zkSaaS

Zero-Knowledge SNARKs as a Service

Sanjam Garg    Aarushi Goel    Abhishek Jain    Guru Vamsi Policharla    Sruthi Sekar

NTTResearch    Johns Hopkins University    Berkeley
zk-SNARKs: Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge

$L \in NP$

$x \in L$

Prover $\rightarrow \pi \rightarrow$ Verifier

Okay, I believe you
zk-SNARKs: Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge

$L \in \mathbb{NP}$

$x \in L$

Verifier should not learn the secret witness

Soundness

Cheating prover cannot give accepting proof if $x \notin L$

Succinctness

Time to verify is smaller than time to compute

Okay, I believe you
zk-SNARKs: Numerous Applications

Anonymous Credentials [Chaum82]

Verifiable Inference of Machine Learning [LKKO20]

Private Smart Contracts [BCGMMW20]

Proving existence of bugs in code [HK20]

Verifying authenticity of images in media [NT16]

Privacy Respecting Cryptocurrency [BCGGMTV14]
zk-SNARKs: Lots of Work on Improving Efficiency

On the Size of Pairing-based Non-interactive Arguments

DIZK: A Distributed Zero Knowledge Proof System

Public-Coin Zero-Knowledge Arguments with (almost) Minimal Time and Space Overheads

New optimization techniques for PlonK’s arithmeticization

Time to generate of zk-SNARKs >>> Time to check the relation directly

Delegating Computations with (almost) Minimal Time and Space Overhead

Scalable, transparent, and post-quantum secure computational integrity

Brakedown: Linear-time and post-quantum SNARKs for R1CS
Gru’s Quest to be a Supervillain

Since Gru has a very long list of despicable achievements, computing a zk-SNARK will take a really long time

I am a great villain and deserve to be part of Vicious 6

Prove this in zero-knowledge

Gru: Rising Villain

Vicious 6: A prolific set of Supervillains
Can Gru Delegate zk-SNARK Computation?

Details of his despicable achievements

zk-SNARK attesting to his achievements

Violates Privacy!!: Requires leaking the entire witness to AWS

Gru: Rising Villain

Vicious 6: A prolific set of Supervillains
Can Gru Delegate zk-SNARK Computation?

Each minion only gets a share of the witness

Delegate to a group of minions who run an MPC to compute the zk-SNARK

Details of his despicable achievements

Gru: Rising Villain

Vicious 6: A prolific set of Supervillains
Can Gru Delegate zk-SNARK Computation?

Collaborative zk-SNARKs [OB22]

Delegate to a group of minions who run an MPC to compute the zk-SNARK

Each minion only gets a share of the witness

Details of his despicable achievements
Collaborative zk-SNARKs [OB22]

Efficient MPC for computing zk-SNARKs
Each party does work proportional to a single prover

[WZCAS18] leverage parallelism to distribute work across machines in a compute cluster to get faster proof generation

Privacy
Wasteful
Not Privacy Preserving
Our Goal

Better utilization of resources of the parties in collaborative zk-SNARKs, for faster proof generation, in a privacy-preserving manner
Our Results: \textbf{zkSaaS}

- **Framework**: For privacy preserving delegation of zk-SNARK computation. Each server is expected to run for a shorter duration than a single local prover.

- **Design**: Design \textbf{zkSaaS} for Groth16 [Gro16], Marlin [CHMMVW20] and Plonk [GWC19].

- **Implementation**: Implement a prototype of \textbf{zkSaaS} for Groth16, Plonk and get \(\approx 22\times\) speed-up when run with 128 parties for \(2^{21} - 2^{25}\) constraints.
**zkSaaS Framework**

### Typical zk-SNARKs

<table>
<thead>
<tr>
<th>Step 1:</th>
<th>Computing Extended Witness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2:</td>
<td>Generating Proof</td>
</tr>
<tr>
<td></td>
<td>(Cryptographic Operations + Field Operations)</td>
</tr>
</tbody>
</table>

**Pre-Processing:** each server gets a part of the correlated randomness

- Secret Shares “Extended Witness”

- **π**

- Client computes Step 1
- Servers collectively compute Step 2

- **Cryptographic Operations** get equally divided amongst all servers
- **Field Operations** get equally divided amongst small servers. **King** does work linear in the number of **field operations**.
Applicability of zkSaaS

To aid users with small devices

For extremely large computations
Designing zkSaaS
General Template [OB22]

Identify basic building blocks in zk-SNARKs + Design custom MPC protocols for each building block with the required efficiency

Combine them to get a zkSaaS for the corresponding zk-SNARK
Building Blocks in Groth16, Marlin, Plonk

Multi-Scalar Multiplications (MSM)

\[ F(g_1, \alpha_1, ..., g_m, \alpha_m) = \prod_{i \in [m]} g_i^{\alpha_i} \]

Fast Fourier Transform (FFT)

For converting between coefficient and evaluation representation of polynomials

Partial Products

\[ F(x_1, ..., x_m) = \left( \prod_{i \in [j]} x_i \right) \]

Polynomial Multiplication and Division

A combination of addition, multiplication and FFT operations
Packed Secret Sharing (PSS) [FY92]

Regular Secret Sharing

1 Value $\rightarrow$ $n$ shares

Corruption threshold: $t < \frac{n}{2}$
Packed Secret Sharing (PSS) [FY92]

Regular Secret Sharing

1 Value $\rightarrow$ $n$ shares

Corruption threshold: $t < \frac{n}{2}$

Packed Secret Sharing

$O(n)$ Values $\rightarrow$ $n$ shares
**Packed Secret Sharing (PSS) [FY92]**

Regular Secret Sharing

- Secret value: $v$
- Shares: $s_1, s_2, s_3, s_4, s_5$
- 1 Value $\rightarrow n$ shares
- Corruption threshold: $t < \frac{n}{2}$

Packed Secret Sharing

- Secret vector: $v_1, v_2, v_3, v_4$
- Shares: $s_1, s_2, s_3, s_4, s_5$
- $O(n)$ Values $\rightarrow n$ shares
- Corruption threshold $t < n\left(\frac{1}{2} - \frac{1}{\varepsilon}\right)$
Experimental Results
zkSaaS for Groth16: Setup

**Local Prover**
- 1vCPU and 4 GB RAM

**zkSaaS Servers**
- 96vCPU and 128 GB RAM
- 1vCPU and 2 GB RAM each

---

N1 GCP Instances
zkSaaS for Groth16

- No. of Servers = 128
- Packing constant = 32
- No. of corrupt servers = 31

**Memory Exhaustion**

Weak servers can handle 16 times more constraints than consumer machine before running out of memory.

**Running Time**

We get ≈ 22× speed-up over consumer machine.

**Why not 32 times?**

1. FFT doesn’t achieve equal division of work
2. Sub-optimal use of Pippenger’s algorithm for MSMs
zkSaaS for Groth16

No. of Constraints = $2^{19}$

Packing constant = $n/4$

No. of corrupt servers = $n/4$
Thanks!
aarushi.goel@ntt-research.com
https://aarushigoel.github.io/