Standards for Zero-Knowledge Proofs and their Relevance to the NIST Threshold Call

Mary Maller, Ethereum Foundation and PQShield

MPTS 2023: NIST Workshop on Multi-party Threshold Schemes 2023 - 27 Sept 2023
What is a Zero-Knowledge Proof

The digital language of truth

- Everything I say in zero-knowledge is true.
- I can choose to say nothing at all.
- Everything I do not say is perfectly hidden.
What is a Zero-Knowledge Proof

The digital language of truth

• Everything I say in zero-knowledge is true.
• I can choose to say nothing at all.
• Everything I do not say is perfectly hidden.

Privacy
Building Blocks
Scalability
What is a Zero-Knowledge Proof

The digital language of truth

- Everything I say in zero-knowledge is true.
- I can choose to say nothing at all.
- Everything I do not say is perfectly hidden.

Privacy

Building Blocks

Scalability

Companies usually resort to trusted hardware. Zero-knowledge would be better solution but is currently viewed as experimental technology.
# ZKPs or Trusted Hardware?

<table>
<thead>
<tr>
<th></th>
<th>ZKPs</th>
<th>Trusted Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacks Detectable</td>
<td>Nope</td>
<td>Nope</td>
</tr>
<tr>
<td>No Trusted Third Party</td>
<td>Yes</td>
<td>Nope</td>
</tr>
<tr>
<td>Easy Bug Fixes</td>
<td>Sometimes</td>
<td>Nope</td>
</tr>
<tr>
<td>Safe from Side Channels</td>
<td>Sometimes</td>
<td>Nope</td>
</tr>
<tr>
<td>Open Source</td>
<td>Yes</td>
<td>Nope</td>
</tr>
<tr>
<td>Widely Used</td>
<td>Nope</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Disclaimer: I may be a little biased here.
Many of today’s engineering efforts are targeting Scalable Blockchains.
Today I am focusing on ZKPs in the context of MPC;

Outside academia, industries, governments and NIST are thinking about advanced cryptographic primitives;

Many cryptographic primitives rely on zero-knowledge.
Multiparty Computation (MPC)

- 3 parties can compute the output of a function.
- 2 parties cannot compute the output.
Threshold Signatures = MPC Special Case

- 3 parties can compute the output of a function.
- 2 parties cannot
Threshold Decryption = MPC Special Case

- 3 parties can compute the output of a function.
- 2 parties cannot
Desirable Properties for Threshold Schemes

- Easy Key Generation
- Active Security
- Identifiable abort
Desirable Properties for Threshold Schemes

- **Trusted key generation**: Sometimes acceptable and sometimes not acceptable.

- **Distributed key generation**: Usually tricky.

- **ZKPs**: Multisignatures with Proof of Possession of secret key are usually easy.
Desirable Properties for Threshold Schemes

- **Passive security:** Output is correct and private when all parties follow the protocol.
- **Active security:** Output is correct and private even if parties behave badly.
- **ZKPs:** If all parties prove honest behaviour in zk in a passively secure scheme, then output is actively secure.
Desirable Properties for Threshold Schemes

- **Liveness**: Want protocol to terminate. If doesn't terminate want to know why.

- **ZKPs**: If all parties prove honest behaviour in zk in a passively secure scheme, then aborts can only be caused by not saying anything at all.
• **Table 12 (Page 53)**: Explicitly expressed interest in zero-knowledge proofs of knowledge of secret key for a selection of schemes.

• **Option 1**: Give special purpose proving scheme for each of the relations.

• **Option 2**: Give general purpose proving scheme and just specialise the constraints.

<table>
<thead>
<tr>
<th>Related type</th>
<th>Related (sub)sub-category: Primitive</th>
<th>Example ZKPoK (including consistency with public commitments of secret-shares, when applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keygen</td>
<td>C1.5.1: ECC keygen</td>
<td>of discrete-log ((s \text{ or } d)) of pub key (Q)</td>
</tr>
<tr>
<td></td>
<td>C1.5.2: RSA keygen</td>
<td>of factors ((p, q)), or group order (\phi), or decryption key (d)</td>
</tr>
<tr>
<td></td>
<td>C1.5.3: AES keygen</td>
<td>of secret key (k) (with regard to secret-sharing commitments)</td>
</tr>
<tr>
<td>PKE</td>
<td>C1.2.1: RSA encryption</td>
<td>of secret plaintext (m) (encrypted)</td>
</tr>
<tr>
<td></td>
<td>C1.2.2: RSA decryption</td>
<td>of secret-shared plaintext (m) (after SSO-threshold decryption)</td>
</tr>
<tr>
<td>Symmetric</td>
<td>C1.4.1: AES enciphering</td>
<td>of secret key (k) (with regard to plaintext/ciphertext pair)</td>
</tr>
<tr>
<td></td>
<td>C1.4.2: Hashing in KDM</td>
<td>of secret pre-image (Z)</td>
</tr>
</tbody>
</table>
Special Purpose

• Fast: No need to do arithmetisation.

• More work: Standardisation process will only be useful for one primitive.

General Purpose

• Slow: Pay full cost of arithmetisation.

• Less work: Standardisation process is useful for all ZKP applications.

But general purpose ZKPs are now fast enough that we can afford it.
ZK Implementations are Becoming Fast

### RISC Zero Datasheet

**Example** | **Cycles**
---|---
Factors | 32 k
Chess | 256 k
Digital Signature | 64 k
EVM | 2048 k
JSON | 64 k
Password Checker | 64 k
SHA | 64 k
Waldo | 6192 k
Worde | 64 k

<table>
<thead>
<tr>
<th>Cycles</th>
<th>Prover Time</th>
<th>RAM</th>
<th>Proof Size</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 k</td>
<td>1.38 s</td>
<td>234.4 MB</td>
<td>201.3 kB</td>
<td>23.7 kHz</td>
</tr>
<tr>
<td>64 k</td>
<td>1.87 s</td>
<td>468.7 MB</td>
<td>213 kB</td>
<td>35 kHz</td>
</tr>
<tr>
<td>128 k</td>
<td>2.80 s</td>
<td>937.4 MB</td>
<td>236 kB</td>
<td>46.9 kHz</td>
</tr>
<tr>
<td>256 k</td>
<td>4.97 s</td>
<td>1.87 GB</td>
<td>247.7 kB</td>
<td>52.7 kHz</td>
</tr>
<tr>
<td>512 k</td>
<td>9.49 s</td>
<td>3.75 GB</td>
<td>259.9 kB</td>
<td>55.3 kHz</td>
</tr>
<tr>
<td>1024 k</td>
<td>17.96 s</td>
<td>7.5 GB</td>
<td>273.2 kB</td>
<td>58.4 kHz</td>
</tr>
<tr>
<td>2048 k</td>
<td>50.13 s</td>
<td>15 GB</td>
<td>297.8 kB</td>
<td>41.8 kHz</td>
</tr>
<tr>
<td>4096 k</td>
<td>1:51.2</td>
<td>30 GB</td>
<td>311.1 kB</td>
<td>37.7 kHz</td>
</tr>
</tbody>
</table>

### Benchmarking ZKP Development Frameworks: the Pantheon of ZKP

*Figure 2. Proof Generation Time for SHA-256 (MBP M1)*

Mo Dong

2nd March 2023
ZKProof Standardisation Effort

- Global movement to standardise and mainstream advanced cryptography by building a community-driven trust ecosystem.
- Formed in 2018 after top researchers and developers saw technology becoming advanced enough for standards.
- I joined the editorial team in 2021.
- We expect this to be a long process as the community jointly learn best practices.
ZKProof Standardisation Effort

- Most ZKPs are formalised only in research papers.
- Research paper != formal specification suitable for deployment.
- Collaboration of industry developers and academics are in the process of writing specifications for a full general purpose proving system.
- This is a lot of work.
- Hopeful that if we can pull it off, then it should be directly applicable to proofs of possession and other threshold related applications.
Final Remarks

• Easier to get support for specification drafts with formal industry support for the application.

• If you are seriously considering using ZKPs in your threshold application then we would love to hear from you.

• Contact us at contact@zkproof.org