A Brief Introduction to Halo 2 and Cycles of Elliptic Curves

Daira Hopwood
@feministPLT
daira#0512 on Discord
https://github.com/daira/halographs

Elliptic curves

- An elliptic curve in affine form is:
  - a *group* of points (x, y)
  - over a *field of definition* $\mathbb{F}_p$
  - satisfying some equation, say $y^2 = x^3 + ax + b$,
  - with (for this equation) a "point at infinity" as the group identity.
- If the group has a prime order q, we'll name the curve $E_{p \to q}$.
- Because it's a group, we can add points, or multiply them by a scalar.
- The *scalar field* is $\mathbb{F}_q$.
- We'll assume that a proof system using $E_{p \to q}$ efficiently supports circuits with arithmetic over $\mathbb{F}_q$.
- A cycle of elliptic curves is a pair $E_{p \to q}$ and $E_{q \to p}$.

Motivation for cycles

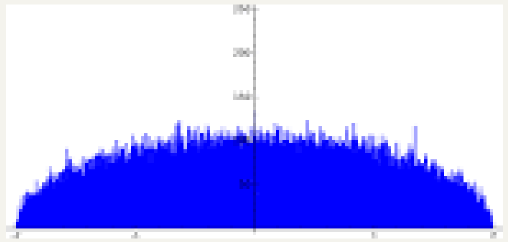
- A proof system using $E_{p \to q}$ directly supports arithmetic over $\mathbb{F}_q$.
- "Wrong-field" arithmetic has an overhead of over 1000 times.
  - This is using the sum of residues method for reduction: Zcash #4093
- Doesn't Plookup solve this?
  - No; Plookup helps but wrong-field arithmetic probably still has an overhead of 10-20 times.
- As we'll see later, it's not quite sufficient for recursive proofs to just do arithmetic in the field of definition of the other proof system. We'll need two instances of the proof system, on $E_{p \to q}$ and on $E_{q \to p}$.

The Tweedle cycle

- The Halo paper gives a pair of curves:
  - $E_{p \to q} : y^2 = x^3 + 5$ is called Tweedledum.
  - $E_{q \to p} : y^2 = x^3 + 5$ is called Tweedledee.
  - $p = 2^{254} +$ `0x38AA1276C3F59B9A14064E200000001`
  - $q = 2^{254} +$ `0x38AA127696286C9842CAFD400000001`
  - Both have 126-bit Pollard rho security, maximal embedding degree, and 2-adicity ≥ 33.
  - They have cubic endomorphisms that are useful for optimization.
  - $\gcd(p - 1, 5) = \gcd(q - 1, 5) = 1$
- These curves can also be used for Halo 2.
- We’re going to briefly explain the construction that found them.

Constructing cycles – the problem

- By the Hasse bound, the order of an elliptic curve over $\mathbb{F}_p$ lies in the range $[\,p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}\,]$. For Tweedle this range is of size ~2^{129}.
- The Sato–Tate conjecture* concerns the distribution of the order in this range. We don't need to go into detail, but here's a picture:

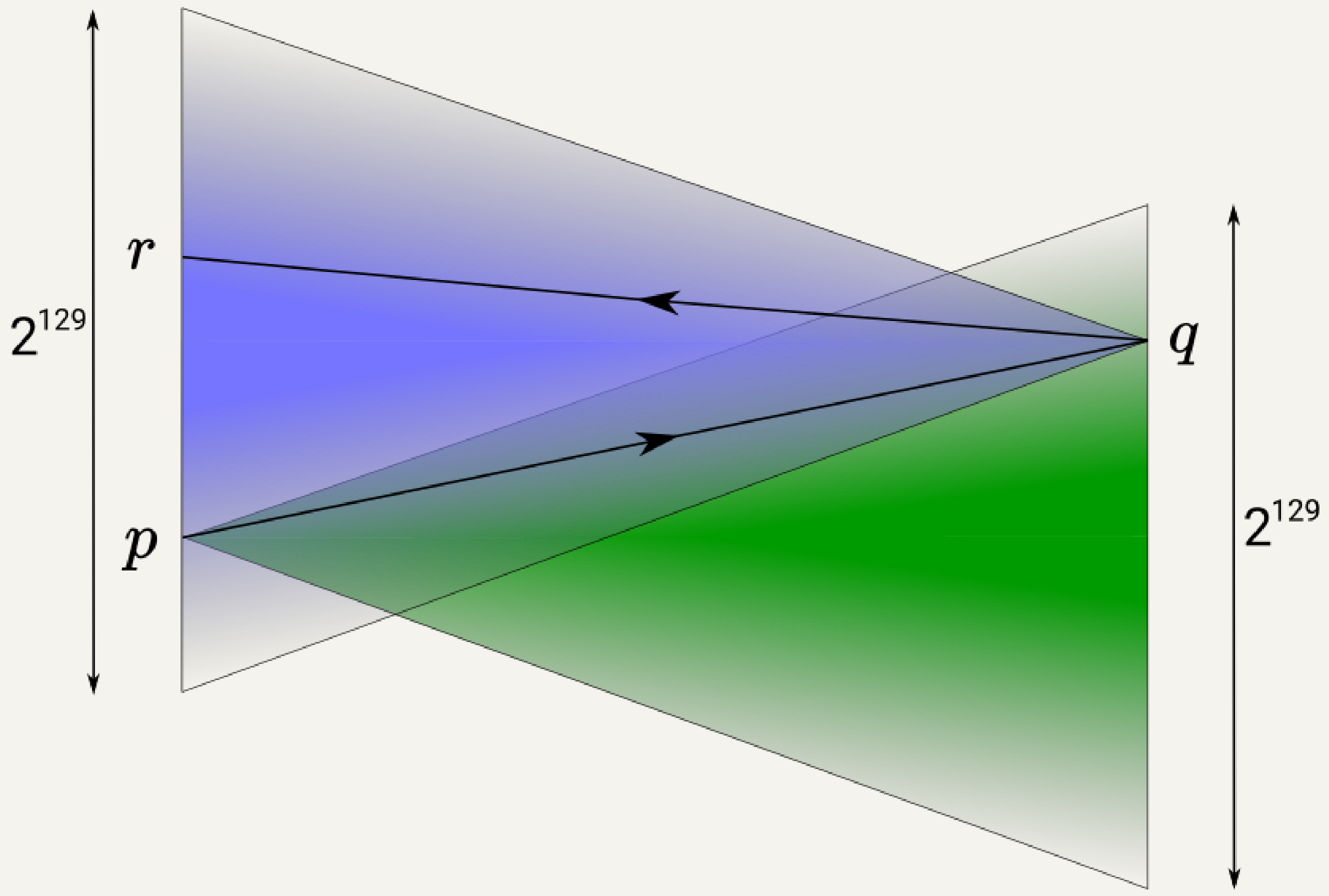


* proven by Taylor et al in 2006.

- That is, the order could be anywhere in the range.
- And (if the order is a prime q) when we construct a curve $E_{q\to r}$ it could also have order anywhere in its Hasse range.
- So, it's exceptionally unlikely that $E_{q\to r}$ has order p.

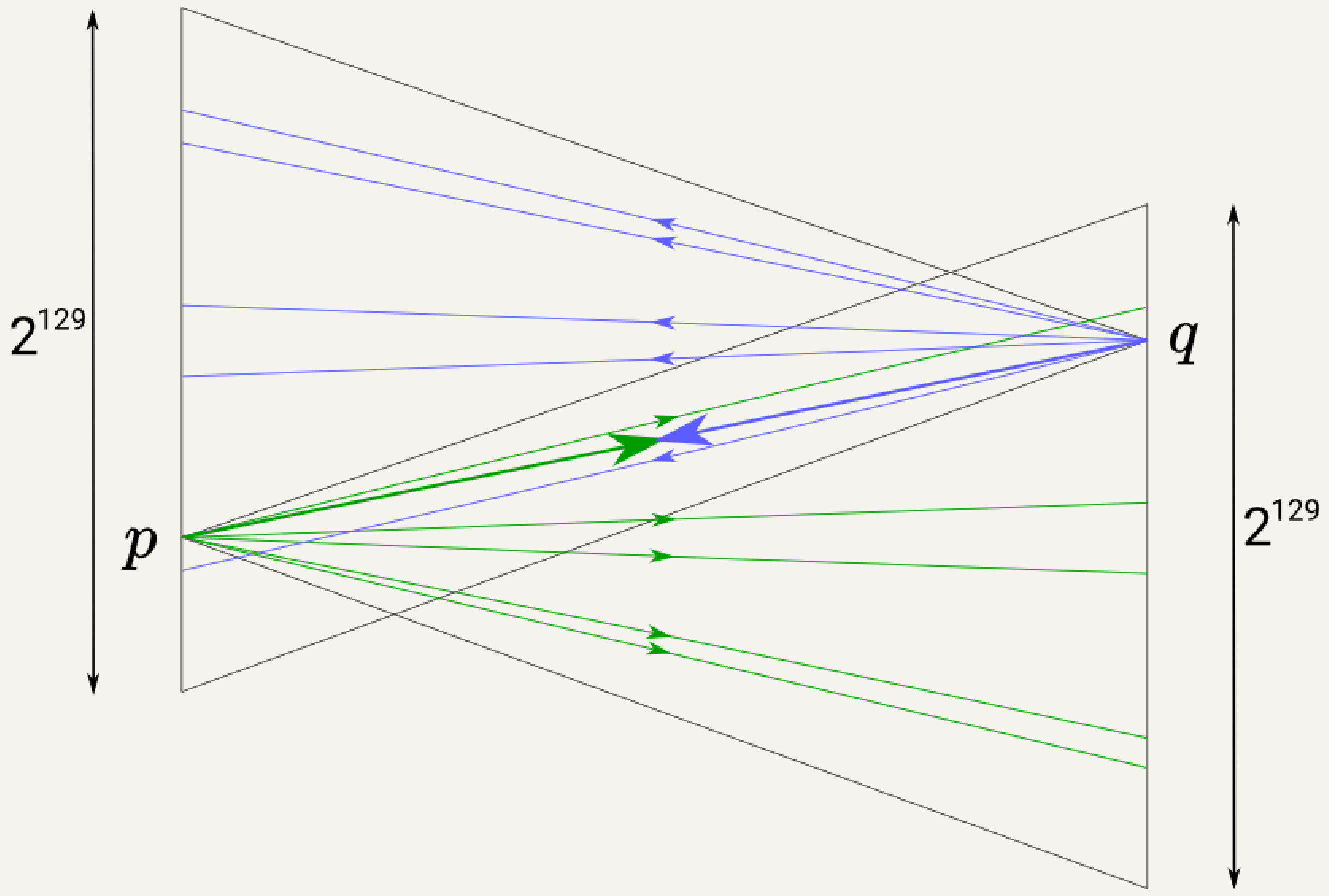
Constructing cycles – the problem

- By the Hasse bound, the order of an elliptic curve over $\mathbb{F}_p$ lies in the range $[\,p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}\,]$. For Tweedle this range is of size ~2^{129}.



Constructing cycles – the solution

- Suppose we were able to restrict the orders to a small number of possibilities, one of which was guaranteed to form a cycle...



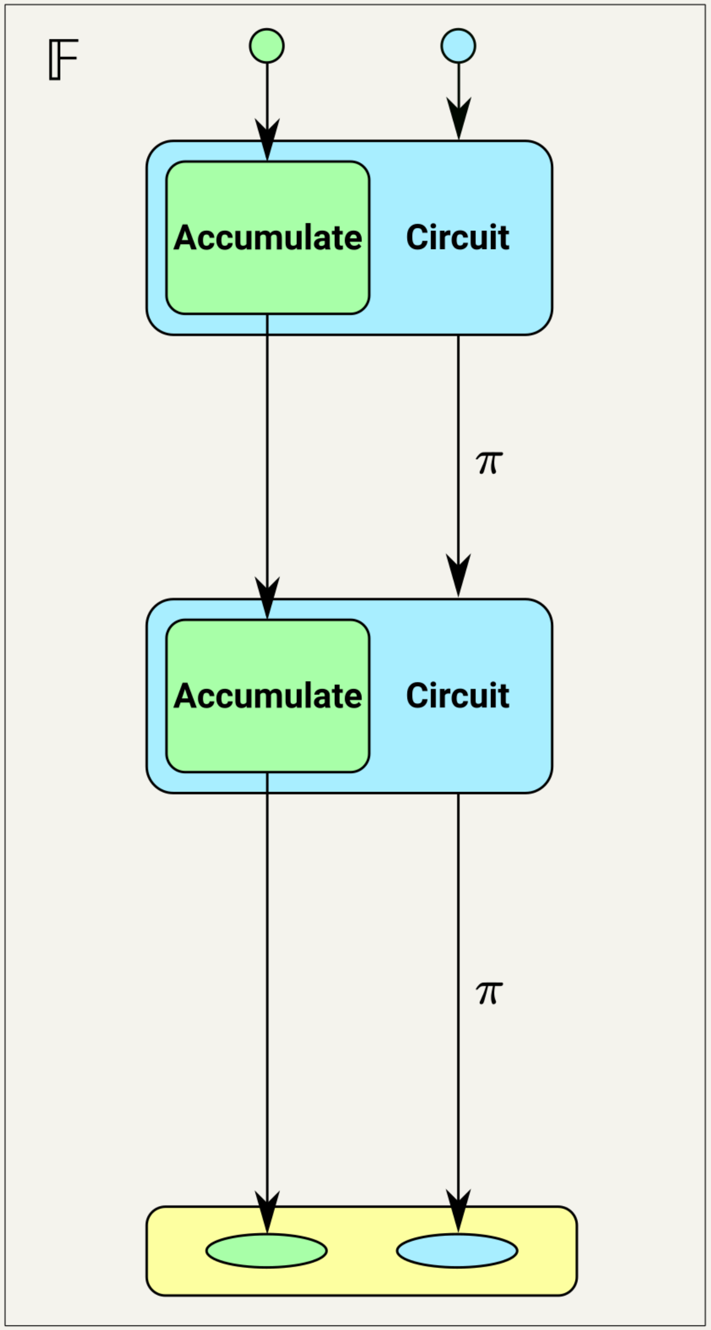
CM curves to the rescue

- Katherine Stange and Joe Silverman noticed that CM curves have precisely that property [SS2011].
- What is a CM curve? Watch my ZK Study Club presentation!
- Here, we just need to know that the subset of CM curves we use in Halo are ordinary curves over $\mathbb{F}_p$ with equation $y^2 = x^3 + b$.
- For a given p, these curves can only have 6 possible orders as b varies.
- And for any $E_p/\mathbb{F}_p$, one of the possible orders of $E_q/\mathbb{F}_q$ is guaranteed to form a cycle.

2-adicity

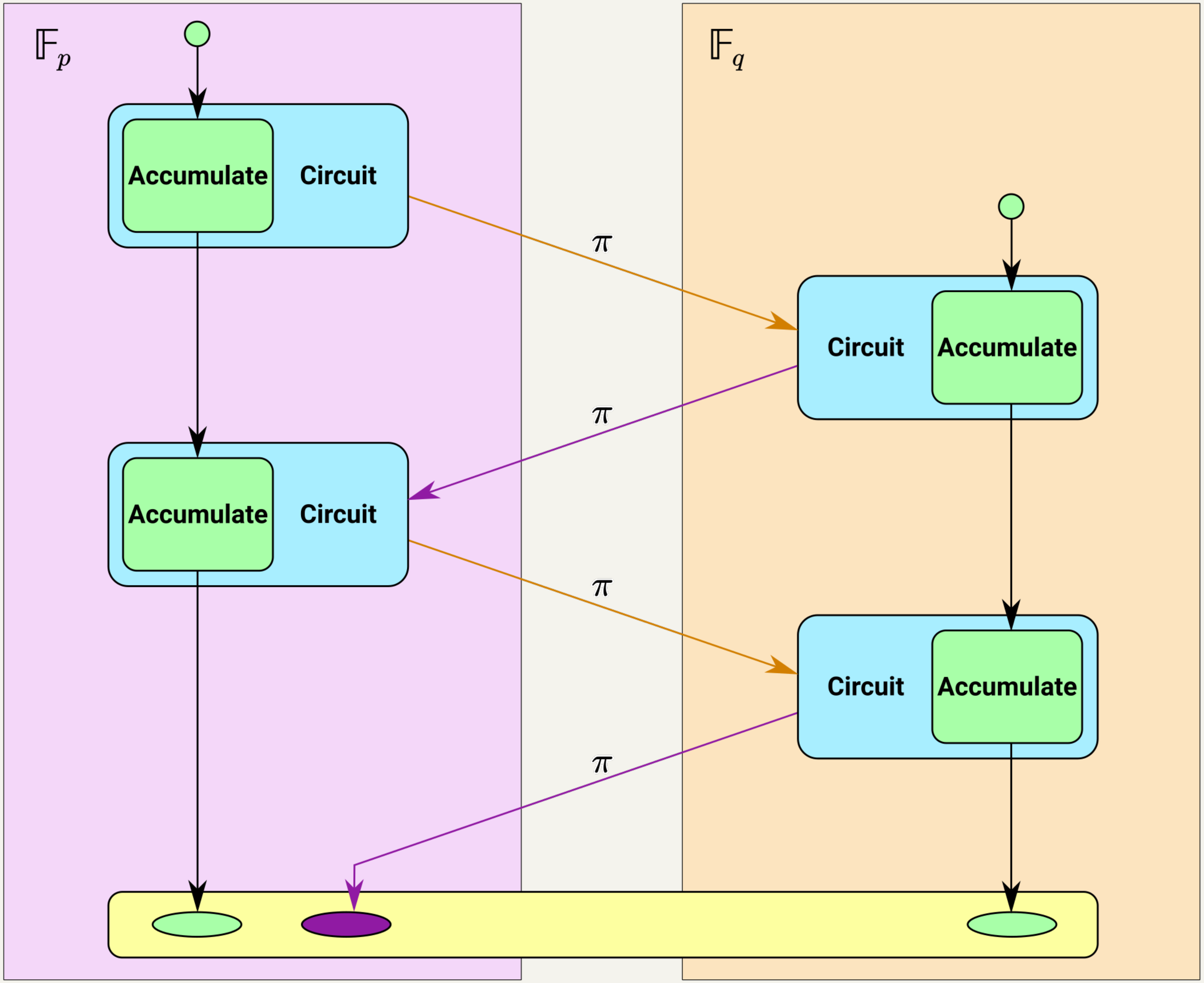
- Protocols that use Lagrange basis, or that need to efficiently multiply polynomials, benefit from $\mathbb{F}_{p,\,q}^{*}$ having a “large enough” multiplicative subgroup of size z^c. The simplest option is z = 2.
- In other words, we need $p \equiv 1$ and $q \equiv 1 \pmod{2^c}$.
- It turns out that the CM method allows us to find such curves almost for free.
- For the details see my ZK Study Club!

Accumulation schemes

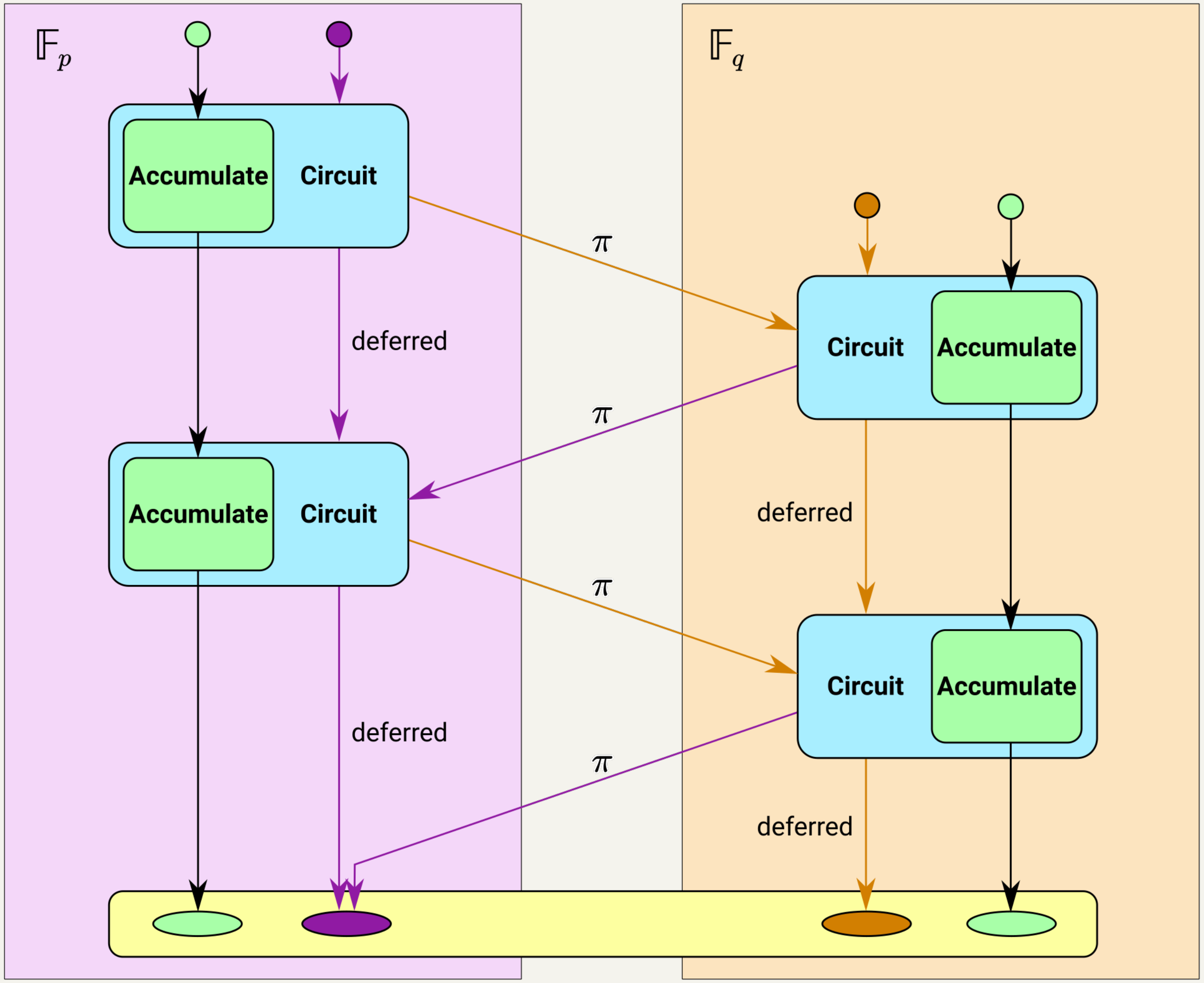


- In an accumulation scheme, it is expensive to “decide” validity of an instance, but cheap to combine two (or more) instances.
- This is how the idea is described in [BCMS2020] (ia.cr/2020/499).
- In practice, accumulation schemes and proof systems use arithmetic in a particular field.
- So if we tried to literally implement what is shown in the diagram, we’d have to do wrong-field arithmetic in the circuit.

Accumulation schemes on 2-cycles

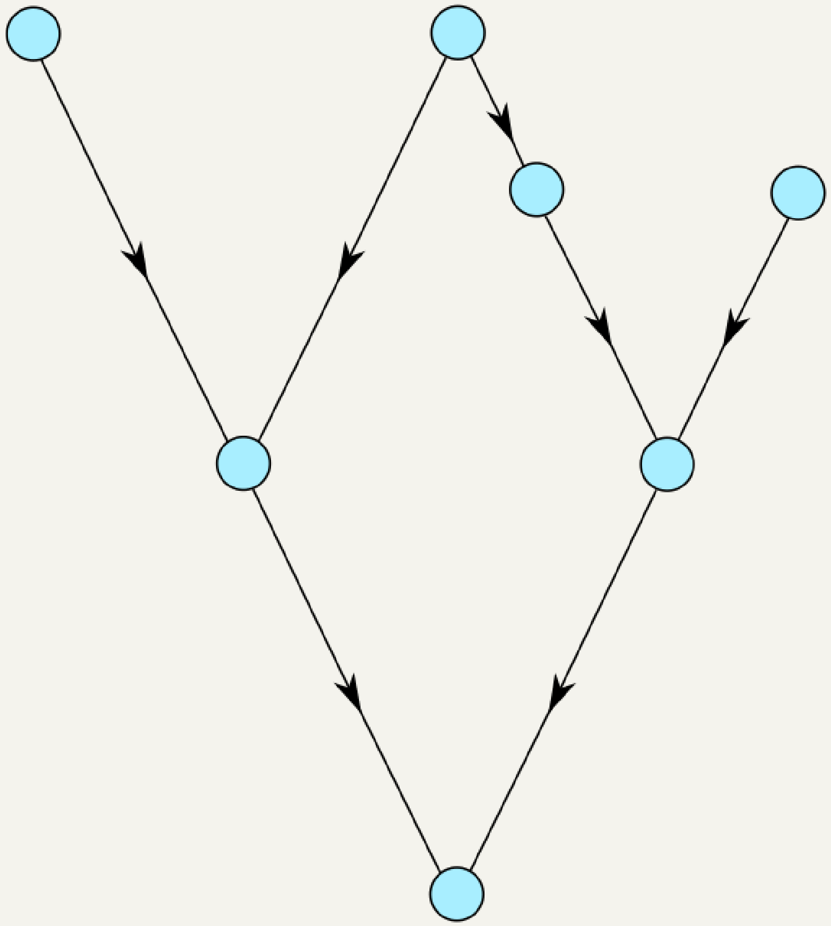


Accumulation schemes on 2-cycles



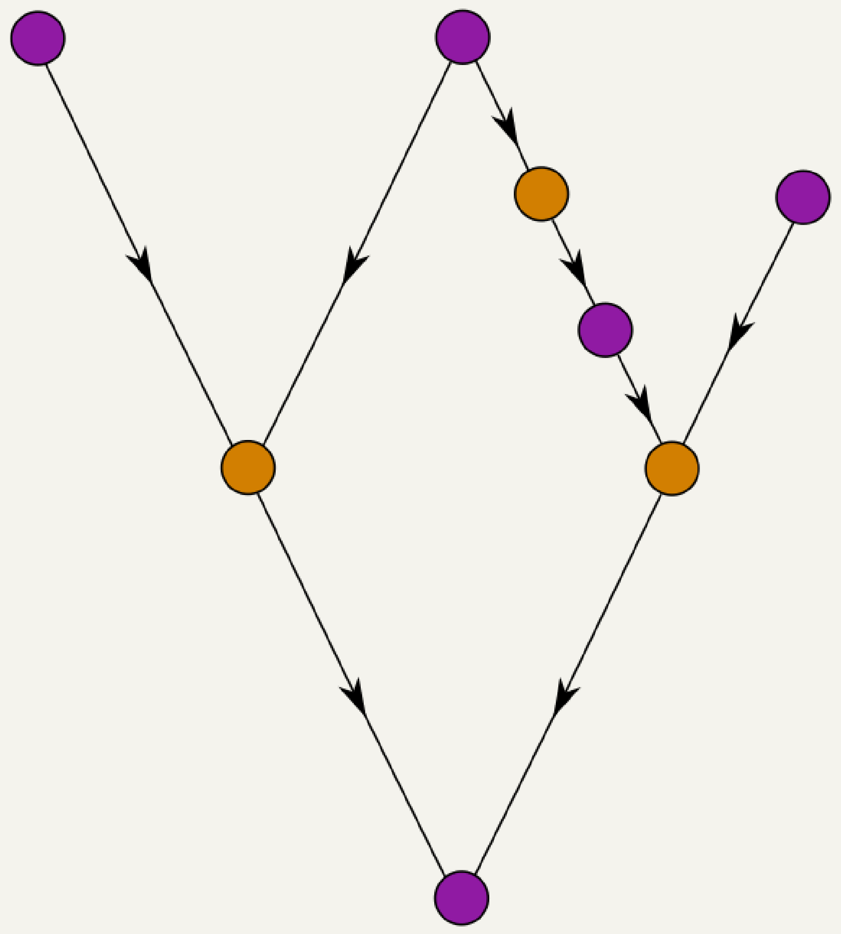
Encoding general Proof-Carrying Data

- Let’s consider how this goes for general PCD, where we have a directed acyclic graph of proofs.



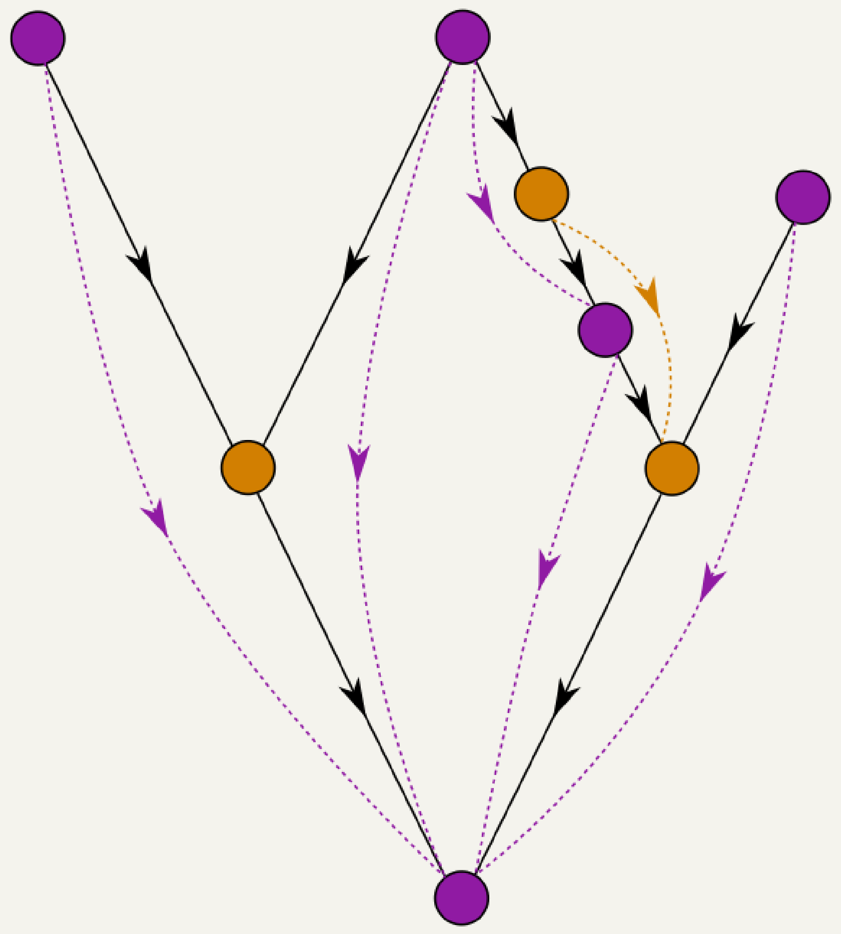
Encoding general Proof-Carrying Data

- We might need to add extra wrapper proofs so that $\mathbb{F}_p$ and $\mathbb{F}_q$ proofs alternate on all paths.



Encoding general Proof-Carrying Data

- We might need to add extra wrapper proofs so that $\mathbb{F}_p$ and $\mathbb{F}_q$ proofs alternate on all paths. Here we also show deferreds.



Arithmetizations

- Any generic proof system needs an “arithmetization”, which is a way of expressing statements as relations between variables.
- In R1CS, each constraint is $A \times B = C$, where A, B, and C are linear combinations.
- Halo 1 used the same arithmetization as Sonic and Bulletproofs. I’ll call it the “Sonic arithmetization”. It has:
  - multiplicative constraints $u \times v = w$, where each of u, v, and w are *fresh* variables;
  - linear constraints, $A = 0$ where A is a linear combination.
- In Halo 2, we use the same arithmetization as PLONK. It has:
  - polynomial gates, which can be customized for the operations used in the circuit;
  - copy constraints, which connect wires together.

The Halo verification circuit

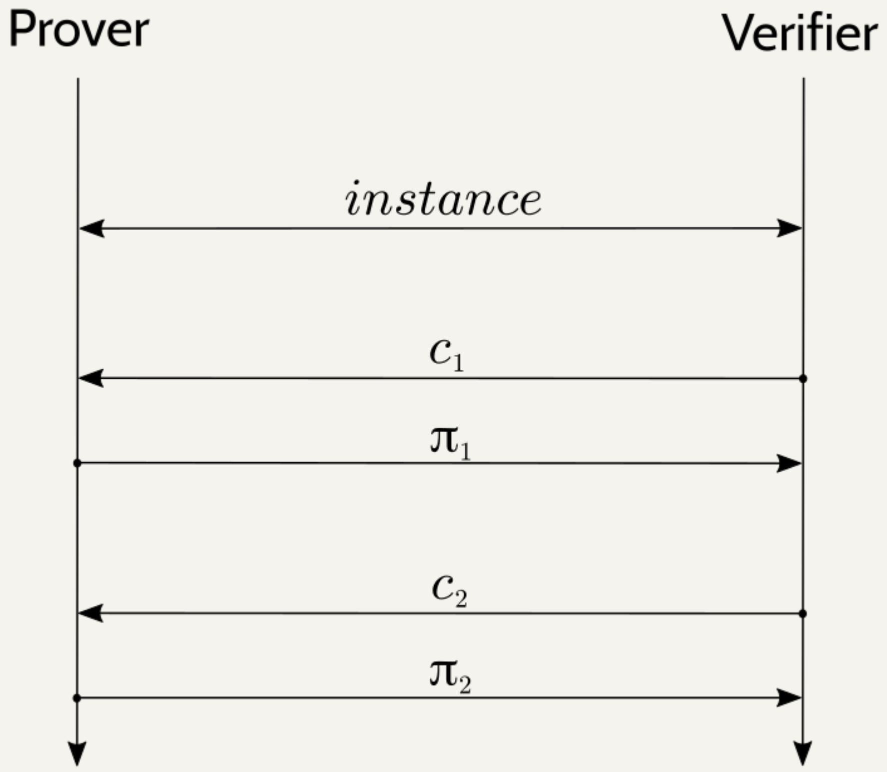
- Practical circuits tend to be dominated by a small number of operations that are repeated many times. In the Halo verification circuit, we need:
  - Multi-scalar multiplications
  - A hash function, to generate Fiat–Shamir challenges.
- By customizing the PLONK polynomial gates, we can obtain large efficiency gains for these operations.
- We also need to include the “application” circuit(s), and so any optimizations to the proof system should be made with that in mind.

Scalar multiplication in circuits

- Halo uses prime-order short Weierstrass curves, $E : y^2 = x^3 + b$.
- We start with a variation on the scalar multiplication algorithm in Zcash #3924, which takes 6 muls per scalar bit.
  - This uses an idea of Kirsten Eisentrager, Kristin Lauter, and Peter Montgomery. Instead of computing $[2]\ A + P$ directly in a double-and-add algorithm, we compute $(A \pm P) + A$.
- It turns out we can improve 6 muls/bit to 3.5 muls/bit, using the endomorphism ϕ we mentioned earlier.
  - An endomorphism is a short-cut to calculating a scalar multiplication.
- We only require the multiplications to be of some scalar with at least 128 bits of entropy (depending on the verifier challenges).
- The basic idea is to add one of $\{ -P,\ P,\ -\phi(P),\ \phi(P) \}$ at each step. This takes 7 muls for every 2 bits of entropy (the extra mul is to conditionally apply ϕ).
- This has the effect of multiplying by $\zeta a + b$, where a and b depend on random $\mathbf{r}$.
- Now we "just" need to prove that $\mathbf{r} \rightarrow \zeta a + b$ is 1:1, and that there are no exceptional cases for the additions (this takes 2½ pages of the Halo paper).

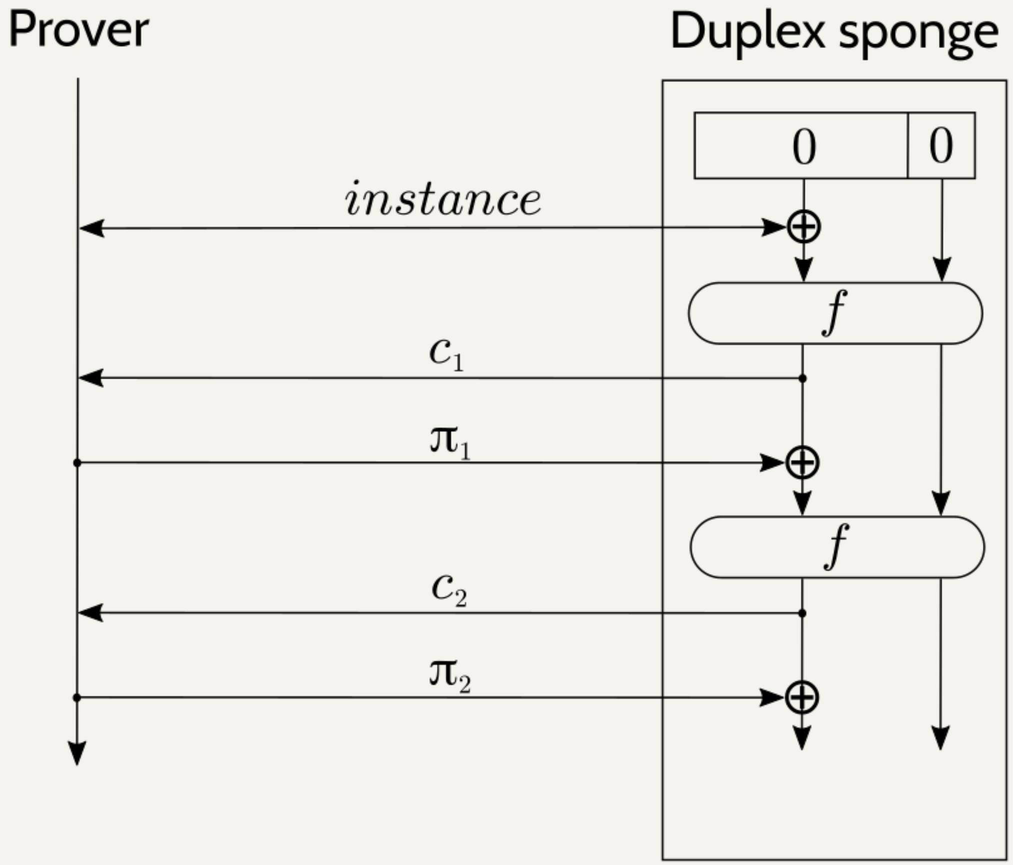
Fiat–Shamir and duplex sponges

- The Fiat–Shamir construction takes an interactive public-coin protocol, ...



Fiat–Shamir and duplex sponges

- The Fiat–Shamir construction takes an interactive public-coin protocol, and replaces the verifier with a hash function.



- Using a duplex sponge basically halves the number of f evaluations relative to other hash constructions.

Optimizations

- Use addition in the field for $\oplus$, rather than XOR.
- Compress the absorbed inputs.
- Pick a “rate” that is just large enough that we only need one f evaluation per round.
- To instantiate f in the duplex sponge, we need a permutation that is efficient in the circuit.
- Rescue is a permutation designed to be efficient in circuits over a prime field.
- Choose curves with gcd(p–1, 5) = 1 so that $x \mapsto x^5$ is a permutation.

Questions?

Daira Hopwood
@feministPLT
@daira#0512 on Discord
https://github.com/daira/halographs