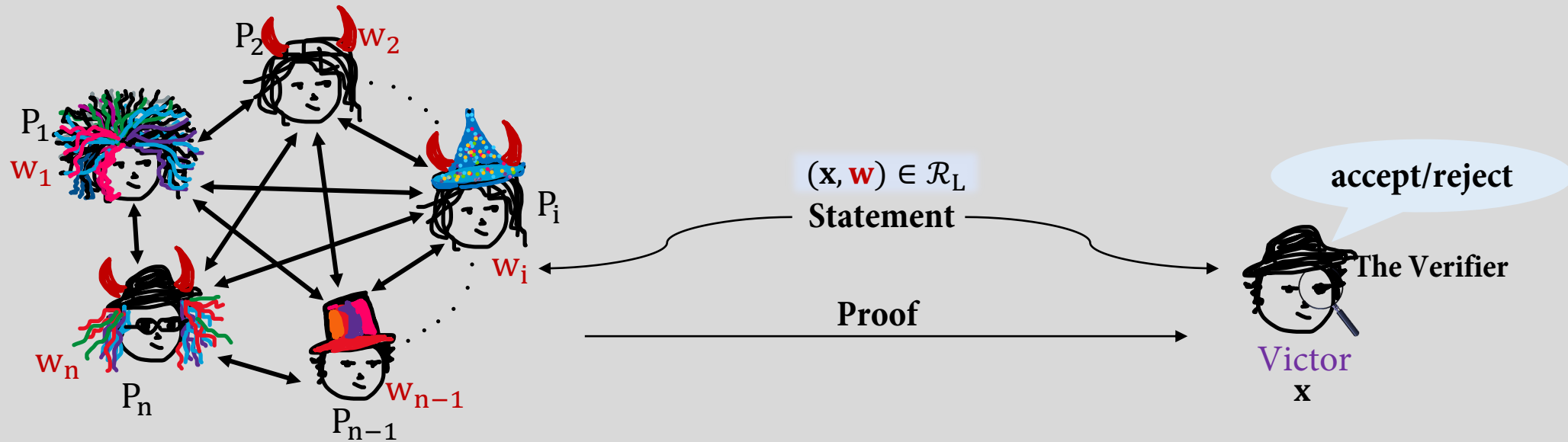




Malicious Security in Collaborative zkSNARKs



Sruthi Sekar



CRYPTO RESEARCHERS' MEET, IIT MADRAS

Joint Work with..



Sanjam Garg
UC Berkeley



Aarushi Goel
Rutgers University



Abhishek Jain
Johns Hopkins and
NTT Research



Bhaskar Roberts
UC Berkeley

Malicious Security in Collaborative zkSNARKs: More than Meets the Eye [CRYPTO 2025]

zkSNARKs: Zero-Knowledge Succinct Non-interactive Arguments of Knowledge



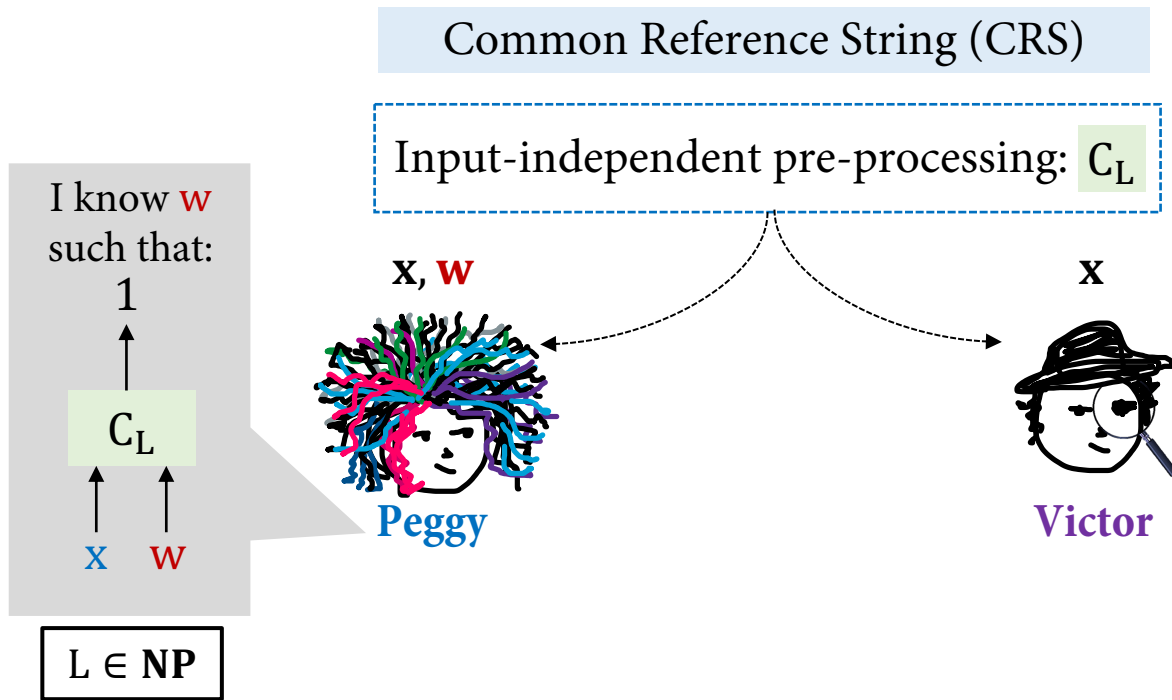
Peggy
(Prover)



Victor
(Verifier)

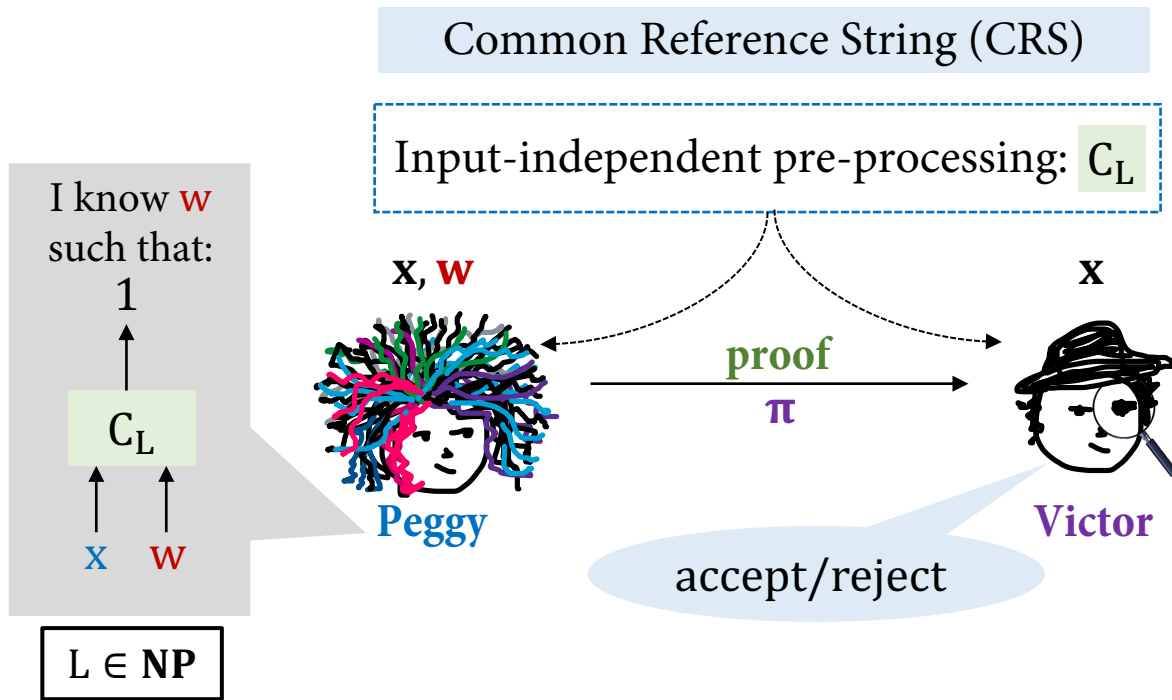
Intuition: *succinct* proof that a certain statement is true.
E.g.: “I know an m such that $\text{SHA256}(m) = 0$ ”

zkSNARKs: Zero-Knowledge Succinct Non-interactive Arguments of Knowledge



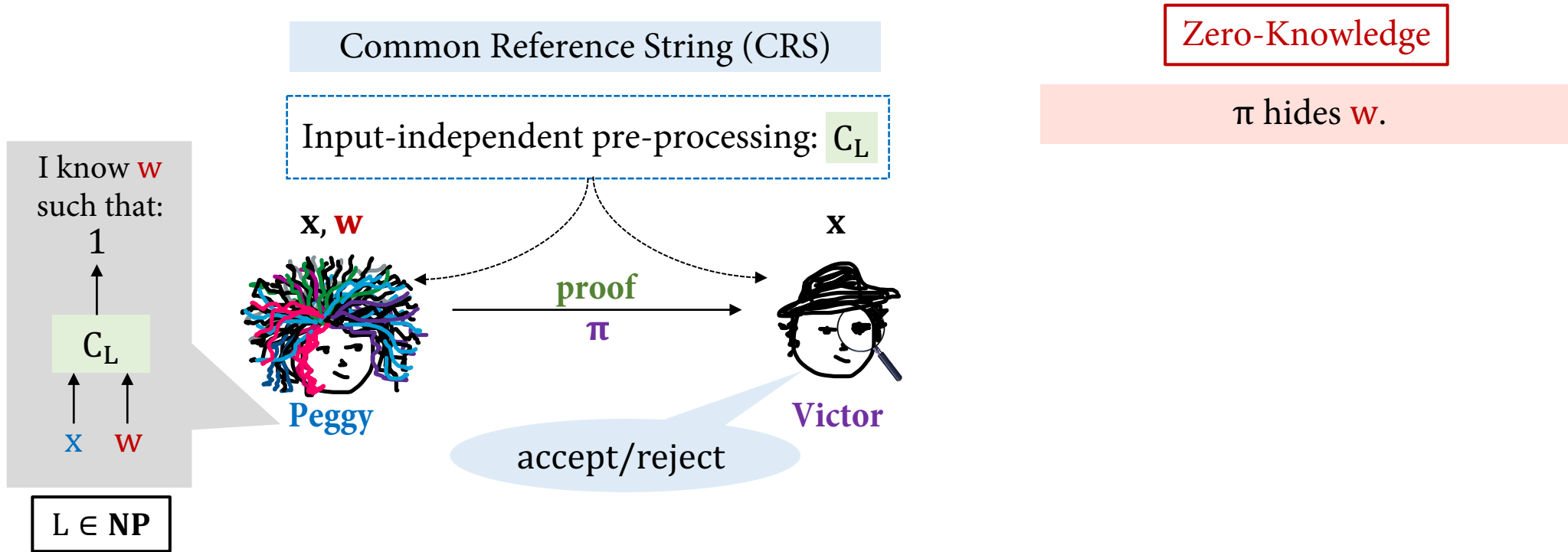
Intuition: *succinct* proof that a certain statement is true.
E.g.: “I know an m such that $SHA256(m) = 0$ ”

zkSNARKs: Zero-Knowledge Succinct Non-interactive Arguments of Knowledge



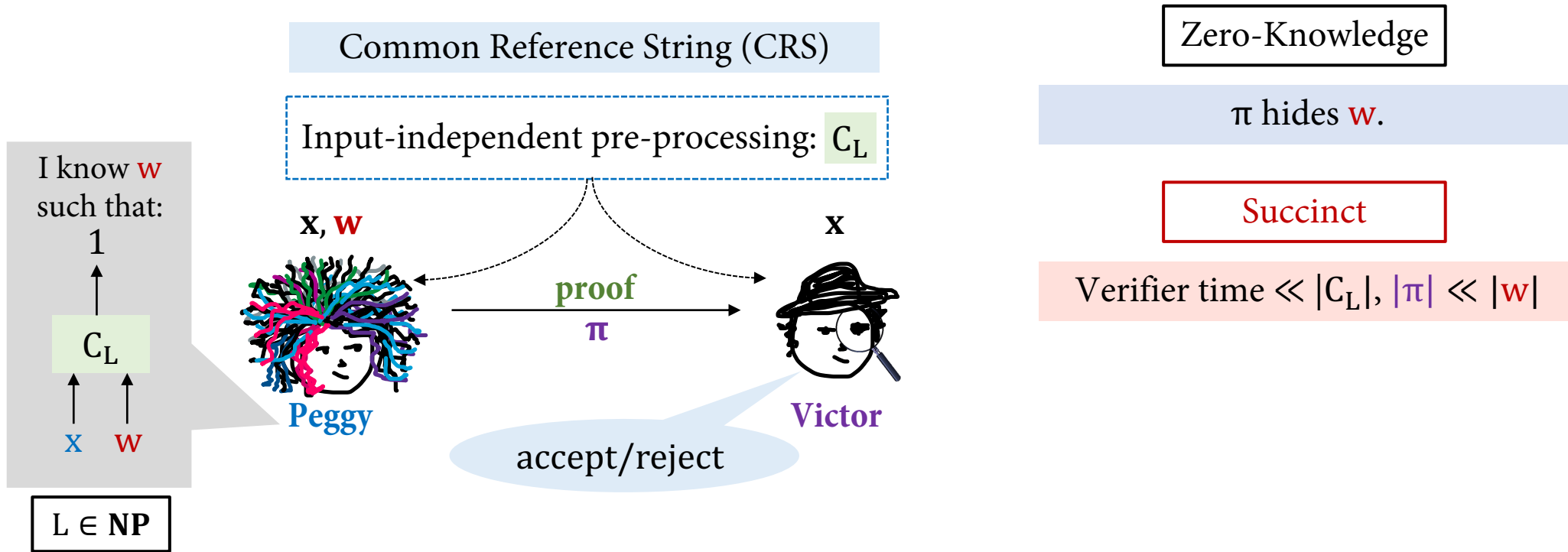
Intuition: *succinct* proof that a certain statement is true.
E.g.: "I know an m such that $SHA256(m) = 0$ "

zkSNARKs: **Zero-Knowledge** Succinct Non-interactive Arguments of Knowledge



Intuition: *succinct* proof that a certain statement is true.
E.g.: "I know an m such that $\text{SHA256}(m) = 0$ "

zkSNARKs: Zero-Knowledge Succinct Non-interactive Arguments of Knowledge

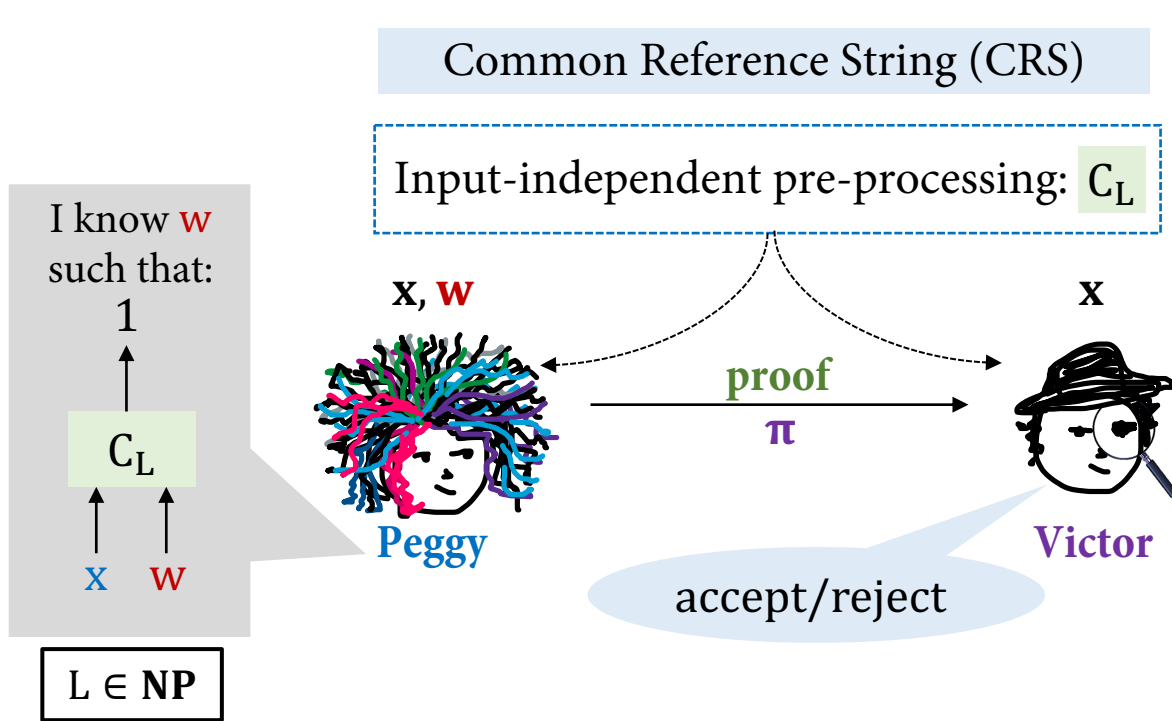


Intuition: *succinct* proof that a certain statement is true.

E.g.: "I know an m such that $SHA256(m) = 0$ "

→ **Short Proof and fast to verify!**

zkSNARKs: Zero-Knowledge Succinct Non-interactive Arguments of Knowledge



Intuition: *succinct* proof that a certain statement is true.
E.g.: "I know an m such that $\text{SHA256}(m) = 0$ "

Zero-Knowledge

π hides w .

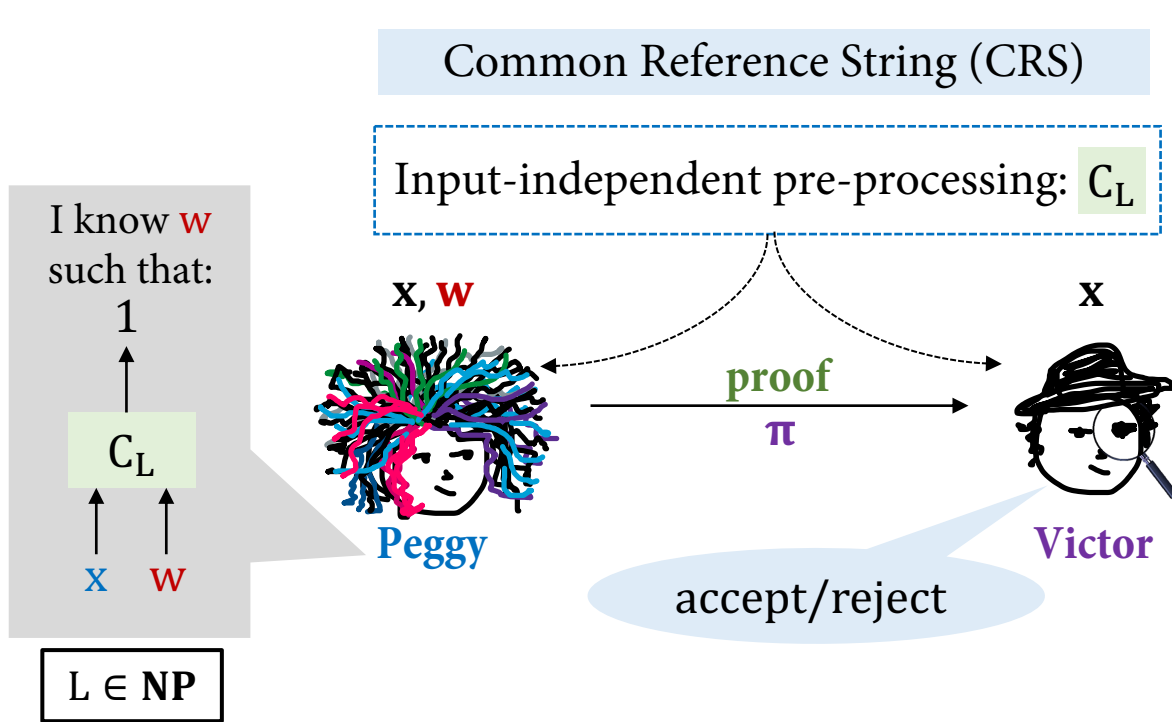
Succinct

Verifier time $\ll |C_L|$, $|\pi| \ll |w|$

Non-interactive

Anyone can verify

zkSNARKs: Zero-Knowledge Succinct Non-interactive Arguments of Knowledge



Intuition: *succinct* proof that a certain statement is true.
E.g.: "I know an m such that $\text{SHA256}(m) = 0$ "

Zero-Knowledge

π hides w .

Succinct

Verifier time $\ll |C_L|$, $|\pi| \ll |w|$

Non-interactive

Anyone can verify

Soundness

π will not be accepted if $\nexists w$.

What If Witness Is Split Across Multiple Provers?

Aggregate Healthcare Statistics

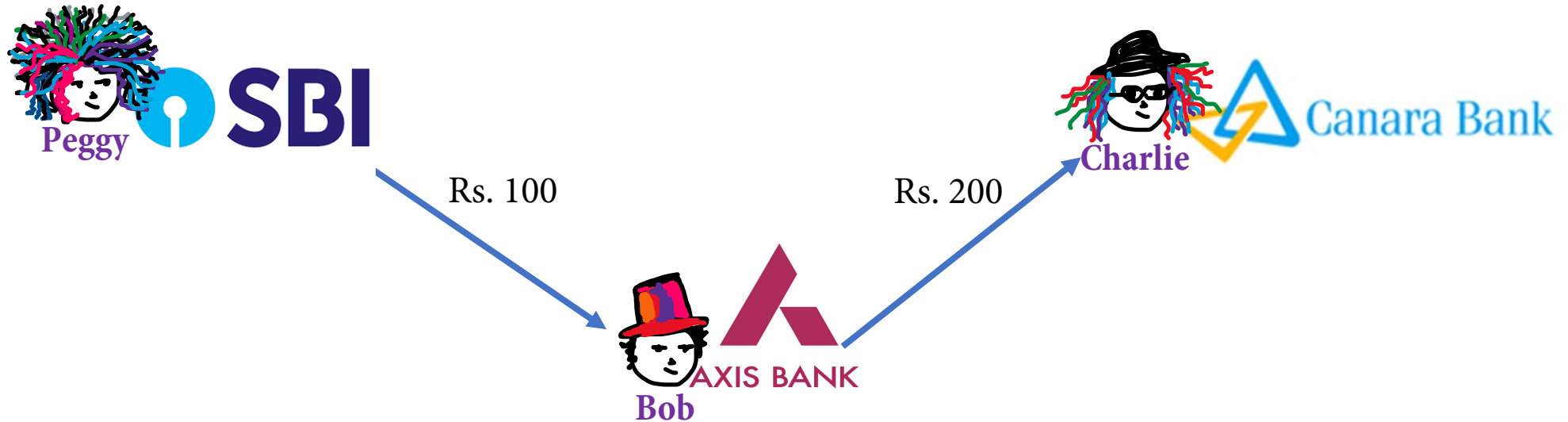


Example:

Collectively prove that they do not charge the same price for insured and uninsured patients **without revealing the patient records.**

What If Witness Is Split Across Multiple Provers?

Private Audit of Financial Transactions

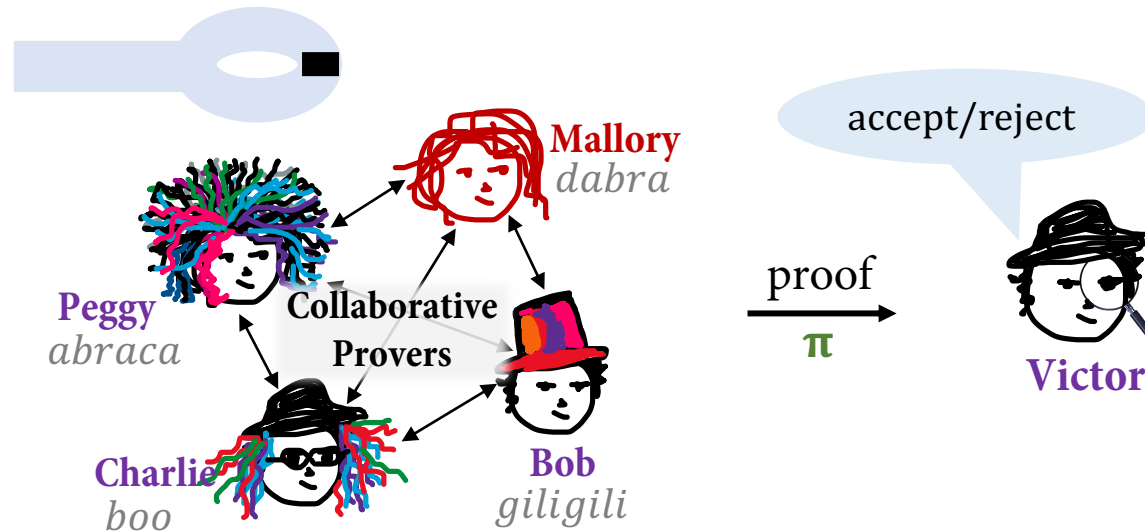


Example:

Collectively prove Peggy transferred money to Bob and Bob to Charlie
without revealing other financial transactions.

Collaborative zkSNARKs

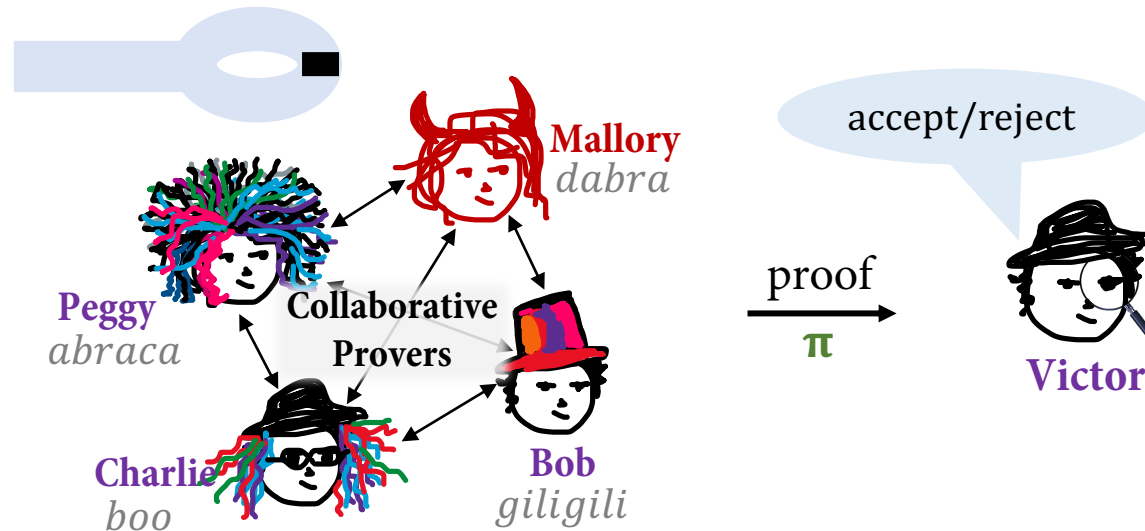
[Ozdemir & Boneh, Usenix Security 2022]



The secret password is held in parts by four people:
Peggy, Charlie, Bob and **Mallory** who can collectively prove that they together hold a secret password.

Collaborative zkSNARKs

[Ozdemir & Boneh, Usenix Security 2022]

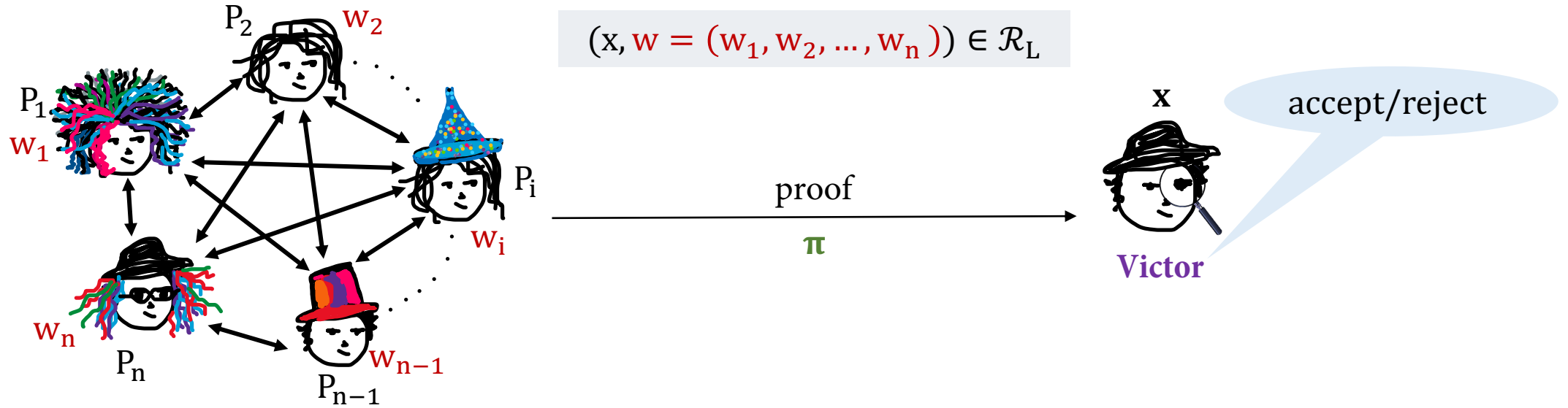


The secret password is held in parts by four people:
Peggy, Charlie, Bob and **Mallory** who can collectively prove that they together hold a secret password.

Zero-Knowledge: Against Victor, and against a malicious **Mallory** who is trying to learn secrets of **Peggy, Charlie** and **Bob**.

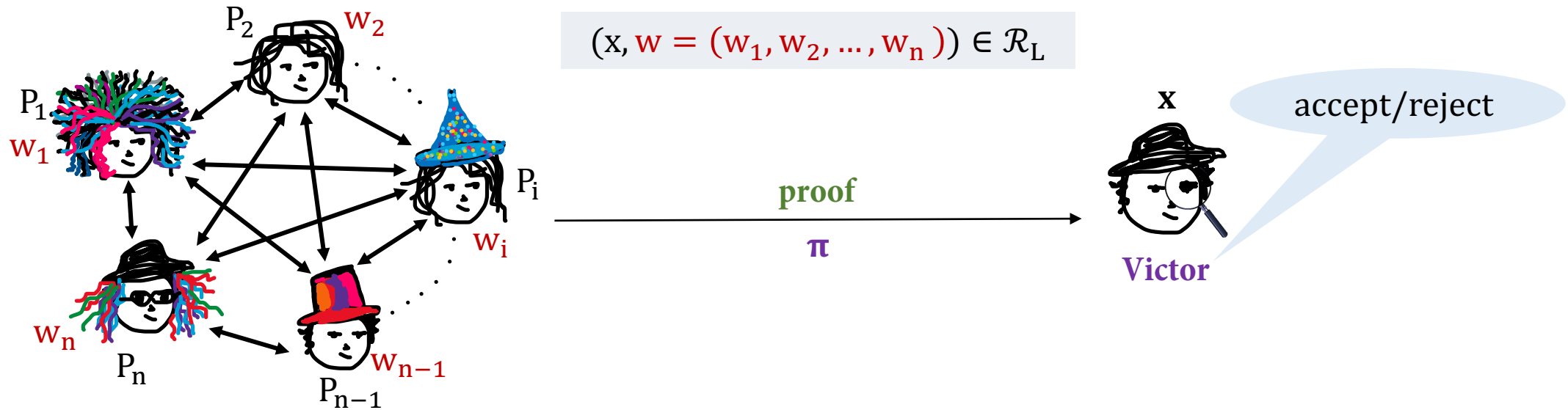
Collaborative zkSNARKs

[Ozdemir & Boneh, Usenix Security 2022]



Collaborative zkSNARKs

[Ozdemir & Boneh, Usenix Security 2022]



Completeness: $(x, w) \in \mathcal{R}_L \Rightarrow$ Victor always accepts

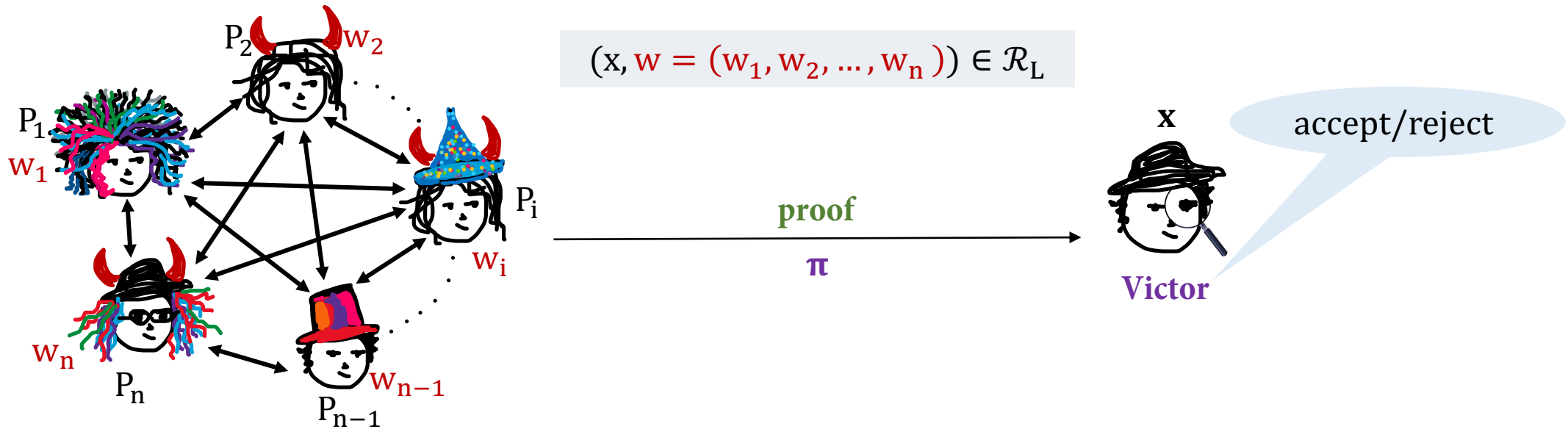
Succinctness: $|\pi|$ and verifier time are $o(|\mathcal{R}_L|)$.

Soundness*: $(x, w) \notin \mathcal{R}_L \Rightarrow$ Victor will accept only with negligible probability.

*it actually satisfies knowledge soundness, but for this talk not defining it.

Collaborative zkSNARKs

[Ozdemir & Boneh, Usenix Security 2022]



Completeness: $(x, w) \in \mathcal{R}_L \Rightarrow$ Victor always accepts

Succinctness: $|\pi|$ and verifier time are $o(|\mathcal{R}_L|)$.

Soundness*: $(x, w) \notin \mathcal{R}_L \Rightarrow$ Victor will accept only with negligible probability.

t-zero-knowledge: A subset of $\leq t$ parties learn nothing about witness of other parties.
(except whether the combined witness is valid)

*it actually satisfies knowledge soundness, but for this talk not defining it.

Constructing Collaborative zkSNARKs



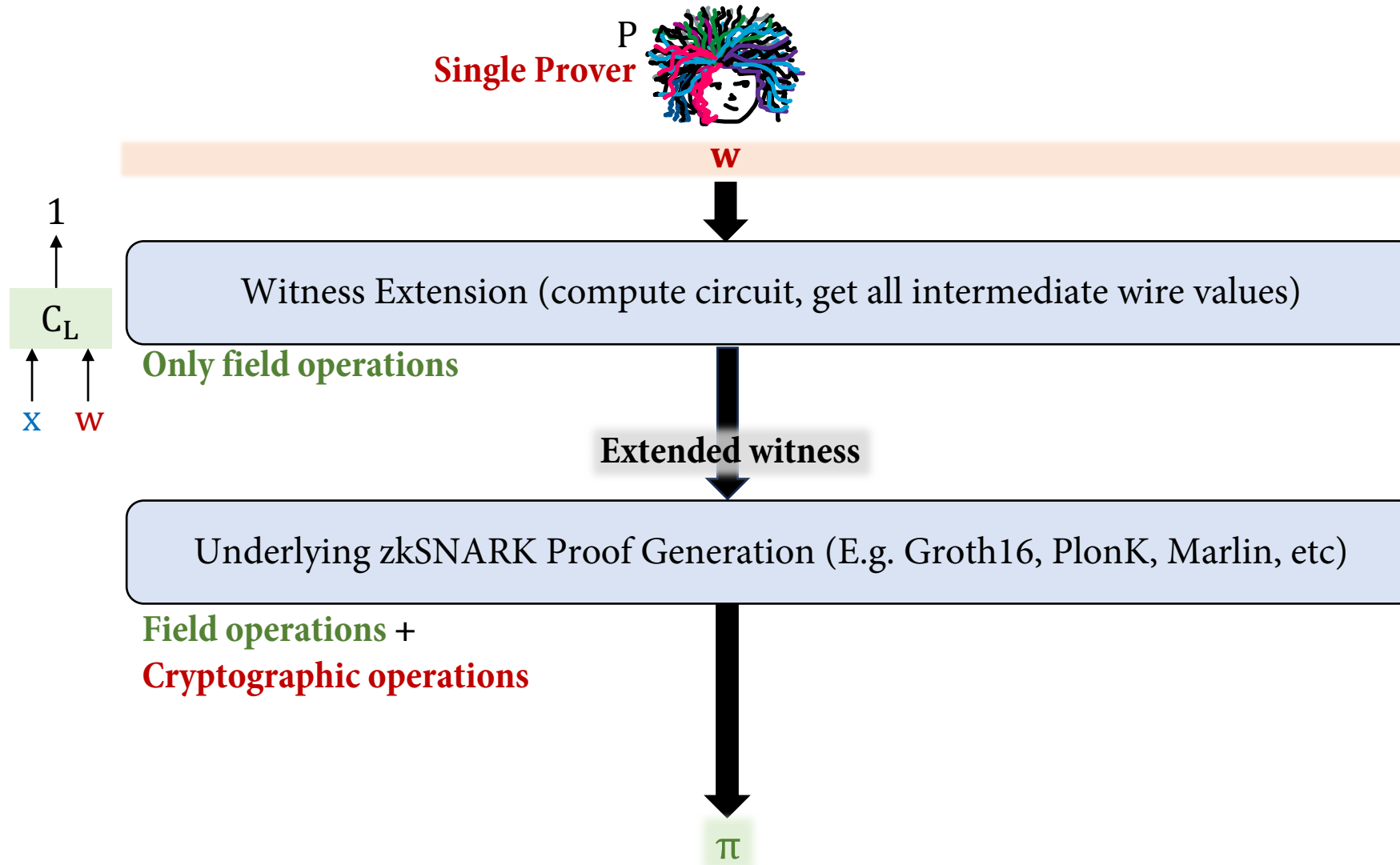
Compute a zk-SNARK using a secure multiparty computation protocol?



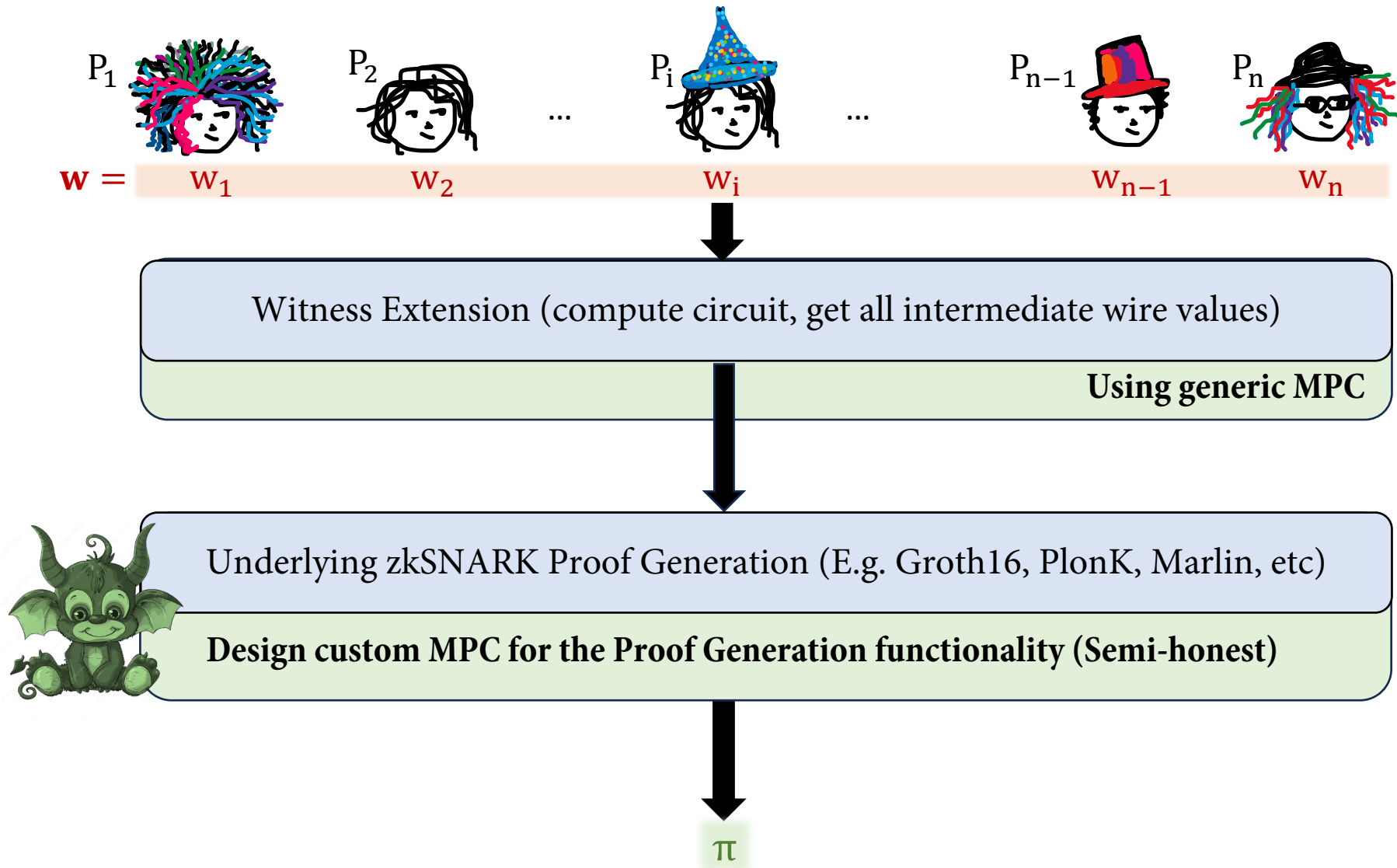
Where is the non-triviality?

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

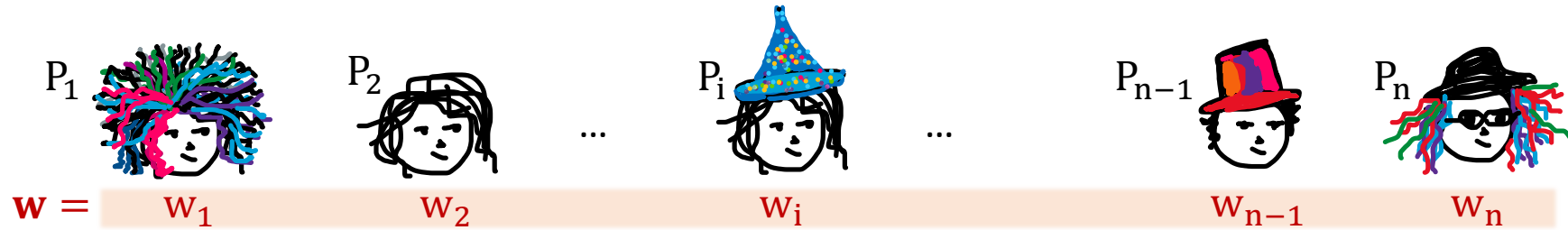
Design Blueprint of zkSNARKs



Design Blueprint of Collaborative zkSNARKs



Design Blueprint of Collaborative zkSNARKs



Witness Extension (compute circuit, get all intermediate wire values)

Using generic MPC

Main novelty of all prior work lies here

Underlying zkSNARK Proof Generation (E.g. Groth16, PlonK, Marlin, etc)

Design custom MPC for the Proof Generation functionality (Semi-honest)



π

Custom MPC for Computing Proof given Extended Witness



Underlying zkSNARK Proof Generation (E.g. Groth16, PlonK, Marlin, etc)

Design custom MPC for the Proof Generation functionality (Semi-honest)

Design Custom semi-honest MPCs specifically for following functionalities

Multi-Scalar Multiplications (MSM)

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

Fast Fourier Transform (FFT)

For converting between coefficient and evaluation representation of polynomials

Partial Products

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

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-Goel-Jain-Policharla-S'23]

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

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

[Garg-Goel-Koloneos-Sinha'25]

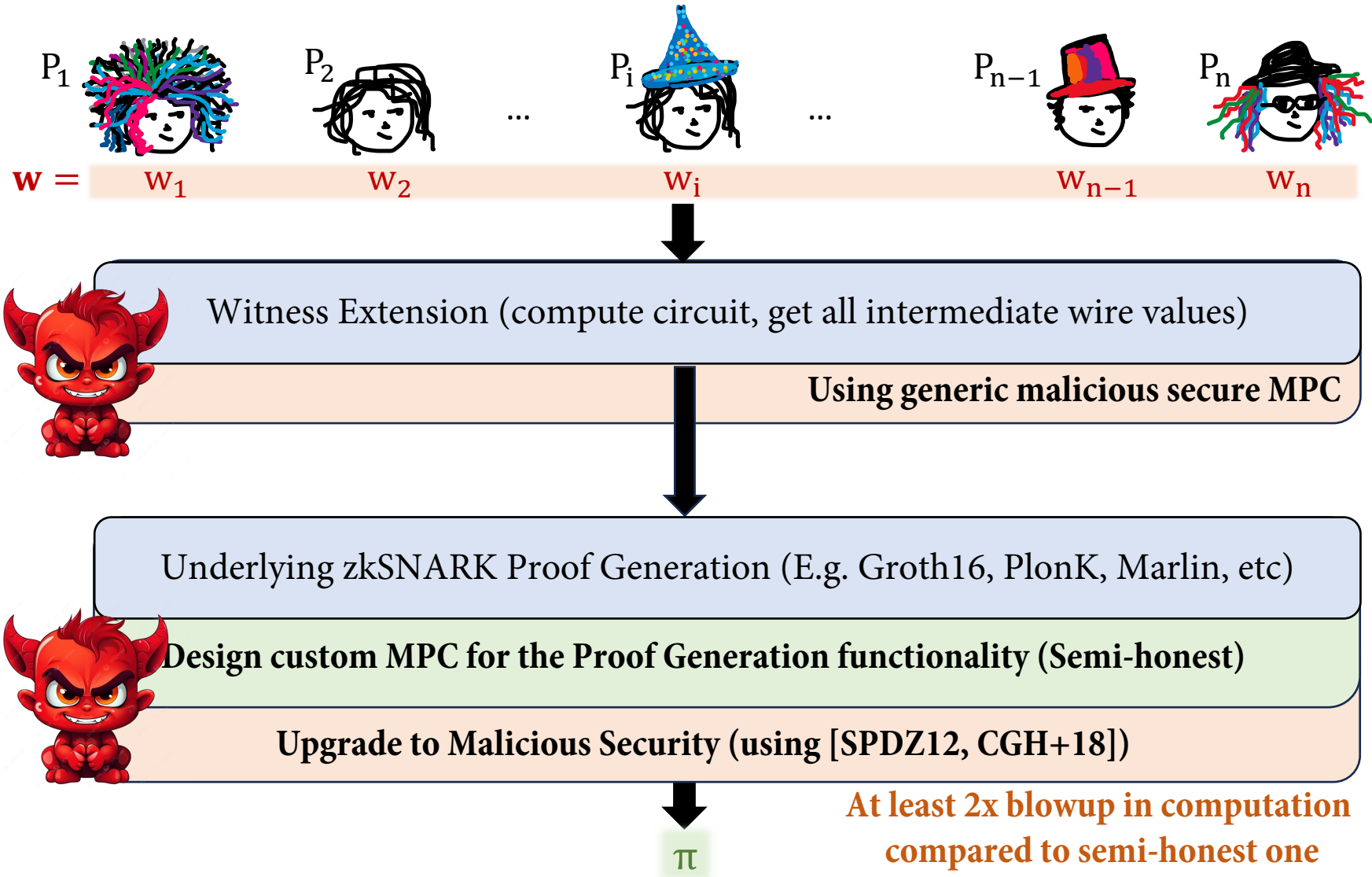
[Fang-Garg-Roberts-Wu-Zhang'25]

Industry

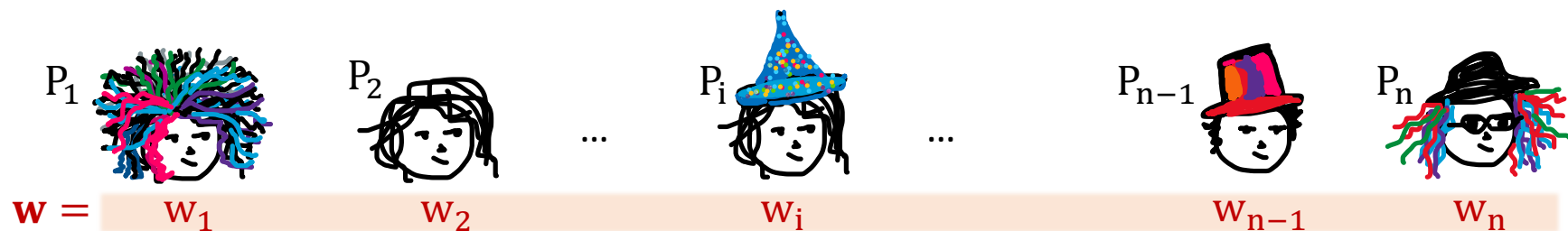


TACED

Collaborative zkSNARKs: Malicious Security



Collaborative zkSNARKs: Malicious Security



Witness Extension (compute circuit, get all intermediate wire values)

[Garg-Goel-Jain-S-Roberts 25]

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

Using generic malicious secure MPC

! Else, loss of input privacy!!

Underlying zkSNARK Proof Generation (E.g. Groth16, PlonK, Marlin, etc)

Design custom MPC for the Proof Generation functionality (Semi-honest)

Upgrade to Malicious Security (using [SPDZ12, CGH+18])

π

At least 2x blowup in computation compared to semi-honest one

Our Main Contribution [Garg-Goel-Jain-**S**-Roberts 25]

Underlying zkSNARK Proof Generation (E.g. Groth16, PlonK, Marlin, etc)



Design custom MPC for the Proof Generation functionality (Semi-honest)

Upgrade to Malicious Security (using [SPDZ12, CGH+18])

At least 2x blowup in computation
compared to semi-honest one

Can we minimize (or eliminate this overhead?)



Semi-honest
Collaborative zk-SNARK

=



Maliciously Secure
Collaborative zk-SNARK

Additive Attack Paradigm

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

Honest majority
setting

In most *secret-sharing based* semi-honest MPC protocols,
the *scope of malicious attacks are limited*.

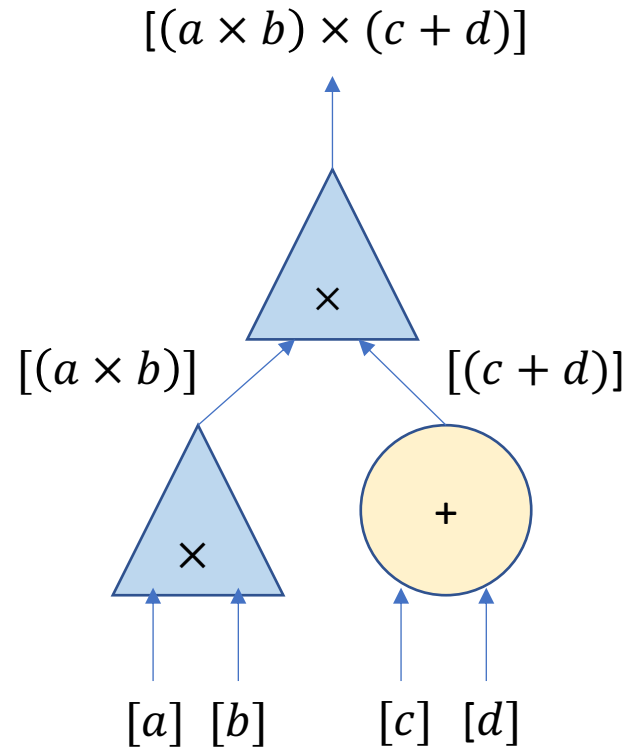


Additive Attack Paradigm

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

Honest majority
setting

In most *secret-sharing based* semi-honest MPC protocols, the scope of malicious attacks are limited.

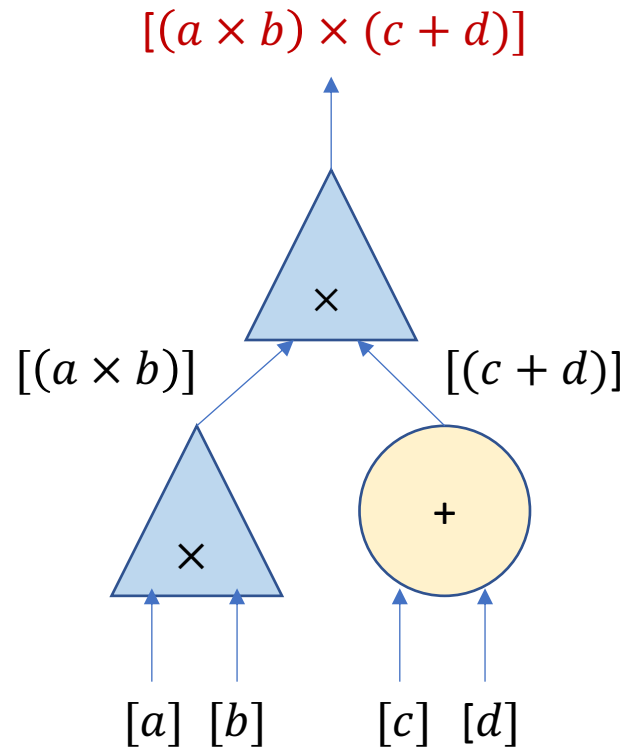


Additive Attack Paradigm

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

Honest majority
setting

In most *secret-sharing based* semi-honest MPC protocols,
the **scope of malicious attacks are limited.**



1

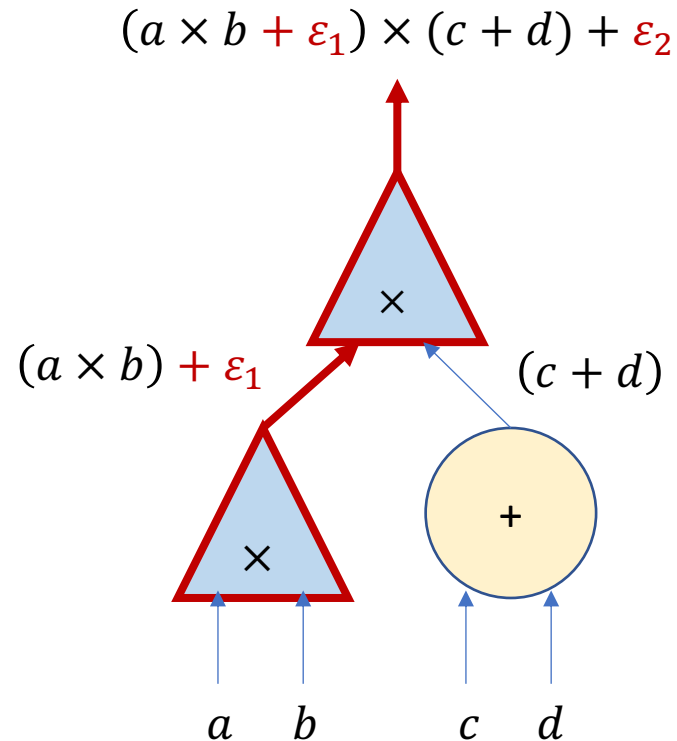
Private until output reconstruction

Additive Attack Paradigm

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

Honest majority
setting

In most *secret-sharing based* semi-honest MPC protocols,
the **scope of malicious attacks are limited.**



1

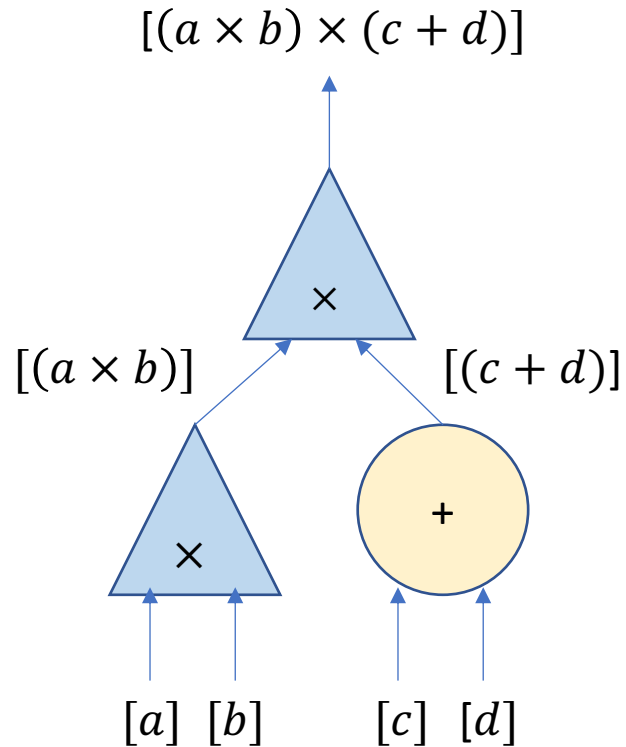
Private until output reconstruction

2

Injecting arbitrary additive errors

Semi-honest $\rightarrow$ Malicious Security

[General Strategy]



Use a semi-honest MPC to compute
secret shares of the output (don't reconstruct yet)



Verify correctness of computation
(by checking consistency of two parallel runs of the
semi-honest protocol before reconstruction)

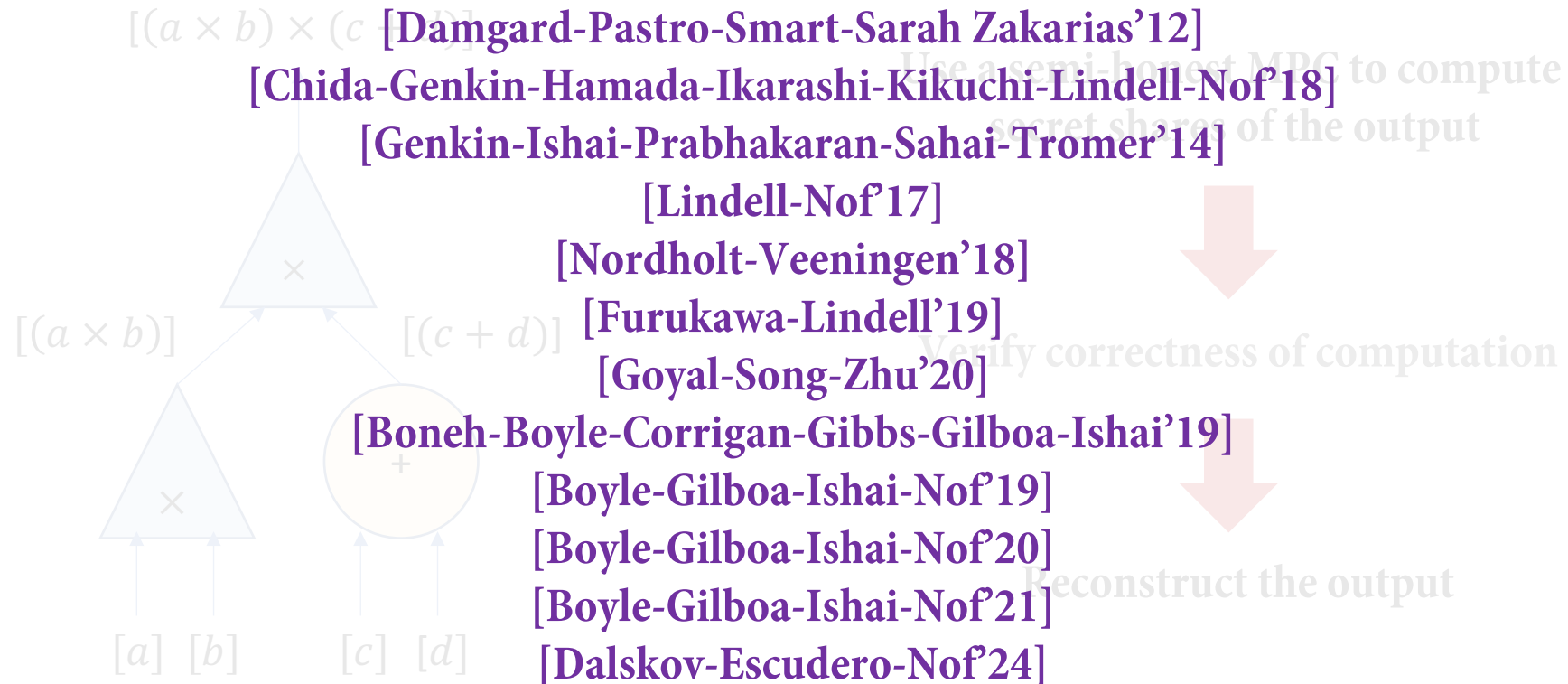


Reconstruct the output

$\approx 2\times$ blowup in computation
compared to semi-honest one

Semi-honest $\rightarrow$ Malicious Security

[General Strategy]



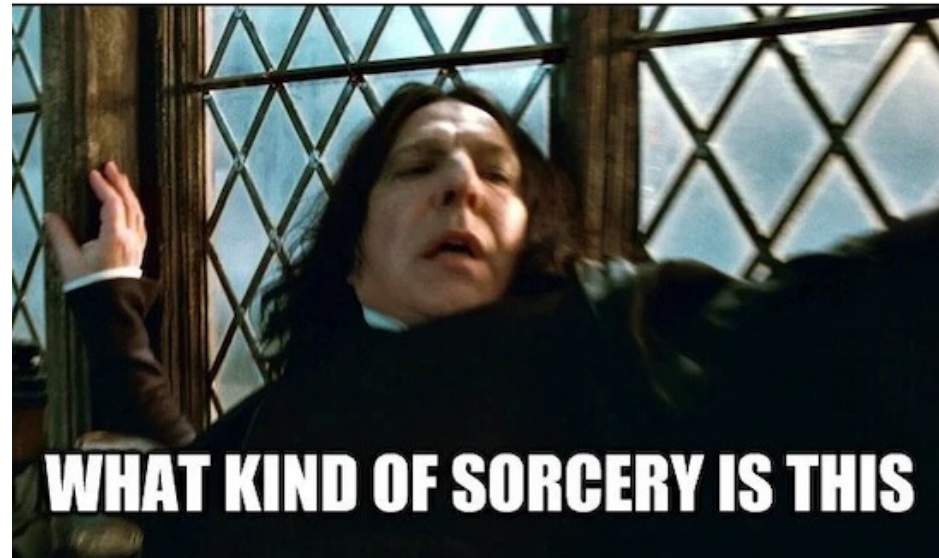


Can we get malicious security for free?

In the honest-majority setting



[Garg-Goel-Jain-**S**-Roberts'25]

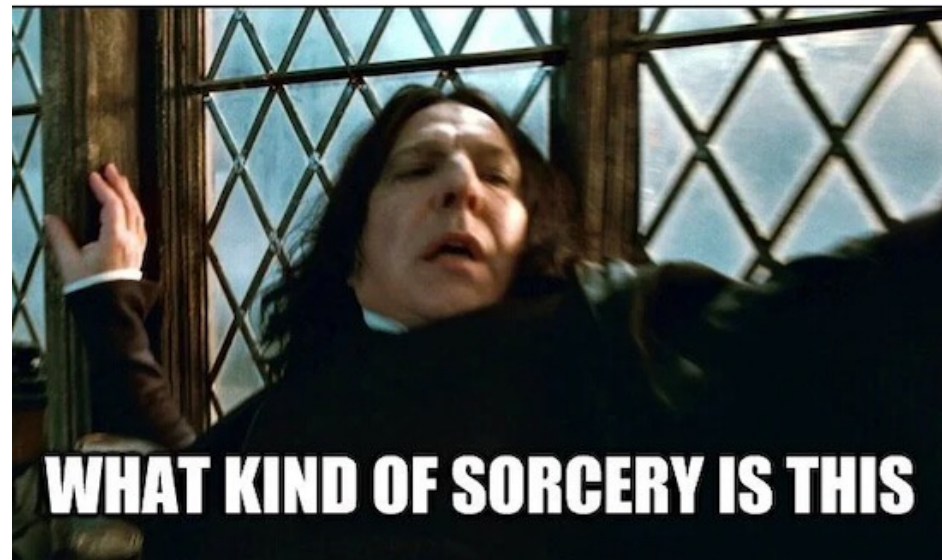




Can we get malicious security for free?

If the adversary can **inject errors**,
how can we even hope to get output correctness?

[Garg-Goel-Jain-S-Roberts'25]



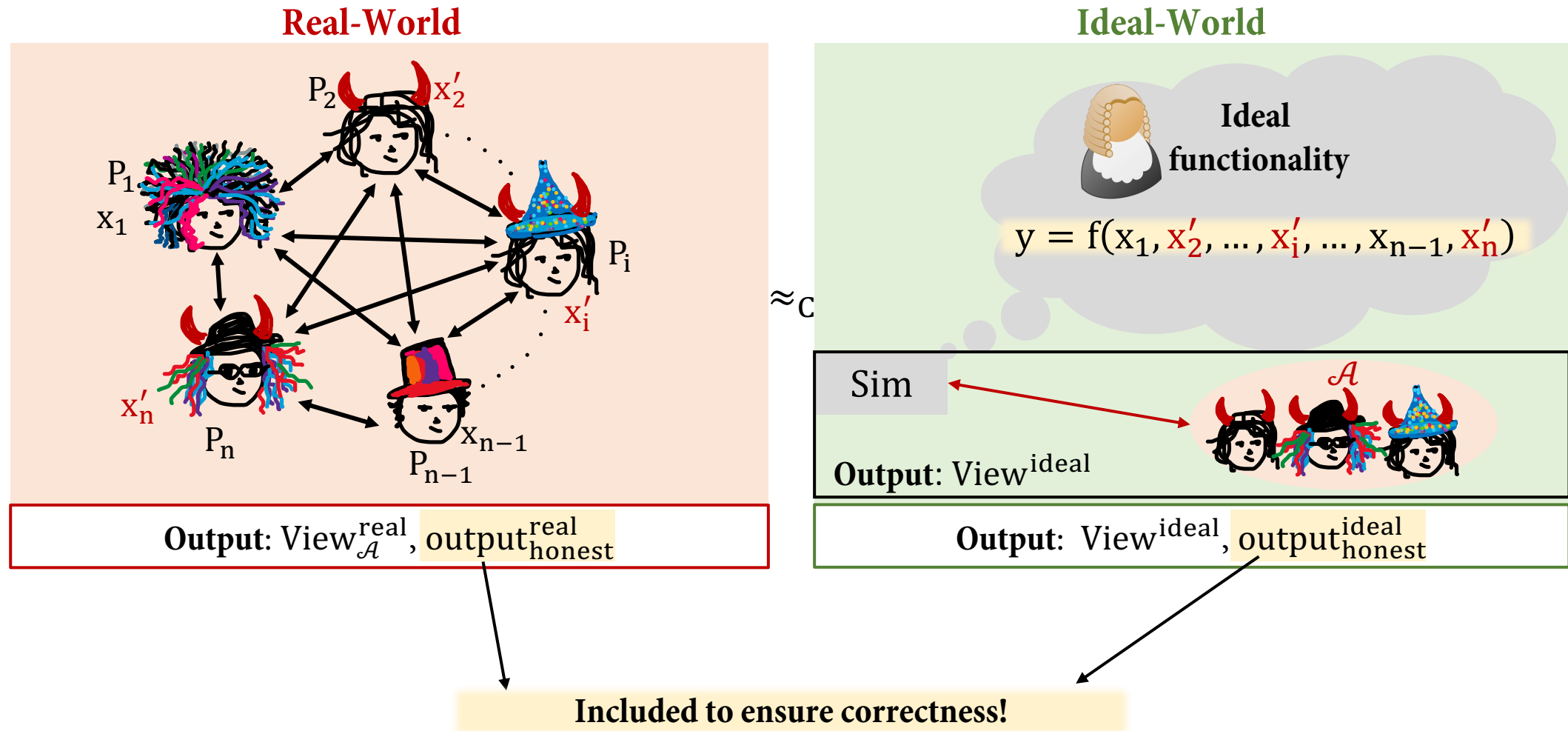
Recall: Want to compute zkSNARKs using MPC



Proofs are self-verifying

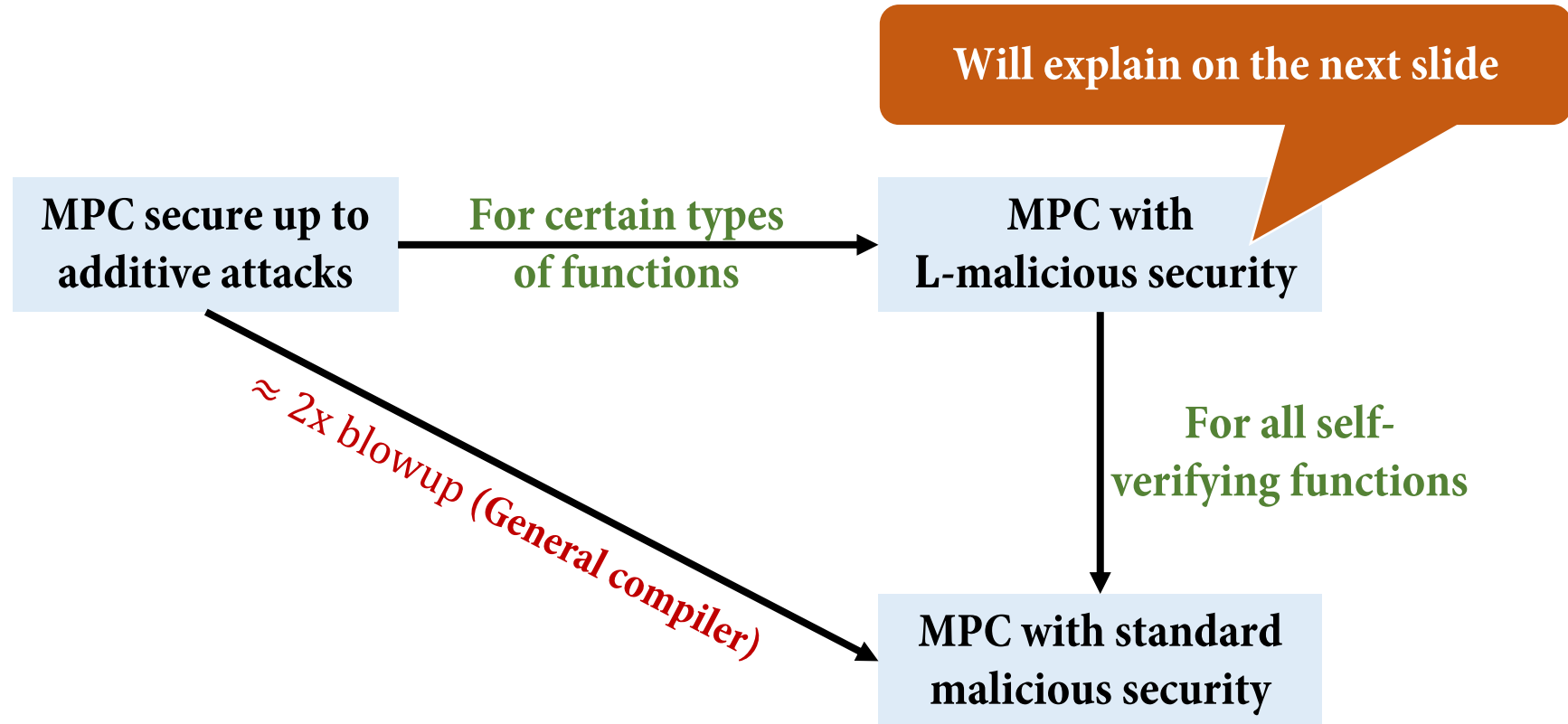
MPC does not need to enforce correctness of computed proof
We can simply check whether the proof successfully verifies

Malicious Security for MPCs: Definition



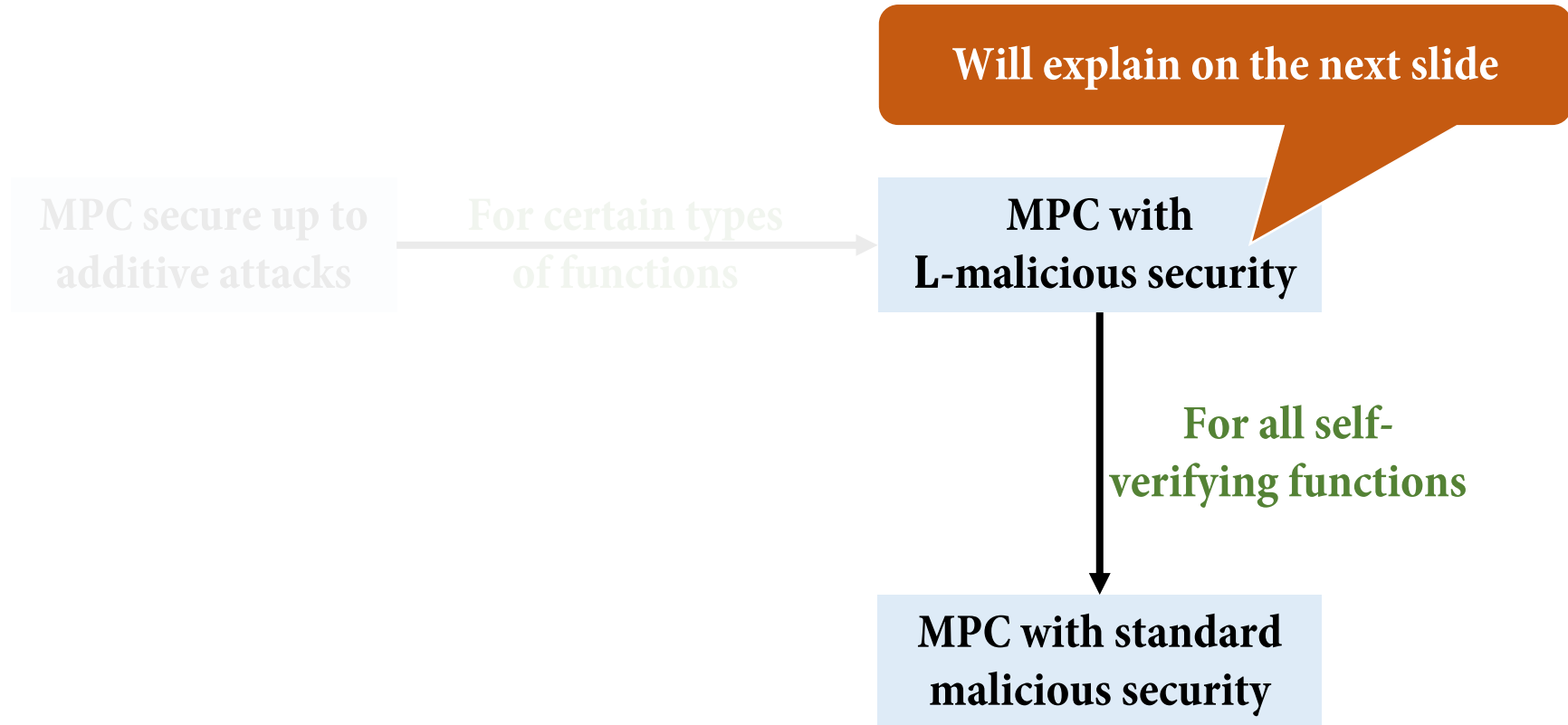


Our Key Idea



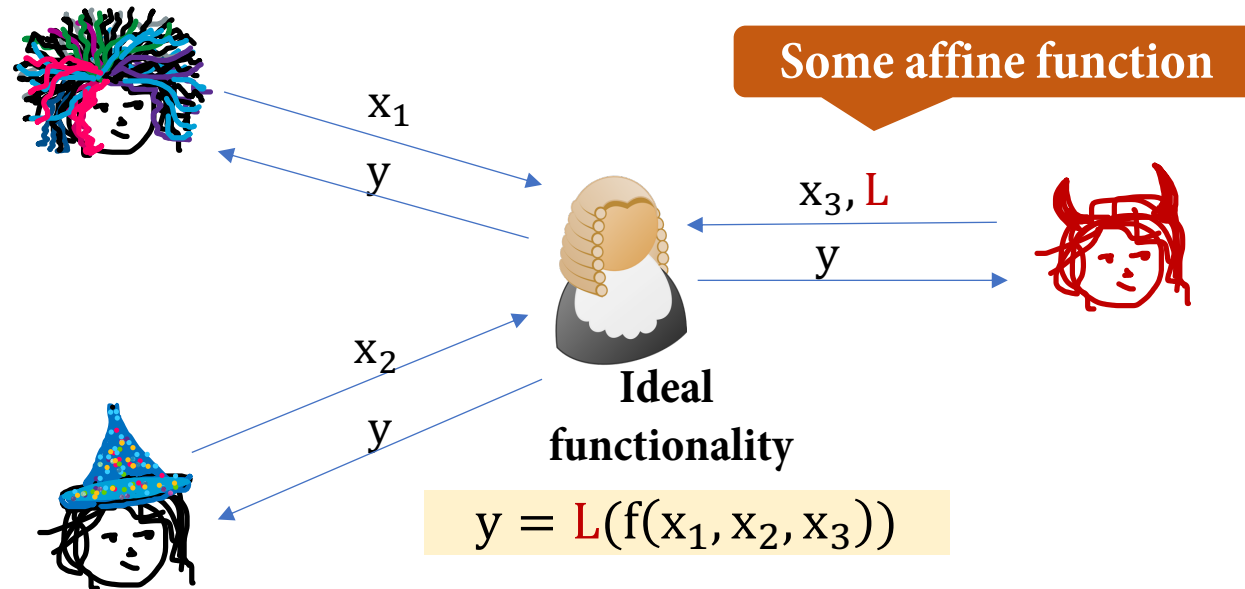


Our Key Idea



Relaxed Notion of Malicious Secure MPC

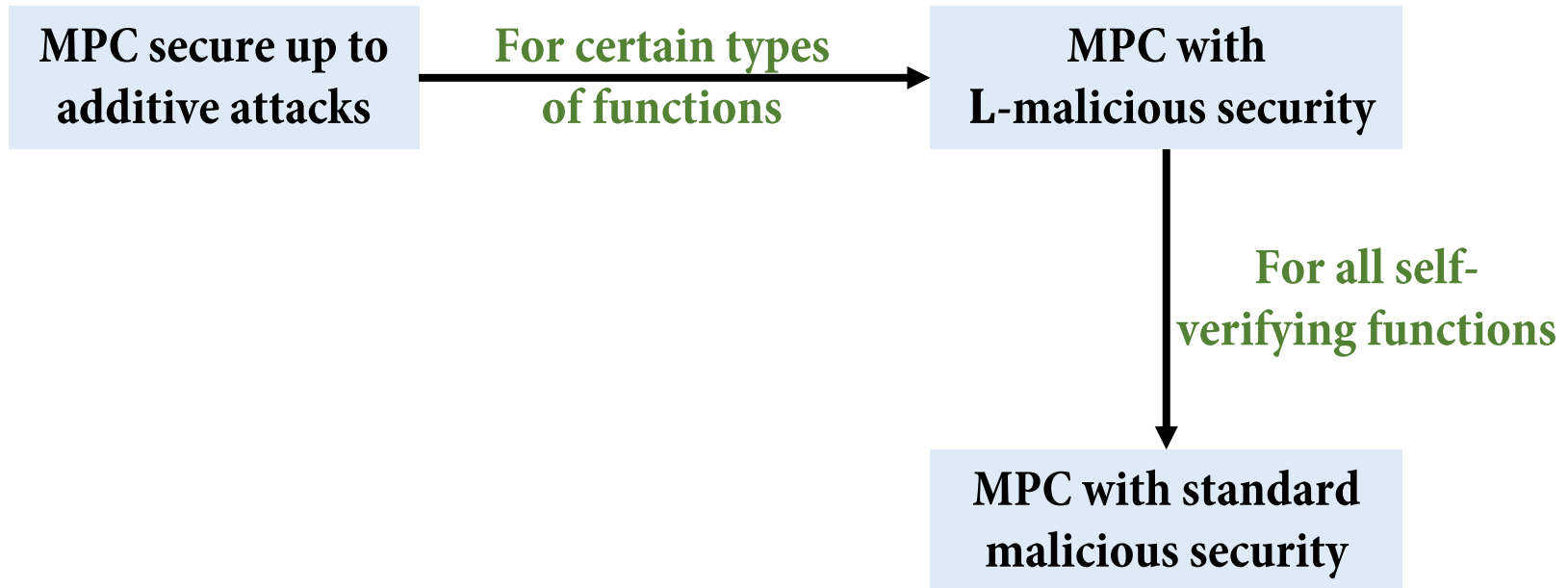
L-Malicious Security



- Note, the relaxation to **L-malicious security** only affects correctness!
- When computing proofs, **an L-Maliciously Secure** MPC implies standard malicious security.

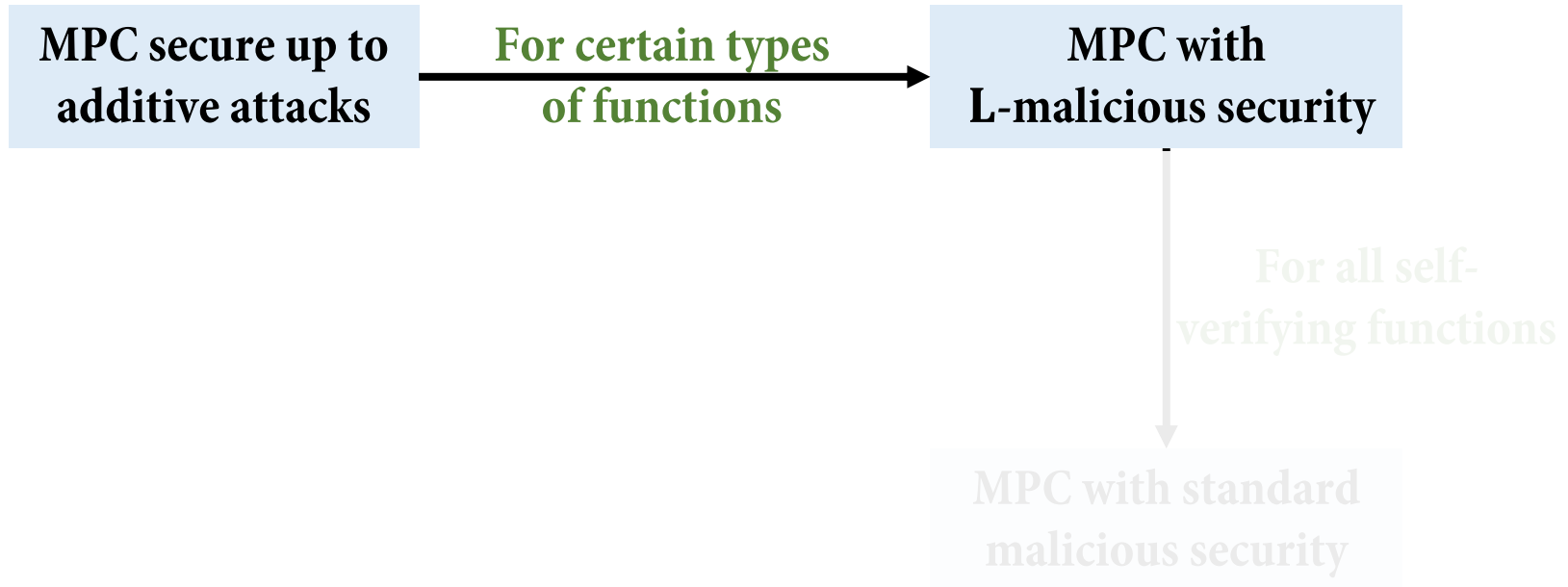


Our Key Idea





Our Key Idea



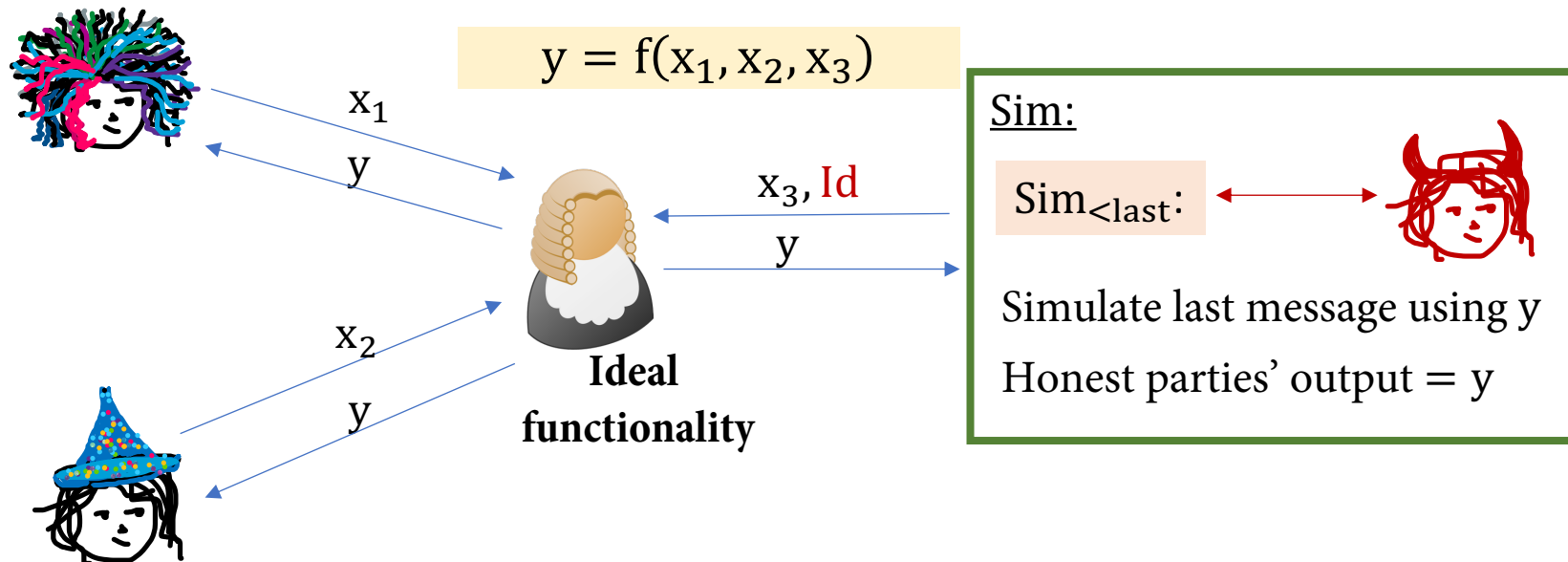
Example I: Affine Functions

$$f_{A,B,C}(x_1, x_2, x_3) = A \cdot x_1 + B \cdot x_2 + C \cdot x_3 + D \quad \text{No errors!}$$

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

How to Simulate?

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



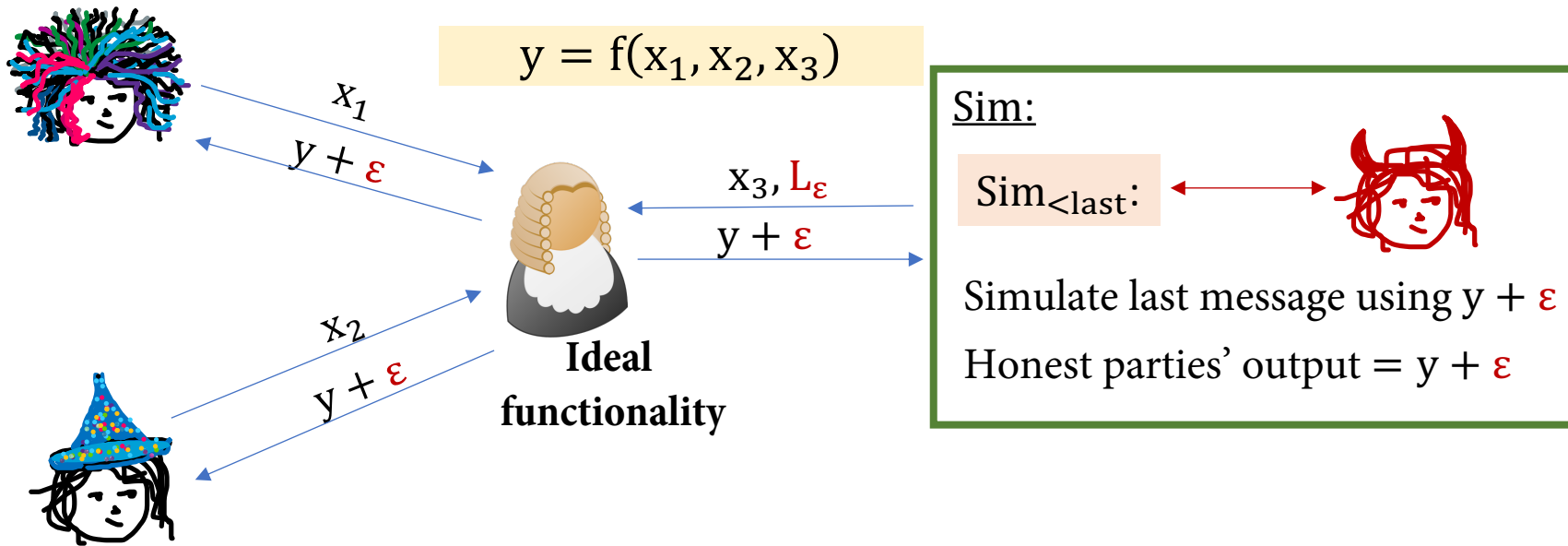
Example II: Degree-Two Computations

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

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.

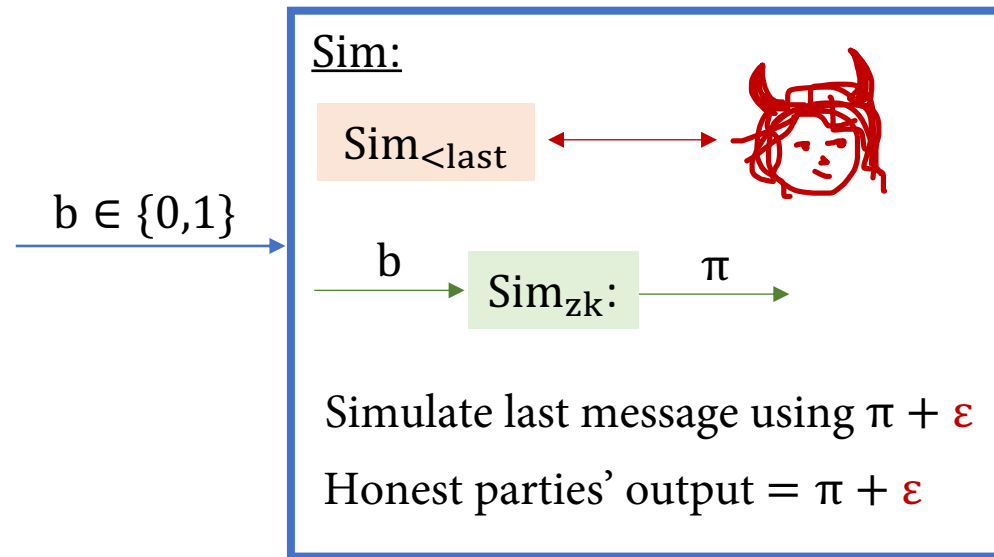
Malicious Secure Collaborative [Groth'16] Proof

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

How to Simulate?

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

(Determined by the MPC for generating extended witness)



Example III: Special Randomized Functions

$$f(x_1, x_2; r) = r \cdot (x_1 \cdot x_2)$$

Assuming x_1, x_2 (arbitrary inputs) and random r are secret-shared

Honestly computed output

$$r \cdot (x_1 \cdot x_2)$$

Adversarially computed output

$$r \cdot (x_1 \cdot x_2 + \epsilon_1) + \epsilon_2$$

Both are uniformly distributed!

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

Example IV: Randomized Encodings

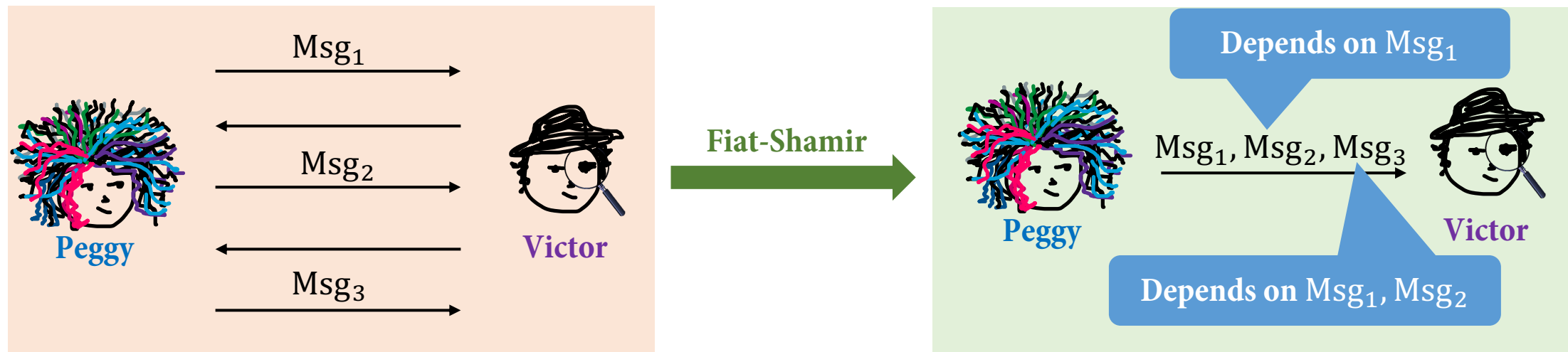
$$f(x_1, x_2, x_3, x_4) = x_1 \cdot x_2 \cdot x_3 \cdot x_4$$

Sequential multiplication over secret-shared values

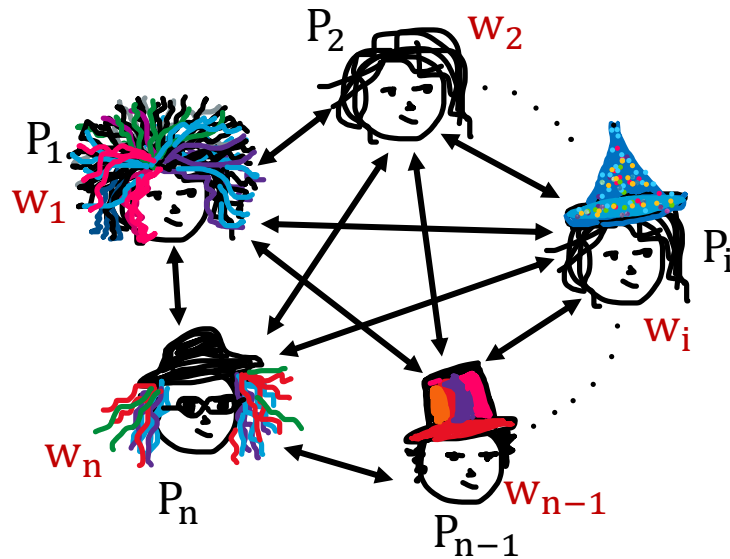
[Bar-Ilan-Beaver'89]: Constant round MPC for sequential multiplications.

Reduces to computing special randomized functions over secret shared values!

zkSNARKs in ROM



Collaborative Variants of zkSNARKs in ROM



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^{||}

¹Stanford University
²University College London
³Blockstream

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, where n is the bit length of the range. Proof generation and verification times are linear in n .

*P*lonK: 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:



Collaborative Version: [\[Ozdemir-Boneh'22\]](#)

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

Open Problems

- [MPC] What other functions can be computed using semi-honest MPC with free malicious security?
- [MPC/SNARK] Can these ideas be extended to the dishonest-majority setting?
- [MPC/SNARK] Exploring malicious security of other types of collaborative zk-SNARKs (e.g., those based on packed secret-sharing [Garg-Goel-Jain-Policharla-S'22]).
- [SNARK] Investigating applications of collaborative zk-SNARKs.
(some ongoing work: application to zk-zkRollups)
- [SNARK] Scalable Collaborative zkSNARKs with fair-exchange (True zkSNARKs-as-a-Service).



[eprint/2025/1026](#)

Thanks!