Collaborative zk-SNARKs

Proving as One over Distributed Secrets

Aarushi Goel

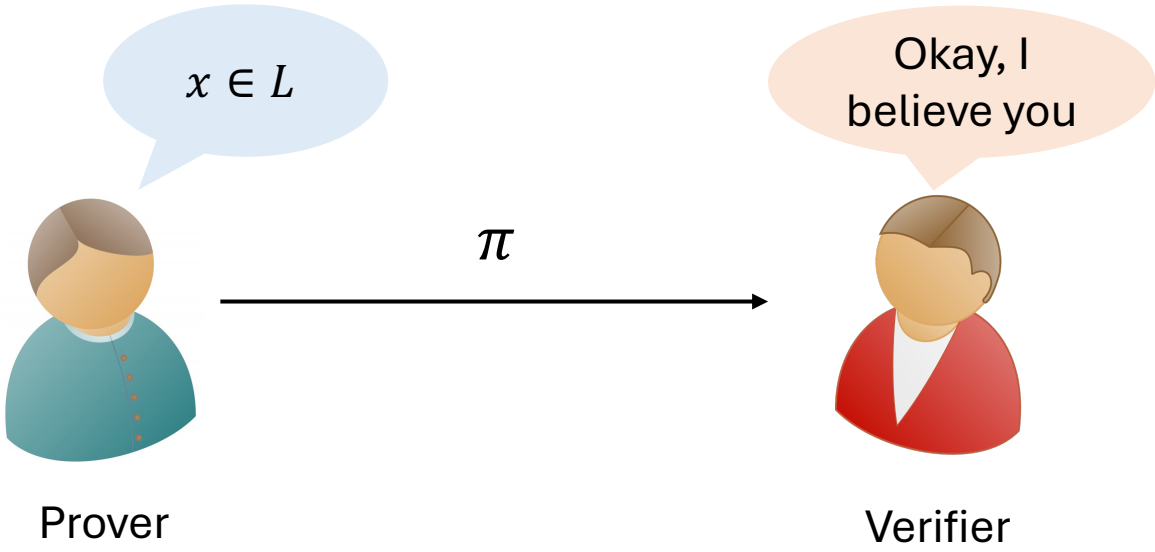
PURDUE UNIVERSITY

Simons – Proofs Workshop | Summer 2025

zk-SNARKs

Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge

$$L \in NP$$



π is computed by a single prover who knows the witness corresponding to x

What if a single prover does not have the entire witness?

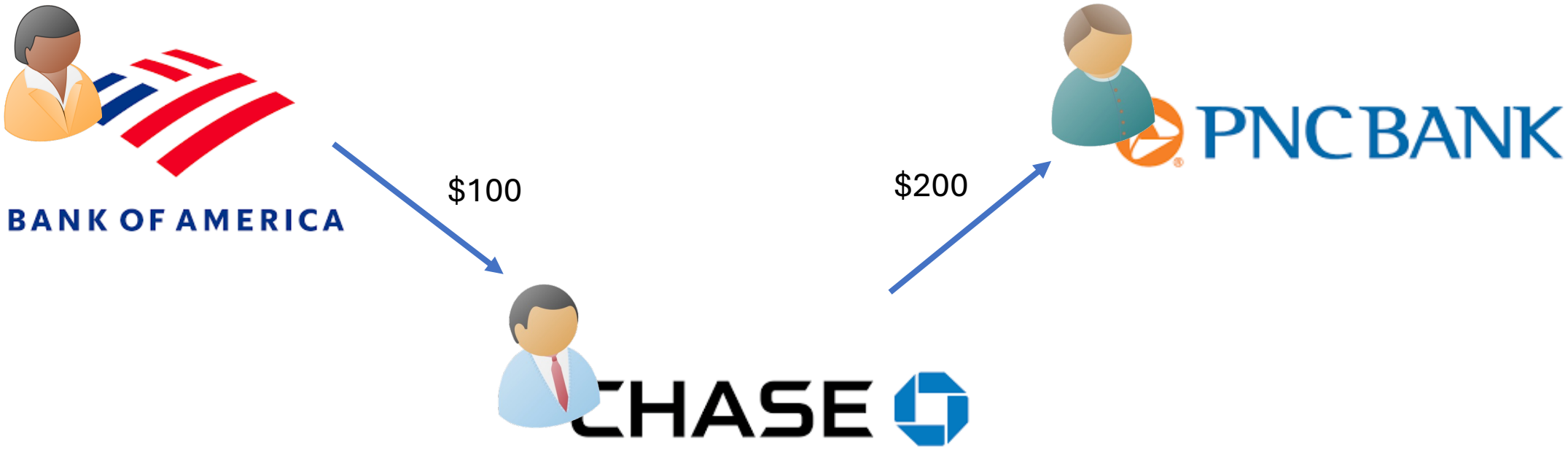
Aggregate Healthcare Statistics

Collectively prove that they do not charge the same price for insured and uninsured patients

Without revealing patient records

What if a single prover does not have the entire witness?

Private Audits of Financial Transactions



Collectively prove whether Alice transferred money to Bob and Bob to Charlie

Without reveal other financial transactions

Common Factors in these Examples

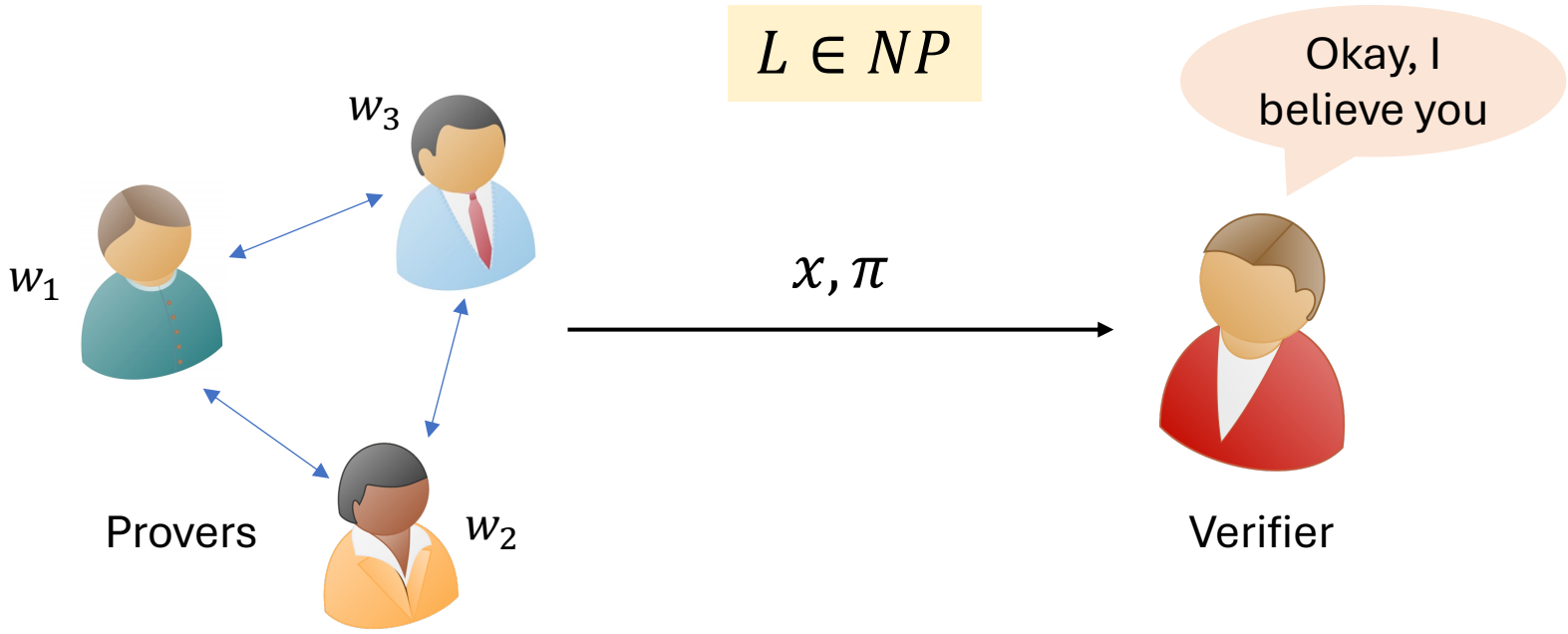
No single entity has the entire witness

No single entity knows whether the statement is true

Multiple entities want to collectively generate a single proof

They do not want to share their private witnesses with each other

Collaborative zkSNARKs [Ozdemir-Boneh'22]



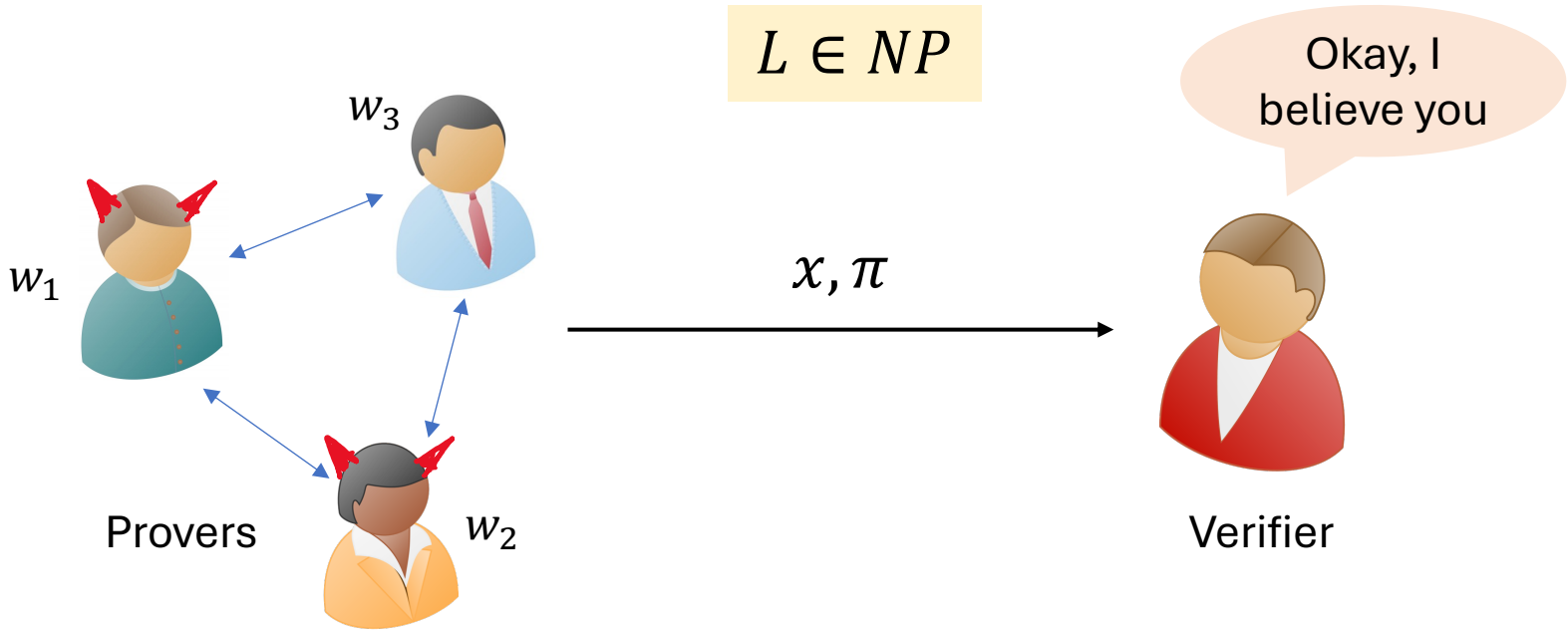
Soundness

π verifies iff $R_L(x, w_1, w_2, w_3) = 1$

Succinctness

π is short

Collaborative zkSNARKs [Ozdemir-Boneh'22]



t-Zero-Knowledge

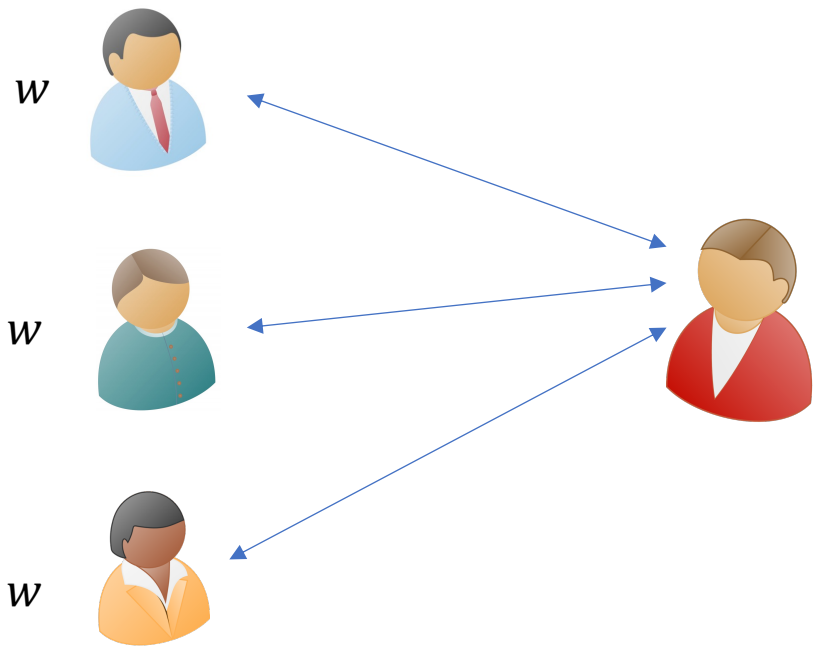
This is stronger than regular zero-knowledge

The Verifier and corrupt provers should not learn w_3 even if $x \notin L$

Different from Other Multi-Prover Notions

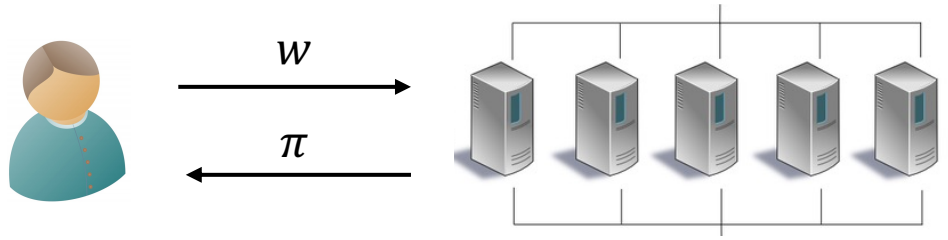
Multi-prover Interactive Proofs (MIPs)

[Ben-Or-Goldwasser-Kilian-Wigderson'88]



Non-privacy preserving distributed proof generation

[Wu-Zhang-Chiesa-AdaPopa-Stoica'18]



Delegating proof computation to a compute cluster

Constructing Collaborative zk-SNARKs

Compute a zk-SNARK using a secure multiparty computation protocol

Where is the non-triviality?

zk-SNARKs are cryptographic computations. Computing them using MPC will incur large overheads.

[OB'22] Approach for Constructing Collaborative zkSNARKs

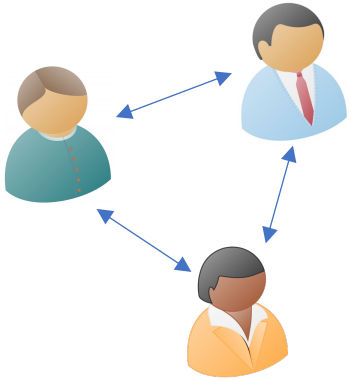
Typical zk-SNARKs

Short witness ⇢ Extended witness
(Simple Field Operations)

Compute proof using extended witness
(Cryptographic Operations + Field Operations)

Collaborative zk-SNARKs

Generic MPC for generating extended witness



Custom MPC for computing proof given
extended witness

[OB'22] Approach for Constructing Collaborative zkSNARKs

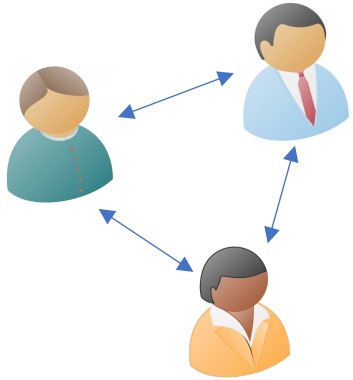
This MPC must check validity of joint witness before proceeding with proof computation.

Collaborative zk-SNARKs

Generic MPC for generating extended witness

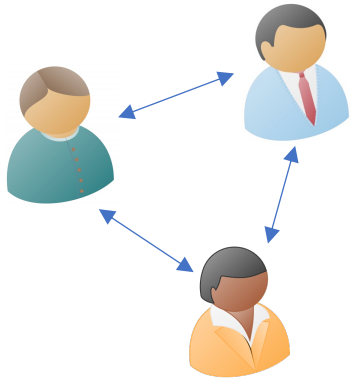
Else, loss of input privacy!!

[Garg-G-Jain-Sekar-Roberts 25]



Custom MPC for computing proof given extended witness

Rest of this talk will focus on



Custom MPC for computing proof given extended witness

Custom MPC for computing proof given extended witness

Main novelty of all prior work lies here

Any Existing Malicious

Security Compiler

MPC Secure against a Semi-honest Adversary

MPC Secure against a Malicious Adversary

Custom MPC for computing proof given extended witness

(MPC Secure against a Semi-honest Adversary)

Prior Work: Designing efficient MPC for functions of the following form

Multi-Scalar Multiplications (MSM)

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

Partial Products

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

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

Custom MPC for computing proof given extended witness

Academia

[Ozdemir-Boneh'22]

[Garg-G-Jain-Policharla-Sekar'23]

[Liu-Zhou-Wang-He-Zhang-Yang-Zhang'24]

[Liu-Zhou-Wang-Zhang-Yang'24]

[Garg-G-Kolonelos-Sinha'25]

Industry

webb

RENEGADE
ON CHAIN DARK POOL

TACEO

Custom MPC for computing proof given extended witness

MPC Secure against a
Semi-honest Adversary

Incurs additional overhead!

Any Existing Malicious

Security Compiler

MPC Secure against a
Malicious Adversary

Can we minimize (or eliminate this overhead?)

[Garg-G-Jain-Sekar-Roberts'25]

Semi-honest
Collaborative zk-SNARK

=

Maliciously Secure
Collaborative zk-SNARK

Additive Attack Paradigm

[Genkin-Ishai-Prabhakaran-Sahai-Tromer'14]

Honest majority setting

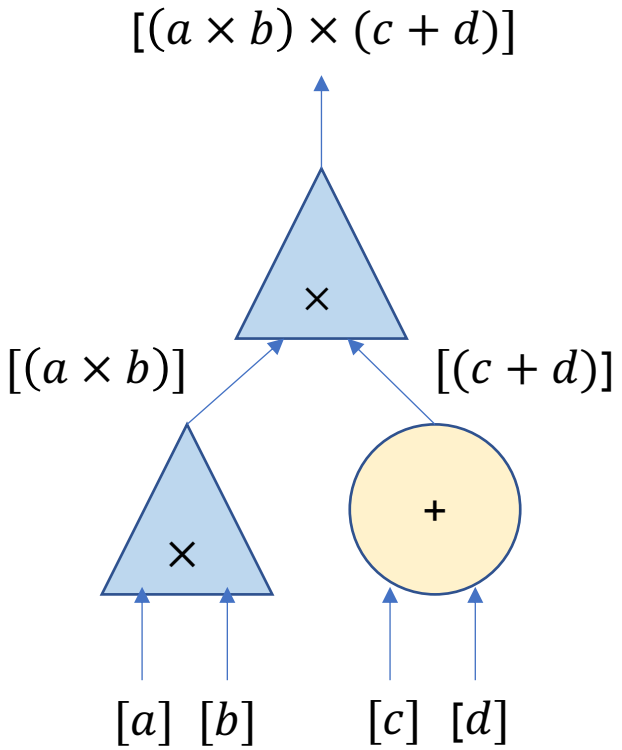
Many secret-sharing based semi-honest MPC protocols are also reasonably secure against malicious adversaries.

Additive Attack Paradigm

[Genkin-Ishai-Prabhakaran-Sahai-Tromer'14]

Honest majority setting

Many secret-sharing based semi-honest MPC protocols are also reasonably secure against malicious adversaries.

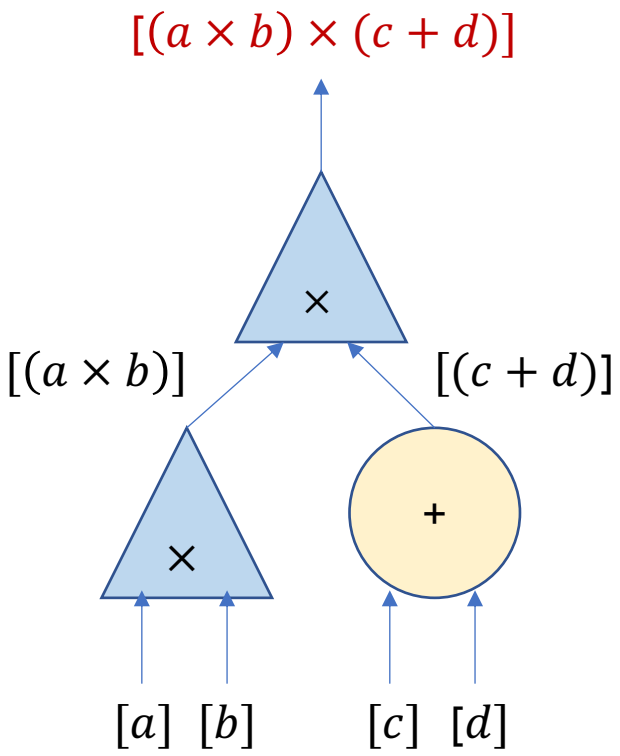


Additive Attack Paradigm

[Genkin-Ishai-Prabhakaran-Sahai-Tromer'14]

Honest majority setting

Many secret-sharing based semi-honest MPC protocols are also reasonably secure against malicious adversaries.



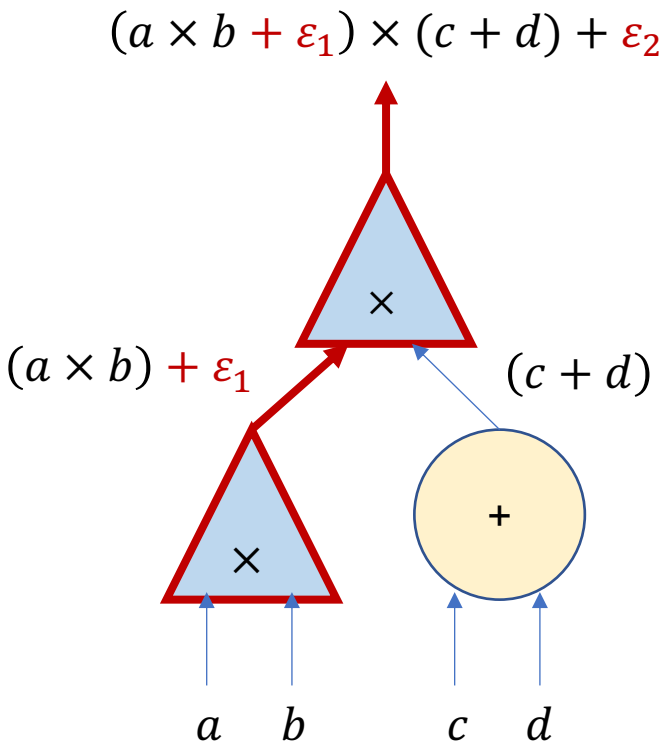
1 Private until output reconstruction

Additive Attack Paradigm

[Genkin-Ishai-Prabhakaran-Sahai-Tromer'14]

Honest majority setting

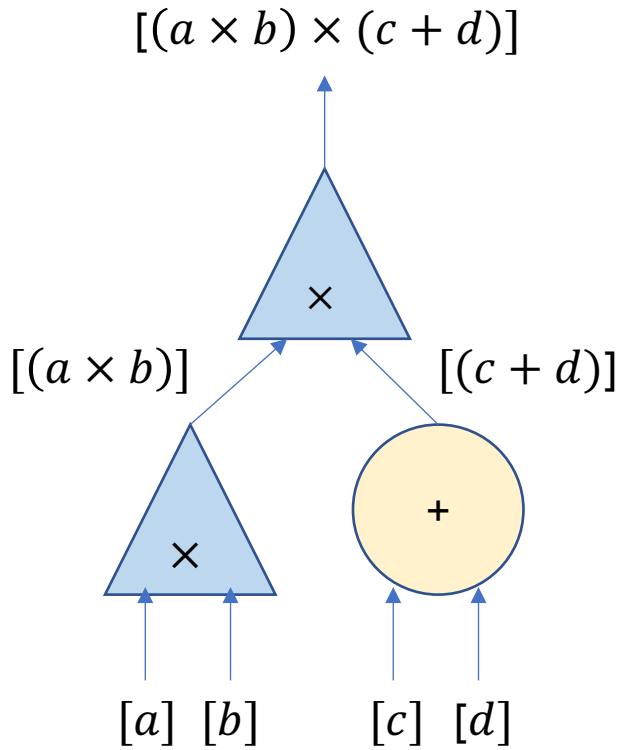
Many secret-sharing based semi-honest MPC protocols are also reasonably secure against malicious adversaries.



1 Private until output reconstruction

2 Injecting arbitrary additive errors

Semi-Honest → Malicious Security
(General Strategy)



Use a semi-honest MPC to compute secret shares of the output

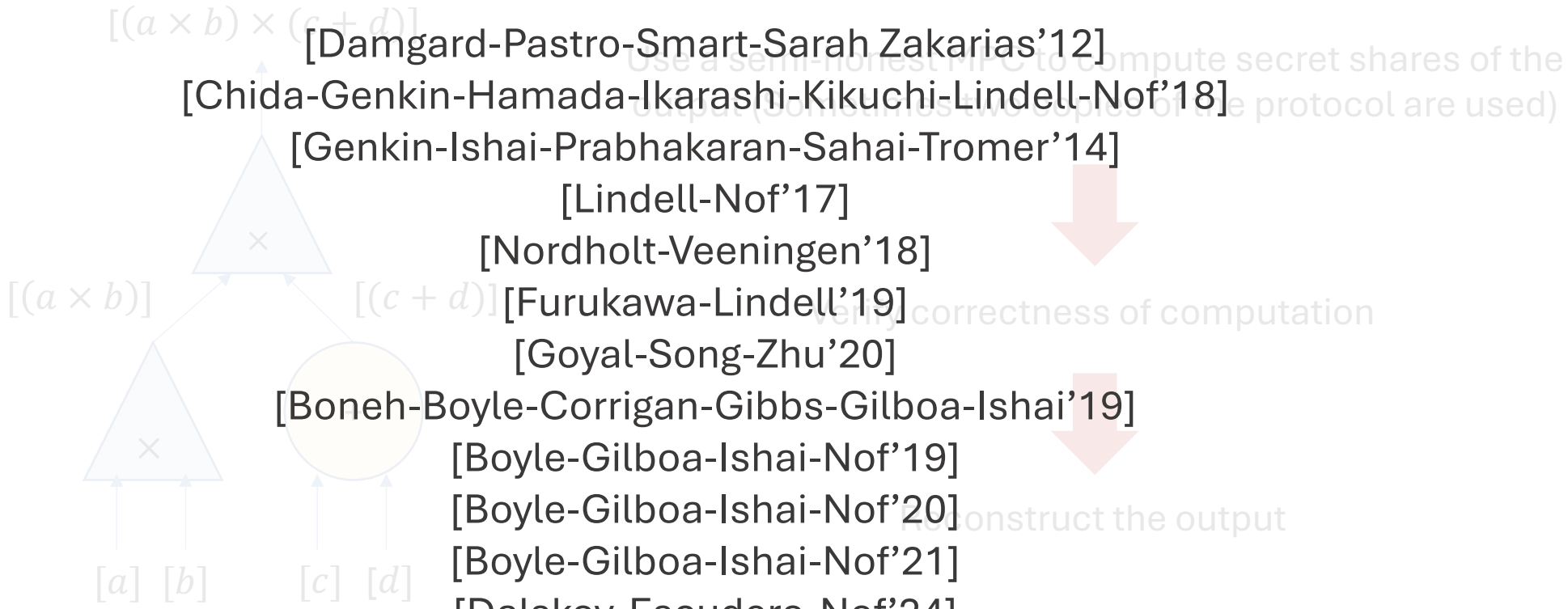
↓

Verify correctness of computation

↓

Reconstruct the output

Semi-Honest → Malicious Security

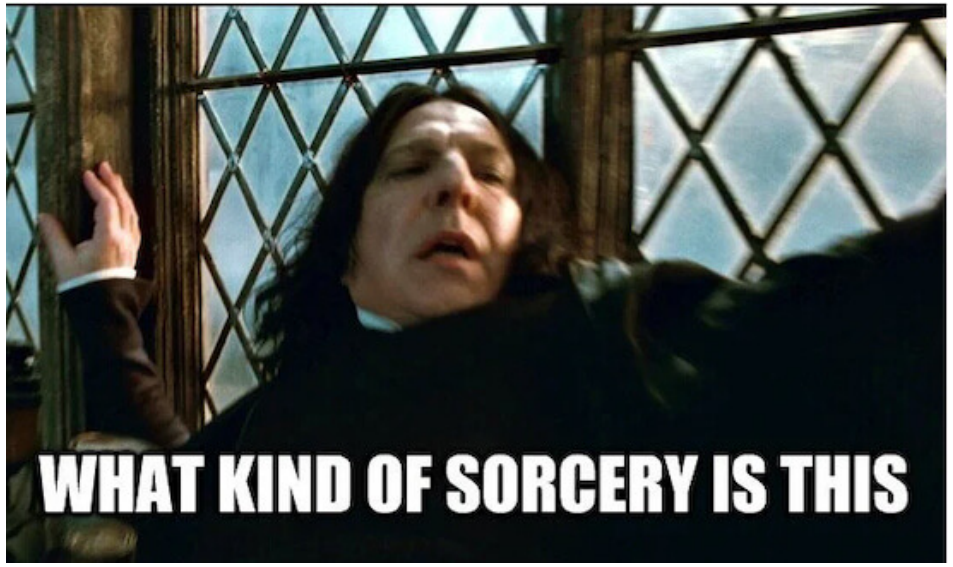


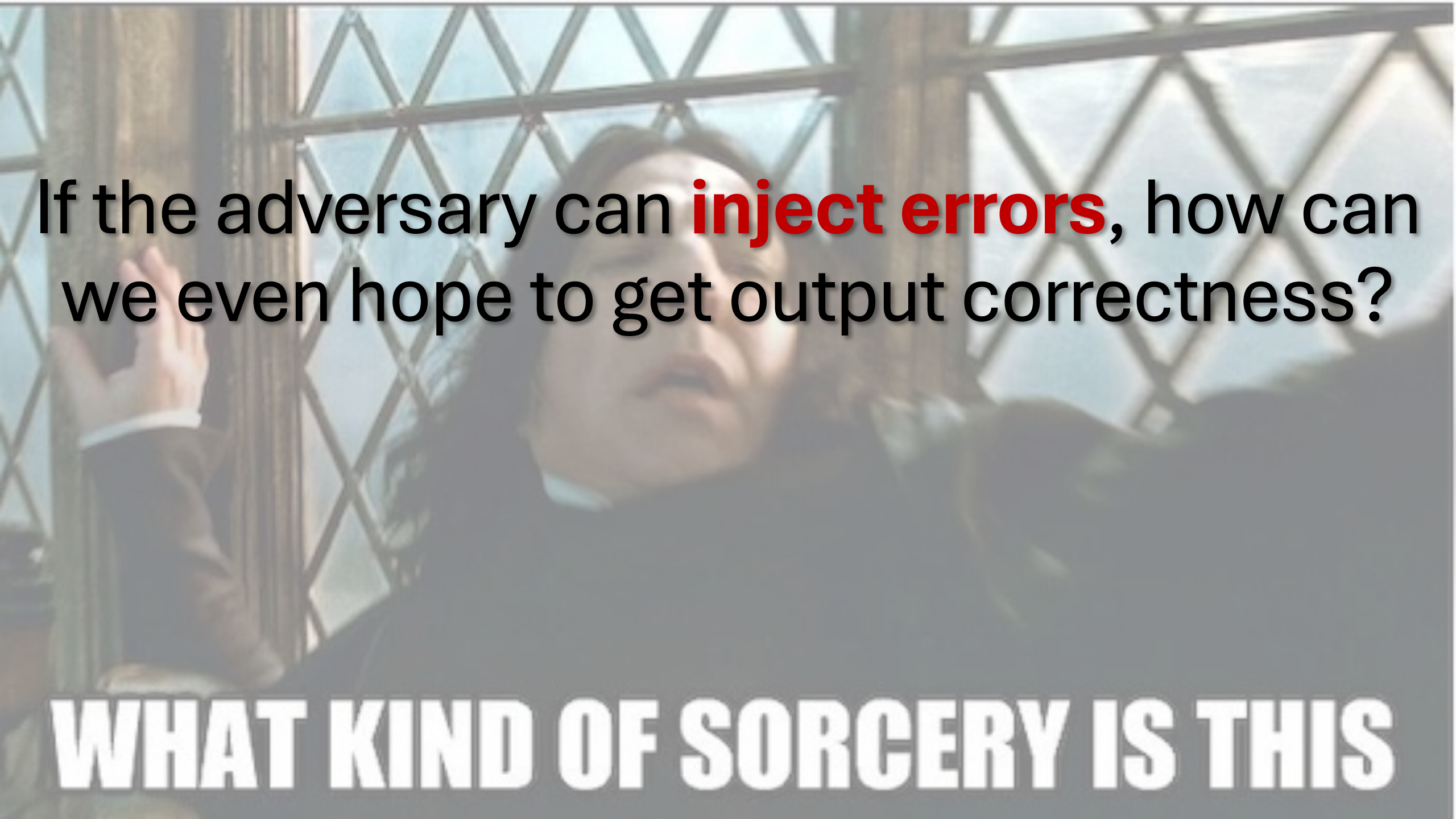
[Damgard-Pastro-Smart-Sarah Zakarias'12]
[Chida-Genkin-Hamada-Ikarashi-Kikuchi-Lindell-Nof'18]
[Genkin-Ishai-Prabhakaran-Sahai-Tromer'14]
[Lindell-Nof'17]
[Nordholt-Veeningen'18]
[Furukawa-Lindell'19]
[Goyal-Song-Zhu'20]
[Boneh-Boyle-Corrigan-Gibbs-Gilboa-Ishai'19]
[Boyle-Gilboa-Ishai-Nof'19]
[Boyle-Gilboa-Ishai-Nof'20]
[Boyle-Gilboa-Ishai-Nof'21]
[Dalskov-Escudero-Nof'24]

Can we get malicious security for free?

In the honest-majority setting

[Garg-G-Jain-Sekar-Roberts'25]




If the adversary can inject errors, how can we even hope to get output correctness?
WHAT KIND OF SORCERY IS THIS

We want to compute zk-SNARKs using MPC

Proofs are self-verifying

MPC does not need to enforce correctness of computed proof
We can simply check whether the proof successfully verifies

Our Idea

Will explain on the next slide

MPC secure up to additive attacks

For certain types of functions

MPC with L-malicious security

For all self-verifying functions

MPC with standard malicious security

Our Idea

Will explain on the next slide

MPC secure up to additive attacks

For certain types of functions

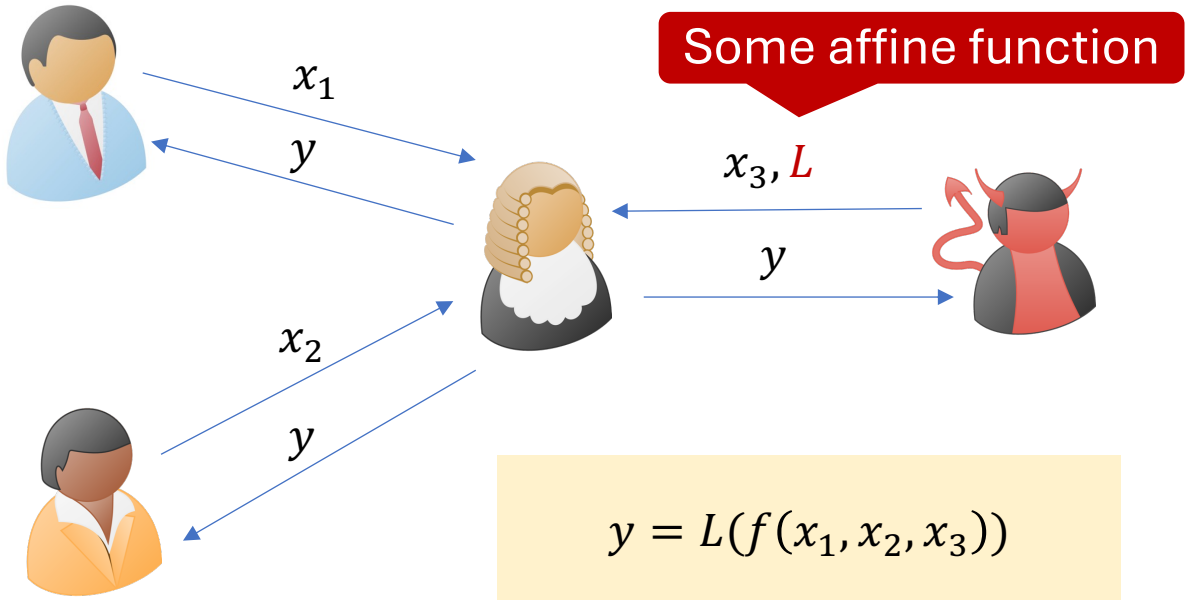
MPC with L-malicious security

For all self-verifying functions

MPC with standard malicious security

Relaxed Notion of Maliciously Secure MPC

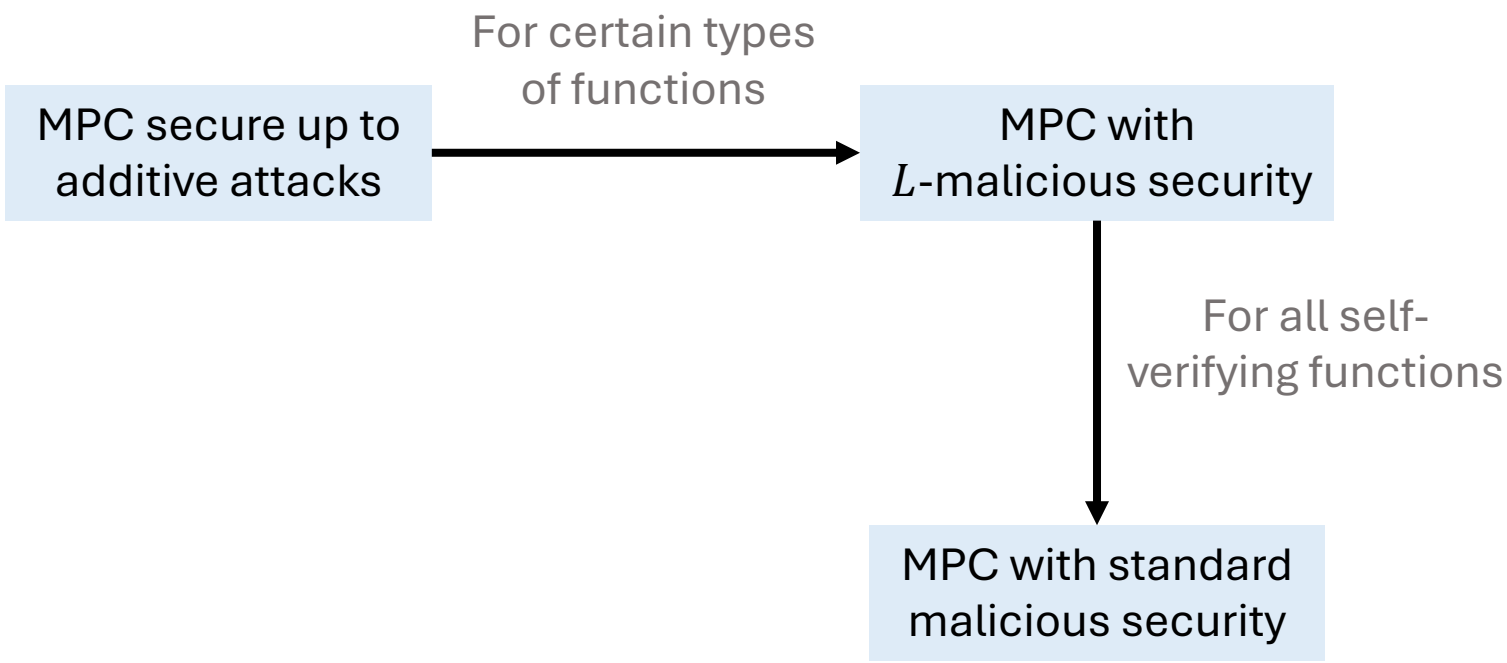
L-Malicious Security



$$y = L(f(x_1, x_2, x_3))$$

When computing proofs, an L-Maliciously Secure MPC implies standard malicious security

Our Idea


MPC secure up to additive attacks
For certain types of functions
MPC with L-malicious security
For all self-verifying functions
MPC with standard malicious security

Our Idea

MPC secure up to additive attacks

For certain types of functions

MPC with L-malicious security

For all self-verifying functions

MPC with standard malicious security

Example 1: Affine Functions

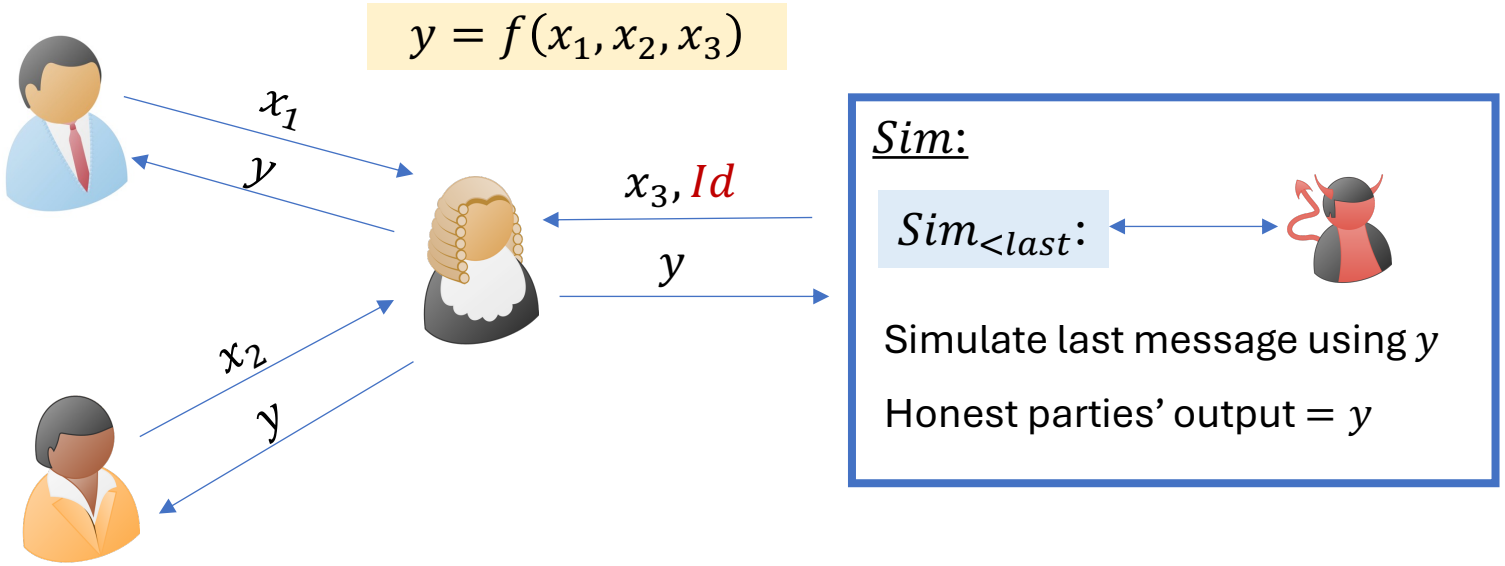
$$f_{A,B,C}(x_1, x_2, x_3) = A.x_1 + B.x_2 + C.x_3 + D$$

No errors!

Assuming x_1, x_2, x_3 are secret shared, while A, B, C, D are constants

How to Simulate?

$Sim_{<last}$: From [GIPST14], this simulator extracts corrupt inputs and errors while generating an indistinguishable view for all but the last round.



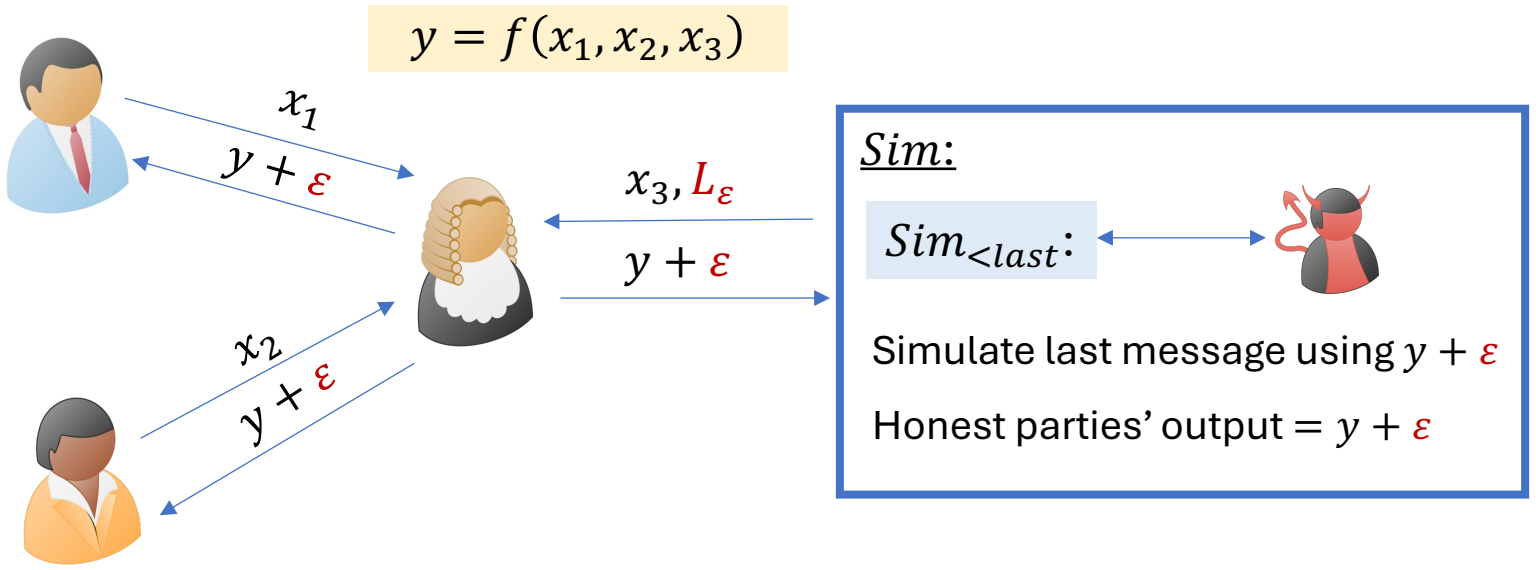
Example 2: Degree-Two Computations

$$f_{A,B}(x_1, x_2, x_3) = x_1 . x_2 + A . x_3 + B$$

Additive error ε can be introduced

Assuming x_1, x_2, x_3 are secret shared, while A, B are constants

How to Simulate?



How is an MPC for such simple functions useful?

Computation of [Groth'16] Proofs can be expressed as a degree-2 computation over the extended witness

t −Zero-Knowledge is different from standard malicious security

In t −zero-knowledge, the simulator does not have oracle access to the proof functionality. It only learns a bit indicating validity of joint witness.

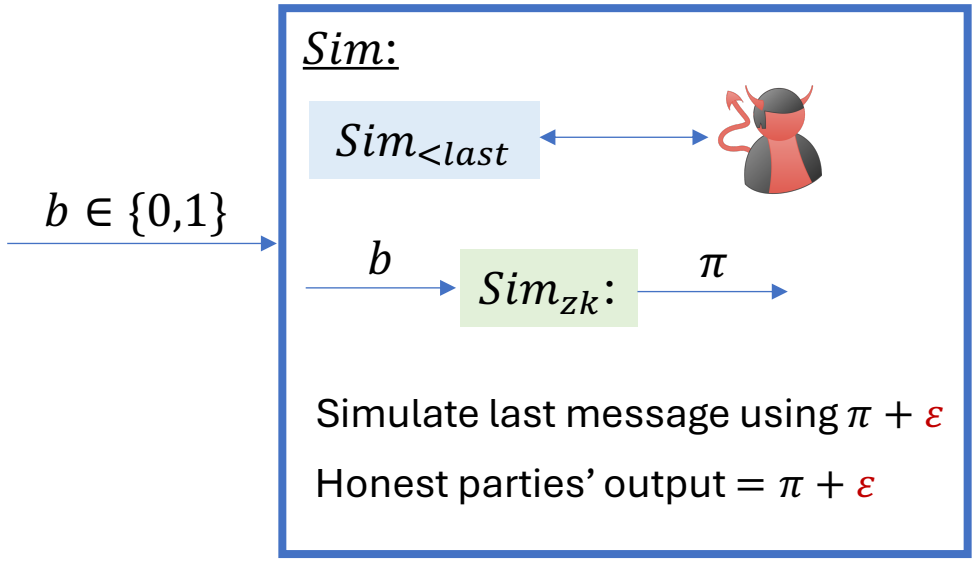
Maliciously Secure Collaborative [Groth16] Proof Generation

Sim_{zk}: this simulator exists from zero-knowledge property of Groth16 zk-SNARKs

How to Simulate?

$$b =^{?}= R_L(x, w_1, w_2, w_3)$$

(Determined by the MPC for generating extended witness)



Example 3: Special Randomized Functions

$$f(x_1, x_2; r) = r.(x_1.x_2)$$

Assuming x_1, x_2 are arbitrary secret shared inputs,
while r is a secret shared random input

Honestly computed output	Adversarially computed output
$r.(x_1.x_2)$	$r.(x_1.x_2 + \varepsilon_1) + \varepsilon_2$

Both are uniformly distributed.

As a result, given $Sim_{<last}$, the adversary's view in the last round of the MPC can be easily simulated

Example 4: Randomized Encodings

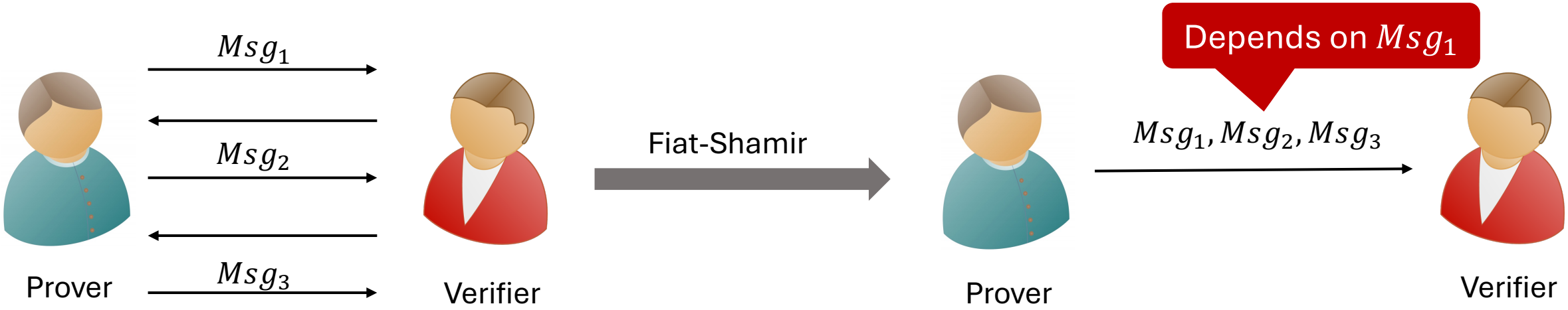
$$f(x_1, x_2, x_3, x_4) = x_1 . x_2 . x_3 . x_4$$

Sequential multiplication over secret shared values

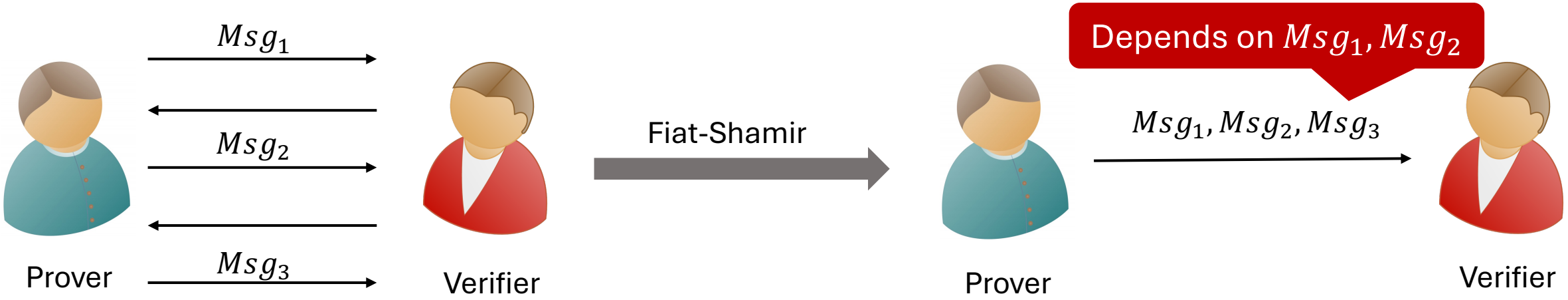
[Bar-Ilan-Beaver'89]: Constant round MPC for sequential multiplications.

Reduces to computing special randomized functions over secret shared values

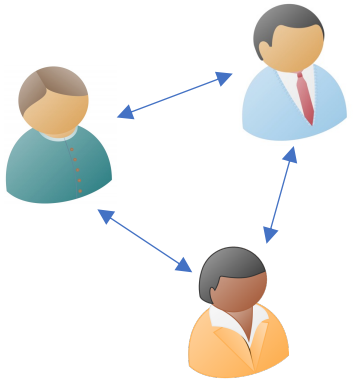
zk-SNARKs in ROM



zk-SNARKs in ROM



Collaborative Variants of zk-SNARKs in ROM



Provers

Run an MPC to compute Msg_1

Reconstruct Msg_1 to query RO

Run an MPC to compute Msg_2

Reconstruct Msg_2 to query RO

.

.

.

If each Msg_i can be computed using one of the example functions, then this overall yields an L −Maliciously secure MPC for reactive functions

Our Results

Bulletproofs: Short Proofs for Confidential Transactions and More

Benedikt Bünz*[1], Jonathan Bootle†[2], Dan Boneh‡[1],
Andrew Poelstra§[3], Pieter Wuille¶[3], and Greg Maxwell‖

[1]Stanford University
[2]University College London
[3]Blockstream

Full Version**

Abstract

We propose Bulletproofs, a new non-interactive zero-knowledge proof protocol with very short proofs and without a trusted setup; the proof size is only logarithmic in the witness size. Bulletproofs are especially well suited for efficient range proofs on committed values: they enable proving that a committed value is in a range using only $2\log_2(n) + 9$ group and field elements,

Collaborative Version: RENEGADE ON CHAIN DARK POOL

PlonK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge

Ariel Gabizon* Aztec
Zachary J. Williamson Aztec
Oana Ciobotaru Pi Squared

July 17, 2025

Abstract

zk-SNARK constructions that utilize an updatable universal structured reference string remove one of the main obstacles in deploying zk-SNARKs[GKM+]. The important work of Maller et al. [MBKM] presented Sonic - the first potentially practical zk-SNARK with fully succinct verification for general arithmetic circuits

Collaborative Version: [Ozdemir-Boneh'22]

We can get free malicious security for these collaborative zk-SNARKs

Open Problems

- What other functions can be computed using semi-honest MPC with free malicious security?
- Can these ideas be extended to the dishonest-majority setting?
- Exploring other types of collaborative zk-SNARKs (e.g., those based on packed secret-sharing).
- Investigating applications of collaborative zk-SNARKs.

Thanks!