libZK: a zero knowledge proof library
(The Longfellow ZK scheme)

Matteo Frigo, abhi shelat

∃ Desire for ZK schemes for “Small” Identity theorems

7.4.3.5.3 Zero-Knowledge Proofs

“Attestation Provider linkability cannot be fully eliminated when using attestation formats based on salted hashes. The only viable mitigation is to adopt Zero-Knowledge Proofs (ZKPs) as a verification mechanism instead of relying on salted-attribute hashes.”

…

“One key area of development is age verification, where the European Commission is actively exploring and testing ZKP-based solutions. The outcomes of this initiative could pave the way for the adoption of ZKPs within the EUDI Wallet ecosystem, further strengthening privacy protections in future implementations.”

https://eu-digital-identity-wallet.github.io/eudi-doc-architecture-and-reference-framework/2.4.0/architecture-and-reference-framework-main/#74353-zero-knowledge-proofs

7.1 Zero-Knowledge Proofs

A next version of the Technical Specifications for Age Verification Solutions will include as an experimental feature the Zero-Knowledge Proof (ZKP) solution described in the paper *"Matteo Frigo and abhi shelat, Anonymous credentials from ECDSA, Cryptology ePrint Archive, Paper 2024/2010, 2024, available at https://eprint.iacr.org/2024/2010"*. This zkSNARK-based solution was selected through a process where several alternatives were evaluated. This solution requires no changes to the attestation issuance process or to the structure of the Proof of Age credential itself. This backward compatibility allows AVIs to gracefully fall back to traditional protocols in environments where ZKPs are not supported.

https://ageverification.dev/Technical%20Specification/architecture-and-technical-specifications/#71-zero-knowledge-proofs

∃ Desire for ZK schemes in IETF

- SD-JWT/Oauth

“anecdotally I do think there is (or would be) general interest in [ZK] techniques [that are pq-secure] from a nontrivial number of folks in the community. So yes, I'd be supportive of such a statement as long as it was properly qualified as such.”

∃ Desire for PQ ZK schemes

Feedback from both the EU and organizations (e.g., Apple, Google) indicates a desire to *avoid* deploying new ECC-based schemes.

The preference is to choose PQ schemes when possible.

Other requirements such as *device-binding* and *legacy support* rule out schemes such as the BBS family.

There is a strong market need to specify PQ ZK schemes for these applications.

The IETF CFRG is one logical place.

Q: is there interest in doing this work here?

Longfellow ZK scheme

- Follows the oldest ZK "recipe" from **BGGHKMR88**

Run an IP ⟶ Transcript.
Commit to Transcript ⟶ Com.
ZK for "Com contains T and IP-verifier(T)=1"

IP: Sumcheck (Prover runs in O(C) time.)

Commit/ZK: Ligero (IOP, only uses SHA256.)

Longfellow ZK is Robust enough for practice

Longfellow handles a statement like this one for ISO MDOC ZK presentment.

< 1s on a phone.

Given the public values (PK_II, “age_over_18”, transcript, time_now), there exists

a 2231 byte string MDL, a hash e_1, a hash h_2, an index X,
a signature sig_1, …, a 32-b nonce, a pk DPK,
a pair of strings time_start, time_end, and a signature sig_2

such that:

```
e_1 = SHA-256(MDL)                                                  and
p256.verify( sig_1, e_1, PK_II ) = true                             and

h_2 = MDL[valueDigests][org.iso.18013.5.1][‘age_over_18’]]          and
h_2 = SHA-256(nonce, ‘age_over_18’, ‘True’)                         and

DPK = MDL[deviceKeyInfo][deviceKey][-2, -3]                         and
p256.verify( sig_2, transcript, DPK )    = true                     and

time_start = MDL[validityInfo][validFrom]                           and
time_end = MDL[validityInfo][validUntil]                            and
time_start < time_now < time_end                                    and
```

Longfellow ZK scheme

- eprint/2024/2010
- Open source
- External security reviews underway
- High quality deployment has been verified at Google & OWF.
  - Code size constraints
  - Memory use constraints
  - efficiency-core CPU constraints
  - Android + iOS
- Best proof/work for “small theorems”

Risks can be systematically addressed

- Proximity gap analysis for soundness is active. Simple proofs easy, state-of-the-art is subtle.

- How can we trust circuits?
  - Formal methods are underway
  - Proposal: Circuits certified in the application WG, scheme certified in CFRG.

- Code complexity

Code Complexity

Ligero + Merkle

Language	files	comment	code
Header	6	252	851
C++	2	11	275
SUM:	8	263	1126

Sumcheck

Language	files	comment	code
Header	8	159	795
C++	2	45	342
SUM:	10	204	1137

ZK

Language	comment	code
Header	232	806
C++	27	292
SUM:	259	1098

bls12_381: 11,299 code lines, 1401 comments
Whir + merkle: 3385 lines; whir/sumcheck: 916 lines

P convinces V that “there exists w such that [illegible]”

Prover [illegible]

Verifier [illegible]

Pick a random pad.
Ligero-Commit(w, pad)

com →

Run Sumcheck-P(C,x,w) ⟷ Run Sumcheck-V(C,x)

When Sumcheck sends message m_i, send m_i + pad_i.
Record transcript T.

…

Send random challenges.
Record transcript T

Run Ligero-Prover(com, Enc-Sumcheck-Verify(T,x,pad)) →

Ligero-Verify(com, Enc-Sumcheck-Verify,T,x,proof)

Hyrax