THINKING BACK 5 MONTHS

7000 GWEI BASEFEE FLASHBACKS
Enter the L2s
Scaling

L2s Scale Compute

EIP 4844

(Proto) Danksharding Scales Data Availability
Checking availability
Serving the data
TWO WEEKS LATER...
Throw it all away!
## Why KZG?

<table>
<thead>
<tr>
<th></th>
<th>Merkle (SHA 256)</th>
<th>Merkle (Arithmetic)</th>
<th>KZG</th>
<th>IPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verification inside</td>
<td>Prohibitive</td>
<td>Doable</td>
<td>Cheap</td>
<td>Cheap</td>
</tr>
<tr>
<td>SNARK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proof Verification</td>
<td>Hard</td>
<td>SNARKs needed for</td>
<td>Easy</td>
<td>O(N) time</td>
</tr>
<tr>
<td>(&amp; low deg. proofs)</td>
<td></td>
<td>easy verify</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proof generation in</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Only with larger proof-size</td>
</tr>
<tr>
<td>O(n*log(n))</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commit linearity</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Proof linearity</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Security assumption</td>
<td>Very safe</td>
<td>Immature hash</td>
<td>Trusted setup + Quantum vuln.</td>
<td>Quantum vuln.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Modified from Vitalik's post: ethresear.ch/t/13863
Wait, what a trusted setup?!
Summoning a secret
Only one honest participant required
There will be bugs!
There will be malicious users
More Participants than Powers!

Note: Log-Log scale
Design decisions - KISS

- Quick and easy $\Rightarrow$ Many participants
- Simple specs $\Rightarrow$ Many implementations (write your own!)
- Few powers $\Rightarrow$ Low resource requirements
- Updatable setup $\Rightarrow$ Future ceremonies can add to it
How does it work?
How does the ceremony work
How does the ceremony work
How does the ceremony work
Enter the curves
Sequencer checking things
What could go wrong?

- Censor contributions
  - Lie about verification
  - Not give you a turn
- But, this is attributable!
How does it really work?
Powers ??

\([s], [s^2], [s^3], \ldots, [s^{2^{12}}]\)
\[ [s], [s^2], [s^3], \ldots, [s^{2^{12}}] \]
Entropy Calculation

\[ [s], [s^2], [s^3], \ldots, [s^{2^{12}}] \]
Doing a heckn’ big calculate

\[ [s], [s^2], [s^3], \ldots, [s^{2^{12}}] \]

\[ x, x^2, x^3, \ldots, x^{2^{12}} \]
Even more math...

\[ [s], [s^2], [s^3], \ldots, [s^{2^{12}}] \]

\[ x, x^2, x^3, \ldots, x^{2^{12}} \]

\[ x[s], x^2[s^2], x^3[s^3], \ldots, x^{2^{12}}[s^{2^{12}}] \]
Even more math...

\[ [s], [s^2], [s^3], \ldots, [s^{2^{12}}] \quad \times \quad x, x^2, x^3, \ldots, x^{2^{12}} \]

\[ x[s], x^2[s^2], x^3[s^3], \ldots, x^{2^{12}}[s^{2^{12}}] \]

\[ [x \cdot s], [(x \cdot s)^2], [(x \cdot s)^3], \ldots, [(x \cdot s)^{2^{12}}] \]
\[ [x \cdot s], [(x \cdot s)^2], [(x \cdot s)^3], \ldots, [(x \cdot s)^{2^{12}}] \]
When can you participate?

**Testnet**
- Next few days - Opening the ceremony up to the public to smooth out the UI.

**Final Audit**
- 7 Nov - The sequencer and the primary contributor implementation audit starts

**Launch**
- End Nov - The ceremony goes live. Runs for at least 2 months.

**EIP-4844**
- 2023 sometime - Up to the client devs and community demand.
 Grants!

Writing an implementation
- 🛠️ Roll your own crypto!
- ⚒️ Implement a client

Crazy randomness generation
- 🌵 Vision trip
- 🌋 Collect weird entropy
- 🔥 Destroy your secret (and PC?!)
Thank you!

Carl Beekhuizen
EF Researcher
carl@ethereum.org

@CarlBeek
A contribution contains

\[ [x. s]_1, [(x. s)^2]_1, [(x. s)^3]_1, \ldots, [(x. s)^{2^{12}}]_1 \]

\[ [s]_2 \quad [x. s]_2 \]

Update check

\[ e([x. s]_1, g_1) \overset{?}{=} e([x]_1, [s]_2) \]

Powers check

\[ \forall i \in \mathbb{Z}^+, i \leq 2^{12} : \quad e([x. s]^i_1, [x. s]_2) \overset{?}{=} e([(x. s)^{i+1}]_1, g_1) \]
<table>
<thead>
<tr>
<th>Verification inside SNARK</th>
<th>Merke (SHA 256)</th>
<th>Merkle (Arithmetic)</th>
<th>KZG</th>
<th>IPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prohibitive</td>
<td>Prohibitive</td>
<td>Doable</td>
<td>Cheap</td>
<td>Cheap</td>
</tr>
<tr>
<td>Hard</td>
<td>Hard</td>
<td>SNARKs needed for easy verify</td>
<td>Easy</td>
<td>O(N) time</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Only with larger proof-size</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Security assumption</td>
<td>Very safe</td>
<td>Immature hash functions</td>
<td>Trusted setup + Quantum vuln.</td>
<td>Quantum vuln.</td>
</tr>
</tbody>
</table>

Why KZG?

- Modified from Vitalik’s post: ethresear.ch/t/13863