Collaborative zkSNARKs

Zero-knowledge proofs for distributed secrets

Alex Ozdemir, Dan Boneh
To appear in USENIX Security’22
Provable properties about secrets

Authentication ✅ zkSNARKs

I prove that I know my password/secret key.
Provable properties about *distributed* secrets

**Money Laundering**
Several banks prove that “Eve” has Colonial Pipeline’s ransom.

**Healthcare Statistics**
Several hospitals prove that procedure prices are “fair”.
This talk

Applications:
• Authentication ✓ zkSNARKs
• Money laundering ❓
• Healthcare statistics ❓

Outline:
1. Why zkSNARKs are insufficient
2. New tool: collaborative zkSNARK
3. Building collaborative zkSNARKs
4. Surprising efficiency
Background

zkSNARKs
Witness relations

\[ P(x, \omega) \quad \cup (x) \quad \exists \omega. (x, \omega) \in R \]

- \( w \) may be large
- not private
zkSNARKs

\[
\begin{align*}
\text{pp} &\leftarrow \text{Setup}() \\
\exists? \omega. (x, \omega) &\in R \\
P(x, \omega) &\leftarrow \text{Prove}(\text{pp}, x, \omega) \\
U(x) &\leftarrow \text{Verify}(\text{pp}, x, \pi) \\
\{0, 1\} &\leftarrow \text{verify} \left( x, \omega, \pi \right)
\end{align*}
\]

- Sound: \(\pi\) proves \(w\) exists
- (zk) zero knowledge: hides \(w\)
- (S) succinct: short \(\pi\), fast Verify
- (N) non-interactive
- (AR) argument: \textit{computationally} sound
- (K) knowledge: \(P\) knows \(w\)
Authentication with a zkSNARK
Existing zkSNARKs

• From pairings & elliptic curves
  • Groth16
  • Marlin (KZG)
  • Plonk (KZG)
  • …
• From hashing & codes
  • Fractal
  • …
  • …

For distributed secret data, who plays the prover?
Collaborative zkSNARKs

Definitions
Collaborative zkSNARKs

Syntax:

- Setup() → pp
- Prove(pp, x, P_1(w_1), ..., P_N(w_N)) → π
- Verify(pp, x, π) → {0, 1}
t-zero-knowledge

Any adversary controlling \( \leq t \) provers learns nothing but whether
\[(x, \overrightarrow{w}) \in R\]

• Formally
  • Adversary corrupts \( \leq t \) provers
  • ZK simulator is given
    • the corrupt witnesses
    • \( x \)
    • \( b = (x, \overrightarrow{w}) \in R \)
  • We use the random oracle model
Knowledge soundness

• *could* mean $P_1$ knows $w_1$, ..., $P_N$ knows $w_N$
• actually means $P_1$, ..., $P_N$ collectively know $w_1$, ..., $w_N$
  • “distributed knowledge” [Halpern, Moses ’90]

• Random oracle
  • extractor programs RO
Our focus: secret-shared R1CS witnesses

R1CS:

• Class of relations
• Generalize arithmetic circuits
• Definition:
  • $R: A, B, C \in \mathbb{F}^{n \times m}$
  • $x \in \mathbb{F}^k, w \in \mathbb{F}^{m-k}$
  • Satisfied when
    • $a \leftarrow w \parallel x$
    • $Aa \circ Ba = Ca$

\[ A, B, C \xrightarrow{\text{Setup}_{R1CS}} [w], x \xrightarrow{\text{Prove}_{R1CS}} \pi \]

secret-shared among provers
Designing co-zkSNARKs

Overview of constructions
Approach: MPC the Prover

GenericMPC(zkSNARK.Prove, [w]) → π

1000x slower 1000x slower

1,000,000x slowdown ?!

avoid. instead achieve 1000x - 2000x slowdown
Potential Bottlenecks

Single-prover bottlenecks:
• Elliptic curve operations
• Fourier transforms

MPC bottlenecks:
• Polynomial divisions
• Partial products
• Merkle tree evaluations

- This talk: a good solution
- MPC-efficient
- MPC-efficient (for SNARK provers)
- special protocol
- This talk: an okay solution
MPC Crash-Course

Computation: arithmetic circuit over a finite field

1. Secret-share wire values among N parties
   
   e.g. \( x = x^{(1)} + x^{(2)} + \ldots + x^{(N)} \)

2. Secure protocols for +, * on shares

3. Evaluate circuit, inputs to outputs

We use two MPCs: SPDZ (authenticated additive shares, malicious majority) and GSZ (Shamir shares, honest majority)
MPC-friendly elliptic curve arithmetic

Elliptic curve addition

\[ g_1 = (x_1, y_1) \]
\[ g_2 = (x_2, y_2) \in E(F) \]

Option 1: Share \((x, y)\) coordinates

\[ [g_1] \oplus [g_2] = (x, y) \]
\[ (x_1, y_1) \oplus (x_2, y_2) \rightarrow y = y_1y_2 + y_1 + y_2 \]

Nasty Formulas

Expensive
- multiple multiplications
- communication

Option 2: Elliptic curve sharing

\[ [g] = g^{(1)} \oplus g^{(2)} \oplus \ldots \oplus g^{(N)} \rightarrow \text{add share-wise} \]

Efficient
- no communication
Merkle Trees

non-linear and expensive

Idea (Riad S. Wahby): reduce with proof composition

proof size (& verifier work) \( \times N \)
Implementation
Implementation Goals

• Three base zkSNARKs
  • Groth16
  • Marlin/KZG
  • Plonk/KZG

• Two base MPCs
  • GSZ: $t < N/2$ (honest majority)
  • SPDZ: $t < N$ (malicious majority)

• Goals:
  • Compete with existing zkSNARKs
    • (well optimized!)
  • Iterate on MPCs, sub-protocols
  • Don’t work too hard
An Opportunity

1. Arkworks has curve-generic provers:

   fn prove\(<E: \text{PairingEngine}>\)(..) {
     ...
   }

2. Curve interfaces define +, *, ...

   trait PairingEngine {
     type ScalarField;
     type Curve;
     fn field_add(\(...\) \);
     fn field_mul(\(...\) \);
     fn curve_add(\(...\) \);
     ...
   }

*Radically oversimplified*
Implementation Strategy

1. Implement MPCs for shared field and curve operations
   1. SDPZ
   2. GSZ
2. Wrap MPCs & implement arkworks interfaces
3. Instantiate zkSNARK prover
   ➢ Mis-appropriates zkSNARK prover as a co-zkSNARK prover!
Performance
Experimental Setup

Measure:
• Wall-clock proving time

Vary:
• $N$: number of provers
• $n$: R1CS size (# constraints)
• $c$: link capacity
• Base: Groth16/Marlin/Plonk
• $t$: security threshold
  • $< N/2$ (honest majority, GSZ)
  • $< N$ (malicious majority, SPDZ)

Simplifications:
• No intra-prover parallelism
• Skip MPC preprocessing
  • Small for our computation
Experiment 1: Good network, few parties

Fix a 3Gb/s link, vary # rank-1 constraints

\[ t < \frac{N}{2} \rightarrow \text{no slowdown} \]
\[ t < N \rightarrow 2x \text{ slowdown} \]
Experiment 2: Many provers

Fix 1024 constraints, 3Gb/s link, Groth16, vary # of provers

Slowdown grows with N; better for SPDZ
Experiment 3: Low-capacity link

Fix 1024 constraints, 2 provers, malicious majority (SPDZ)

Slowdowns grow, but far better than 1000x
Discussion, Future Work

• Bandwidth is the bottleneck for many provers, low link capacity
  • **Bad news**: 2-prover co-zkSNARK (additive sharing) → $\Omega(n)$ communication
    • From randomized 2-party communication complexity of DISJOINT
    • Conjecture: $\Omega(\lambda n)$ (needed: generalize DISJOINT from $\{0,1\}$ to $\mathbb{F}$)

• Exploit intra-prover parallelism

• A nicer post-quantum co-zkSNARK with $o(N)$ proof-size?
Collaborative zkSNARKs

Alex Ozdemir, Dan Boneh

Conclusions:

1. **Collaborative zkSNARKs** support distributed secrets
   - Multiple users, hospitals, banks, ...

2. Very efficient
   - $(N/2)$-ZK → no slowdown
   - $(N - 1)$-ZK → 2x slowdown

3. Far better than MPC for typical computations → ~1000x slowdown

Groth16 co-zkSNARK proving time

![Graph showing Groth16 co-zkSNARK proving time](https://github.com/alex-ozdemir/multiprover-snark)

## Code

<table>
<thead>
<tr>
<th>Component</th>
<th>Lines (Rust)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Library</td>
<td>~700</td>
</tr>
<tr>
<td>Arkworks adapters</td>
<td>~2000</td>
</tr>
<tr>
<td>MPC protocols</td>
<td>~3000</td>
</tr>
<tr>
<td>Plonk</td>
<td>~1200</td>
</tr>
</tbody>
</table>