ECLIPSE*: Better Commit-and-Prove SNARKs with Universal SRS

Diego F. Aranha¹, Emil Madsen Bennedsen², Matteo Campanelli³, Chaya Ganesh⁴, Claudio Orlandi¹, Akira Takahashi¹

*Enhanced Compiling Method for Pedersen-Committed zkSNARK Engines

https://ia.cr/2021/934

¹ Aarhus University
² Concordium
³ Protocol Labs
⁴ Indian Institute of Science
Our Setting

• Succinct and non-interactive ZK (SNARKs)
• Commit-and-Prove (CP-SNARK)
• Universal Trusted Setup
Succinct and Non-Interactive ZK

e.g., $x = \text{msg}$, $w = \text{signature}$
OK, P must know $w$ such that $R(w)$ holds.

“Trusting someone else’s claims on data that you have not seen”
In CP-ZK we prove $R$ and we open a commitment.
Motivation for CP

Compression/Fingerprinting

Commit-ahead-of-time

Modular/efficient composition of proofs

[e.g., $SHA(g^x) = y$]

Efficient Proof Scheme

Some proof

Some other proof

My “credentials”

[Public ML models]

[Time]

[Sensitive DB]

[Proofs of correct training]
Some Applications

• Anonymous Credentials

• Blockchains:
  • with privacy properties
  • proofs on data posted on blockchains

• Generally: anywhere data need to be referenced to (privately or succinctly)
Syntax: SNARKs vs CP-SNARKs

Setting on the right is a special case of the other. Then why care??

Efficiency & interoperability

e.g., $x = \text{msg}$, $w = \text{signature}$
e.g., $c_i \text{ commit to DBs}$
Clarifying (our) CP-SNARK Setting

Desiderata ([CFQ18,ZKProof]:
- Efficient ZK opening
- Interoperable commitments (as standard as possible)

**Unsatisfactory Solutions:**
- Use Merkle trees or Pedersen to commit, then open in circuit
- ⭕ Standard commitments ✖ Expensive

$$P \xrightarrow{\mathsf{a}} \mathcal{R}, \mathcal{R} \rightarrow \mathcal{V} \rho_j (x, c_1, \ldots, c_k)$$
Clarifying (our) CP-SNARK Setting

Desiderata ([CFQ18, ZKProof]):
- Efficient ZK opening
- Interoperable commitments (as standard as possible)

**UNSATISFACTORY SOLUTIONS:**
- Use Merkle trees or Pedersen to commit, then open in circuit

### STANDARD COMMITMENTS
- AS ABOVE BUT WITH SMART ARITHMETIC/ECCLEPTIC CURVES
  (e.g., ZCash [IJPJ16], [COCO], [VCS16], [JABBERWOCKY])

### EXPENSIVE
- More expansive

### EFFICIENT
- Curve dependent

\[ P \xrightarrow{\gamma} y \]
\[ f_y(x, e_1, \ldots, e_k) \]
Trust Models in SNARKs (and CP-SNARKs)

- **Transparent :-))** (Bulletproofs, Hyrax, DARK…)
  - no trusted setup

- **SRS (Structured Reference String) :-|** (Pinocchio, Groth16…)
  - Keygen(R) -> srs_R

- **Universal SRS (USRS) :-)** (GKMM18, LegoSNARK, Sonic, Marlin, PLONK,…)
  - Keygen(maxSize) -> srs_gen
  - Specialize(srs_gen, R) -> srs_R
  - Often also **updatable** (anyone can rerandomize srs_gen)
Eclipse results from $10^9$ feet:
new ways to construct CP-SNARKs with a Universal SRS generically
Summary of Our Results

- General Compiler into CP-SNARKs with Universal SRS
  - Your favorite SNARK* with USRS -> CP-SNARK
  - * in "information-theoretic" form (more on that later)
- CP versions of Marlin, PLONK, and SONIC
  - commitment type = Pedersen
- All with small overhead (next slide)
Resulting USRS CP-SNARKs—Efficiency

|                  | $|\pi|$      | Prove (time)              | Verify (time)         |
|------------------|------------|---------------------------|-----------------------|
| ECLIPSE [ABC+21] | $O(\log(\ell \cdot d))$ | $O(n + \ell \cdot d)$ | $O(\ell \cdot d)$ |
| Lunar [CFF+20]   | $O(\ell)$      | $O(n + \ell \cdot d)$   | $O(\ell)$            |
| LegoUAC [CFQ19]  | $O(\ell \log^2(n))$ | $O(n) + \ell \cdot \tilde{O}(d)$ | $O(\ell \log^2(n))$ |

Time is in group operations. Above, $n$ is roughly # of multiplication gates.

In practice the two family of systems show a tradeoff in verification time/proof size.
Constructing (USRS) SNARKs
Compilers from idealized information-theoretic objects

Information-theoretic Object + Crypto primitive → Cryptographic Proof System

Compilation
# Practical* SNARKs with Universal SRS

| zkSNARK          | size $|\mathcal{vk}_R|, |\pi|$ | time | Prove                  | Verify                |
|------------------|------------------|------|----------------|------------------|
| Sonic            | $G_1$            | 20   | 273$n$        | 7 pairings       |
|                  | $G_2$            | 3    | $O(m \log m)$ | $O(\ell + \log m)$ |
|                  | $\mathbb{F}$    | 16   |                |                  |
| MARLIN           | $G_1$            | 12   | 14$n + 8m$    | 2 pairings       |
|                  | $G_2$            | 2    |                |                  |
|                  | $\mathbb{F}$    | 8    |                |                  |
| PLONK (small proof) | $G_1$          | 8    | 11$n + 11a$   | 2 pairings       |
|                  | $G_2$            | 1    |                |                  |
|                  | $\mathbb{F}$    | 7    |                |                  |
| PLONK (fast prover) | $G_1$          | 8    | $O((n + a)\log(n + a))$ | $O(\ell + \log(n + a))$ |
|                  | $G_2$            | 9    |                |                  |
|                  | $\mathbb{F}$    | 1    |                |                  |
|                  |                  | 7    |                |                  |

Roughly:
- $n$: # MUL gates
- $a$: # ADD gates
- $m$: # wires

*practical + focus is on $O(1)$ proof size
Idealized protocols for USRS SNARKs

Algebraic Holographic Proofs (AHPs)

- Interactive
- Prover holds polynomials "encoding" the witness
- It gives oracle access to their evaluations
A picture of the idealized protocol

Queries Q:
Evaluations of polynomial (e.g. $p_1(x^*) = t^*$)
Compiling to USRS SNARKs: Ingredients

- Underlying compiler in Marlin/DARK/Lunar/PLONK

Main tool is a **Polynomial Commitment** PC:

- with *compressing* commitment to polynomials
- Allows proving efficiently (and succinctly) in ZK:
  - \( p(x) = y \) (evaluation)
  - (plus degree bounds: \( \deg(p) \leq D\text{bound} \))

**Notation** (circles for polynomial commitment)

NB: different from these commitments!
Compiling to USRS SNARKs
Compiling to USRS SNARKs

Proves queries $Q$ are satisfied by poly commitments $c_1, \ldots, c_N$
The Resulting USRS SNARKs

• Use Fiat-Shamir for non-interaction

• Why is the SRS *Universal*?
  • Because we can define
    
    \[ \text{SNARK.Setup(maxSize)} \rightarrow \text{srs\_gen := PC.Setup(maxPolyDeg)} \]
    
    • Where maxPolyDeg depends on maxSize
Compiling into CP-SNARKs
Compiling into CP-SNARKs
Compiling into CP-SNARKs

It proves "linking", or: knowledge of \( w \) s.t:

1) \([c]\) opens to (parts of) \( w \)
2) \([c_i]\) opens to \( p_i \), forall \( i \)
3) \( w \) is "consistent with the execution"
Challenge 1: depending on only part of the witness

It proves "linking", or: knowledge of \( w \) s.t:
1) \([c]\) opens to (parts of) \( w \)
2) \((c_i)\) opens to \( p_i\), forall \( i \)
3) \( w \) is "consistent with the execution"

Our solution:
showing that \((c_i)\) can be additively decomposed in our SNARKs of interest

Definition 9 (Decomposable witness-carrying polynomials). Let \( W \) be an index set of witness-carrying polynomials of AHP. We say that polynomials \((p_{i,j}(X))_{(i,j) \in W}\) of AHP are decomposable if there exists an efficient function \( \text{Decomp}((p_{i,j}(X))_{(i,j) \in W}, I) \rightarrow (p_{i,j}^{(1)}(X), p_{i,j}^{(2)}(X))_{(i,j) \in W} \) such that it satisfies the following properties for any \( I \subset [n] \):
- Additive decomposition: \( p_{i,j}(X) = p_{i,j}^{(1)}(X) + p_{i,j}^{(2)}(X) \) for \((i,j) \in W\).
- Degree preserving: \( \deg(p_{i,j}^{(1)}(X)) \) and \( \deg(p_{i,j}^{(2)}(X)) \) are at most \( \deg(p_{i,j}(X)) \) for \((i,j) \in W\).
- Non-overlapping: Let \( w = \text{WitExt}((p_{i,j}(X))_{(i,j) \in W}), w^{(1)} = \text{WitExt}((p_{i,j}^{(1)}(X))_{(i,j) \in W}), \) and \( w^{(2)} = \text{WitExt}((p_{i,j}^{(2)}(X))_{(i,j) \in W}). \) Then
  \[ (w_i)_{i \in I} = (w_i^{(1)})_{i \in I}, \]  \[ (w_i)_{i \notin I} = (w_i^{(2)})_{i \notin I}, \]  \[ (w_i^{(1)})_{i \notin I} = 0, \]  \[ (w_i^{(2)})_{i \in I} = 0. \]
Challenge 2: efficient and succinct proof of linking

It proves "linking", or: knowledge of \( w \) s.t:
1) [c] opens to (parts of) \( w \)
2) (c_i) opens to p_i, forall i
3) \( w \) is "consistent with the execution"

From previous slide

- Our solution:
  - Prove through an (amortized) Sigma-protocol a “squashing” of the input commitments
    \[
    C = g^{wh^\alpha}, \hat{C}_i = G^{w_i}H^{\beta_i}, \ w = [w_1, \ldots, w_\ell]
    \]
  - naively requires \( O(|w| \cdot \#\text{commitments}) \) communication, but we then compress it through Compressed-Sigma techniques [AC20] to \( O(\log(|w| \cdot \#\text{commitments})) \)
Comparison with Lunar (CFFQ21)

- Similar blueprint
- Lunar uses a different pairing-based protocol for "linking"
- Different tradeoffs in efficiency (see also table in the next slide)
- Lunar uses a more general formalization (PHP); our work can be easily formalized in the same framework
Open Questions

- Better asymptotics:
  - $O(\ell)$ is inherent in verification time, but can we achieve constant proof size?

- Maybe with one-level of (specialised) recursion?

- Different techniques for “linking” and/or finding other applications for those in ECLIPSE?

|          | $|\pi|$     | Prove (time)  | Verify (time) |
|----------|------------|---------------|---------------|
| ECLIPSE  | $O(\log(\ell \cdot d))$ | $O(n + \ell \cdot d)$ | $O(\ell \cdot d)$ |
| Lunar    | $O(\ell)$  | $O(n + \ell \cdot d)$ | $O(\ell)$     |
| Future?  | $O(1)$     | $O(\ell)$     | $O(\ell)$     |
https://ia.cr/2021/934

Thanks!

For questions/contacts:
matteo@protocol.ai
binarywhales.com