{l} |C| \quad \text{(Weak Succinct)} \\ |C| + |w| \quad \text{(Strong Succinct)} \end{array}\right.$

**Transparent:** no trusted setup
State-of-the-art Transparent ZKPs

MPC-in-the-Head based protocols:
- KKW [KKW18], Limbo [dOT21], etc.
- Pros: Support arbitrary fields.
- Cons: Not succinct.

Virgo [ZXZS19] / Virgo++ [ZLWZ+21]
- Using GKR [GKR08] and LDT
- Pros: Strong succinctness
- Cons: Constraints on choices of fields

Question:
Is it possible to construct a concretely-efficient succinct ZKPoK that can easily support computations on arbitrary fields?
Dubhe: Summary of Contributions

1. Succinct proof in the number of gates.
2. No restriction on the underlying fields.
3. Applications:
   - Identification / digital signature schemes
   - Ring identification / signature schemes
   - All schemes based on unmodified use of AES
### Observations and Dubhe’s Goals

<table>
<thead>
<tr>
<th></th>
<th>ZK</th>
<th>Fast Verifier</th>
<th>Short Proof</th>
<th>Non-linear gates</th>
<th>Linear gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>KKW</td>
<td>☝️</td>
<td>☹️</td>
<td>☹️</td>
<td>☹️</td>
<td>☝️</td>
</tr>
<tr>
<td>GKR</td>
<td>☹️</td>
<td>☝️</td>
<td>☝️</td>
<td>☹️</td>
<td>☹️</td>
</tr>
<tr>
<td>FLPCP</td>
<td>☝️</td>
<td>☹️</td>
<td>☝️</td>
<td>☝️</td>
<td>N/A</td>
</tr>
<tr>
<td>Dubhe</td>
<td>☝️</td>
<td>☝️</td>
<td>☝️</td>
<td>☝️</td>
<td>☝️</td>
</tr>
</tbody>
</table>
Dubhe’s Approach

$d \cdot m$
Polynomial Gates

$O(d \cdot \log m)$
Linear Gates

$O(d)$
Polynomial Gates

$d \cdot \log m$
Linear Gates

$W_{GKR}$
FLPCP transform

$W_{GKR}$
GKR transform

$O(d \cdot \log m)$
Linear Gates

$O(d)$
Linear Gates
Proof of AES

The only non-linear operation: special inverse in SubBytes

\[
\begin{align*}
    b &= a^{-1}, \quad a \neq 0 \\
    b &= 0, \quad a = 0
\end{align*}
\]

Banquet [BdKO+21] / Limbo’s approach: required non-zero inputs to all SubBytes

With extra witness:

\[a \cdot b = 1 \lor (a = 0 \land b = 0) \iff a(ab + 1) = 0 \land b(ba + 1) = 0\]

Without extra witness:

- Treat SubBytes as 8 table-lookups, each has 256 entries.
- Encode each table as an 8-variate polynomial.
Counter-Mode AES

![Graphs showing various metrics for AES blocks with and without extra-witness.](image-url)
AES based Identification / Signature

Identification (interactive):

- Keygen: \( pk \leftarrow AES_{sk}(ID_u) \)
- Proof of Identity: \( w = sk, \ x = (ID_u, pk) \)
- Circuit: AES with extra witness

Signature (non-interactive through Fiat-Shamir Transform):

- Keygen: \( pk \leftarrow AES_{sk}(ID_u) \)
- Signature: a ZKP of \( w = sk, \ x = (ID_u, pk) \) with \( H(m) \) as randoms.
- Circuit: AES with extra witness, skip GKR to reduce number of rounds.
## AES based Identification (100-bit stat. sec.)

<table>
<thead>
<tr>
<th>Identification</th>
<th>P time (ms)</th>
<th>V time (ms)</th>
<th>Comm. (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuickSilver [YSWW21]</td>
<td>334</td>
<td>334</td>
<td>1644</td>
</tr>
<tr>
<td>Virgo++</td>
<td>751</td>
<td>36</td>
<td>132</td>
</tr>
<tr>
<td>Virgo</td>
<td>2265</td>
<td>21.4</td>
<td>174</td>
</tr>
<tr>
<td>Limbo (n=16)</td>
<td>2.7</td>
<td>2.5</td>
<td>10</td>
</tr>
<tr>
<td>Dubhe (n=16)</td>
<td>2.8</td>
<td>2.0</td>
<td>9.2</td>
</tr>
<tr>
<td>Limbo (n=256)</td>
<td>12</td>
<td>11</td>
<td>5.8</td>
</tr>
<tr>
<td>Dubhe (n=256)</td>
<td>6.6</td>
<td>6.0</td>
<td>6.1</td>
</tr>
</tbody>
</table>
## AES based Signature (~128-bit comp. sec.)

<table>
<thead>
<tr>
<th>Sec.</th>
<th>Signature</th>
<th>S time (ms)</th>
<th>V time (ms)</th>
<th>Sign. size (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Virgo</td>
<td>2265</td>
<td>21</td>
<td>174</td>
</tr>
<tr>
<td>103</td>
<td>Virgo++ (243 layers)</td>
<td>49</td>
<td>55</td>
<td>775</td>
</tr>
<tr>
<td>101</td>
<td>Virgo++ (9 layers)</td>
<td>409</td>
<td>32</td>
<td>129</td>
</tr>
<tr>
<td>127</td>
<td>Limbo</td>
<td>3.6</td>
<td>2.5</td>
<td>21</td>
</tr>
<tr>
<td>128</td>
<td>Dubhe</td>
<td>4.8</td>
<td>4.0</td>
<td>30</td>
</tr>
<tr>
<td>133</td>
<td>SPHINCS+-128 (smaller)</td>
<td>164</td>
<td>0.4</td>
<td>29</td>
</tr>
<tr>
<td>128</td>
<td>SPHINCS+-128 (faster)</td>
<td>17</td>
<td>0.7</td>
<td>49</td>
</tr>
</tbody>
</table>
Ring Identification / Signature

**Ring Identification**: Prove in ZK one’s identity belongs to a predefined group.

**Ring Signature**: Sign messages on behalf of a group without revealing the signer’s identity

\[ AES_{sk}(0) = ID \quad \land \quad ID \in \{ID_i, i \in [m]\} \]

Proof of membership: multiplication tree.

\[ ID \in \{ID_i, i \in [m]\} \iff \prod (ID - ID_i) = 0 \]
Ring Identification

- **Prover Time / ms**
  - 16: 1.7
  - 128: 3.8
  - 1024: 18
  - 8192: 160

- **Verifier Time / ms**
  - 16: 1.3
  - 128: 2.6
  - 1024: 11
  - 8192: 79

- **Proof Size / KB**
  - 16: 5.9
  - 128: 7.6
  - 1024: 10
  - 8192: 23

**Ring Size**

- 1
- 10
- 100
- 1000
Ring Signature

- Sign Time / ms
- Verify Time / ms
- Signature Size / KB

Comparison between Dubhe and KKW:
- Dubhe
- KKW
Thank you!

We invite you to read our paper for details and play with our implementation at

https://github.com/zkPrfs/dubhe