zkSaaS

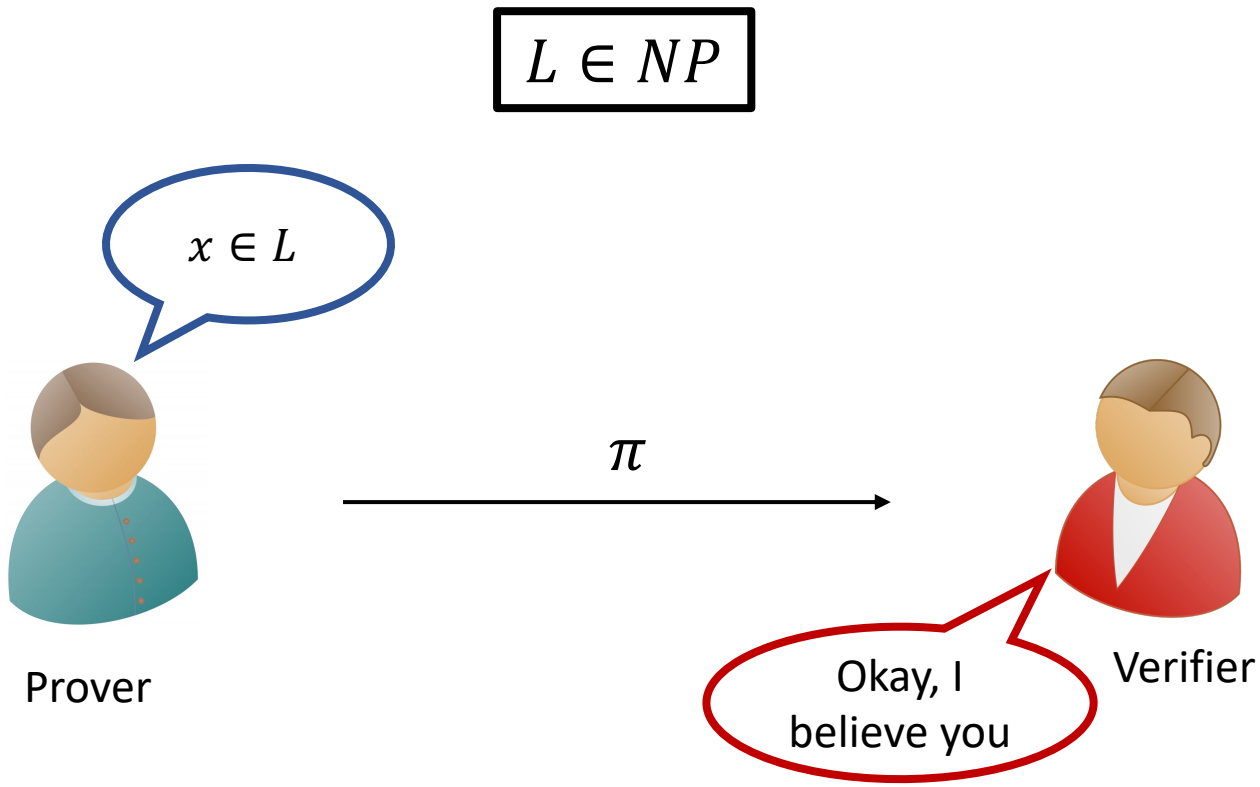
Zero-Knowledge SNARKs as a Service

Sanjam Garg Aarushi Goel Abhishek Jain Guru Vamsi Policharla Sruthi Sekar

NTT Research Johns Hopkins University Berkeley University of California

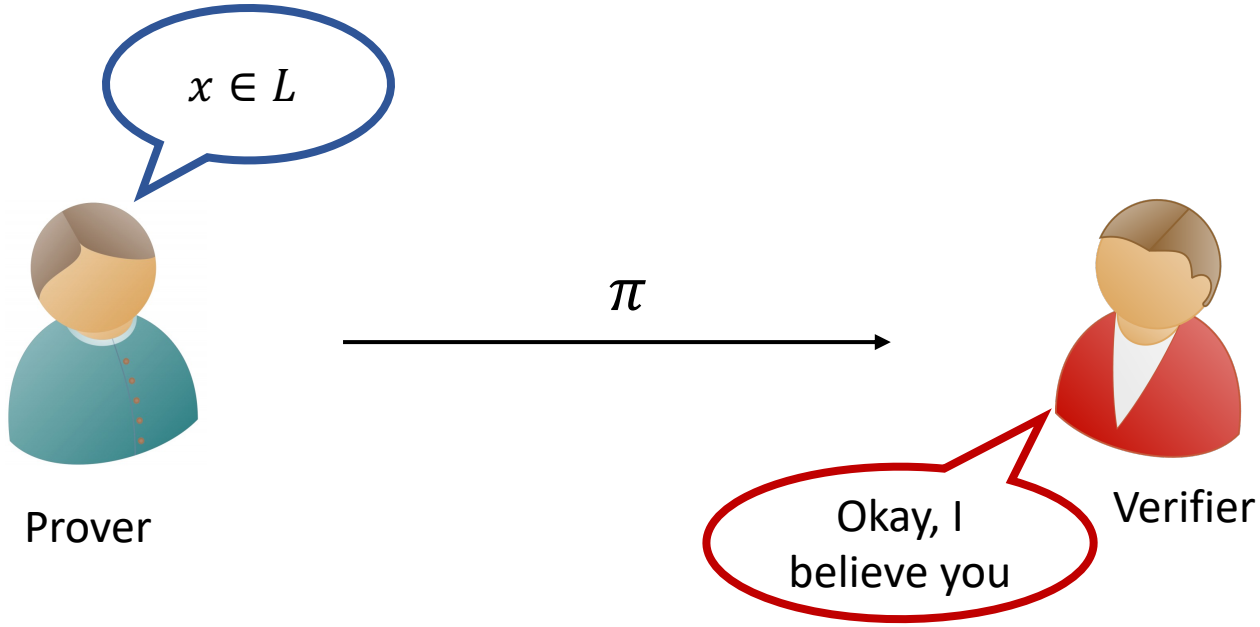
USENIX Security 2023

zk-SNARKs: Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge



zk-SNARKs: Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge

$L \in NP$



Anyone can verify

Zero knowledge

Verifier should not learn the secret witness

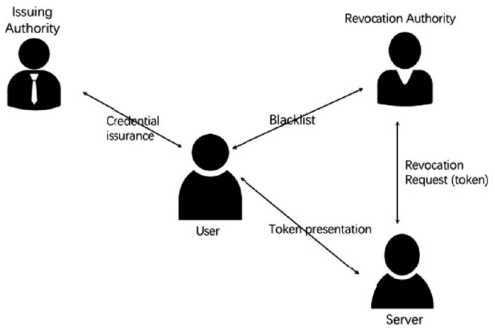
Soundness

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

Succinctness

Time to verify is smaller than time to compute

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

Orion: Zero Knowledge Proof with Linear Prover Time

Tiancheng Xie[1], Yupeng Zhang[2], and

[1] University of California, Berk
{tianc.x,dawnsong}@berkeley
[2] Texas A&M University
zhangyp@tamu.edu

On the Size of Pairing-based Non-interactive Arguments*

Jens Groth**

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

Alexander R. Block[1], Justin Holmgren[2], Alon Rosen[3], Ron D. Rothblum[4], and Pratik Soni[5]

[1] Purdue University, block9@purdue.edu
NTT Research, justin.holmgren@ntt-research.com
[3] IDC Herzliya, alon.rosen@idc.ac.il
[4] Technion, rothblum@cs.technion.ac.il
rsity of California, Santa Barbara, pratik_soni@cs.ucsb.edu

Gemini: Elastic SNARKs for Diverse Environments

Jonathan Bootle
jbt@zurich.ibm.com
IBM Research

Alessandro Chiesa
alessandro.chiesa@epfl.ch
EPFL

Yuncong Hu
yuncong_hu@berkeley.edu
UC Berkeley

Michele Orrù
michele.orru@berkeley.edu
UC Berkeley

DIZK: A Distributed Zero Knowledge Proof System*

Howard Wu
howardwu@berkeley.edu
UC Berkeley

Wenting Zheng
wzheng@eecs.berkeley.edu
UC Berkeley

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Raluca Ada Popa
raluca.popa@berkeley.edu
UC Berkeley

Ion
istoica@b
UC B

New optimization techniques for $\mathcal{PlonK}$'s arithmetization

Miguel Ambrona, Anne-Laure So

Nomadic Labs, Paris, France
name.surname@nomadic-labs.com

HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates

Dan Boneh
Zhenfei Zhang

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

$\mathcal{PlonK}$: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge

Ariel Gabizon*
Aztec

Zachary J. Williamson
Aztec

Oana Ciobotaru

Delegating Computations with (almost) Minimal Time and Space Overhead

Justin Holmgren*
Ron D. Rothblum†

September 13, 2018

Time- and Space-Efficient Arguments from Groups of Unknown Order*

Alexander R. Block†
Justin Holmgren‡
Alon Rosen§
Ron D. Rothblum¶
Pratik Soni‖

September 21, 2021

Aurora: Transparent Succinct Arguments for R1CS

Eli Ben-Sasson
eli@cs.technion.ac.il
Technion

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Michael Riabzev
mriabzev@cs.technion.ac.il
Technion

Nicholas Spooner
nick.spooner@berkeley.edu
UC Berkeley

Madars Virza
madars@mit.edu
MIT Media Lab

Nicholas P. Ward
npward@berkeley.edu
UC Berkeley

May 8, 2019

Scalable, transparent, and post-quantum secure computational integrity

Eli Ben-Sasson*
Iddo Bentov†
Yinon Horesh*
Michael Riabzev*

March 6, 2018

MARLIN:
Preprocessing zkSNARKs with Universal and Updatable SRS

Alessandro Chiesa
lexch@berkeley.edu
UC Berkeley

Yuncong Hu
yuncong_hu@berkeley.edu
UC Berkeley

Mary Maller
mary.maller.15@ucl.ac.uk
UCL

Pratyush Mishra
pratyush@berkeley.edu
UC Berkeley

Psi Vesely
psi@ucsd.edu
UCL

Nicholas Ward
npward@berkeley.edu
UC Berkeley

May 27, 2020

Brakedown: Linear-time and post-quantum SNARKs for R1CS

Alexander Golovnev
Georgetown University

Jonathan Lee
Nanotronics Imaging

Srinath Setty
Microsoft Research

Justin Thaler
Georgetown University

Riad S. Wahby
Algorand and Carnegie Mellon University

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

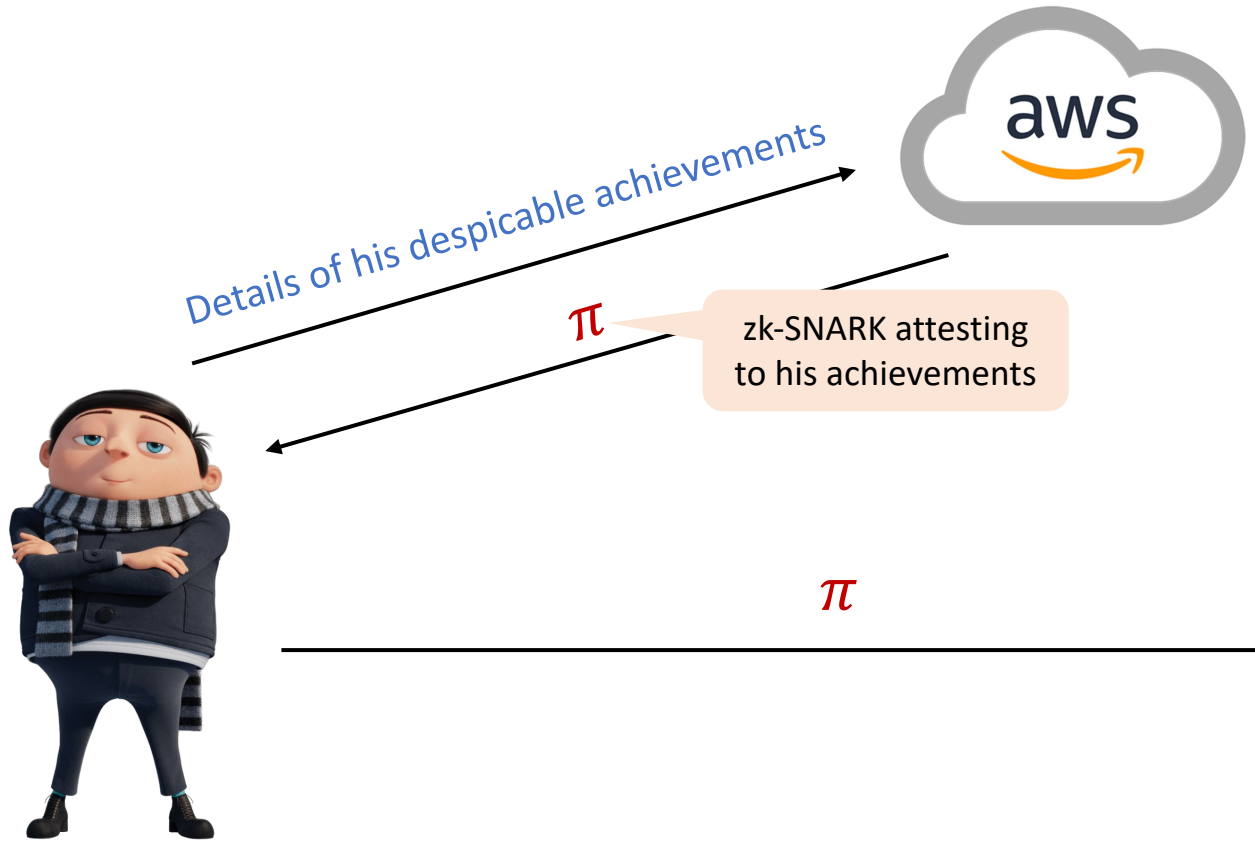
Prove this in zero-knowledge

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

Gru: Rising Villain

Vicious 6: A prolific set of Supervillains

Can Gru Delegate zk-SNARK Computation?

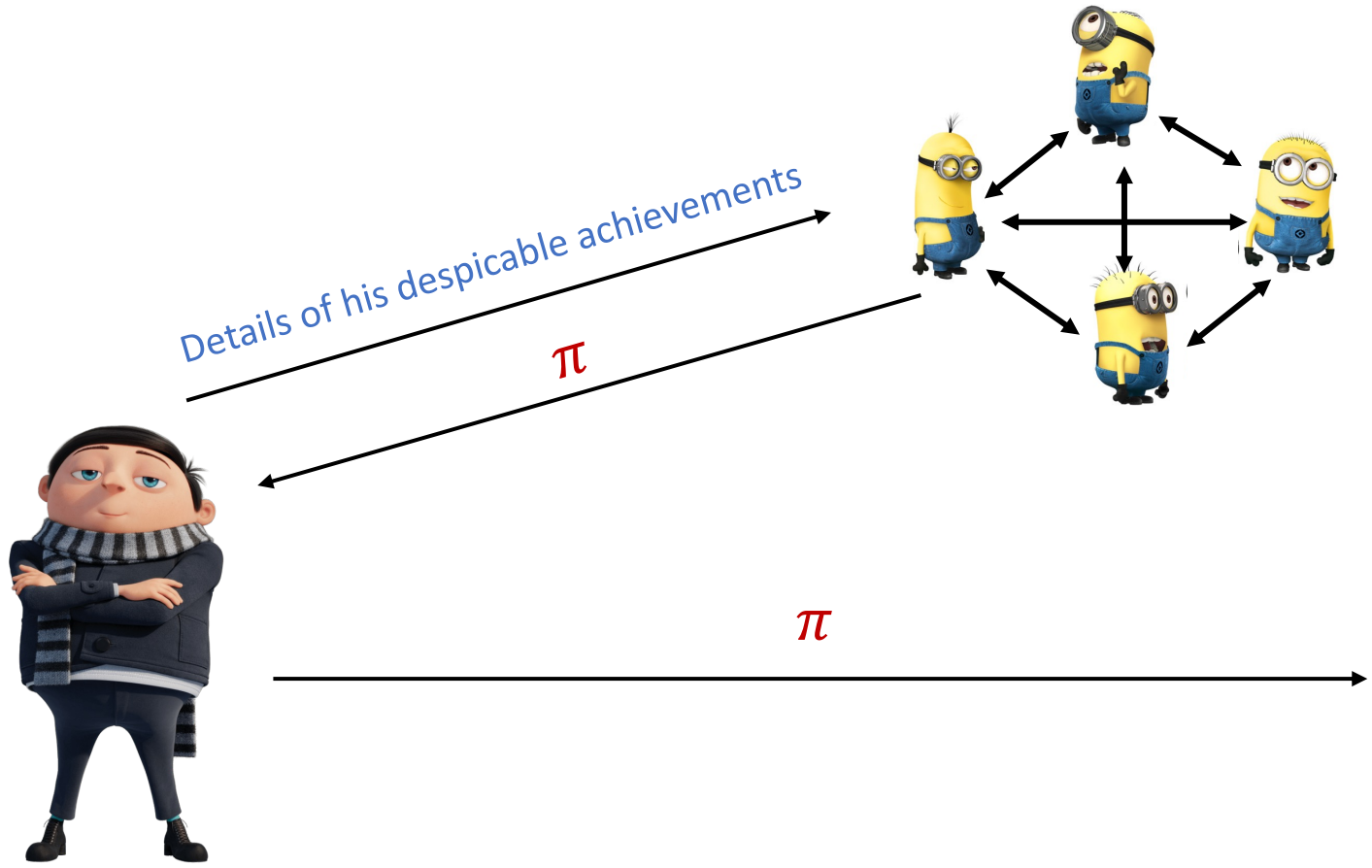


Violates Privacy!!: Requires leaking the entire witness to AWS

Gru: Rising Villain

Vicious 6: A prolific set of Supervillains

Can Gru Delegate zk-SNARK Computation?



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

Each minion only gets a share of the witness

Gru: Rising Villain

Vicious 6: A prolific set of Supervillains

Can Gru Delegate zk-SNARK Computation?

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

π

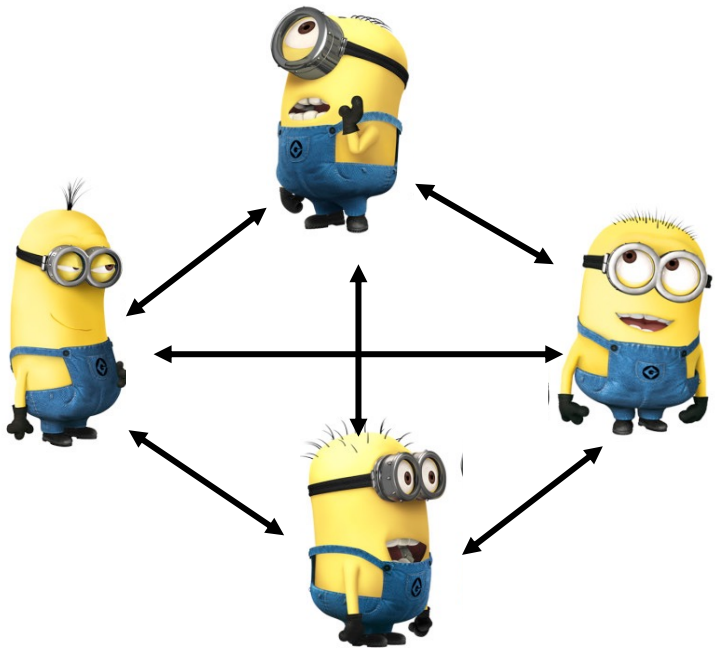
Collaborative zk-SNARKs [OB22]

π

Gru: Rising Villain

Vicious 6: A prolific set of Supervillains

Collaborative zk-SNARKs [OB22]



Efficient MPC for computing zk-SNARKs

Privacy

Each party does work proportional to a single prover

Wasteful

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

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: zkSaaS

Framework	For privacy preserving delegation of zk-SNARK computation. Each servers is expected to run for a shorter duration than a single local prover.
Design	Design zkSaaS for Groth16 [Gro16], Marlin [CHMMVW20] and Plonk [GWC19].
Implementation	Implement a prototype of zkSaaS for Groth16, Plonk and get $\approx$ 22× speed-up when run with 128 parties for $2^{21} - 2^{25}$ constraints

zkSaaS Framework

Typical zk-SNARKs

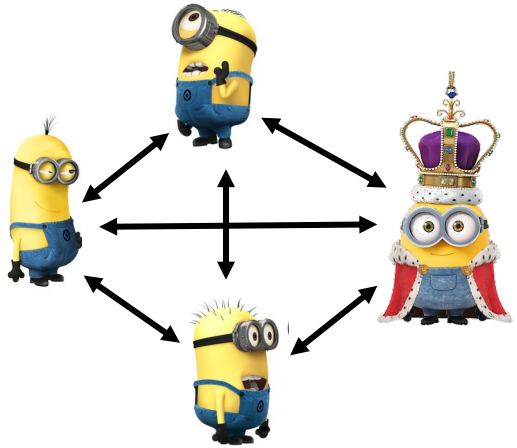
Step 1: Computing Extended Witness

Step 2: Generating Proof
(Cryptographic Operations + Field Operations)

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

Secret Shares "Extended Witness"

π



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.

Client computes Step 1

Servers collectively compute Step 2

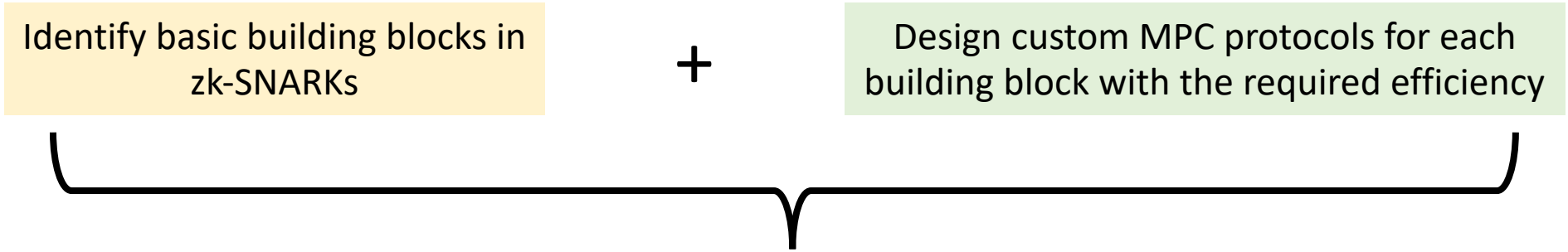
Applicability of zkSaaS

To aid users with small devices

For extremely large computations

Designing zkSaaS

General Template [OB22]



Building Blocks in Groth16, Marlin, Plonk

Multi-Scalar Multiplications (MSM)

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

Partial Products

$$F(x_1, \dots, x_m) = \left(\prod_{i \in [j]} x_i \right)_{j \in [m]}$$

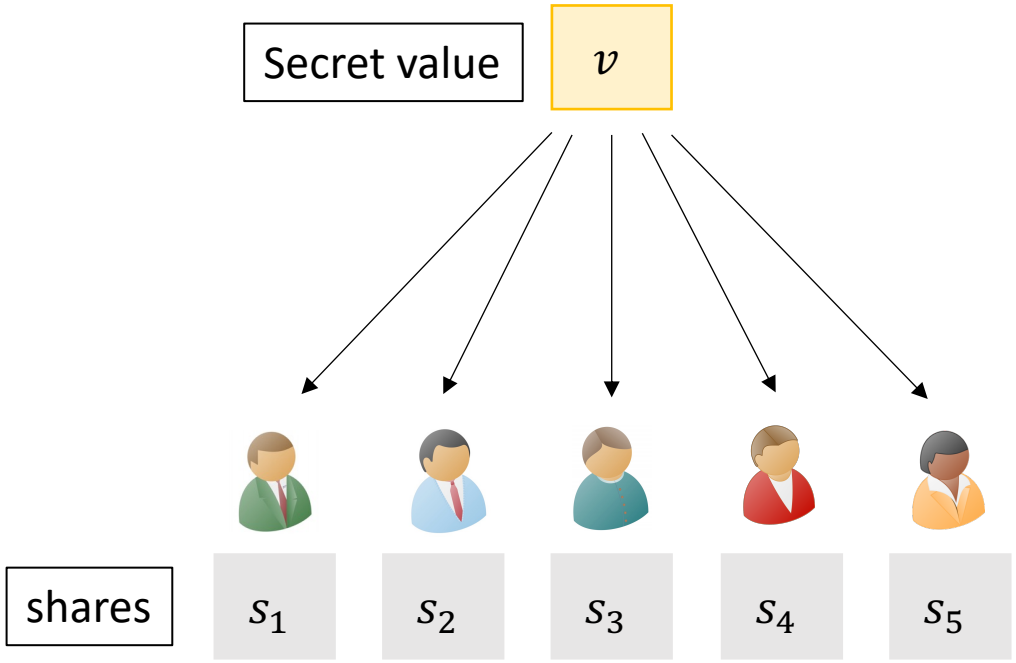
Fast Fourier Transform (FFT)

For converting between coefficient and evaluation representation of polynomials

Polynomial Multiplication and Division

A combination of addition, multiplication and FFT operations

Packed Secret Sharing (PSS) [FY92]

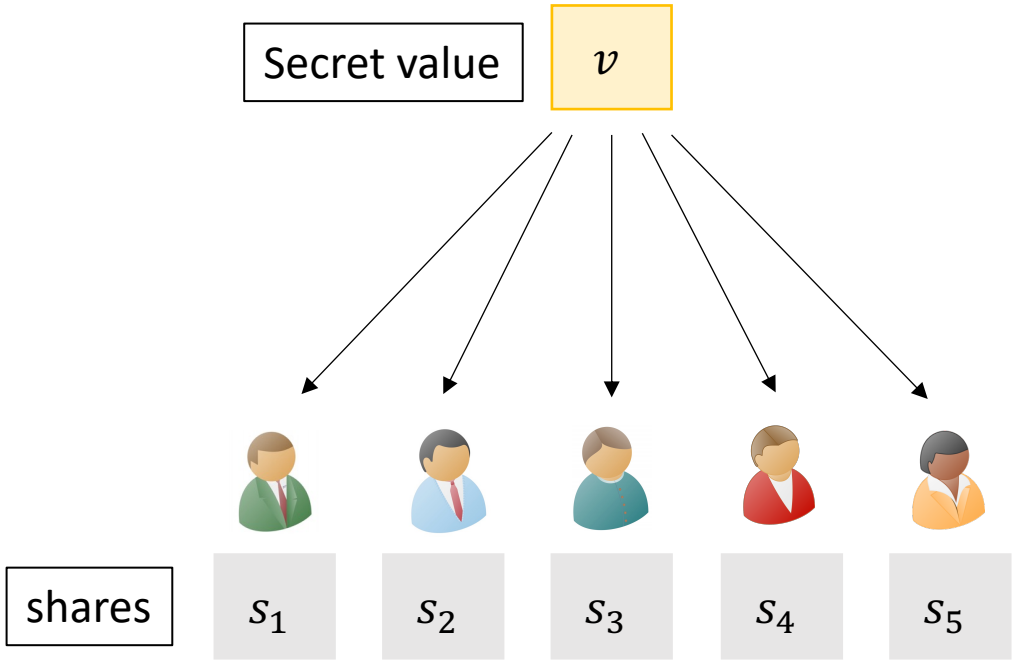


Regular Secret Sharing

1 Value → n shares

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

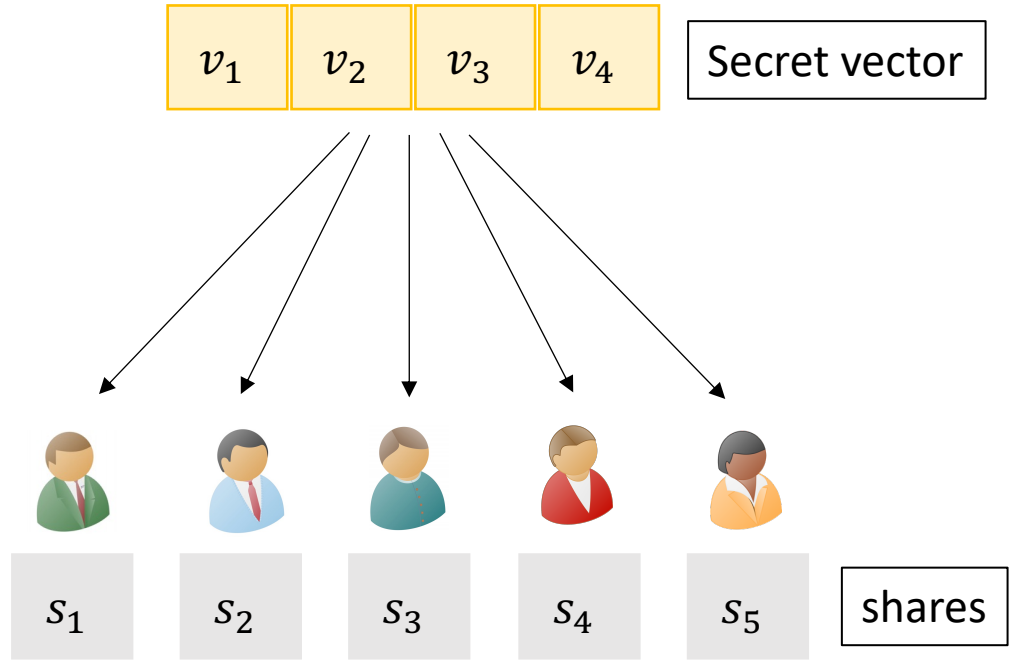
Packed Secret Sharing (PSS) [FY92]



Regular Secret Sharing

1 Value → n shares

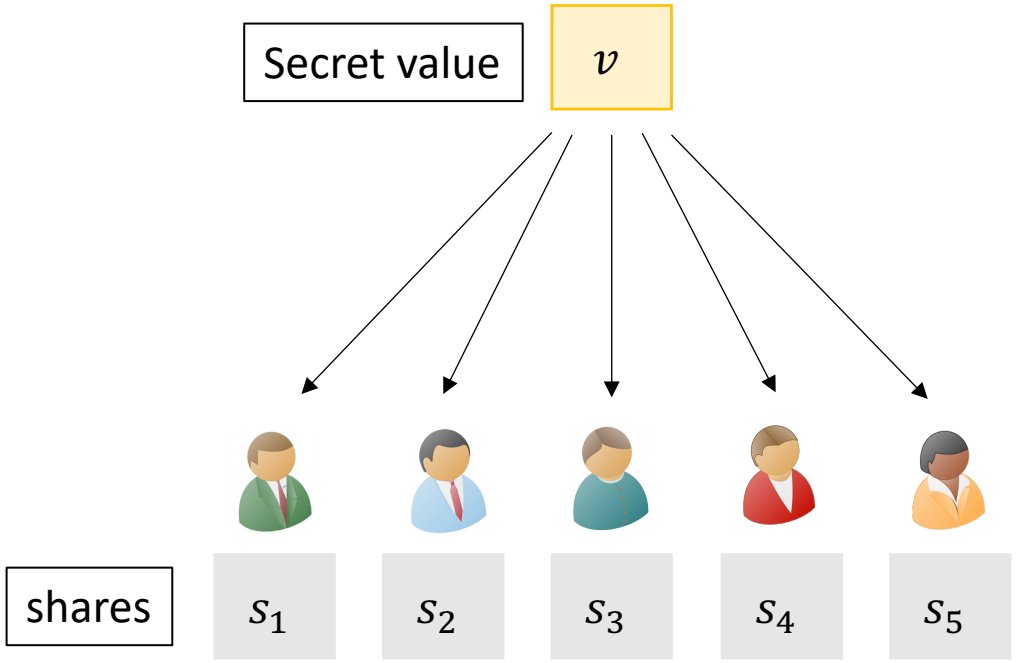
Corruption threshold: $t < \frac{n}{2}$



Packed Secret Sharing

$O(n)$ Values → n shares

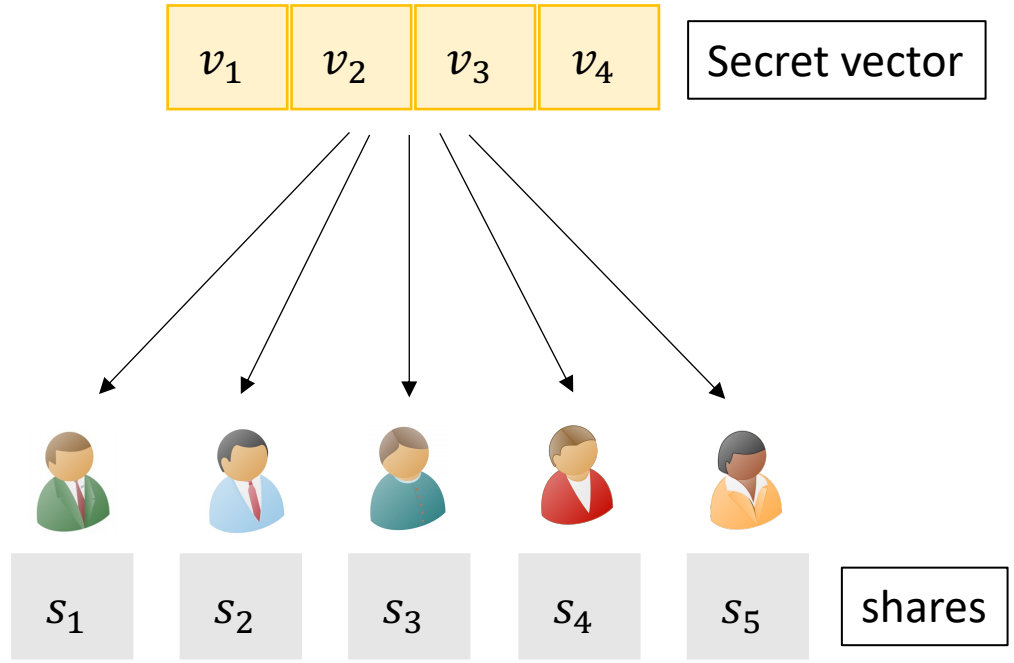
Packed Secret Sharing (PSS) [FY92]



Regular Secret Sharing

1 Value → n shares

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



Packed Secret Sharing

$O(n)$ Values → n shares

Corruption threshold $t < n(\frac{1}{2} - \frac{1}{\varepsilon})$

Experimental Results

zkSaaS for Groth16: Setup

N1 GCP Instances

Local Prover

1vCPU and 4 GB RAM

zkSaaS Servers

96vCPU and 128 GB RAM

1vCPU and 2 GB RAM each

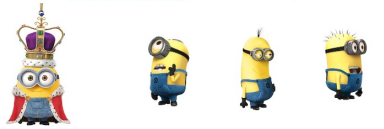
zkSaaS for Groth16

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

Local Prover

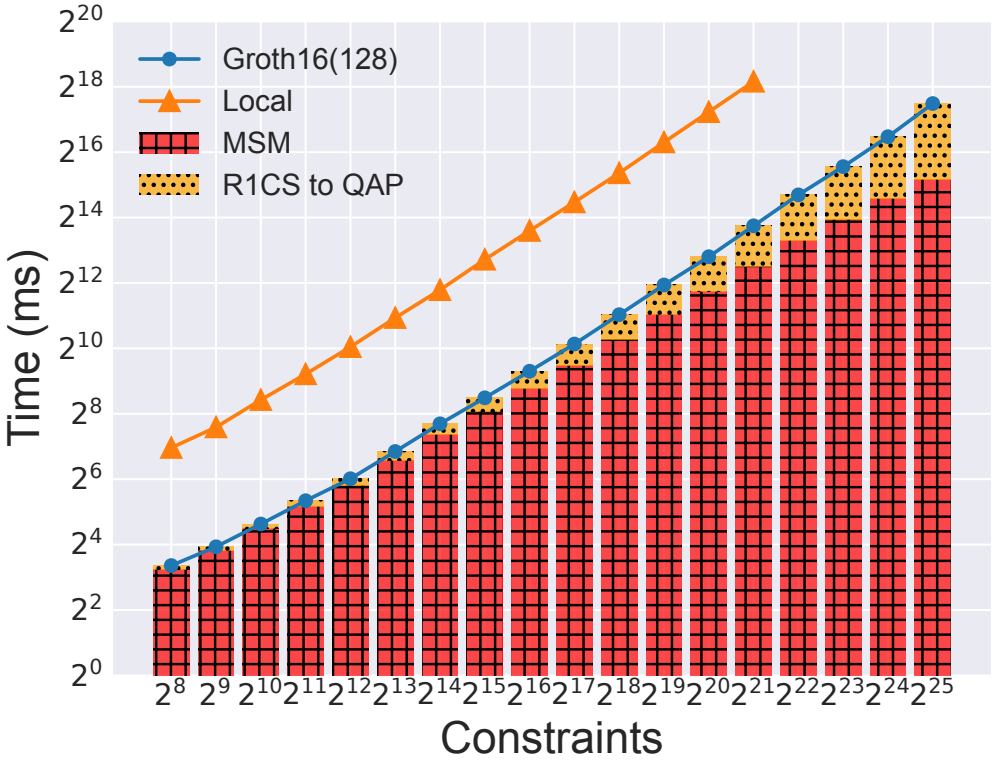
1vCPU and 4 GB RAM

zkSaaS Servers



96vCPU and 128 GB RAM

1vCPU and 2 GB RAM each



Memory Exhaustion

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

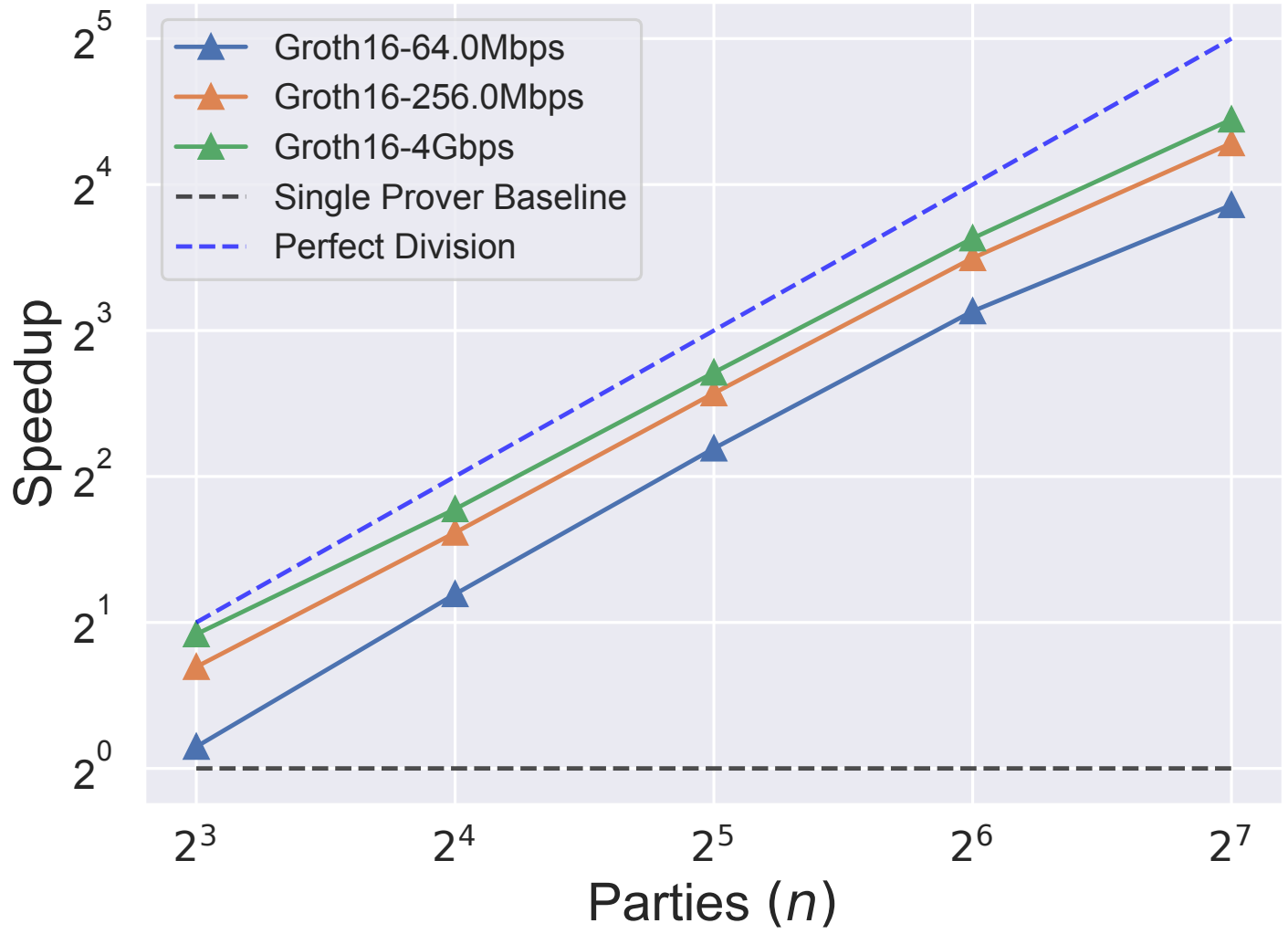
Running Time

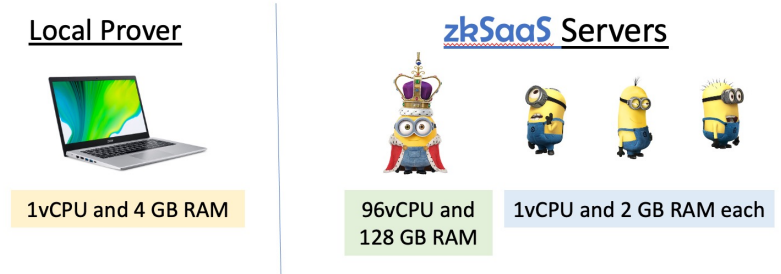
We get ≈ 22× speed-up over consumer machine

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

Paper

Code

Thanks!

aarushi.goel@ntt-research.com

https://aarushigoel.github.io/