Conclusions of standards working groups

Ying Tong Lai, Mary Maller, Michele Orrù

ZKProof6 - 24th May 2024

Why "standards"?

Standards Committee and Working Groups

- The ZKProof Standards Committee was formed in 2023 to develop a process for community promotion of specifications and standards, and oversee the creation of corresponding working groups (WG).

- These processes are in an early stage of implementation.

- The processes will be adaptable to integrate learnings.

Process for creating a ZKProof Working Group

- **Abstract submission:** Propose an abstract to the ZKProof Standards Committee that includes:
  - Title, team and public contact address
  - **WG Liaison:** WG member responsible for keeping the Standards Committee aware of relevant WG updates.
  - **Structure:**
    - Expected structure of future initial draft specification.
    - Planned time-frame.
    - Expected complementary documentation (e.g., open-source reference implementation, test vectors, code instruction).
  - **References:** A list of technical/scientific references related to the WG goal.

Process for creating a ZKProof Working Group

- **Abstract Review/ Approval**
  - Upon internal analysis, Standards Committee will seek initial advisory opinion from the ZKProof Steering Committee and the ZKProof Editors Team.

  - The follow up is either
    - an approval (possibly with editorial suggestions)
    - (ii) a request for improvement/adjustment based on provided feedback, and subsequent resubmission of the WG abstract.

  - Revisions submitted under the same process.

Process for creating a ZKProof Working Group

- **WG Presentation:**
  - After an initial review, the proposed WG will be invited for a video-conference presentation about the goals and plan.
  - The Editors and Steering Committe observers will also be invited to attend, for possible further feedback.
  - In some cases this step can be replaced by a related public presentation at a ZKProof event.

Process for creating a ZKProof Specification (ZSpec)

- **Public abstract.**
- **ZSpec-Proposal:** as close as possible to a complete specification. Should be submitted for oral presentation at a public ZKProof workshop/event.
- **ZSpec-Draft:** Updated proposal based on feedback. Must wait at least 90 days for feedback.
- **ZCall for Comments.** Official ZKProof Call for Comments on a ZSpec-Draft with an open period of at least 90 days for comments.
- **Approved ZSpec.** ZKProof Standards Committee will, after consultation with other teams, issue a determination, either of approval of a new (final) **ZSpec** or a request for further reviews.
- **ZSpec Publication.** Publish approved ZSpec.
- **Updated Versions**: Updates follow similar process.

Plonkish Working Group Conclusions

What is the Plonkish Working Group

- An **arithmetisation** is a language that a proof system uses to express statements. A circuit is a program in this language.
- The associated computation has been computed correctly if and only if all of the constraints in the circuit are satisfied.
- The primary purpose is to specify a particular arithmetisation: the **Plonkish arithmetisation** used in the **Halo 2 proving system**.

Important Links

Repository contains:

- The wg-abstract
- An initial draft for a ZSpec-Proposal for simple Plonkish relation with no shift/ rotation constraints or row orderings.
- A started ZSpec-Proposal for optimised Plonkish relation (not nearly ready).
- Placeholder links for reference implementations.

Wg-plonkish public repo

https://github.com/zkpstandard/wg-plonkish/

Yesterday's Working Session

- Walked through simpler relation spec in lots of detail. This highlighted areas that were poorly explained and typos.
- Briefly showed WIP on optimised relation. Discussed the challenges.
- Actionables:
    - Generalise the equivalence relation
    - Multivariate polynomial outputs should be vectors

Notes by Ying Tong Lai

https://hackmd.io/kyPxyaMaSWeYpJE8Lst_sA

Some Conclusions from Yesterday

- **Rotations:**
  - Rotations are alternative to copy constraints that can be more efficient.
  - Multivariate IOPs can often only support shifts and not rotations.
  - We do not want separate univariate and multivariate constraints system.
  - We will aim for the optimised Plonkish relation to support shifts only.

- **Clap:** Recently released paper could provide useful insights https://arxiv.org/abs/2405.12115

Oracle Compiler Working Group Conclusions

Abstract approved; now recruiting

We have an **abstract** outlining the initial draft specification:

Initial contributors:

Abhiram Kothapalli (CMU)
Adrian Hamelink (independent)
Nick Ward (Polygon Zero)
Han (PSE)
Ignacio Manzur (Nethermind)
Pratyush Mishra (UPenn)
Sarah Meiklejohn (UCL)
Ying Tong (Geometry Research)

- compatibility (IOP -> commitment scheme)
  - univariate/multilinear
  - field size
- security (soundness, zero-knowledge)
- efficiency (succinctness; batch commitment/verification)
- reference API / default implementation
  - reference spec in pseudocode / Lean?
  - default implementation in Rust?

Join us in writing the spec!

e.g. arkworks-rs

```
/// Describes the interface for a polynomial commitment scheme that allows a sender to commit to multiple
/// polynomials and later provide a succinct proof of evaluation for the corresponding commitments at a query set
/// `Q`, while enforcing per-polynomial degree bounds.
pub trait PolynomialCommitment<F: PrimeField, P: Polynomial<F>, S: CryptographicSponge>: Sized
{
   fn setup<R: RngCore>(max_degree: usize, num_vars: Option<usize>, rng: &mut R) -> Result<Self::UniversalParams,
Self::Error>;
   fn commit<'a>(...) -> Result<(Vec<LabeledCommitment<Self::Commitment>>,Vec<Self::CommitmentState>), Self::Error>>
   fn open<'a>(...) -> Result<Self::Proof, Self::Error>
   fn check<'a>(...) -> Result<bool, Self::Error>
   fn batch_open<'a>(...) -> Result<Self::BatchProof, Self::Error>
   fn batch_check<'a, R: RngCore>(...) -> Result<bool, Self::Error>
   fn open_combinations<'a>(...) -> Result<BatchLCProof<F, Self::BatchProof>, Self::Error>
   fn check_combinations<'a, R: RngCore>(...) -> Result<bool, Self::Error>
}
```

Univariate: SonicKZG10, MarlinKZG10, InnerProductArgPC
Multilinear: MarlinPST13

https://github.com/arkworks-rs/poly-commit/blob/master/poly-commit/src/lib.rs

e.g. han0110/plonkish

```rust
pub trait PolynomialCommitmentScheme<F: Field>: Clone + Debug {
    type Polynomial;
    type Commitment;

    fn setup(poly_size: usize, batch_size: usize, rng: impl RngCore) -> Result<Self::Param, Error>;
    fn commit(...) -> Result<Self::Commitment, Error>;
    fn batch_commit<'a>(...) -> Result<Vec<Self::Commitment>, Error>
    fn batch_commit_and_write<'a>(...) -> Result<Vec<Self::Commitment>, Error>
    fn open(...) -> Result<(), Error>;
    fn batch_open<'a>(...) -> Result<(), Error>
    fn verify(...) -> Result<(), Error>;
    fn batch_verify<'a>(...) -> Result<(), Error>;
}
```

Univariate: IPA, KZG, Hyrax
Multilinear: IPA, KZG, Hyrax, Zeromorph, Brakedown, Gemini

https://github.com/han0110/plonkish

Yesterday’s Working Session

minimal assumptions of the IOP:

- separately handle univariate and multilinear IOPs; i.e. no need to support univariate <-> multilinear conversions in the oracle compiler
- field size: some IOPs have a min. requirement on field size (e.g. involving log derivative) => cannot use small-field PCS
- support for rotations?
- zero-knowledge ⇒ requires hiding PCS

efficiency vs compatibility: a scheme should document the choices of components for which it's efficient

Some conclusions from yesterday

- separation of concerns: “sub-IOPs” and their composition should not be visible to the oracle compiler
  - (recruiting for IOP Working Group!)
- WLOG we can assume that an IOP uses only a single PCS; else, it should be expressed as a commit-and-prove scheme

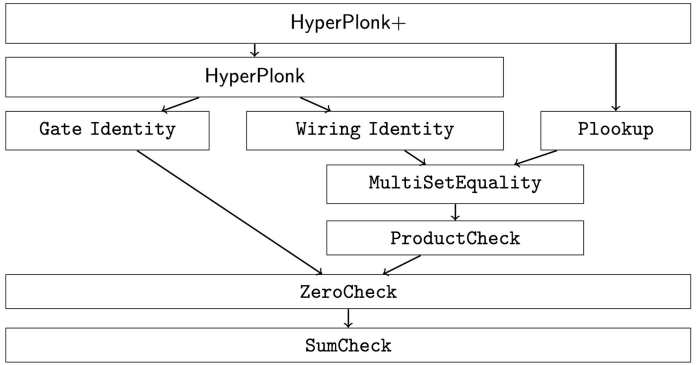

Figure 1: The multilinear polynomial-IOPs that make up HyperPlonk.

Fiat-Shamir Working Group Conclusions

Fiat-Shamir Working Group Conclusions

- Technical Report out
- New Rust library compatible with `arkworks-rs` and `group`

```
$ cargo add nimue
```

- Started specification document

fiat-shamir+subscribe@zkproof.org

Σ-Protocols Working Group Conclusions

Σ-Protocols Working Group Conclusions

- The OG working group
- NGI Zero funding
- Moved to IETF format

sigmaprotocols+subscribe@zkproof.org