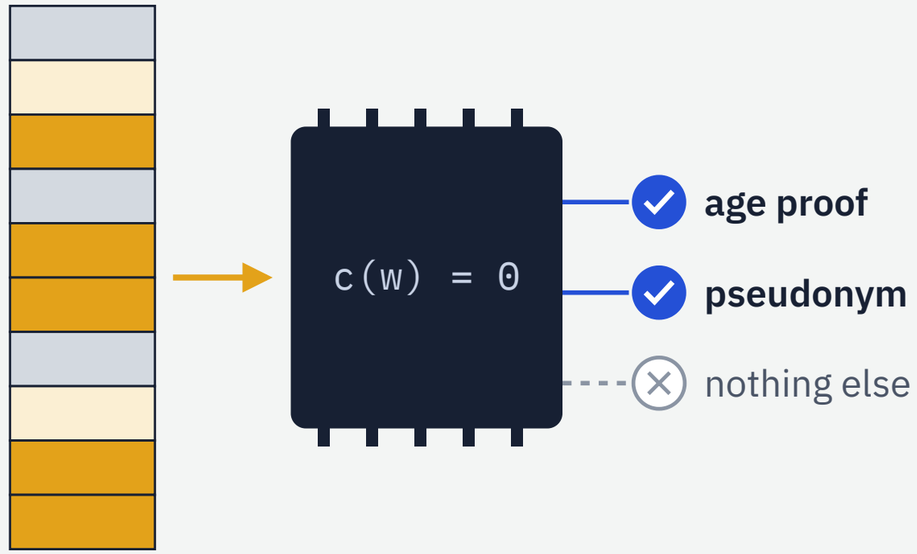
Learn Nothing

ZKP in Age Verification and Pseudonyms



Peter Altmann[1] **Frédéric Nieto**[1] **Gareth Oliver**[2]

[1] SIROS Foundation, {first name}@siros.org [2] Google

EUDI Wallets Launchpad 2026, Brussels

Background

Child safety duties

Keep adults out of spaces meant for minors (e.g., **13–17**)

Age-gate adult content at **18+**

Pseudonyms

Registration

- Fully anonymous accounts
- Bot prevention (proof of human)
- Known-offender matching

Authentication

- Recognise returning users

Account recovery

Services need different things, and some needs contradict each other.

Pseudonyms

One mechanism that can meet many different requirements.

Age

Any threshold or range.

Bound to the holder’s identity and device, so it cannot be shared.

Pseudonyms as property configurations

The relying party asks for a pseudonym with the properties that match its service needs.

Service C

Site-specific
A different value at each site

Known-offender matching
Enforce a ban

Issuer may unmask
For example, under a court order

Cannot be unmasked
Fully anonymous

Recoverable
Get a lost account back

Property N
...

Pseudonym generated by the wallet at presentation time

Site-specific | Known-offender matching | Issuer may unmask

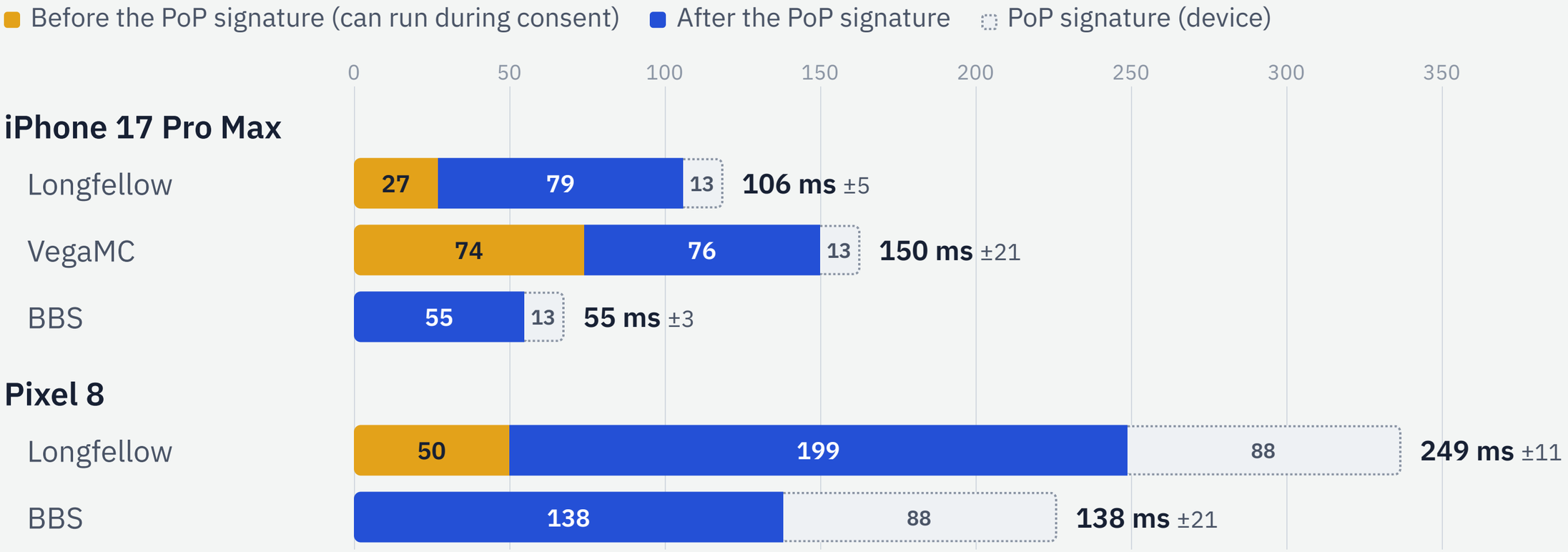
Generating pseudonym values

```
pseudonym = PRF(issuer_input, user_input, verifier_input)
```

`issuer_input`	From the identity attestation. When the issuer knows it, the issuer can remap the pseudonym to an identity. When it does not, no one can.
`user_input`	Full unlinkability, or an alias the user chooses.
`verifier_input`	The verifier's identity: site-specific values and rate limiting.

Prover benchmarks

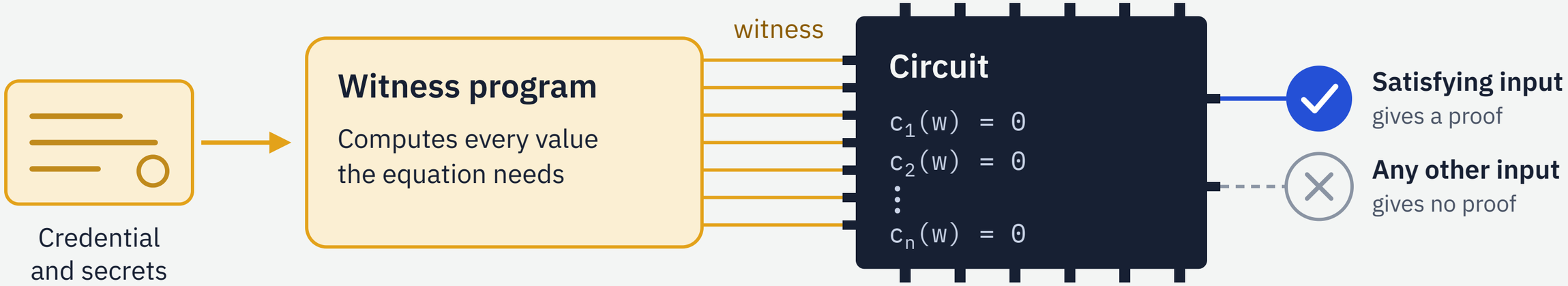
One proof: pseudonym generation, age verification, P-256 device binding (Secure Enclave / StrongBox)



Median prover time in ms, ± standard deviation. Totals exclude the PoP signature (dotted).

Deep dive into circuits

A circuit is one big equation



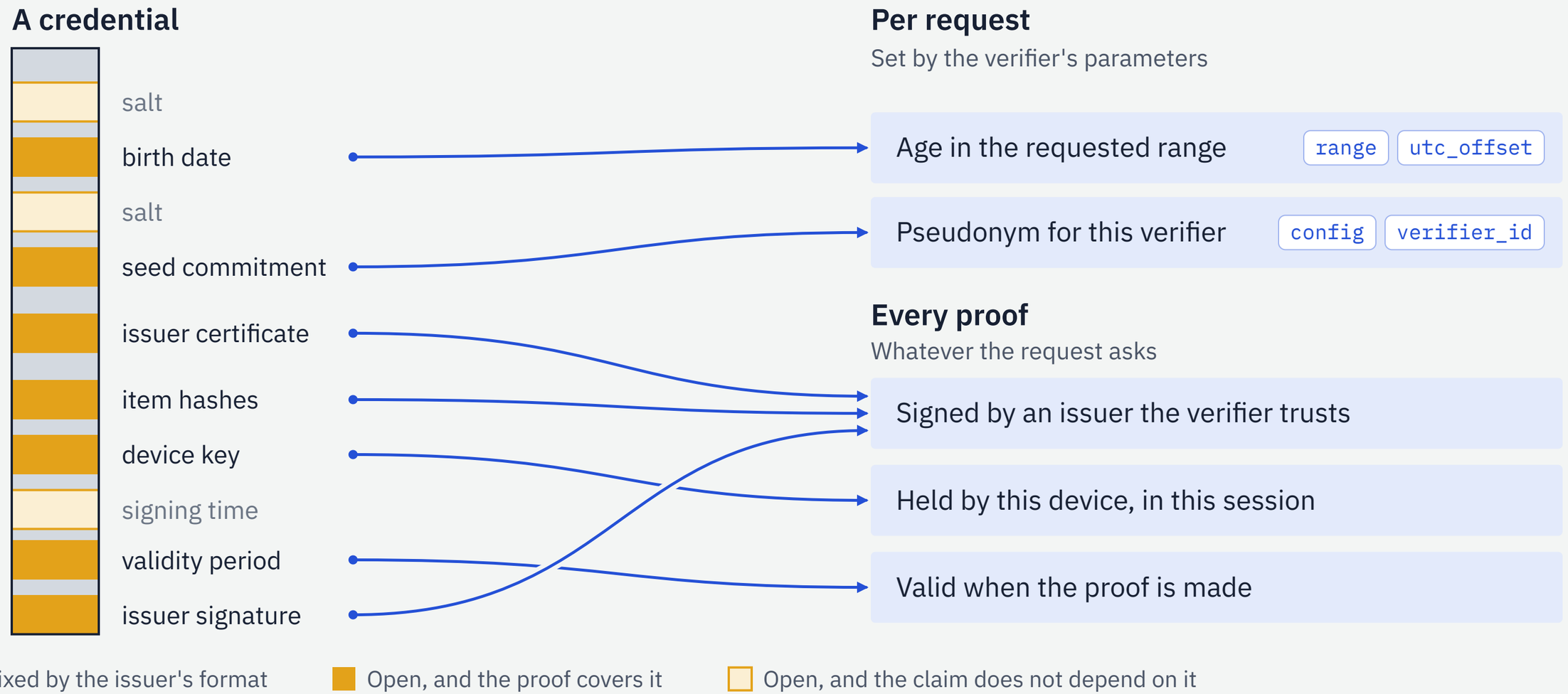
Example:
an age range

- **Program:** computes the birth date's distance to each bound.
- **Circuit:** checks that neither distance is negative.

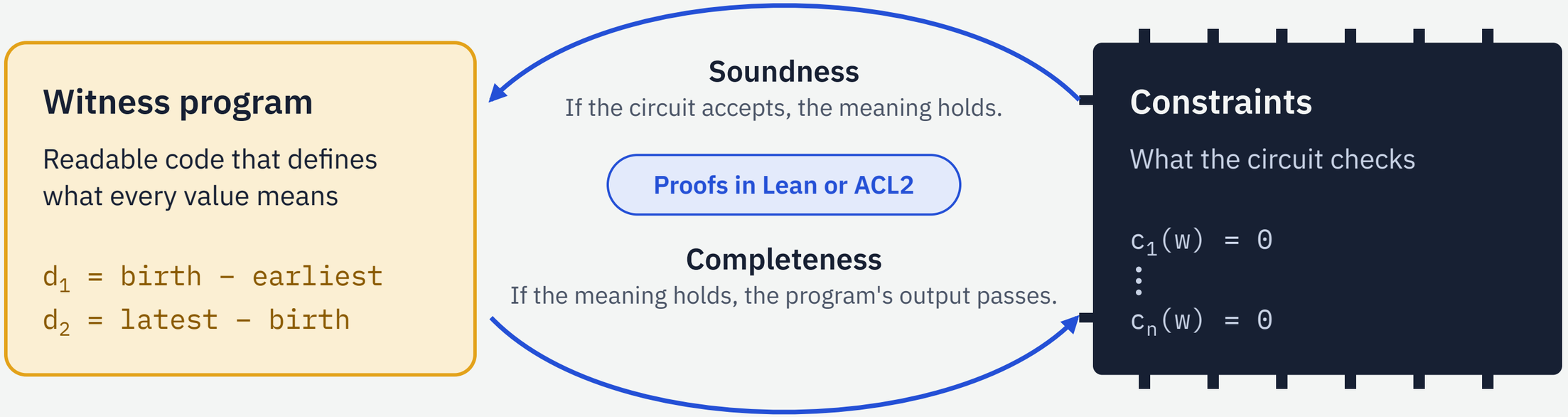
Example:
a pseudonym

- **Program:** computes SHA-256(DST ‖ seed ‖ prover_input ‖ verifier_input).
- **Circuit:** checks the seed opens the signed commitment and the hash is the pseudonym value.

What the proof covers



The witness program is the specification



Status: proofs for large parts of the circuit are ready, and we are closing the gaps.

Without this proof, a missing constraint does not cause an error, and the proof is erroneously verified.

One unified way to request

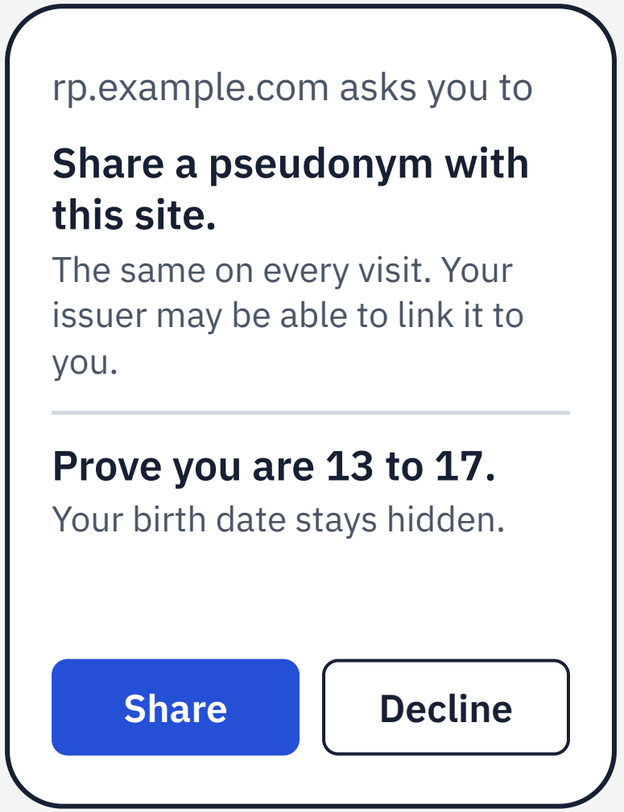
1 The verifier sends a request

```
"_req": [
  { "fn_id": "pseudonym",
    "return": "nym",
    "params": {
      "config": "known-offender-matching-ssc",
      "ver": 1,
      "prf": "hashdh-bls12381g1" } },
  { "fn_id": "age_check",
    "params": { "range": "13-17" } }
]
```

fn_id and **params** in every call

return only for non-assertions

2 The wallet asks for consent



3 The wallet proves it



Open challenges and paths forward

Standardise beyond the ZK formats

Request and response changes
For example, zkp_req in OpenID4VP

Issuance augmentation

What we learned shipping it

Easy	Requesting, verifying, the protocol layer
Harder	Keeping provers updatable and performant on native
Necessary	Interoperable, robust ZK prover implementations

Also in progress: every constraint machine-checked, and more pseudonym configurations.

Questions

Peter Altmann[1] **Frédéric Nieto**[1] **Gareth Oliver**[2]

[1] SIROS Foundation, {first name}@siros.org [2] Google

Try the demo

zkp-demo-verifier.fly.dev