SNARKPack
Practical Groth16 Aggregation

Joint work with
Mary Maller

Anca Nitulescu
Nicolas Gailly
Protocol Labs
In Brief
Groth16

\[ e(A, B) = e(C, D) \]
Many SNARKs

Proofs

\[ = \{ A_i, B_i, C_i \}_{i=1,n} \]

Verification \((D = g^d)\)

\[ e(A_i, B_i) = e(C_i, D) \]
Many SNARKs

Proofs

\( \pi \)

\[ \{ A_i, B_i, C_i \} \]

\( i = 1, n \)

Verification \((D = g^d)\)

\[ e(A_1, B_1) = e(C_1, D) \]

\[ e(A_2, B_2) = e(C_2, D) \]

\[ \ldots \]

\[ e(A_n, B_n) = e(C_n, D) \]
\[ e(A_1, B_1) = e(C_1, D) \]
\[ e(A_2, B_2) = e(C_2, D) \]
\[ \ldots \]
\[ e(A_n, B_n) = e(C_n, D) \]
SNARK Batching

Verification

\[ e(A_1, B_1) = e(C_1, D) \]
\[ e(A_2, B_2) = e(C_2, D) \]
\[ e(A_n, B_n) = e(C_n, D) \]

\[ r \leftarrow \$ \]

\[ e(A_1, B_1) = e(C_1, D) \]
\[ e(A_2, B_2) = e(C_2, D) \]
\[ e(A_n, B_n) = e(C_n, D) \]

\[ r \]
\[ r^2 \]
\[ r^n \]
SNARK Batching

Verification

\[ e(A_1, B_1) = e(C_1, D) \]
\[ e(A_2, B_2) = e(C_2, D) \]
\[ \ldots \]
\[ e(A_n, B_n) = e(C_n, D) \]

Batch Verification

\[ r \leftarrow $ \]
\[ \prod e(A_i, B_i)^{r_i} = \prod e(C_i, D)^{r_i} \]
SNARK Aggregation

Batch Verification

\[ \prod e(A_i, B_i)^{r_i} = \prod e(C_i, D)^{r_i} \]

\[ \prod e(A_i, B_i r_i) = e(\prod C_i r_i, D) \]

\[ \langle g \rangle = G, \langle \tilde{g} \rangle = \tilde{G} \]
\[ e : G \times G \rightarrow \tilde{G} \]
\[ e(g^a, g^b) = \tilde{g}^{ab} \]
SNARK Aggregation

Batch Verification

\[ \prod e(A_i, B_i)^{r_i} = \prod e(C_i, D)^{r_i} \]

\[ \iff \]

\[ \prod e(A_i, B_i^{r_i}) = e(\prod C_i^{r_i}, D) \]
SNARK Aggregation

Batch Verification

\[ \prod e(A_i, B_i)^{r_i} = \prod e(C_i, D)^{r_i} \]

\[ \prod e(A_i, B_i^{r_i}) = e(\prod C_i^{r_i}, D) \]

Aggregation

\[ Z_{AB} = \prod e(A_i, B_i^{r_i}) \]

\[ Z_C = \prod C_i^{r_i} \]
**SNARK Aggregation**

Batch Verification

$$\prod e(A_i, B_i)^{r_i} = \prod e(C_i, D)^{r_i}$$

$$Z_{AB} = e(Z_C, D)$$

Aggregation

$$Z_{AB} = \prod e(A_i, B_i)^{r_i}$$

$$Z_C = \prod C_i^{r_i}$$
SNARK Aggregation

Batch Verification

\[ \prod e(A_i, B_i)^{r_i} = \prod e(C_i, D)^{r_i} \]

\[ Z_{AB} = e(Z_C, D) \]

Aggregation

\[ Z_{AB} = \prod e(A_i, B_i^{r_i}) \]

\[ Z_C = \prod C_i^{r_i} \]
Construction
Tools: MIPP & TIPP

Proofs for Inner Pairing Products and Applications - Bünz, Maller, Mishra, Tyagi, Vesely

\[ \langle A, b \rangle = \prod A_i^{b_i} \]

\[ \langle A, B \rangle = \prod e(A_i, B_i) \]

\[ A_i, B_i \in G, \quad b_i \in \mathbb{Z}_q \]
Tools: MIPP & TIPP

Proofs for Inner Pairing Products and Applications - Bünz, Maller, Mishra, Tyagi, Vesely

\[ \langle A, b \rangle = \prod A_i^{b_i} \]

\[ \langle A, B \rangle = \prod e(A_i, B_i) \]

\[ Z_c = \prod C_i^{r_i} \]

\[ Z_{AB} = \prod e(A_i, B_i^{r_i}) \]

Aggregation
Tools: MIPP & TIPP

Proofs for Inner Pairing Products and Applications - Bünz, Maller, Mishra, Tyagi, Vesely

\[ \langle A, b \rangle = \prod A_i^{b_i} \]

\[ \langle A, B \rangle = \prod e(A_i, B_i) \]

\[ Z_C = \langle C, r \rangle \]

\[ Z_{AB} = \langle A, B^r \rangle \]

Aggregation
Tools: MIPP & TIPP

Proofs for Inner Pairing Products and Applications - Bünz, Maller, Mishra, Tyagi, Vesely

\[ \langle C, r \rangle = \prod C_i^{r_i} \]

\[ \langle A, B^r \rangle = \prod e(A_i, B_i^{r_i}) \]

\[ Z_C = \langle C, r \rangle \]

\[ Z_{AB} = \langle A, B^r \rangle \]

Aggregation
MIPP & TIPP Strategy

Proofs for Inner Pairing Products and Applications - Bünz, Maller, Mishra, Tyagi, Vesely

$Z_C = \langle C, r \rangle$

$Z_{AB} = \langle A, B^r \rangle$
\[ \langle A, B \rangle = e(A_1, B_1) e(A_2, B_2) \cdots e(A_n, B_n) \]

\[ A = (A_1, A_2, \ldots A_n) \]

\[ B = (B_1, B_2 \ldots B_n) \]
Split & Collapse

\[ \langle A, B \rangle = e(A_1, B_1) e(A_2, B_2) \ldots e(A_n, B_n) \]

\[ A = (A_1, A_2, \ldots A_n) \]

\[ B = (B_1, B_2 \ldots B_n) \]

\[ A_{\text{left}} \quad A_{\text{right}} \]

\[ B_{\text{left}} \quad B_{\text{right}} \]
Split & Collapse

Split

\[ A = (A_{\text{left}}, A_{\text{right}}) \]
\[ B = (B_{\text{left}}, B_{\text{right}}) \]

Collapse

\[ A = (A_{\text{left}}, A_{\text{right}}) \]
\[ B = (B_{\text{left}}, B_{\text{right}}) \]
Split & Collapse

Split

\[ A = (A_{\text{left}}, A_{\text{right}}) \]
\[ B = (B_{\text{left}}, B_{\text{right}}) \]

Collapse

\[ A_{\text{left}} A_{\text{right}}, B_{\text{left}} B_{\text{right}} \]

\[ L = \langle A_{\text{right}}, B_{\text{left}} \rangle \]
\[ R = \langle A_{\text{left}}, B_{\text{right}} \rangle \]

\[ H_{A,B} = L \langle A, B \rangle R \]

\[ (A, B) = e(A_1, B_1) e(A_2, B_2) \ldots e(A_n, B_n) \]
Split & Collapse

A = (A\text{left}, A\text{right})
B = (B\text{left}, B\text{right})

H_{A,B} = \langle A\text{left} A\text{right} , B\text{left} B\text{right} \rangle

L = \langle A\text{right} , B\text{left} \rangle
R = \langle A\text{left} , B\text{right} \rangle

H_{A,B} = L \langle A , B \rangle R

\langle A , B \rangle = e(A_1 , B_1) e(A_2 , B_2) \ldots e(A_n , B_n)
Trusted Setup

CRS
Aggregation from existing CRS
SNARK Aggregation

Filecoin: Powers of Tau

Zcash: Powers of Tau
Commitments

\[ \text{Com}(\mathbf{C}) = \langle \mathbf{C}, \mathbf{v}_1 \rangle, \quad \langle \mathbf{C}, \mathbf{v}_2 \rangle \]
Commitments

\[
\text{Com}(A, B) = \langle A, v_1 \rangle \langle w_1, B \rangle
\]
Commitments

\[
\text{Com}(A, B) = \langle A, v_1 \rangle \langle w_1, B \rangle, \\
\langle A, v_2 \rangle \langle w_2, B \rangle
\]
SNARK Aggregation

\[
\text{Aggregation} \quad \{ \begin{array}{c} A_i \quad B_i \quad C_i \end{array} \} \quad i=1,n
\]
SNARK Aggregation

\[ \text{Aggregation} \]

\[ \{ A_i, B_i, C_i \} \]

\[ i=1,n \]

\[ \text{Com}(A, B) \]

\[ \text{Com}(C) \]
**SNARK Aggregation**

\[
\text{Aggregation: } \{\text{Ai, Bi, Ci}\}_{i=1,n} \rightarrow \text{Com(A, B)} \rightarrow \text{Com(C)}
\]

**MIPP:**
\[
\langle C, r \rangle = \prod C_i^{r_i}
\]

**TIPP:**
\[
\langle A, B^r \rangle = \prod e(A_i, B_i^{r_i})
\]
Aggregation

\[ \text{TIPP}: \left\langle A, B^r \right\rangle = \prod e(A_i, B_i^r) \]

\[ \text{MIPP}: \left\langle C, r \right\rangle = \prod C_i^r_i \]

\[ Z_{AB} = \left\langle A, B^r \right\rangle \]

\[ Z_C = \left\langle C, r \right\rangle \]

Aggregation

\[ \text{Com}(A, B) \]

\[ \text{Com}(C) \]

log n

\[ i=1,n \]
**SNARK Aggregation**

Aggregation:

\[
\{ A_i, B_i, C_i \} \quad i=1,n
\]

\[
Z_C = \langle C, r \rangle
\]

\[
Z_{AB} = \langle A, B^r \rangle
\]

Verification:

\[
Z_{AB} = e(Z_C, D)
\]

MIPP + TIPP
Application: Filecoin
Proof of storage

Proves 32GB

ProveCommit (10 SNARKs)

submit on chain

Block n

ProveCommit (10 SNARKs)

ProveCommit (10 SNARKs)

Block n+1
Problem: a block is finite

Proves 32GB

ProveCommit (10 SNARKs)

submit on chain

Block n

Block n+1

Currently hovering around 6 millions SNARK per day!!
Solution (1) : Batch verification

Operations can efficiently be batched for faster verification

Block n → ProveCommit → ProveCommit → ProveCommit → Valid proofS (20ms)
In practice, we start to go up to x200 more ProveCommits on chain (~x2000 SNARKs)
Implementation
Library

- Coded in **Rust**, available at [https://github.com/filecoin-project/bellperson](https://github.com/filecoin-project/bellperson) branch feat-ipp2
- Initial code from the **arkworks library** [https://github.com/arkworks-rs/ripp/](https://github.com/arkworks-rs/ripp/)
- Ported & optimized in the **bellman** framework (bellperson fork)
- Using **BLS12-381 curves** from the **blst library** [https://github.com/supranational/blst](https://github.com/supranational/blst)
- **SRS** combined from Filecoin & Zcash power of taus
  - Code at [https://github.com/nikkolasg/taupipp](https://github.com/nikkolasg/taupipp)
  - Up to $2^{19}$
- Benchmark performed on 32c/64t **AMD Raizen Threadripper**
- **Audited** by NCC and second audit in progress by Matteo Campanelli
Verifier Time

- Verifying aggregate proofs becomes **faster** from *32 proofs*.
- **8192 proofs** in **33ms**
  - "ratio" of 0.004 ms per proof
- **Including unserialization**
- **Optimizations:**
  - Relies heavily on parallelism
  - MIPP/TIPP combined
  - Randomized pairing checks, a single final exponentiation
Proof size

- Use **compression** of $G_T$ points
  - Based on Taurus compression
  - From RELIC library implementation

- Turnover at 128 proofs
  - 23kB for aggregated
  - 24kB for “all proofs”
Aggregation Time

- 8.7s for 8192 proofs
- Relies heavily on parallelism
- $2^{17}$ proofs in ~2mn
Questions / Discussion

- **Trusted Setup**: Main feature is to rely on existing Groth16 CRS - at the cost of slightly more expensive scheme
- **Library**: currently only in bellperson/bellman framework: interested in standalone library?
- **Compatibility**: Not compatible with arkworks library currently
- **Extension**: Could we extend this scheme to other pairing-based primitives? Currently only supports Groth16
- **Standardization**?
Thanks!

eprint.iacr.org/2021/529
Credits

Special thanks to all those who made and released these resources for free:

- Presentation template by SlidesCarnival
- Illustrations by Iconfinder