

Homomorphic Secret Sharing with Verifiable Evaluation

Arka Rai Choudhuri

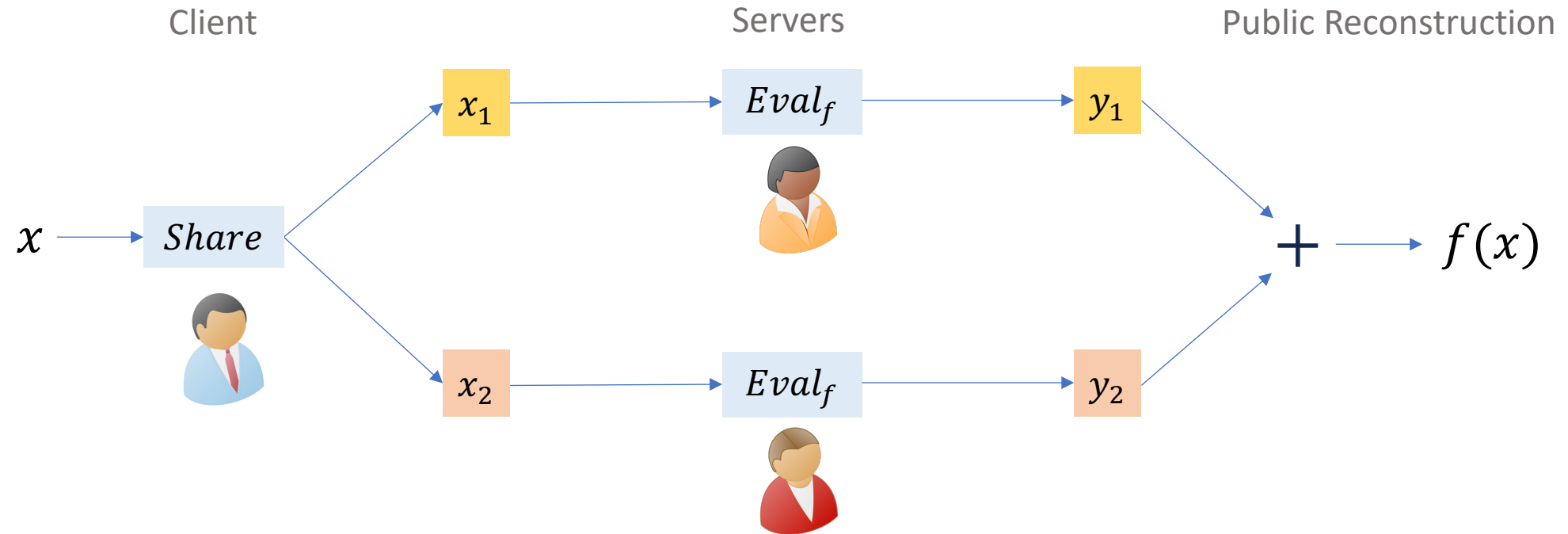
Aarushi Goel

Aditya Hegde

Abhishek Jain



Homomorphic Secret Sharing [Boyle-Gilboa-Ishai 16]



Correctness

$$\text{Eval}_f(x_1) + \text{Eval}_f(x_2) = f(x)$$

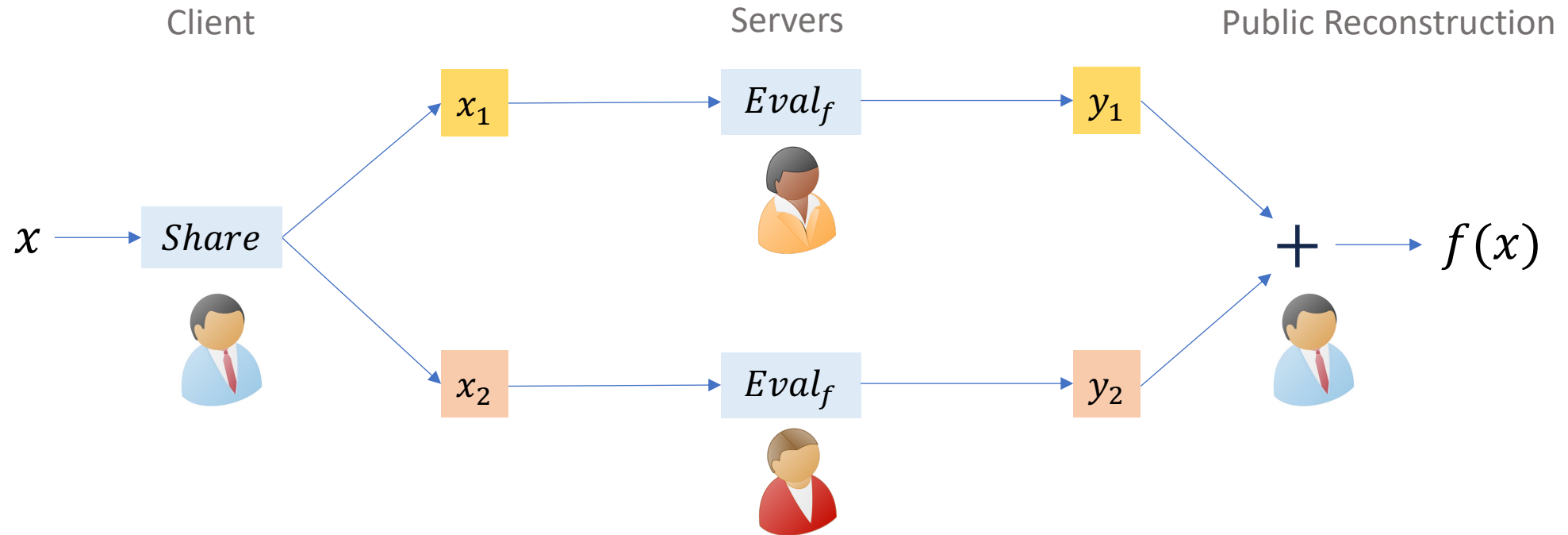
Security

x_1 , x_2 hide x

Succinctness

x_1 , x_2 , y_1 , y_2 are succinct

Homomorphic Secret Sharing [Boyle-Gilboa-Ishai 16]

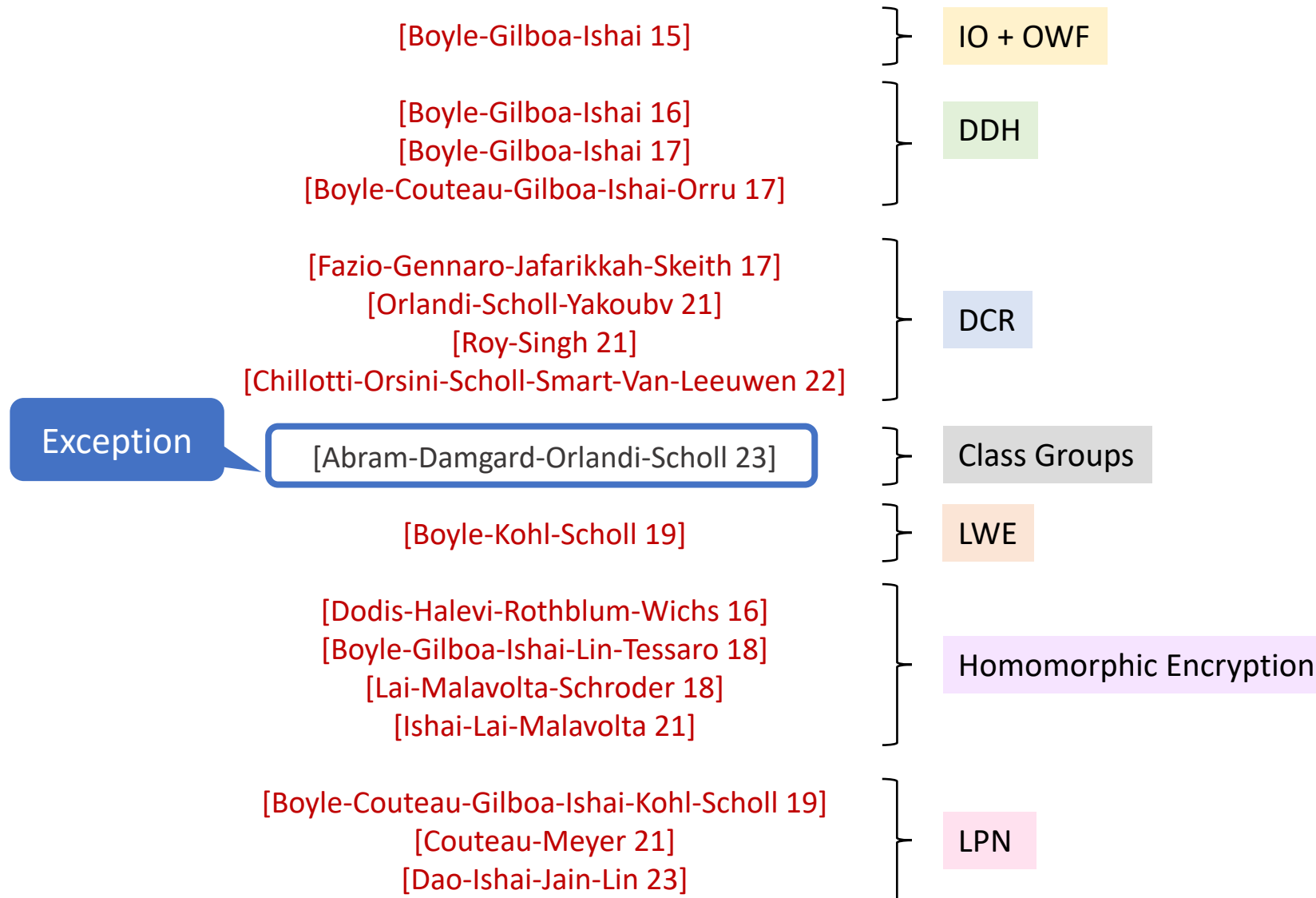


Enables private delegation of computation

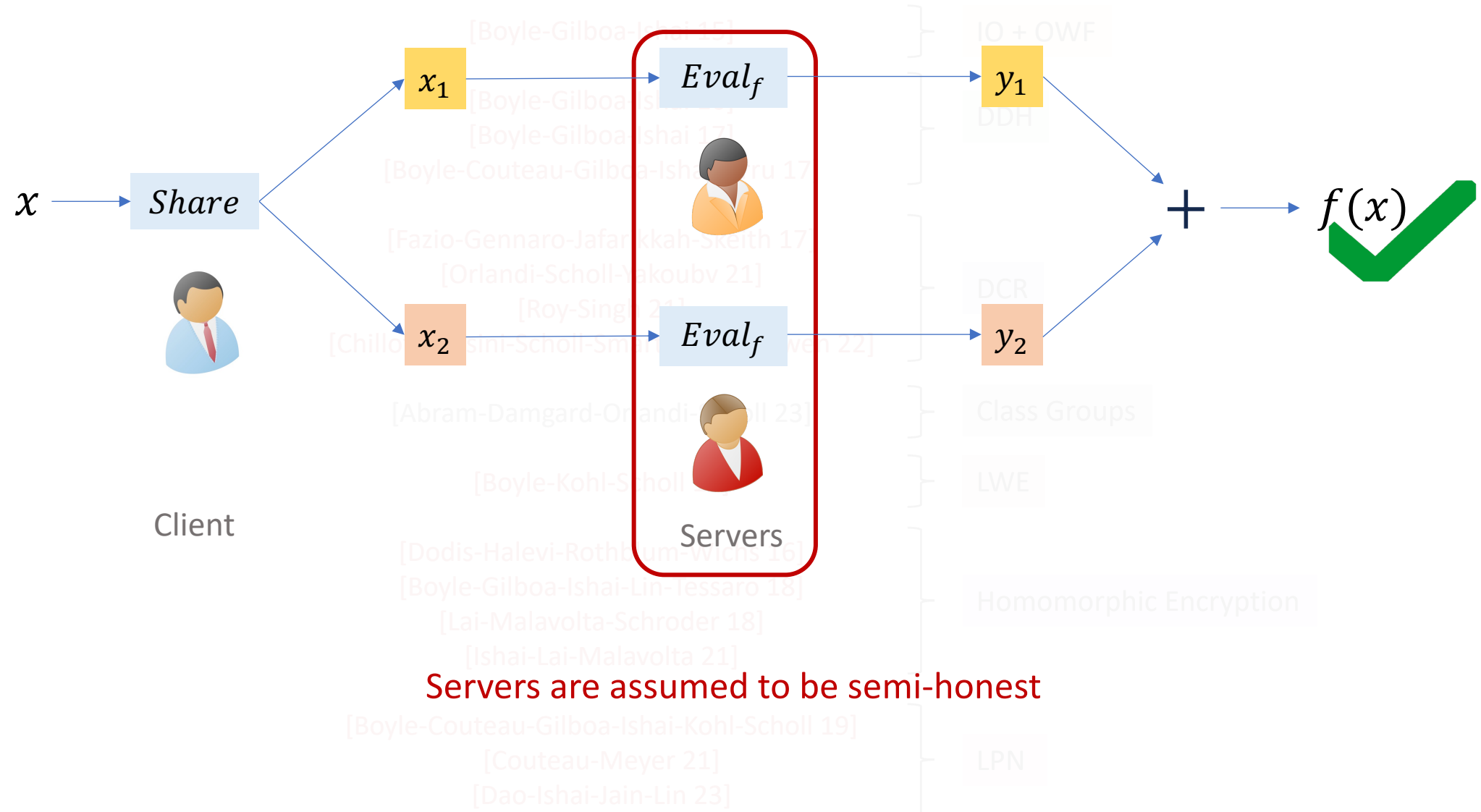
Constructions of HSS

[Boyle-Gilboa-Ishai 15]	}	IO + OWF
[Boyle-Gilboa-Ishai 16]	}	DDH
[Boyle-Gilboa-Ishai 17]		
[Boyle-Couteau-Gilboa-Ishai-Orru 17]		
[Fazio-Gennaro-Jafarikkah-Skeith 17]	}	DCR
[Orlandi-Scholl-Yakoubv 21]		
[Roy-Singh 21]		
[Chillotti-Orsini-Scholl-Smart-Van-Leeuwen 22]	}	
[Abram-Damgard-Orlandi-Scholl 23]	}	Class Groups
[Boyle-Kohl-Scholl 19]	}	LWE
[Dodis-Halevi-Rothblum-Wichs 16]	}	Homomorphic Encryption
[Boyle-Gilboa-Ishai-Lin-Tessaro 18]		
[Lai-Malavolta-Schroder 18]		
[Ishai-Lai-Malavolta 21]		
[Boyle-Couteau-Gilboa-Ishai-Kohl-Scholl 19]	}	LPN
[Couteau-Meyer 21]		
[Dao-Ishai-Jain-Lin 23]		

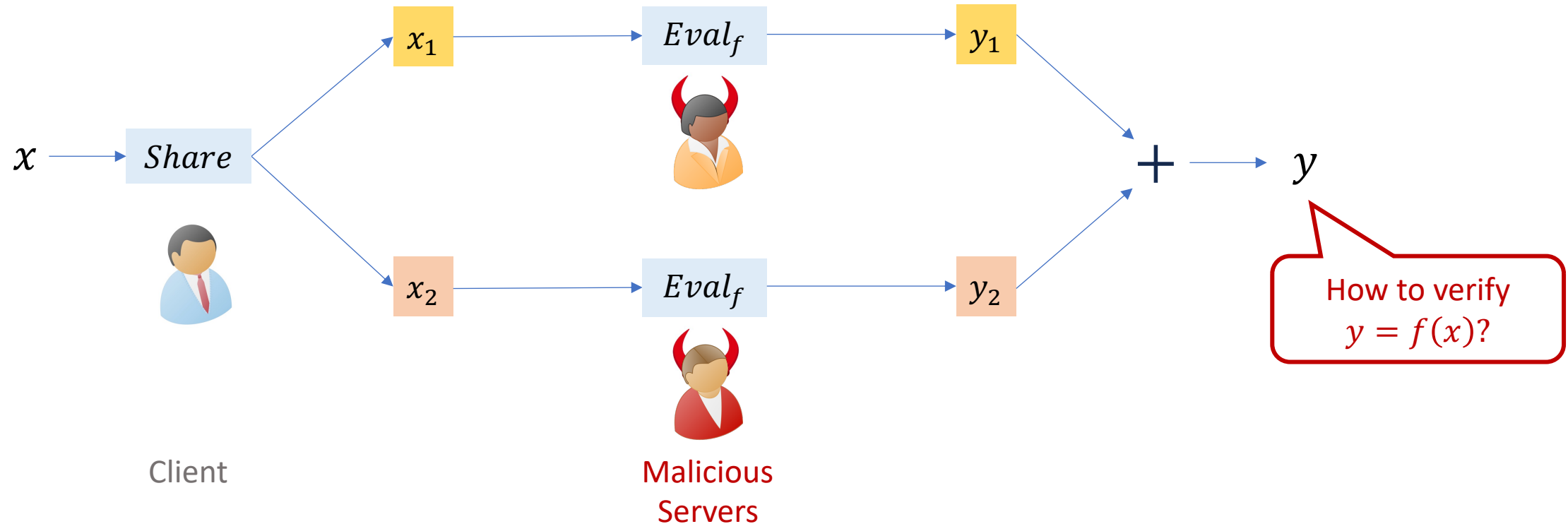
Constructions of Semi-Honest HSS



Constructions of Semi-Honest HSS



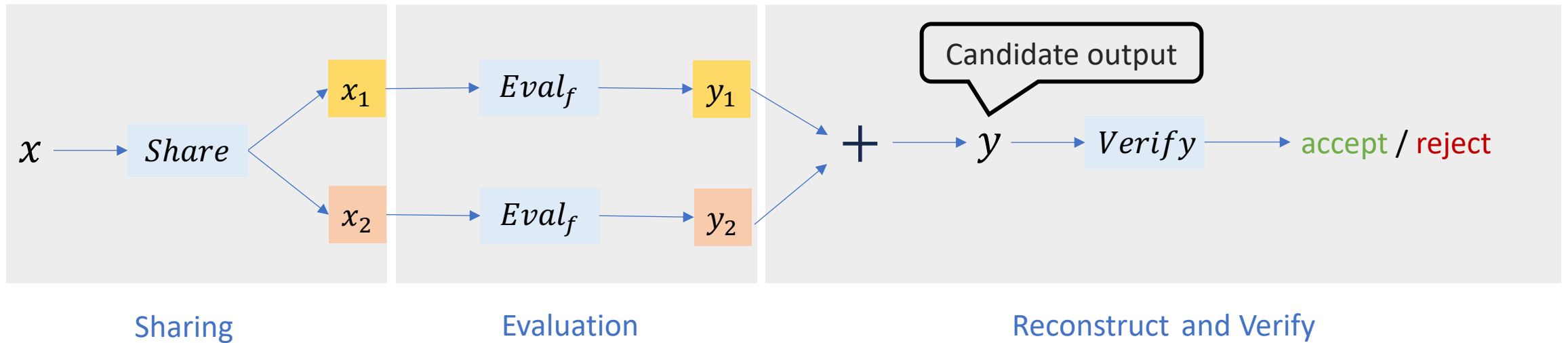
Our Goal: Handling Malicious Servers



Motivation: Private and verifiable delegation of computation

Black box solution!

HSS with Verifiable Evaluation (ve-HSS)

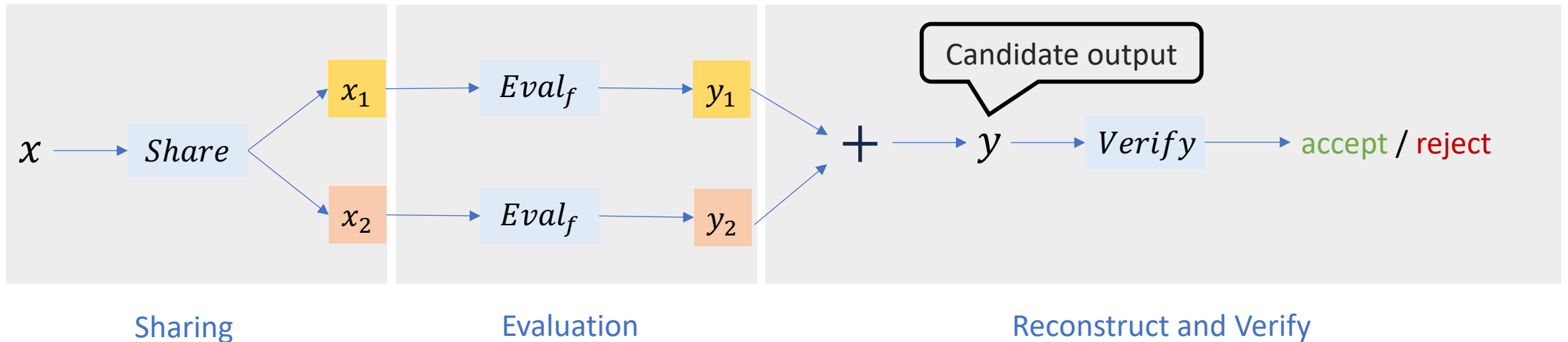


Correctness If everyone behaves honestly, then *Verify* → **accept** and $y = f(x)$

Succinctness x_1 , x_2 , y_1 , y_2 , *Verify* are succinct

Security x_1 , x_2 & output of *Verify* hide x

HSS with Verifiable Evaluation (ve-HSS)



Local Soundness

Verify \rightarrow **accept**, if and only if $y = f(x)$

Public Soundness

Let $x = (x_{priv}, x_{pub})$, then *Verify* \rightarrow **accept** if and only if $\exists x_{priv}$, such that $y = f(x_{priv}, x_{pub})$

Soundness should hold even when both servers are corrupt

Comparison with Prior Work

[ADOS 23]

Maliciously secure sublinear MPC based on HSS

[TLM18, TM19, TBM20, YO19, CZ21, CZ20, ZW22, Che23, HZ20, MTG22]

Varying notions of verifiability in HSS. Most of these don't consider soundness when all parties are corrupt.

Our Results

We design a **general framework** for transforming semi-honest HSS to *HSS with verifiable evaluation* using **certain kinds of zkSNARGs**.

Applications:

Black-box approach for private and verifiable delegation of computation.

Extension:

Multi-Client HSS with verifiable evaluation.

Our Results (1st Instantiation)

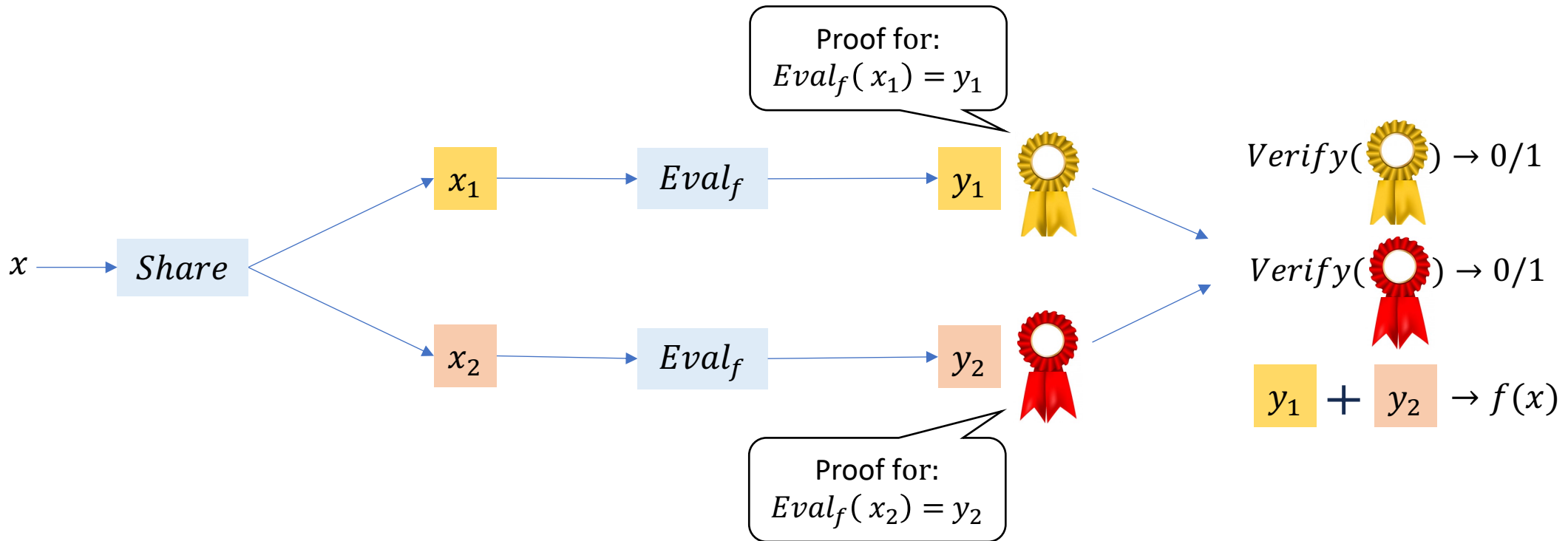
Semi-Honest HSS		SNARG	ve-HSS Function Class
[Orlandi-Scholl-Yakoubv 21] [Roy-Singh 21]	DCR	[Groth16] Generic Group Model	NC^1
[Abram-Damgard-Orlandi-Scholl 23]	Class Groups		
[Boyle-Kohl-Scholl 19]	LWE		
[Dodis-Halevi-Rothblum-Wichs 16] [Boyle-Gilboa-Ishai-Lin-Tessaro 18]	FHE		$P/Poly$
[Boyle-Gilboa-Ishai 15]	IO + OWF		

Our Results (2nd Instantiation)



Semi-Honest HSS	SNARG	ve-HSS Function Class
[Orlandi-Scholl-Yakoubv 21] [Roy-Singh 21] DCR	zkBARG based on [Waters-Wu 22] Subgroup decision assumption	SIMD-NC ¹
[Abram-Damgard-Orlandi-Scholl 23] Class Groups		
[Boyle-Kohl-Scholl 19] LWE		
[Dodis-Halevi-Rothblum-Wichs 16] [Boyle-Gilboa-Ishai-Lin-Tessaro 18] FHE		SIMD-P/Poly
[Boyle-Gilboa-Ishai 15] IO + OWF		

Our Construction

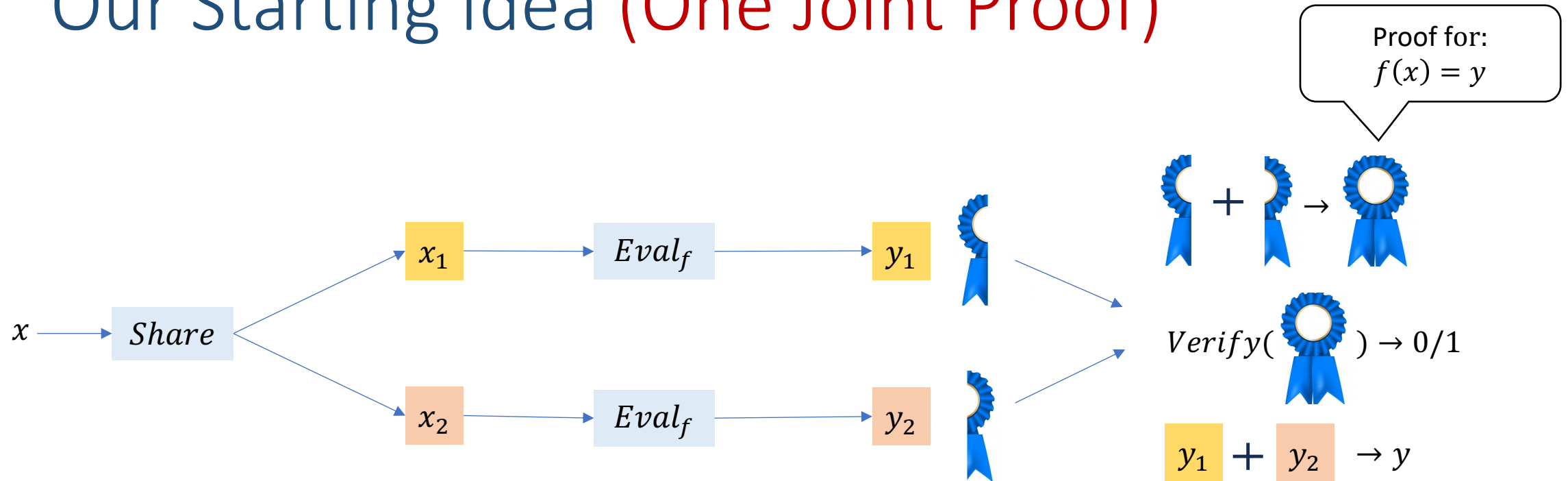
Strawman Approach ([Goldreich-Micali-Wigderson 87] Inspired)



$Eval_f$ is a **cryptographic** function!

Generating ,  will require **non-black-box use of cryptography**

Our Starting Idea (One Joint Proof)



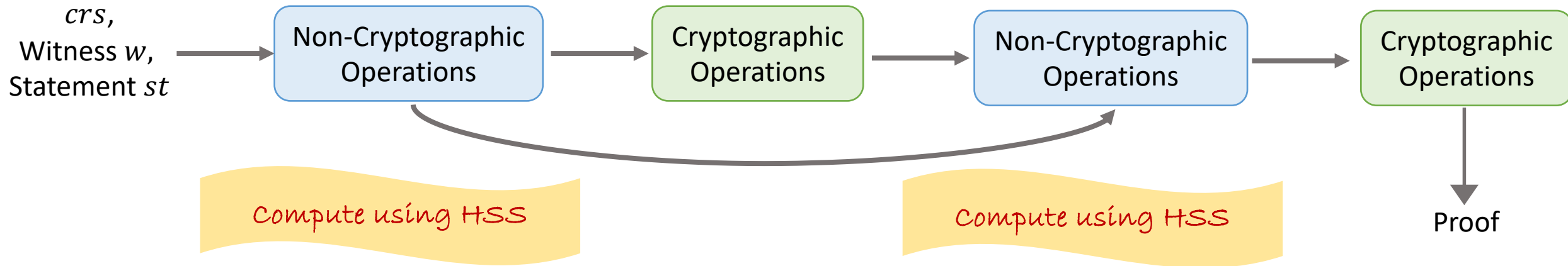
How can we generate these proof shares?

Prover algorithm is a **cryptographic** function!

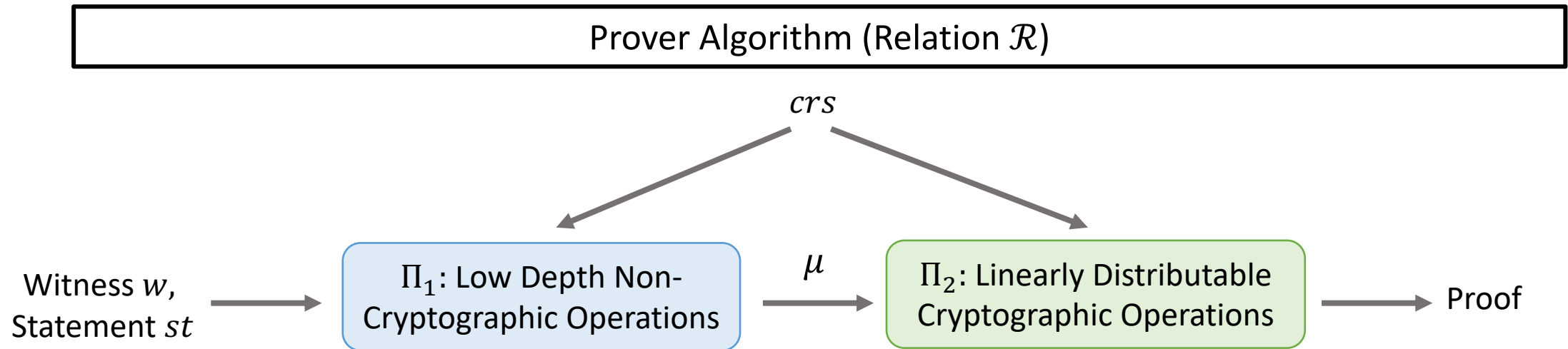
Computing it using HSS will require **non-black-box use of cryptography**

General SNARG Computation

Prover Algorithm (Relation \mathcal{R})



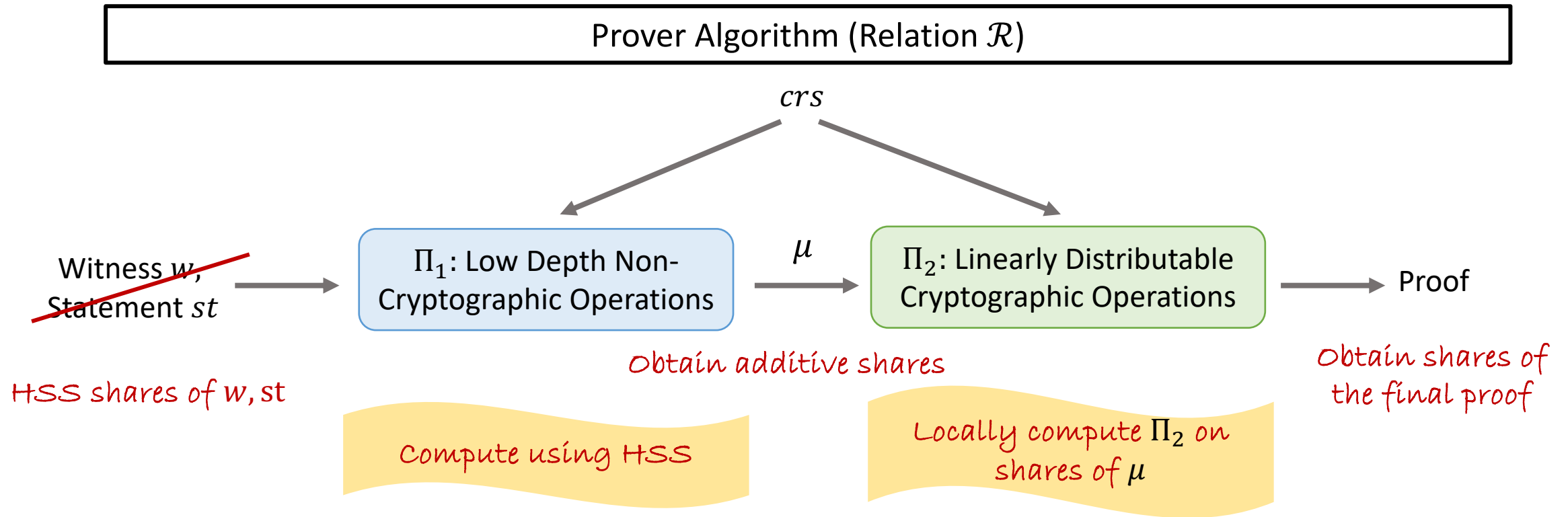
Splittable SNARGs



Let $\mu = \mu_1 + \mu_2$, then $\Pi_2(\text{crs}, \text{st}, \mu) = \Pi_2(\text{crs}, \text{st}, \mu_1) \boxtimes \Pi_2(\text{crs}, \text{st}, \mu_2)$

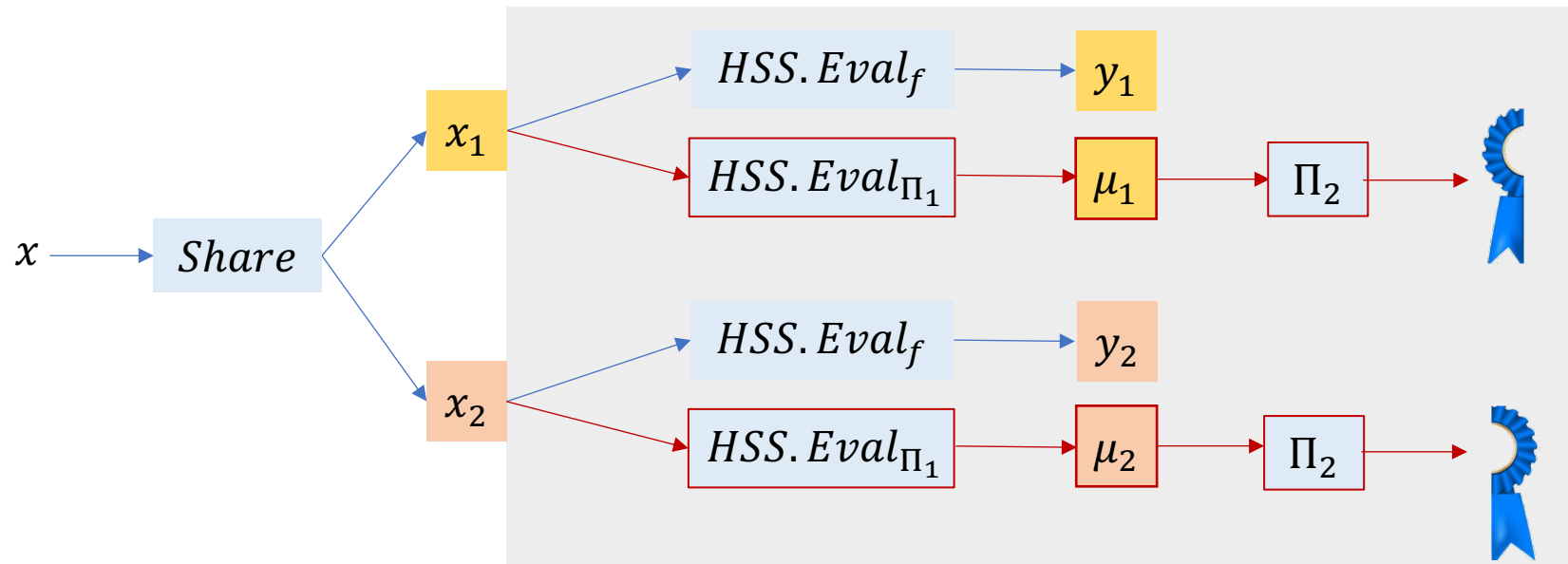
\exists some function

Splittable SNARGs



Candidate Approach using Splittable SNARGs

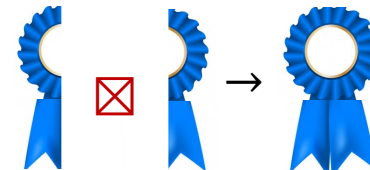
Computing shares of a proof for the deterministic relation: $f(x) = y_1 + y_2$



Evaluation in ve-HSS

Reconstruction and
Verification by the client:

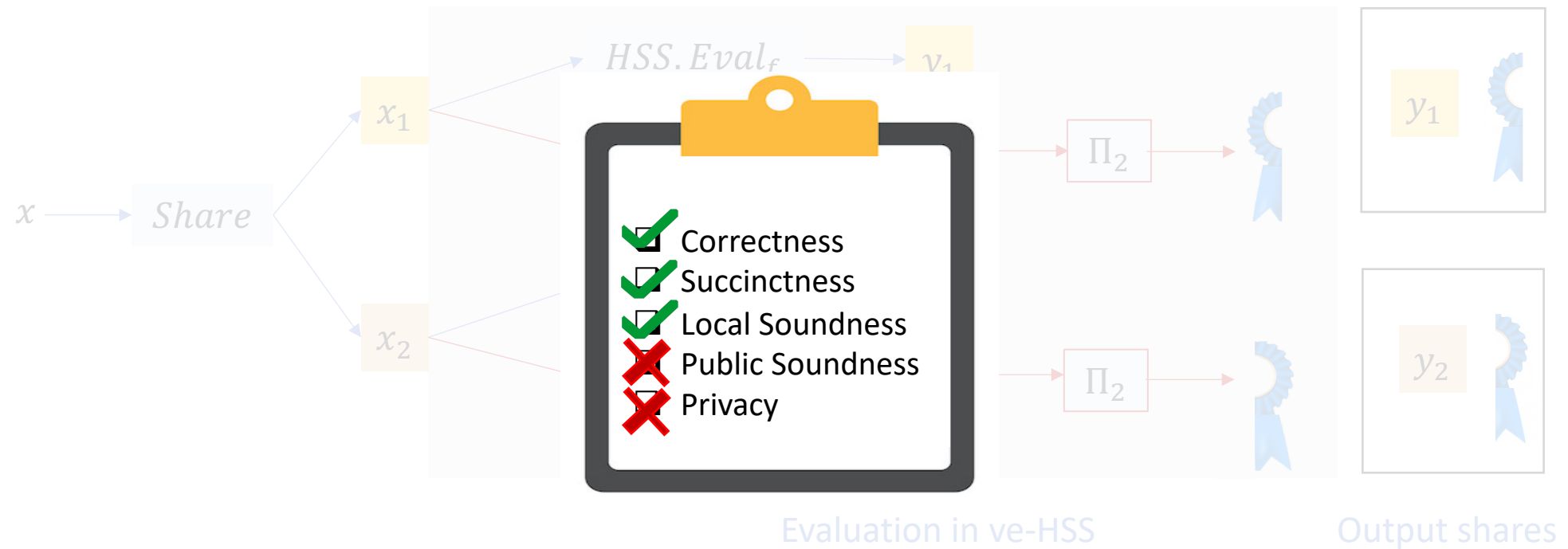
$$y_1 + y_2 \rightarrow y$$



$$Verify(x, y, \text{ribbon}) \rightarrow 0/1$$

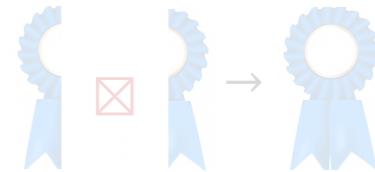
Candidate Approach using Splittable SNARGs

Compute shares of a proof for the deterministic relation: $f(x) = y_1 + y_2$



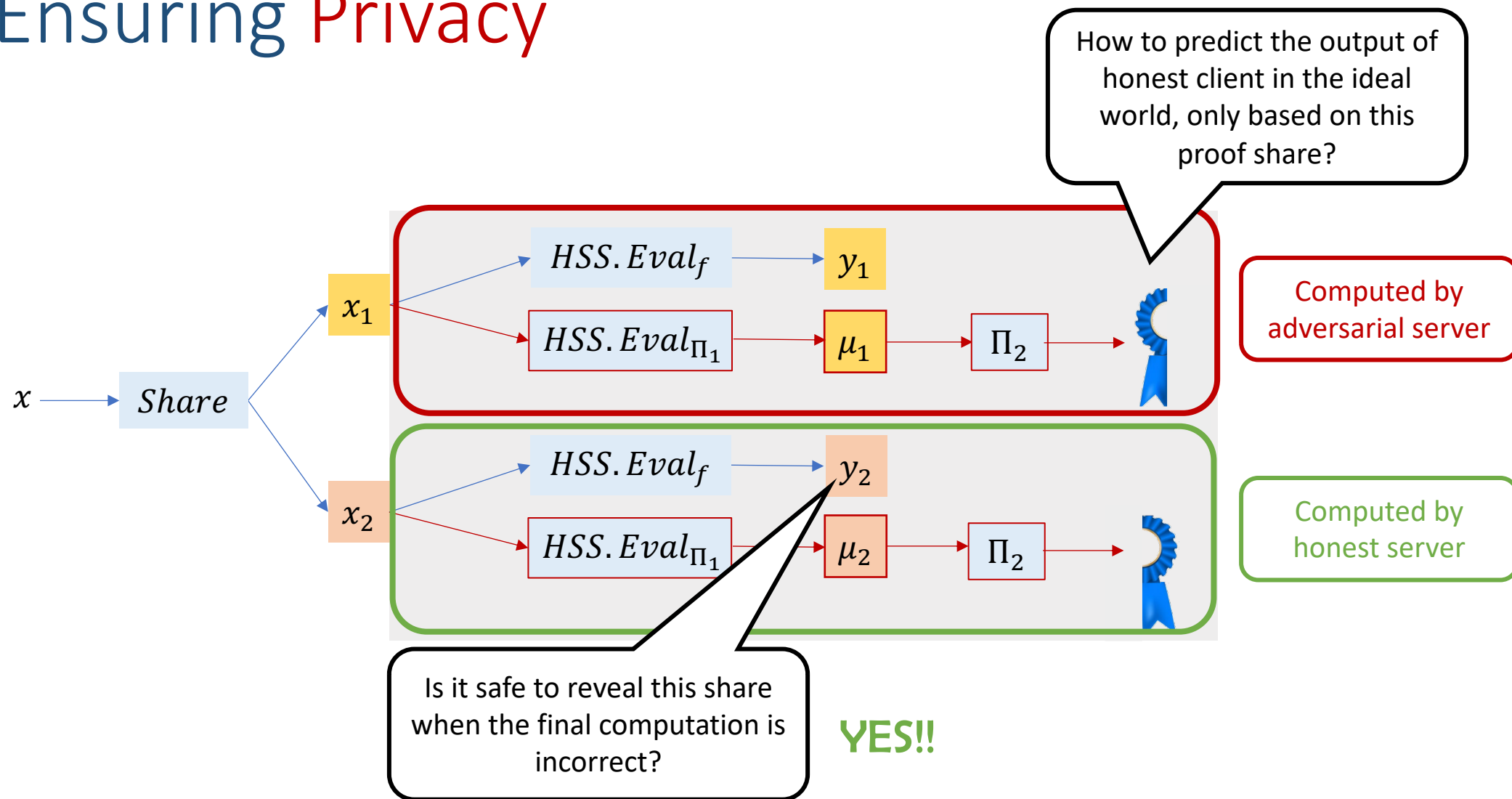
Reconstruct and Verify:

$$y_1 + y_2 \rightarrow y$$









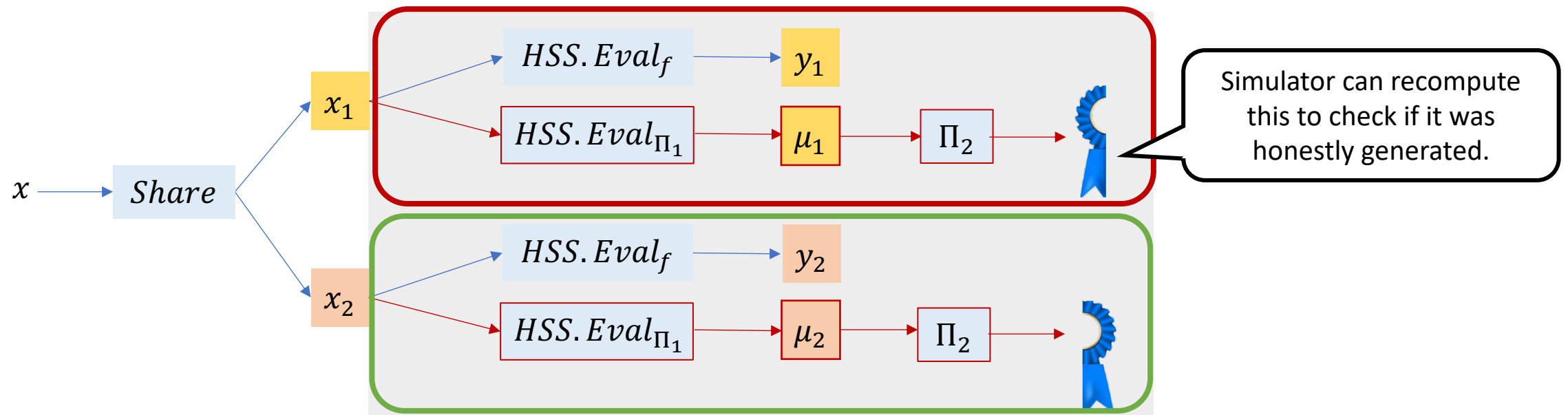
$$Verify(x, y, \text{blue ribbon}) \rightarrow 0/1$$

Ensuring Privacy



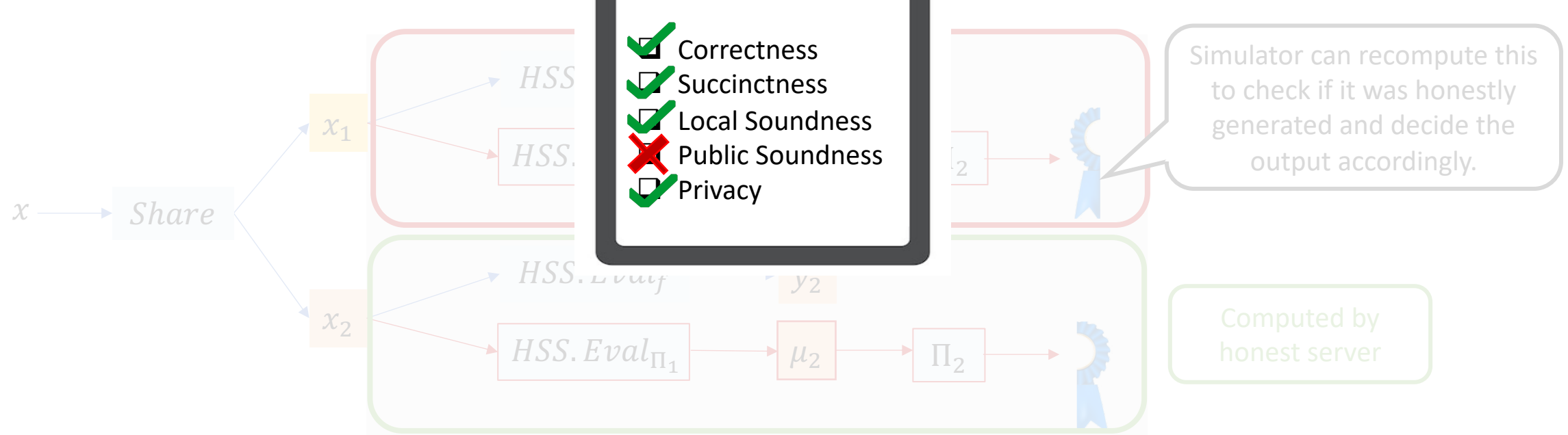
Distributed Prover Robust Verification

For a fixed  computed by the honest server,
a PPT adversary can only find one  such that    \rightarrow  verifies.



Distributed Prover Robust Verification

For a fixed  computed by the honest server,
a PPT adversary can only find one  such that    \rightarrow  verifies.



Adding Public Soundness

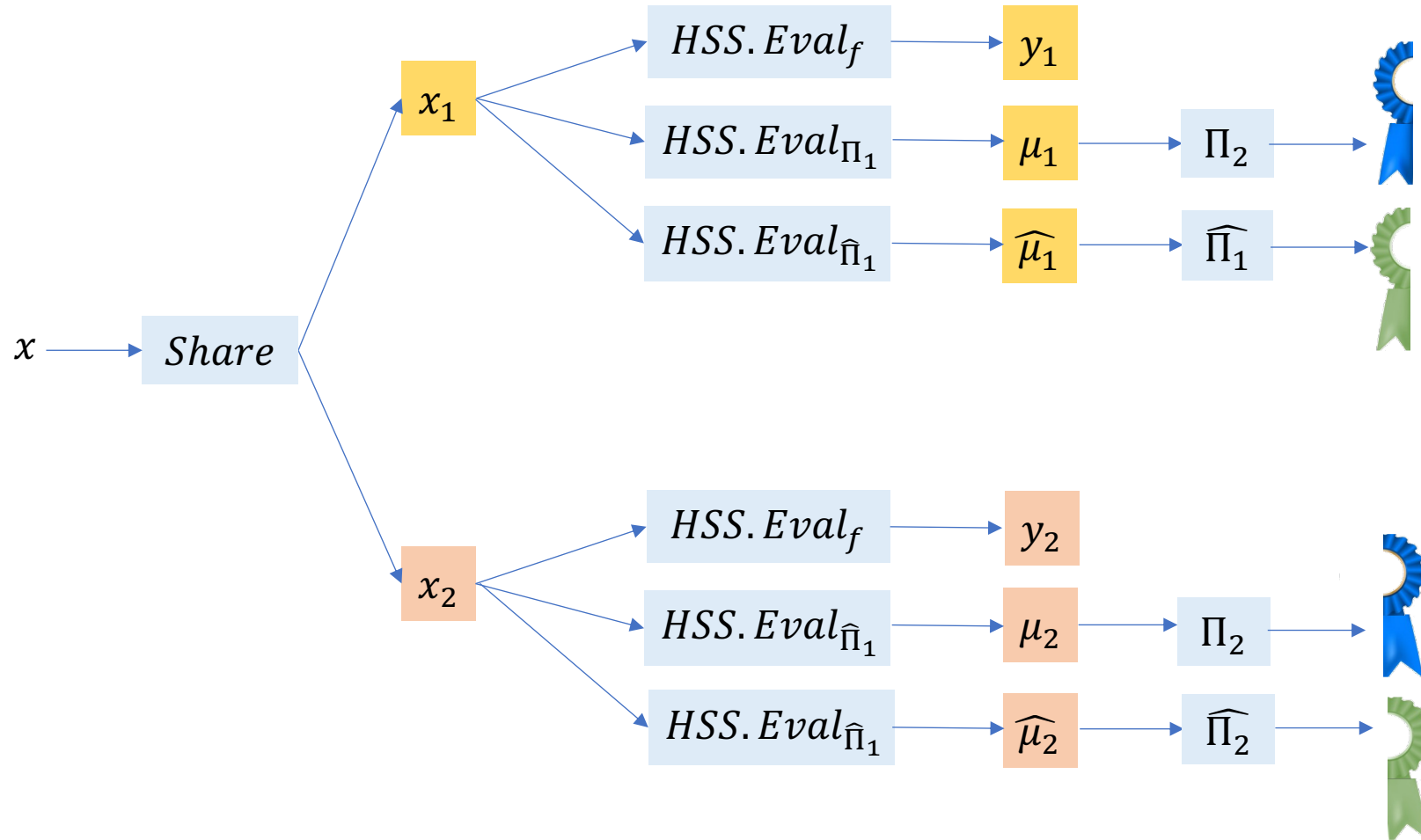
Compute two proofs:

Local Proof: For the deterministic relation: $\mathcal{R}_{local} = \{(f, x, y_1, y_2) \mid st.f(x) = y_1 + y_2\}$

Public Proof: For the relation: $\mathcal{R}_{pub} = \{(f, x_{pub}, y_1, y_2) \mid \exists x_{priv}, st.f(x_{priv}, x_{pub}) = y_1 + y_2\}$



Summary of Construction



Examples of Splittable zkSNARKs

[Groth16] zkSNARKs are splittable

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

Jens Groth**

University College London, UK
j.groth@ucl.ac.uk

Abstract. Non-interactive arguments enable a prover to convince a verifier that a statement is true. Recently there has been a lot of progress both in theory and practice on constructing highly efficient non-interactive arguments with small size and low verification complexity, so-called succinct non-interactive arguments (SNARGs) and succinct non-interactive arguments of knowledge (SNARKs). Many constructions of SNARGs rely on pairing-based cryptography. In these constructions a proof consists of a number of group elements and the verification consists of checking a number of pairing product equations. The question we address in this article is how

Our zero-knowledge version of
[Waters-Wu 22] BARGs are splittable

Batch Arguments for NP and More from Standard Bilinear Group Assumptions

Brent Waters
UT Austin and NTT Research
bwaters@cs.utexas.edu

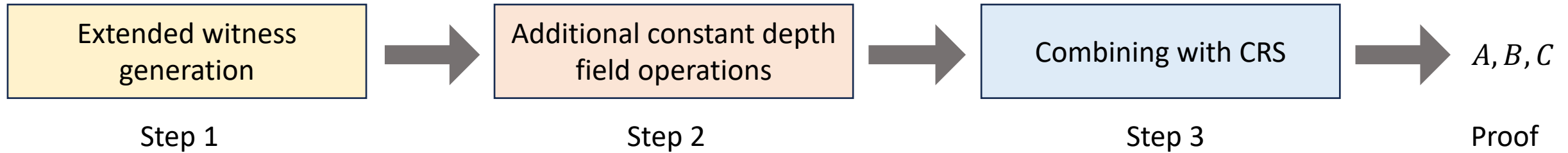
David J. Wu
UT Austin
dwu4@cs.utexas.edu

Abstract

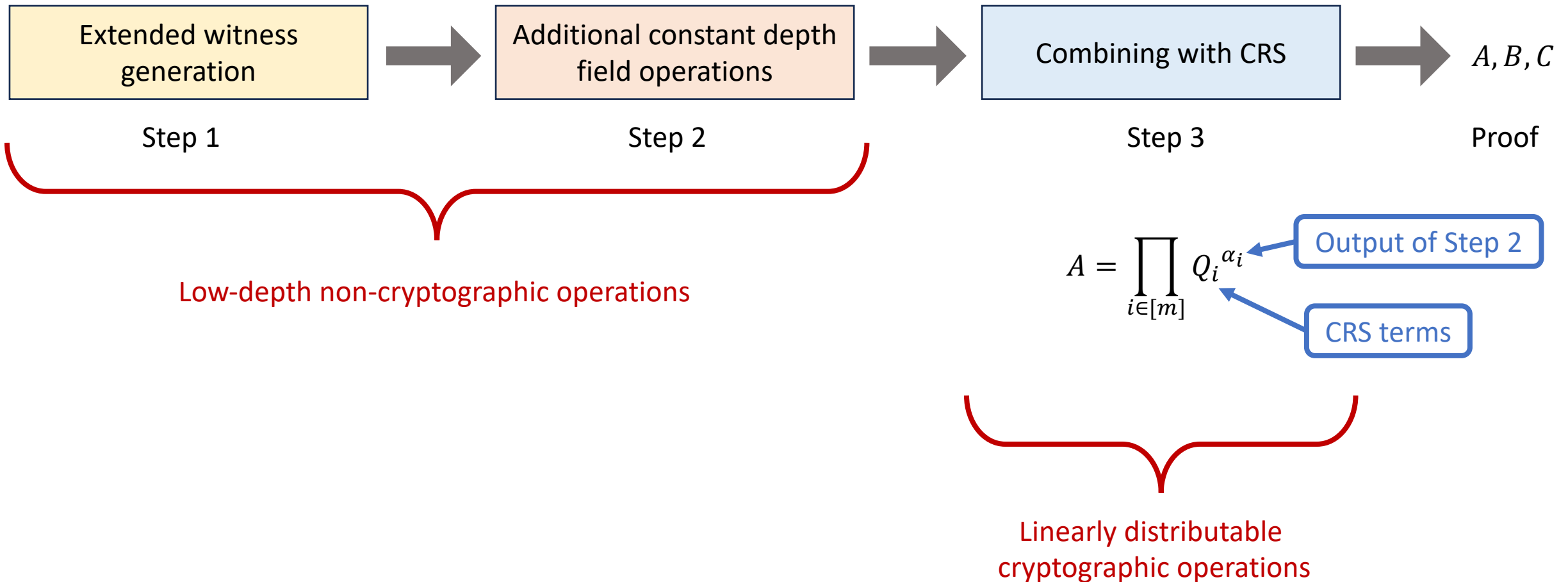
Non-interactive batch arguments for NP provide a way to amortize the cost of NP verification across multiple instances. They enable a prover to convince a verifier of multiple NP statements with communication much smaller than the total witness length and verification time much smaller than individually checking each instance.

In this work, we give the first construction of a non-interactive batch argument for NP from standard assumptions on groups with bilinear maps (specifically, from either the subgroup decision assumption in composite-order groups or from the k -Lin assumption in prime-order groups for any $k \geq 1$). Previously, batch arguments for NP were only known from LWE, or a combination of multiple assumptions, or from non-standard/non-falsifiable assumptions. Moreover, our work introduces a new direct approach for batch verification and avoids heavy tools like correlation.

[Groth16] zkSNARKs are Splittable



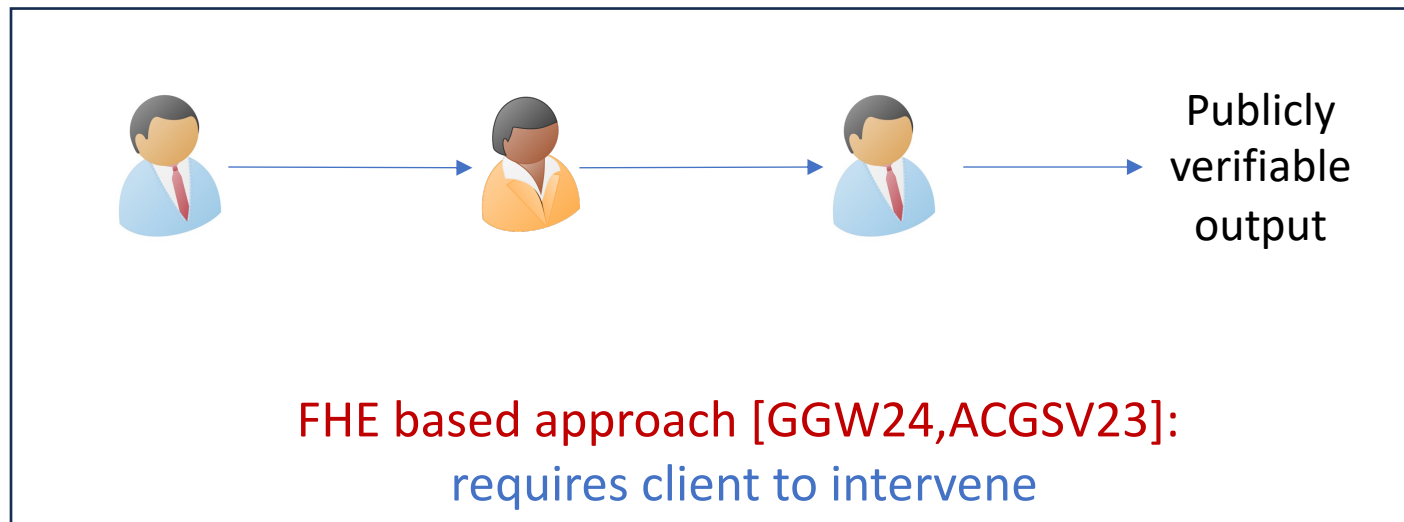
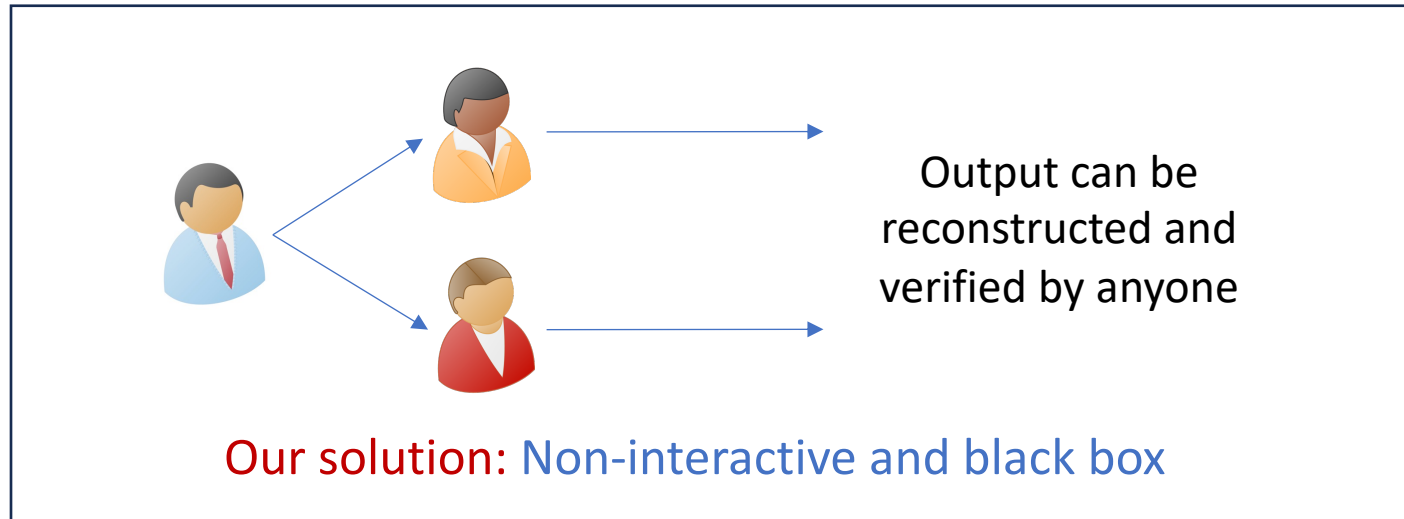
[Groth16] zkSNARKs are Splittable



Applications

(Private and Verifiable Delegation of Computation)

Delegating Non-Cryptographic Functions

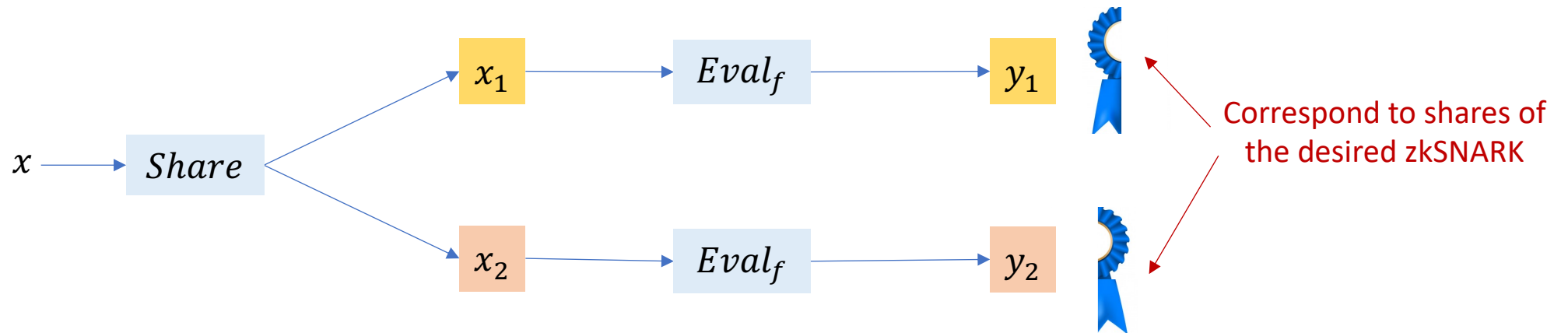


Delegating zkSNARK Computation

Client wants to outsource the computation of a (splittable) zkSNARK for the relation:

$$\mathcal{R} = \{(f, x = x_{pub}, 1) \mid \exists x_{priv} = w, st. f(x, w) = 1\}$$

Use our ve-HSS for computing function f



Non-interactive and black box solution

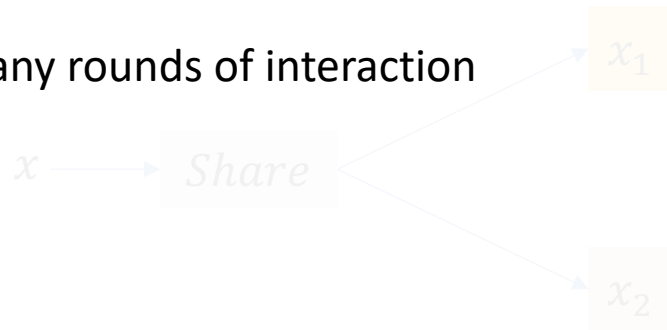
Delegating zkSNARK Computation

Client wants to outsource computation for (possibly) SNARK for the relation:

$$\mathcal{R} = \{(f, x = x_{pub}, 1) \mid \exists x_{priv} = w, \text{st. } f(x, w) = 1\}$$

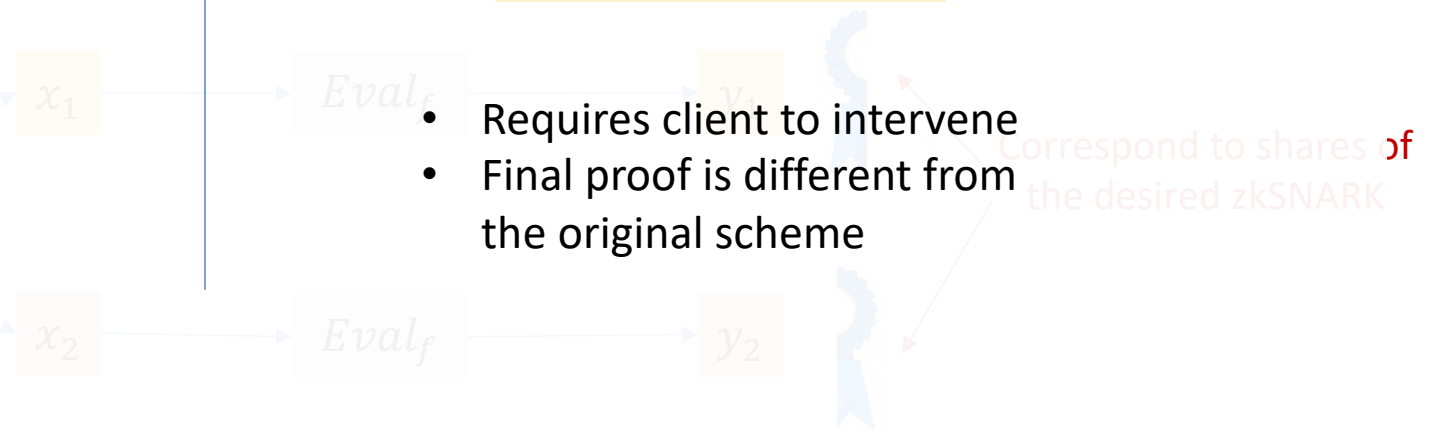
MPC based approaches
[GGJ+23, CLMZ23, LZW+24]

- Many rounds of interaction



FHE based approach
[GGW24]

- Requires client to intervene
- Final proof is different from the original scheme



Non-interactive and black box solution

Summary

A new notion of HSS with verifiable evaluation

A general framework for adding verifiability to semi-honest HSS using splittable SNARGs

Instantiations of splittable SNARGs

Applications to private and verifiable delegation of non-cryptographic and zkSNARK computations

Extension to multi-client HSS with verifiable evaluation

Open Questions

A framework using HSS schemes that have non-negligible correctness error

Other examples and applications of splittable SNARGs

Distributed prover robust verification using MACs

Thank you!