Zero-Knowledge Proofs on Secret-Shared Data

via Fully Linear PCPs

Dan Boneh
Stanford

Elette Boyle
IDC Herzliya

Henry Corrigan-Gibbs
Stanford

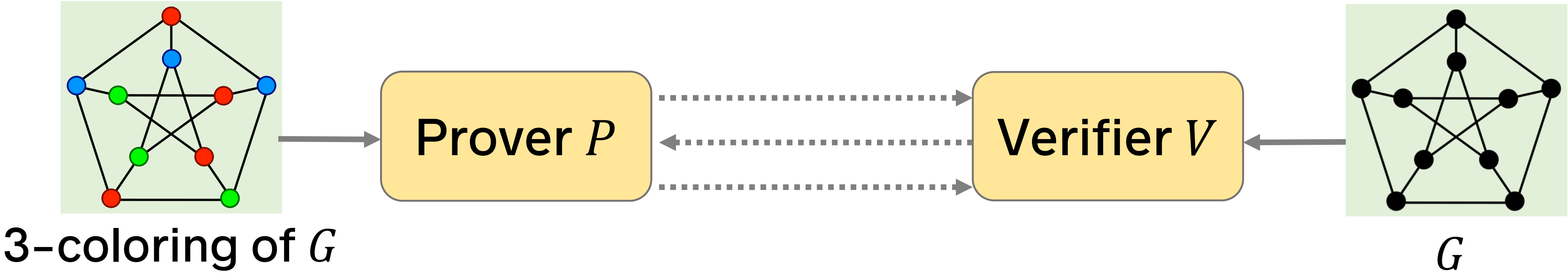
Niv Gilboa
Ben-Gurion University

Yuval Ishai
Technion

Review

Zero-knowledge proofs

[GMR89]



Complete. Honest P convinces honest V.

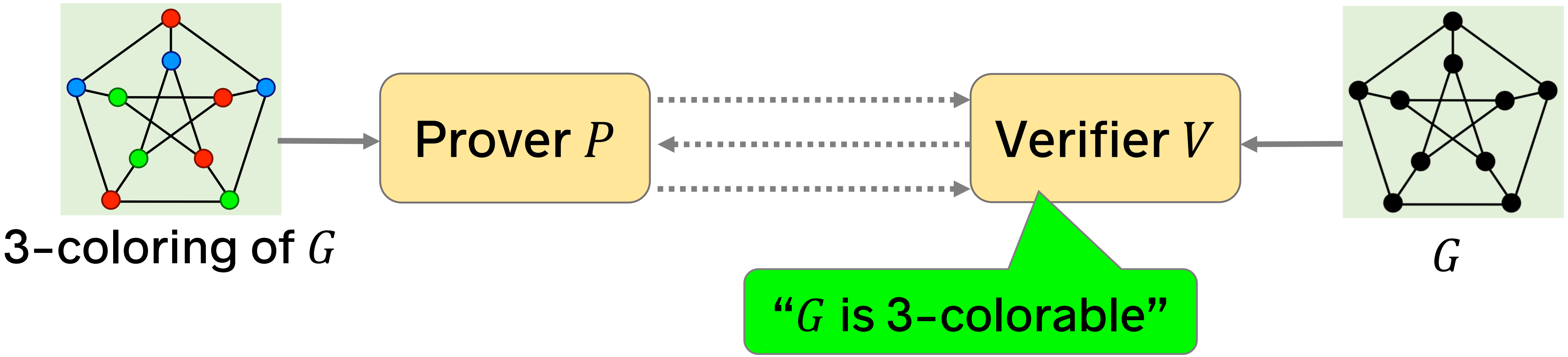
Sound. Dishonest P^* rarely fools honest V.

ZK. Dishonest V^* learns only that $G \in \mathrm{3COL}$.
→ V^* **learns nothing else about** G

Review

Zero-knowledge proofs

[GMR89]



Complete. Honest P convinces honest V.

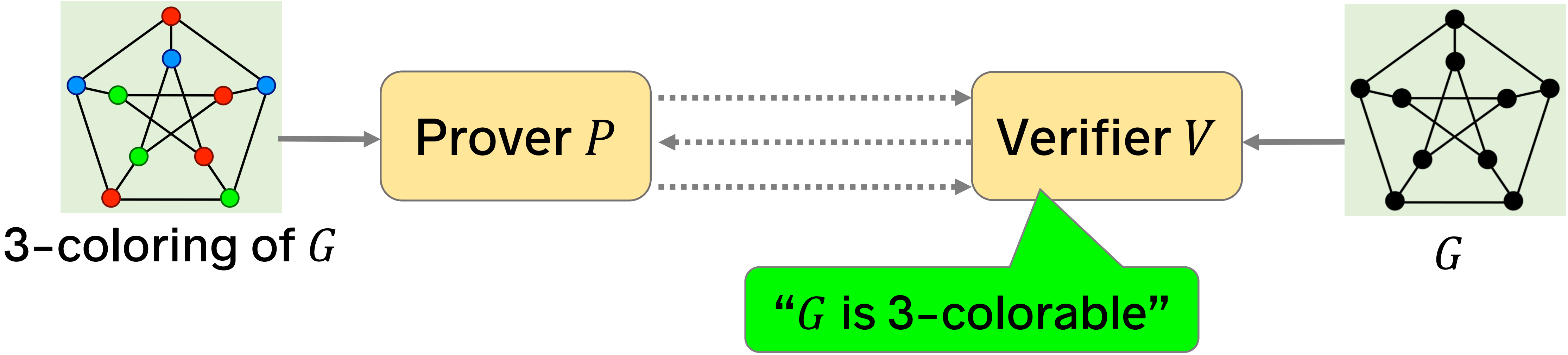
Sound. Dishonest P^* rarely fools honest V.

ZK. Dishonest V^* learns only that $G \in \mathrm{3COL}$.
→ V^* learns nothing else about G

Review

Zero-knowledge proofs

[GMR89]



Complete. Honest P convinces honest V.

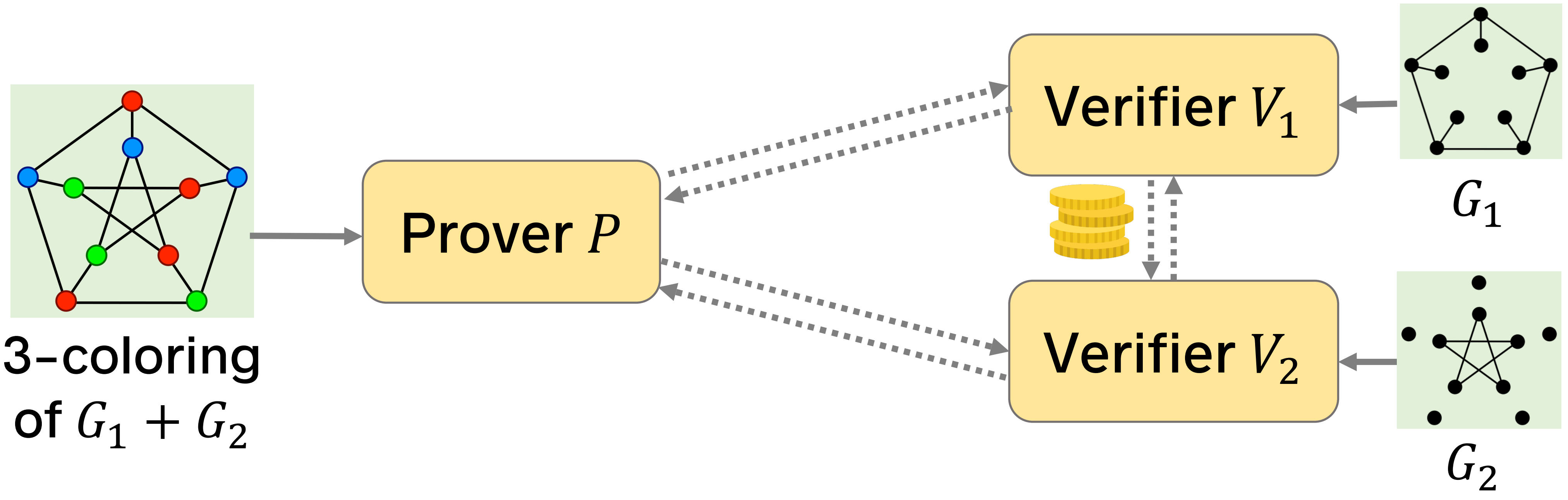
Sound. Dishonest P^* rarely fools honest V.

ZK. Dishonest V^* learns only that $G \in \mathrm{3COL}$.

→ V^* **learns nothing else about** G

This paper

Zero-knowledge proofs on **distributed** data



Complete. Honest P convinces honest (V_1, V_2).

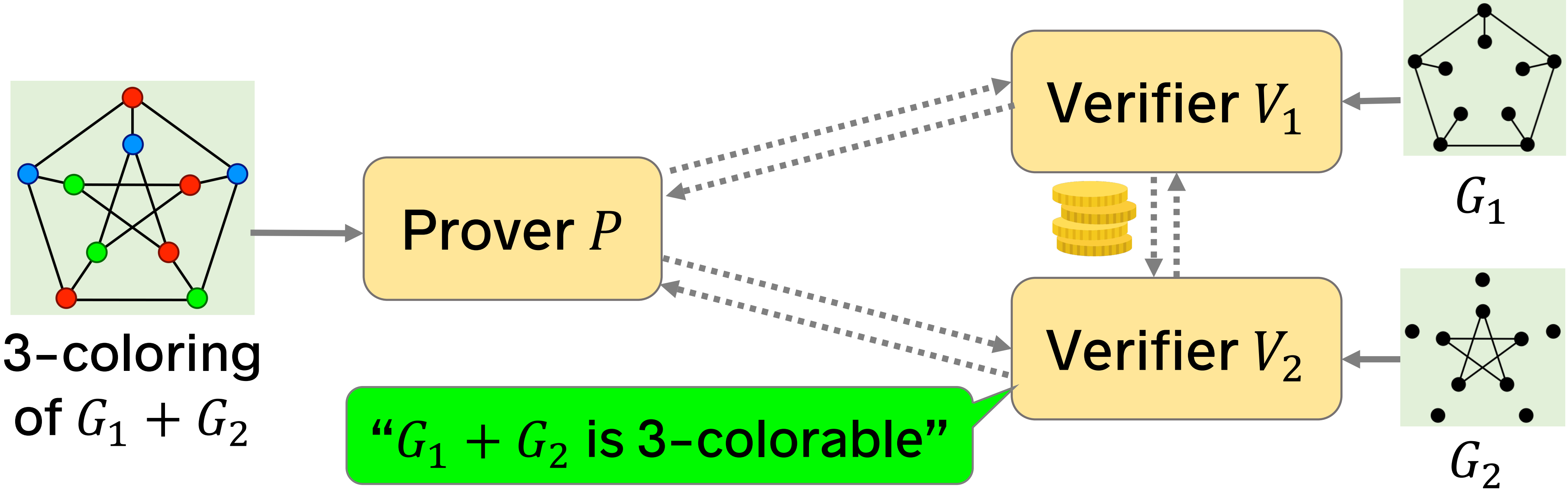
Sound. Dishonest P^* rarely fools honest (V_1, V_2).

Strong ZK. Dishonest V_1^* (or V_2^*) learns only that $G_1 + G_2 \in \mathrm{3COL}$.

→ V_1 learns nothing else about G_2

This paper

Zero-knowledge proofs on distributed data



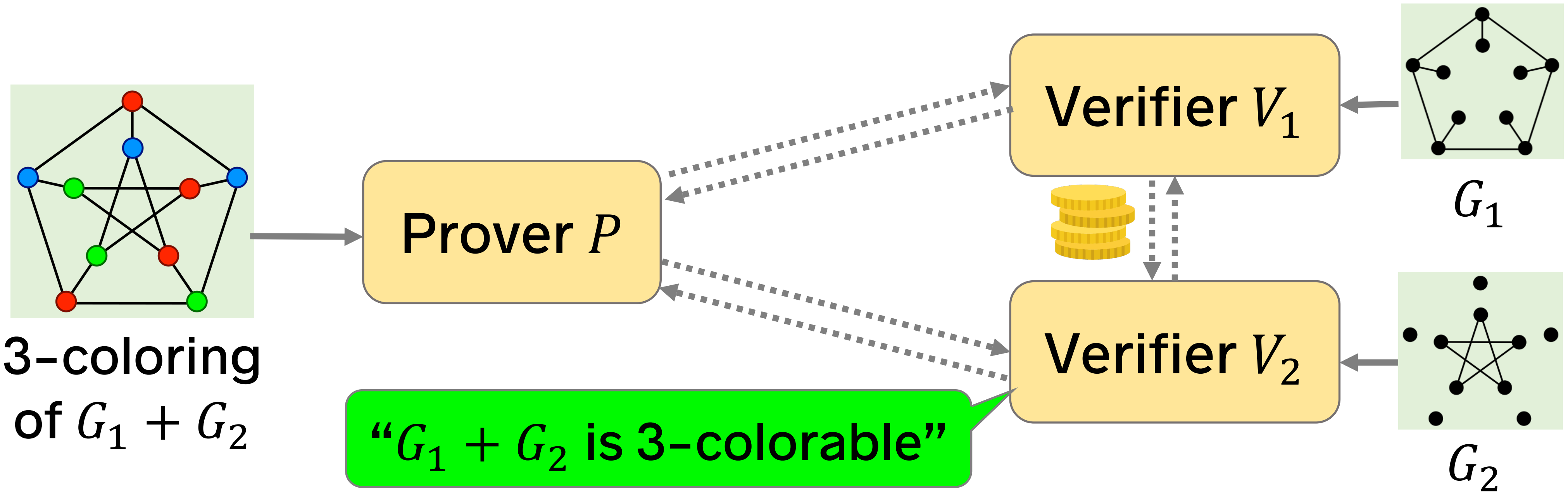
Complete. Honest P convinces honest (V_1, V_2).

Sound. Dishonest P^* rarely fools honest (V_1, V_2).

Strong ZK. Dishonest V_1^* (or V_2^*) learns only that $G_1 + G_2 \in 3\mathrm{COL}$.
→ V_1 learns nothing else about G_2

This paper

Zero-knowledge proofs on distributed data



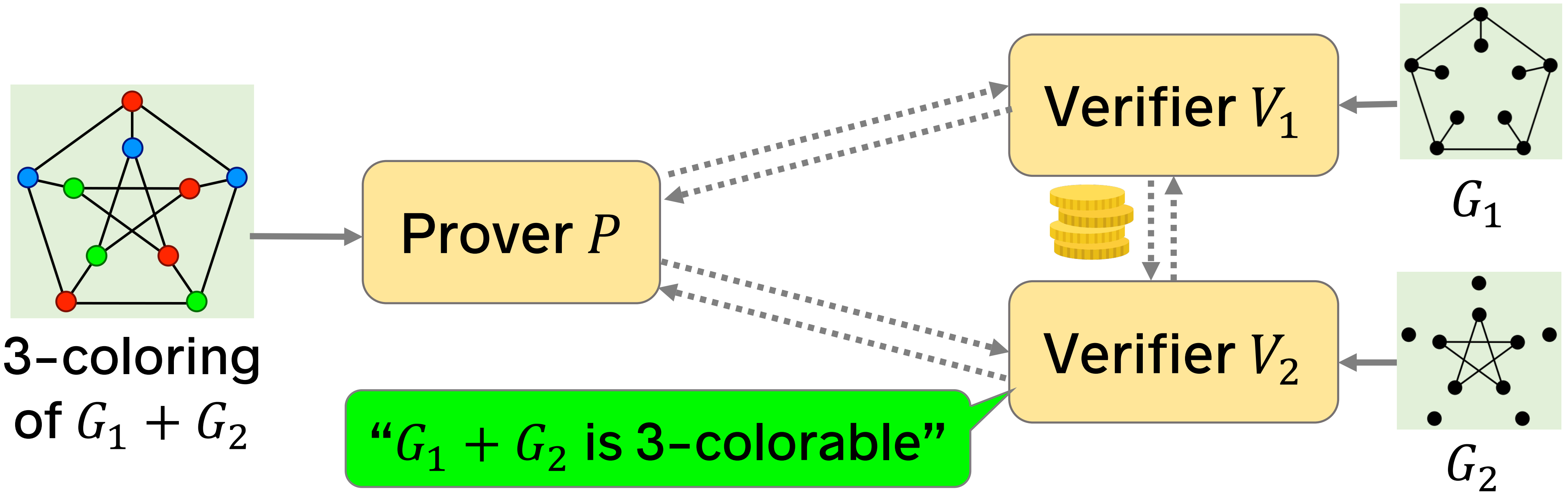
Complete. Honest P convinces honest (V_1, V_2).

Sound. Dishonest P^* rarely fools honest (V_1, V_2).

Strong ZK. Dishonest V_1^* (or V_2^*) learns only that $G_1 + G_2 \in \mathrm{3COL}$.
→ V_1 **learns nothing else about** G_2

This paper

Zero-knowledge proofs on distributed data



k-round protocol = As in other multiparty protocols

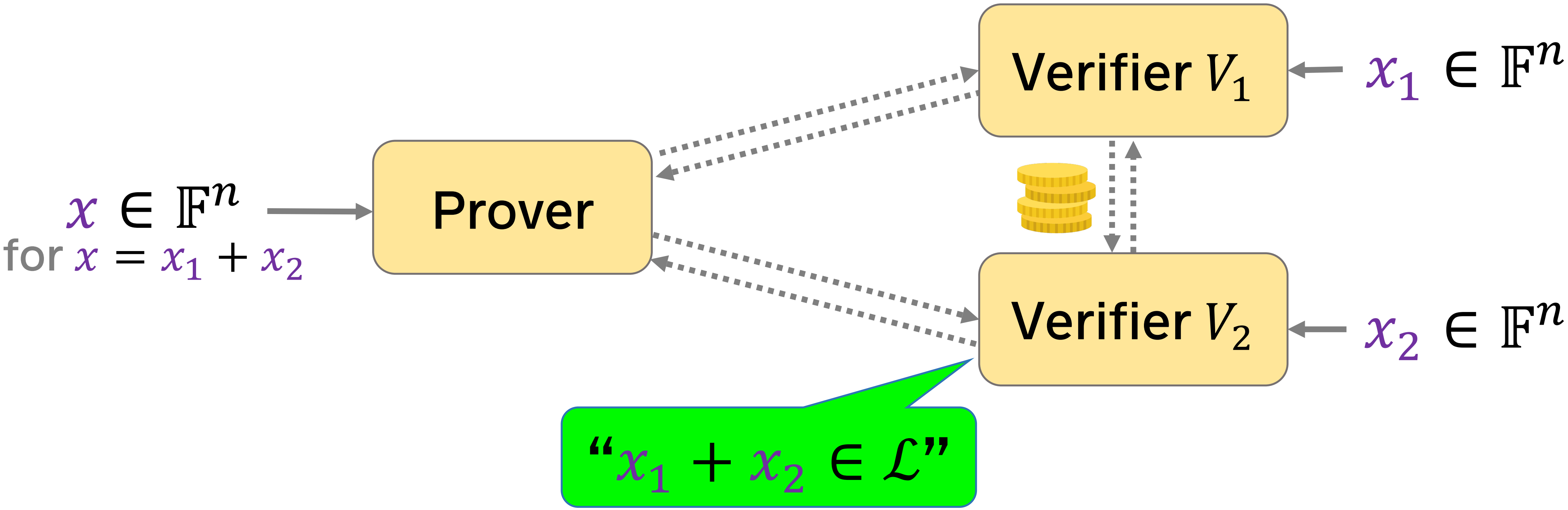
Public coin = Verifiers' messages to prover are random strings

More than two verifiers

Special case

Zero-knowledge proofs on secret-shared data

Language $\mathcal{L} \subseteq \mathbb{F}^n$, for finite field $\mathbb{F}$.



ZK proofs on distributed data
Applications and prior implicit constructions

Application	Language $\mathcal{L}$	Communication Cost: Prior	Communication Cost: This work
PIR writing, private messaging [OS97], [BGI16], Riposte, ...	Weight-one vectors in $\mathbb{F}^n$	$\Omega(n)$	$O(1)$
Private statistics, private ad targeting Adnostic, Adscale, Prio, ...	$\{0,1\}^n \subseteq \mathbb{F}^n$ for large $\mathbb{F}$	$\Omega(n)$	$O(\log n)$

Also: New application to malicious-secure MPC.

ZK proofs on distributed data
Applications and prior implicit constructions

Communication Cost

Application	Language $\mathcal{L}$	P	
PIR writing, private messaging [OS97], [BGI16], Riposte, …	Weight-one vectors in $\mathbb{F}^n$	Ω	
Private statistics, private ad targeting Adnostic, Adscale, Prio, …	$\{0,1\}^n \subseteq \mathbb{F}^n$ for large $\mathbb{F}$	$\Omega(n)$	$O(\log n)$

Used in the Firefox browser

Also: New application to malicious-secure MPC.

ZK proofs on distributed data
Applications and prior implicit constructions

Application	Language $\mathcal{L}$	Communication Cost: Prior	Communication Cost: This work
PIR writing, private messaging [OS97], [BGI16], Riposte, …	Weight-one vectors in $\mathbb{F}^n$	$\Omega(n)$	$O(1)$
Private statistics, private ad targeting Adnostic, Adscale, Prio, …	$\{0,1\}^n \subseteq \mathbb{F}^n$ for large $\mathbb{F}$	$\Omega(n)$	$O(\log n)$

Also: New application to malicious-secure MPC.

Selected results: New ZK proofs

Let $\mathbb{F}$ be a finite field. Let $\mathcal{L} \subseteq \mathbb{F}^n$ be a language. $(n \ll \mathbb{F})$

Theorem. If $\mathcal{L}$ is recognized by circuits of size $|\mathcal{C}|$, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:

- $O(1)$ rounds and
- communication cost $O(|\mathcal{C}|)$. (elements of $\mathbb{F}$)

Theorem. If $\mathcal{L}$ has a degree-two arithmetic circuit, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:

- $O(\log n)$ rounds and
- communication cost $O(\log n)$. (Improves: $\Omega(n)$ [BC17])

Selected results: New ZK proofs

Let $\mathbb{F}$ be a finite field. Let $\mathcal{L} \subseteq \mathbb{F}^n$ be a language. $(n \ll \mathbb{F})$

Theorem. If $\mathcal{L}$ is recognized by circuits of size $|\mathcal{C}|$, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:

- $O(1)$ rounds and
- communication cost $\boldsymbol{O(|\mathcal{C}|)}$. (elements of $\mathbb{F}$)

Theo... e is a ... $\mathcal{L}$ with: ...) [BC17])

- Generalizes special-purpose schemes. [CB17]
- Non-trivial extension to setting in which prover and some verifiers collude.

Selected results: New ZK proofs

Let $\mathbb{F}$ be a finite field. Let $\mathcal{L} \subseteq \mathbb{F}^n$ be a language. ($n \ll \mathbb{F}$)

Theorem. If $\mathcal{L}$ is recognized by circuits of size $|\mathcal{C}|$, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:

- $O(1)$ rounds and
- communication cost $\boldsymbol{O(|\mathcal{C}|)}$. (elements of $\mathbb{F}$)

Theorem. If $\mathcal{L}$ has a **degree-two** arithmetic circuit, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:

- $O(\log n)$ rounds and
- communication cost $\boldsymbol{O(\log n)}$. (Improves: $\Omega(n)$ [BC17])

Selected results: New ZK proofs

Let $\mathbb{F}$ be a finite field. Let $\mathcal{L} \subseteq \mathbb{F}^n$ be a language. $(n \ll \mathbb{F})$

Theorem. If $\mathcal{L}$ is recognized by circuits of size $|\mathcal{C}|$, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:

- $O(1)$ rounds and
- communication cost $\boldsymbol{O(|\mathcal{C}|)}$. (elements of $\mathbb{F}$)

Theorem. If $\mathcal{L}$ has a **degree-two** arithmetic circuit, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:

- $O(\log n)$ rounds and
- communication cost $\boldsymbol{O(\log n)}$. (Improves: $\Omega(n)$ [BC17])

Selected results: New ZK proofs

Let $\mathbb{F}$ be a finite field. Let $\mathcal{L} \subseteq \mathbb{F}^n$ be a language. ($n \ll \mathbb{F}$)

Theorem. If $\mathcal{L}$ is recognized by circuits of size $|\mathcal{C}|$, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:

- $O(1)$ rounds and
- communication cost $\boldsymbol{O(|\mathcal{C}|)}$. (elements of $\mathbb{F}$)

Theorem. If $\mathcal{L}$ has a **degree-two** arithmetic circuit, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:

- $\boldsymbol{k}$ rounds and
- communication cost $\boldsymbol{n^{O(1/k)}}$ (Improves: $\Omega(n)$ [BC17])

Selected results: New ZK proofs

Let $\mathbb{F}$ be a finite field. Let $\mathcal{L} \subseteq \mathbb{F}^n$ be a language. ($n \ll \mathbb{F}$)

Theorem. If $\mathcal{L}$ is recognized by circuits of size $|C|$, there is a pub...

- O...
- co...

Our proofs apply to a much larger class of "structured" languages (see paper)

- Circuits with degree $O(1)$ or repetition or ...

Theorem. If $\mathcal{L}$ has a **degree-two** arithmetic circuit, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:

- $\boldsymbol{k}$ rounds and
- communication cost $\boldsymbol{n^{O(1/k)}}$ (Improves: $\Omega(n)$ [BC17])

This talk

- **ZK proofs on distributed data**
- Fully linear PCPs
- Application: Three-party computation

This talk

- ZK proofs on distributed data
- **Fully linear PCPs**
- Application: Three-party computation

Constructing ZK proofs on distributed data

Step 1. Define "fully linear PCPs"

- A strengthening of linear PCPs [IKO07]
- We then show:

Efficient fully linear PCP for $\mathcal{L}$ → implies → Efficient ZK proof on distributed data for $\mathcal{L}$

Step 2. Construct new fully linear PCPs

Linear probabilistically checkable proofs (PCPs)

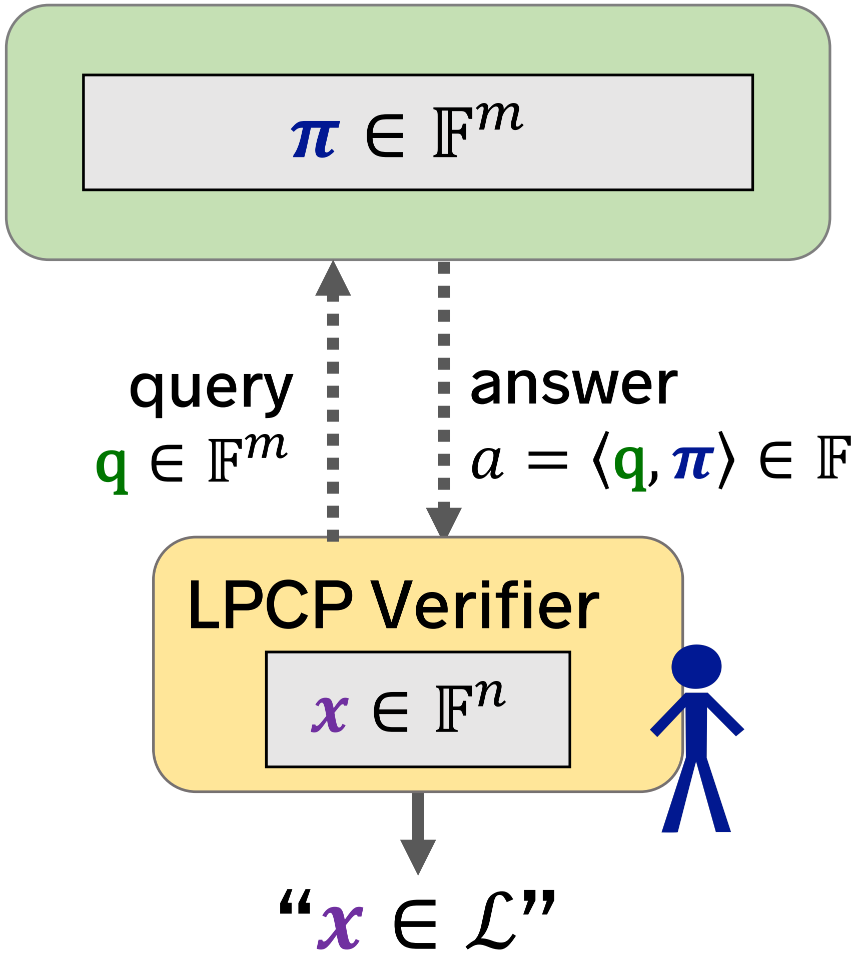
[IKO07]

Finite field $\mathbb{F}$, language $\mathcal{L} \subseteq \mathbb{F}^n$

Linear PCP proof is a vector $\boldsymbol{\pi}$.

Linear PCP verifier
- takes $\boldsymbol{x}$ as input,
- makes $O(1)$ linear queries to $\boldsymbol{\pi}$.

Must satisfy notions of completeness, soundness, and zero knowledge.



Fully linear probabilistically checkable proofs (PCPs)

[This work]

Finite field $\mathbb{F}$, language $\mathcal{L} \subseteq \mathbb{F}^n$

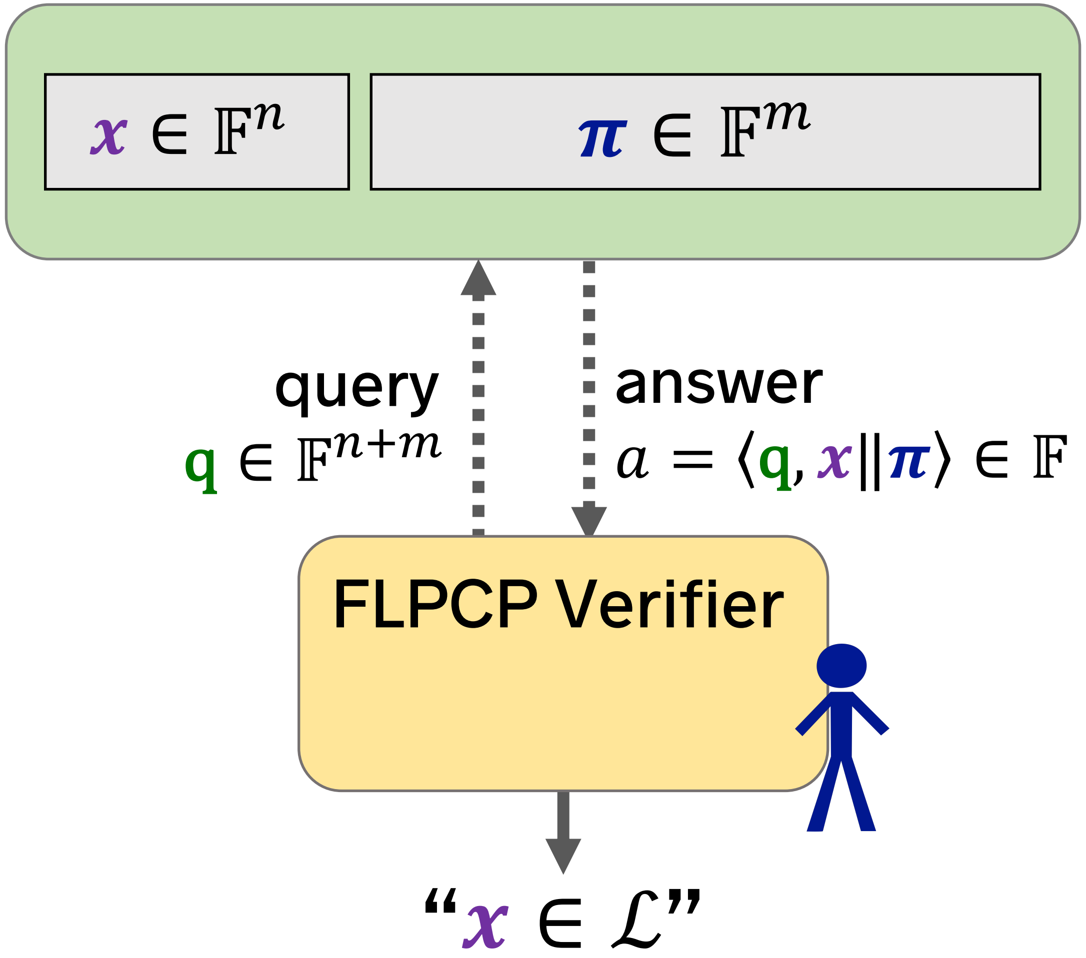
Fully linear PCP proof is a vector $\boldsymbol{\pi}$.

Fully linear PCP verifier

– ~~takes x as input,~~

– makes $O(1)$ linear queries to $(x \| \pi)$.

Must satisfy notions of completeness, soundness, and zero knowledge.



If language $\mathcal{L}$ has an efficient fully linear PCP, it has an efficient ZK proof on distributed data.

1. Generate FLPCP proof and split it using secret sharing.

Verifier V_1 — $\boldsymbol{x_1} \in \mathbb{F}^{n/2}$

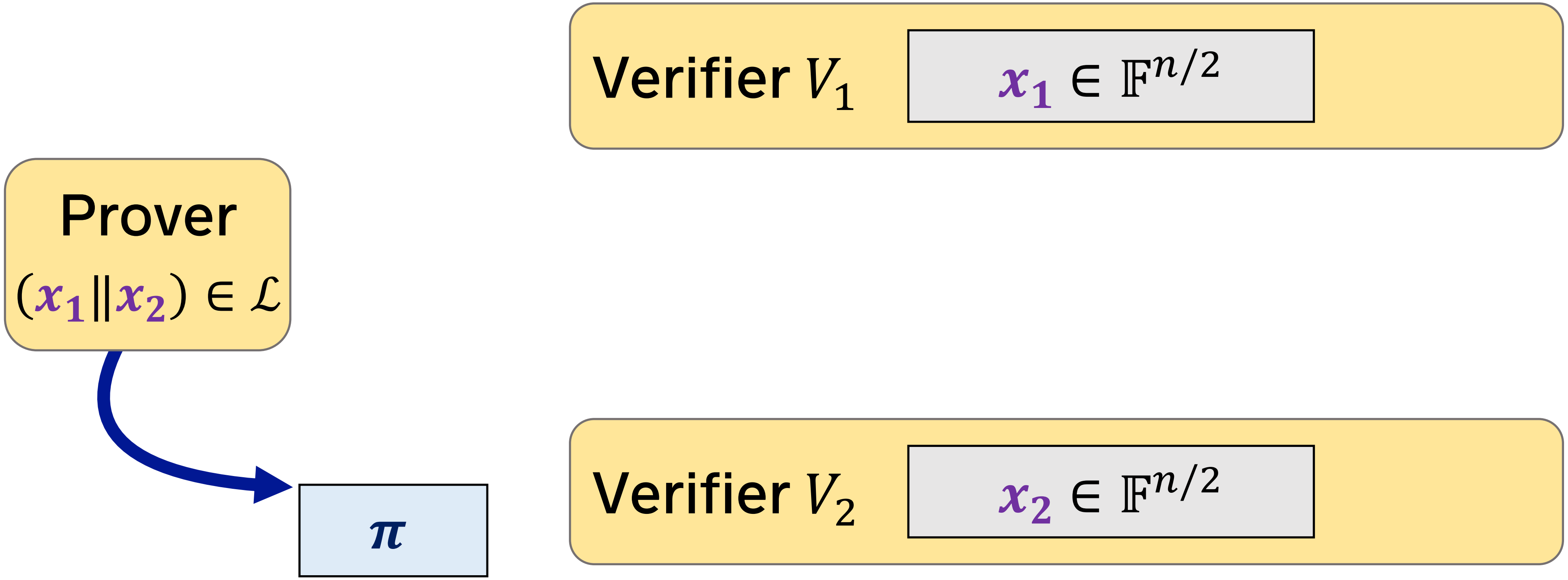
Prover

$(\boldsymbol{x_1} \| \boldsymbol{x_2}) \in \mathcal{L}$

Verifier V_2 — $\boldsymbol{x_2} \in \mathbb{F}^{n/2}$

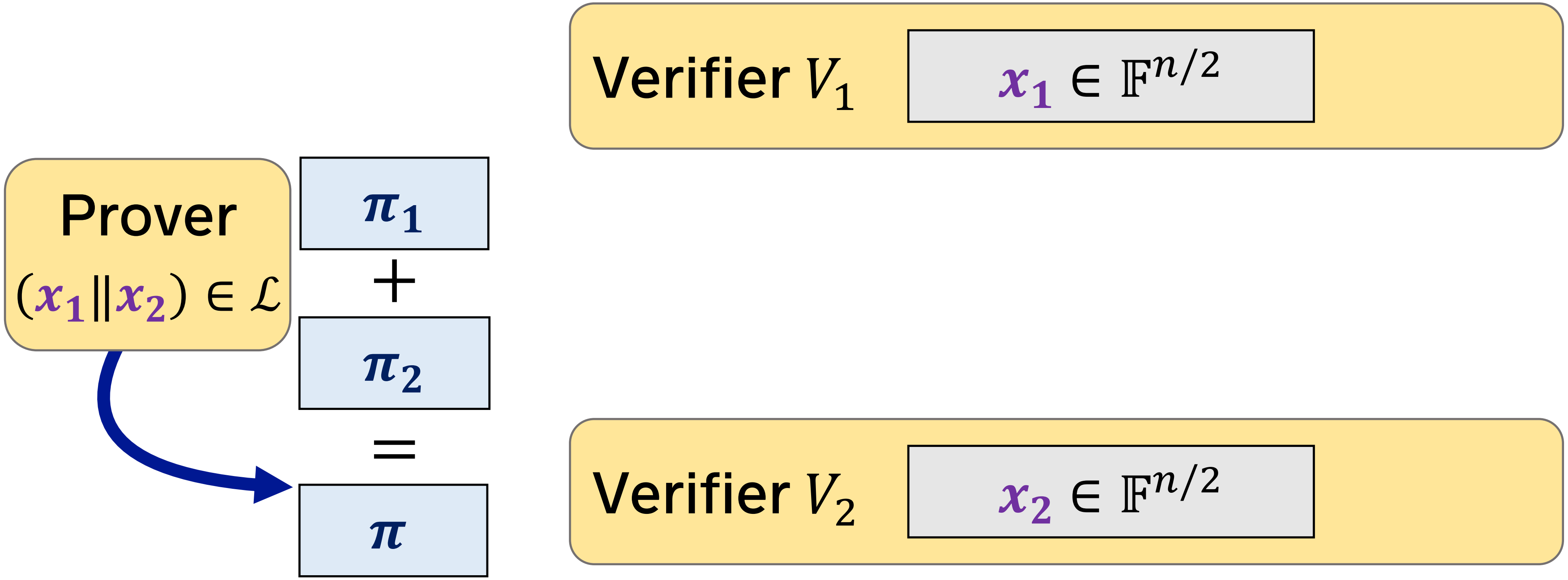
If language $\mathcal{L}$ has an efficient fully linear PCP, it has an efficient ZK proof on distributed data.

1. Generate FLPCP proof and split it using secret sharing.



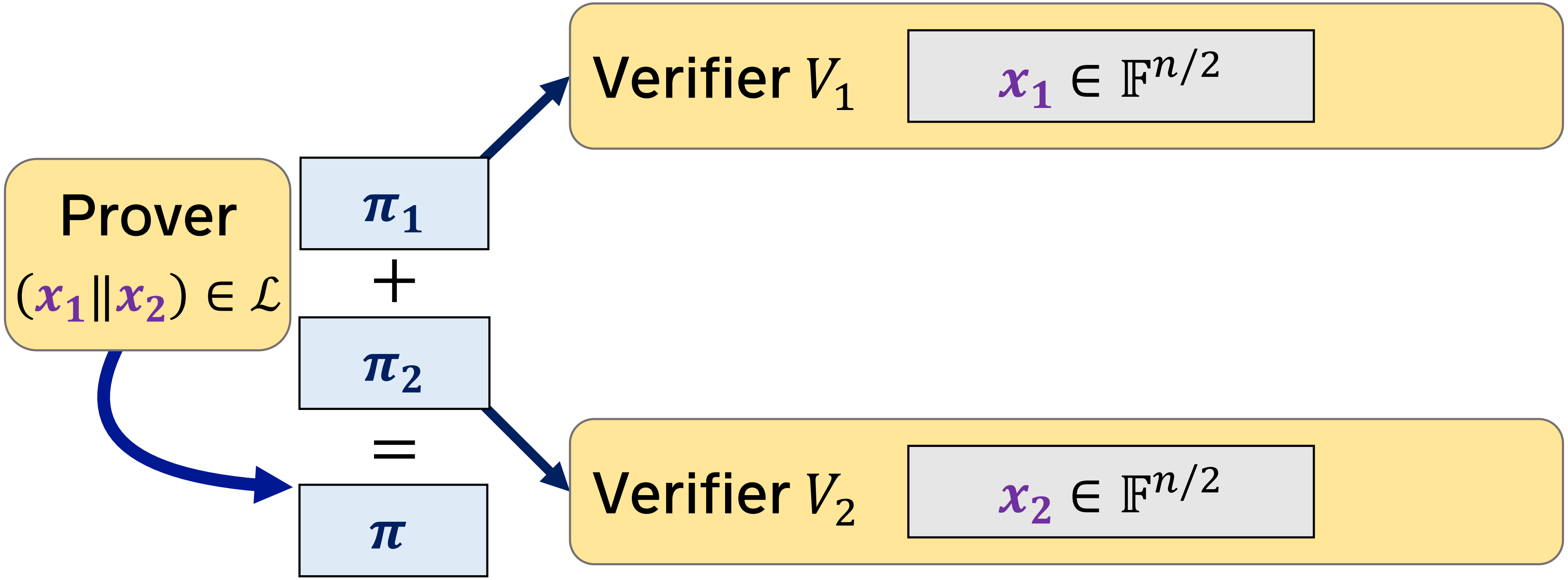
If language $\mathcal{L}$ has an efficient fully linear PCP,
it has an efficient ZK proof on distributed data.

1. Generate FLPCP proof and split it using secret sharing.



If language $\mathcal{L}$ has an efficient fully linear PCP, it has an efficient ZK proof on distributed data.

1. Generate FLPCP proof and split it using secret sharing.



If language $\mathcal{L}$ has an efficient fully linear PCP, it has an efficient ZK proof on distributed data.

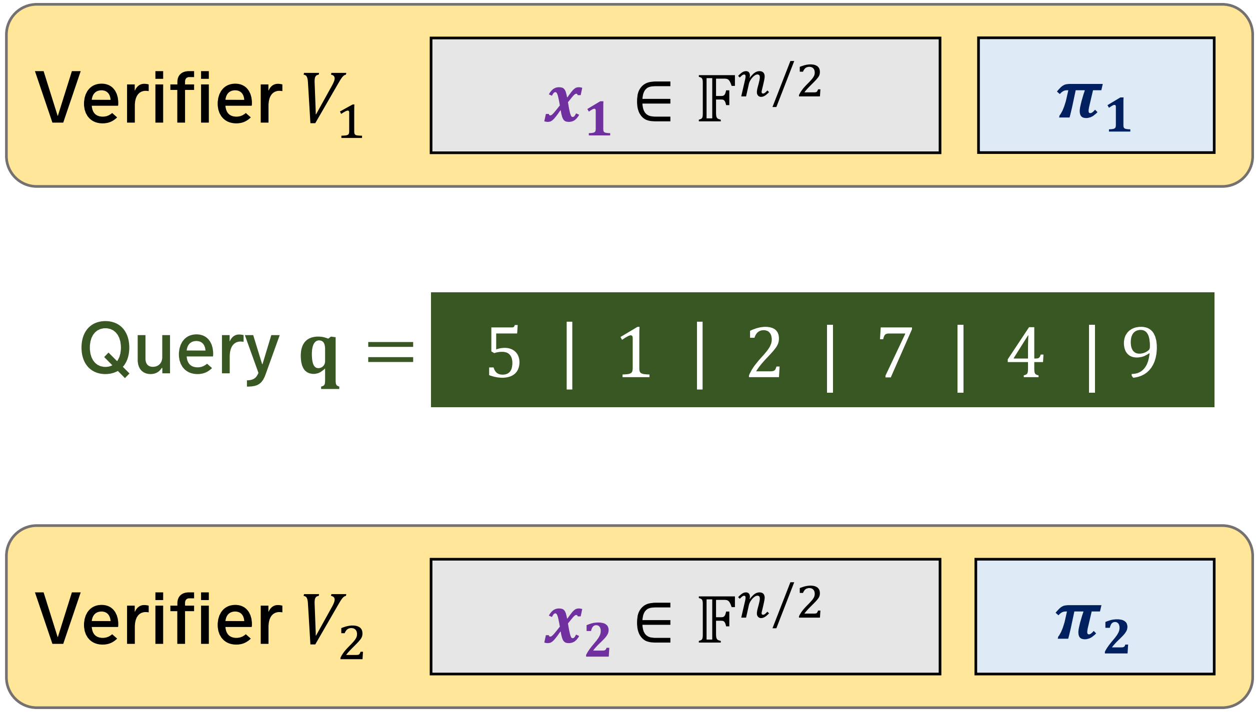
2. Sample query vectors using common randomness.

Verifier V_1 | $\boldsymbol{x_1} \in \mathbb{F}^{n/2}$ | $\boldsymbol{\pi_1}$

Verifier V_2 | $\boldsymbol{x_2} \in \mathbb{F}^{n/2}$ | $\boldsymbol{\pi_2}$

If language $\mathcal{L}$ has an efficient fully linear PCP, it has an efficient ZK proof on distributed data.

2. Sample query vectors using common randomness.



If language $\mathcal{L}$ has an efficient fully linear PCP, it has an efficient ZK proof on distributed data.

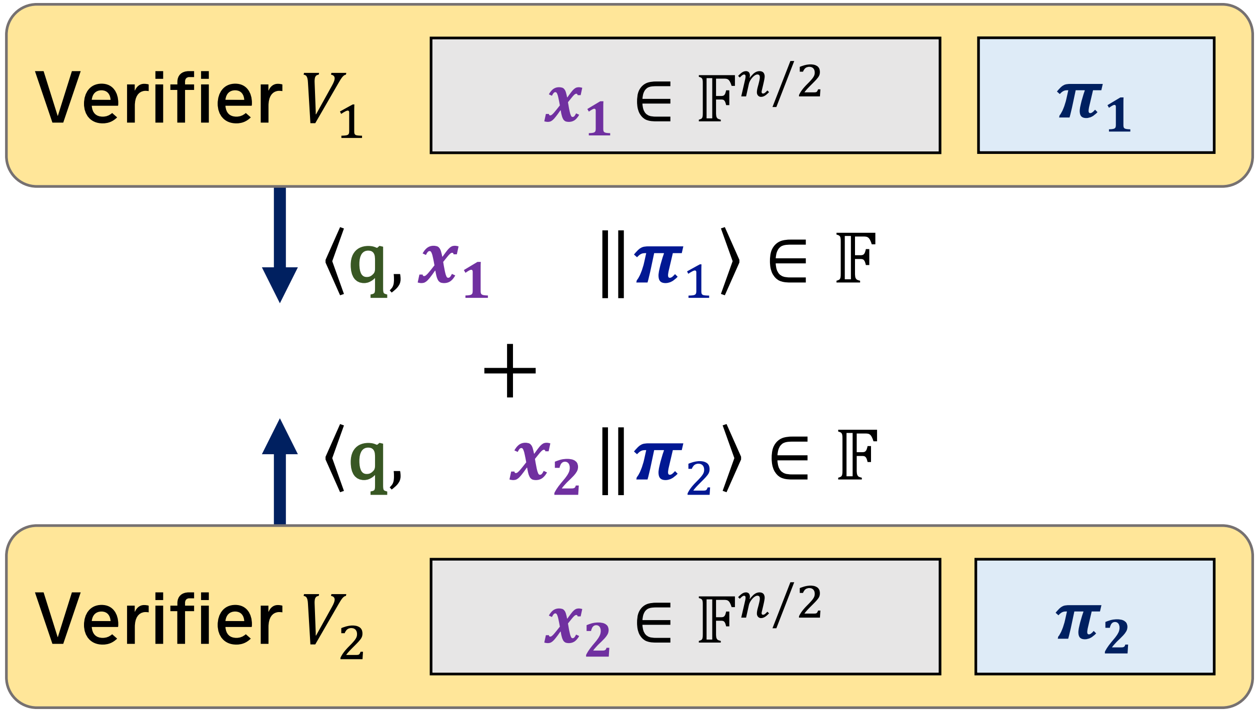
3. Publish shares of query answers and reconstruct.

Verifier V_1 | $\boldsymbol{x_1} \in \mathbb{F}^{n/2}$ | $\boldsymbol{\pi_1}$

Verifier V_2 | $\boldsymbol{x_2} \in \mathbb{F}^{n/2}$ | $\boldsymbol{\pi_2}$

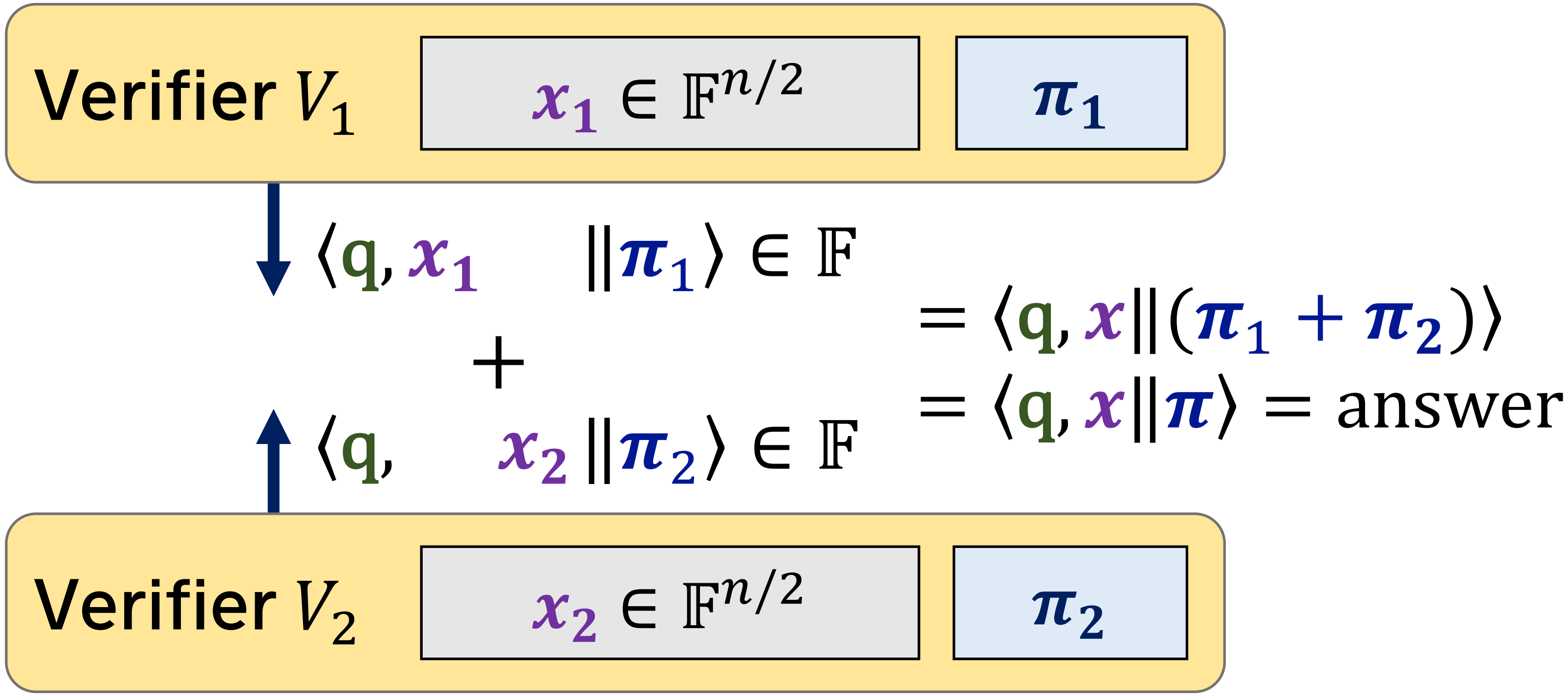
If language $\mathcal{L}$ has an efficient fully linear PCP, it has an efficient ZK proof on distributed data.

3. Publish shares of query answers and reconstruct.



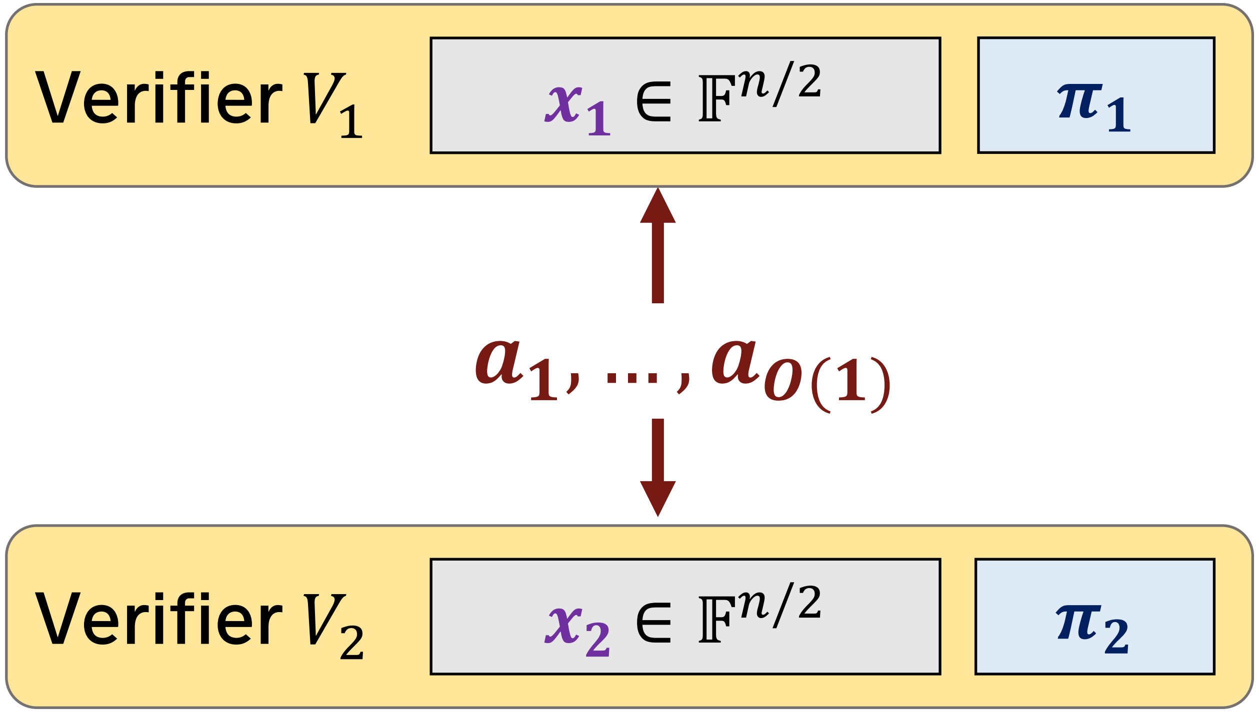
If language $\mathcal{L}$ has an efficient fully linear PCP, it has an efficient ZK proof on distributed data.

3. Publish shares of query answers and reconstruct.



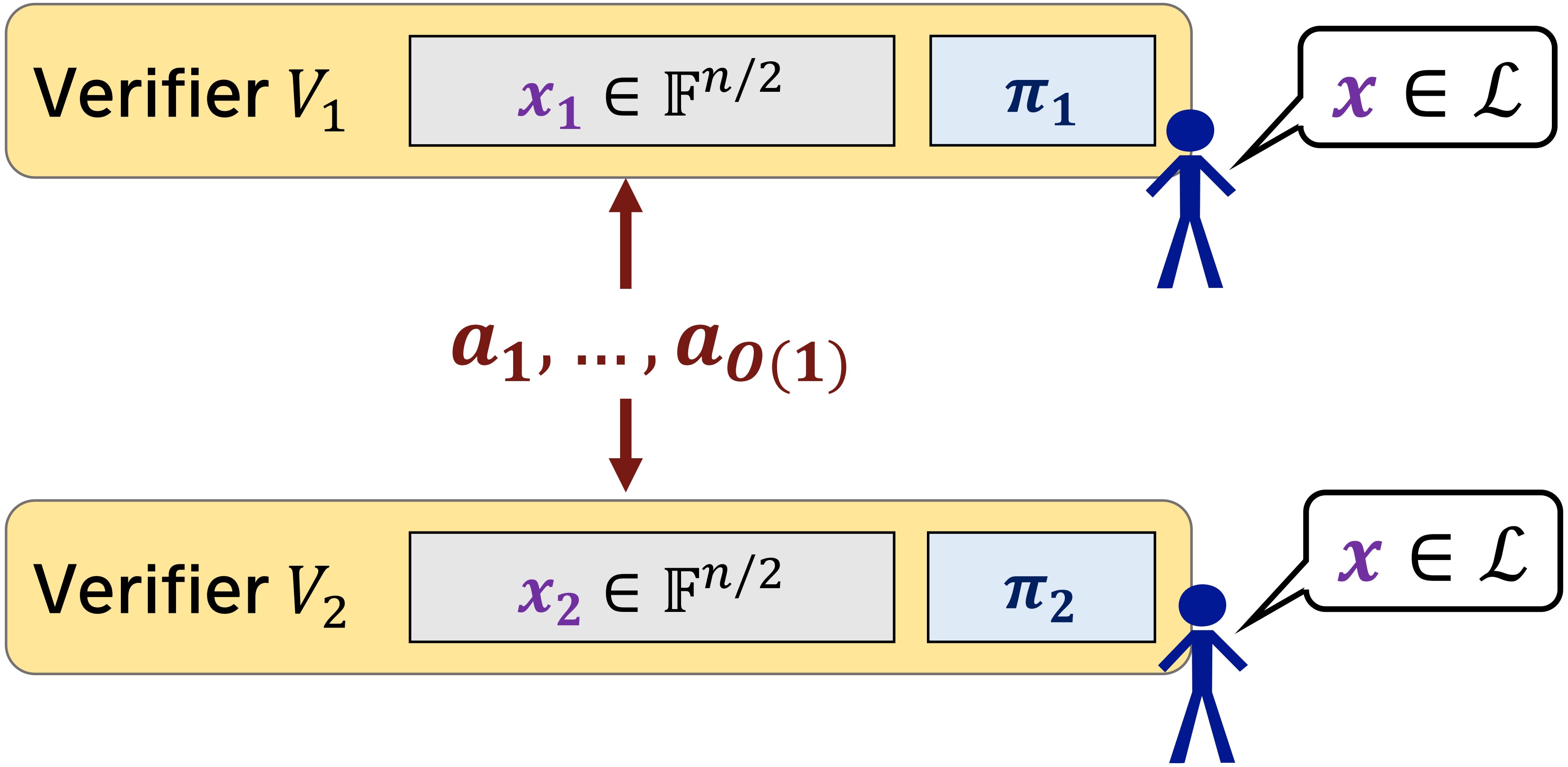
If language $\mathcal{L}$ has an efficient fully linear PCP, it has an efficient ZK proof on distributed data.

4. Recover $O(1)$ query answers, run FLPCP verifier.



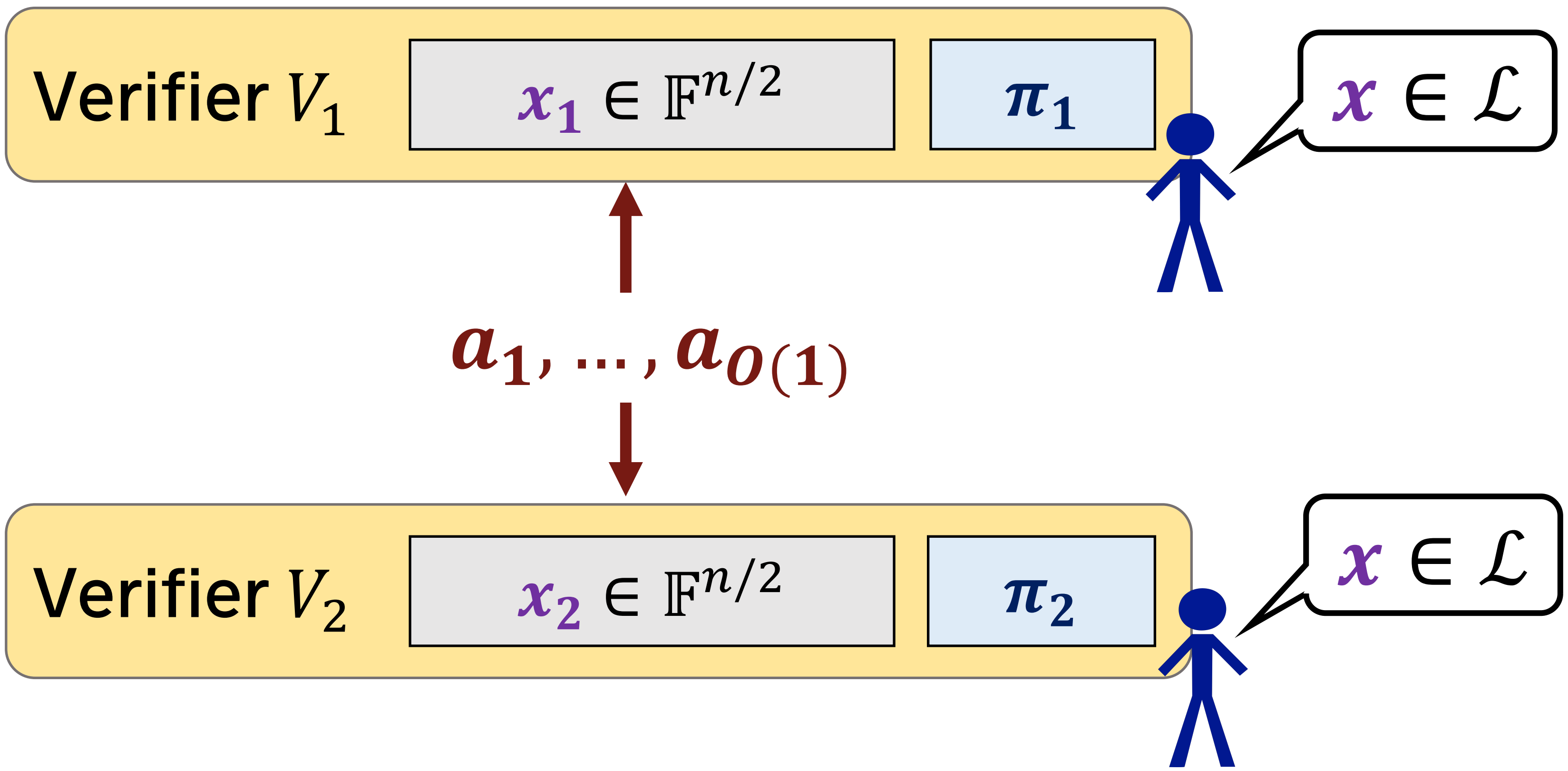
If language $\mathcal{L}$ has an efficient fully linear PCP, it has an efficient ZK proof on distributed data.

4. Recover $O(1)$ query answers, run FLPCP verifier.



If language $\mathcal{L}$ has an efficient fully linear PCP, it has an efficient ZK proof on distributed data.

4. Recover $O(1)$ query answers, run FLPCP verifier.



Communication:
$2|\text{proof}| + O(1)$

Fully linear PCPs: Constructions

- Many existing linear PCPs are also fully linear
  - Linear PCPs [IKO07], Pepper [SMBW12], [GGPR13], [BCIOP13], ...
  - Downside: for circuit size $|\mathcal{C}|$, proof size $\Omega(|\mathcal{C}|)$.
- **We get new shorter proofs using interaction**
  - Applies to "structured" languages
- **Our proofs are closely related to:**
  - Aaronson-Wigderson protocol in comm. complexity [AW09]
  - Interactive PCP and oracle proofs [KR08], [BCS16], [RRR16]
  - Sum-check-like proof systems [BFLS91], [GKR08], [W16]

Short proofs for degree-two circuits

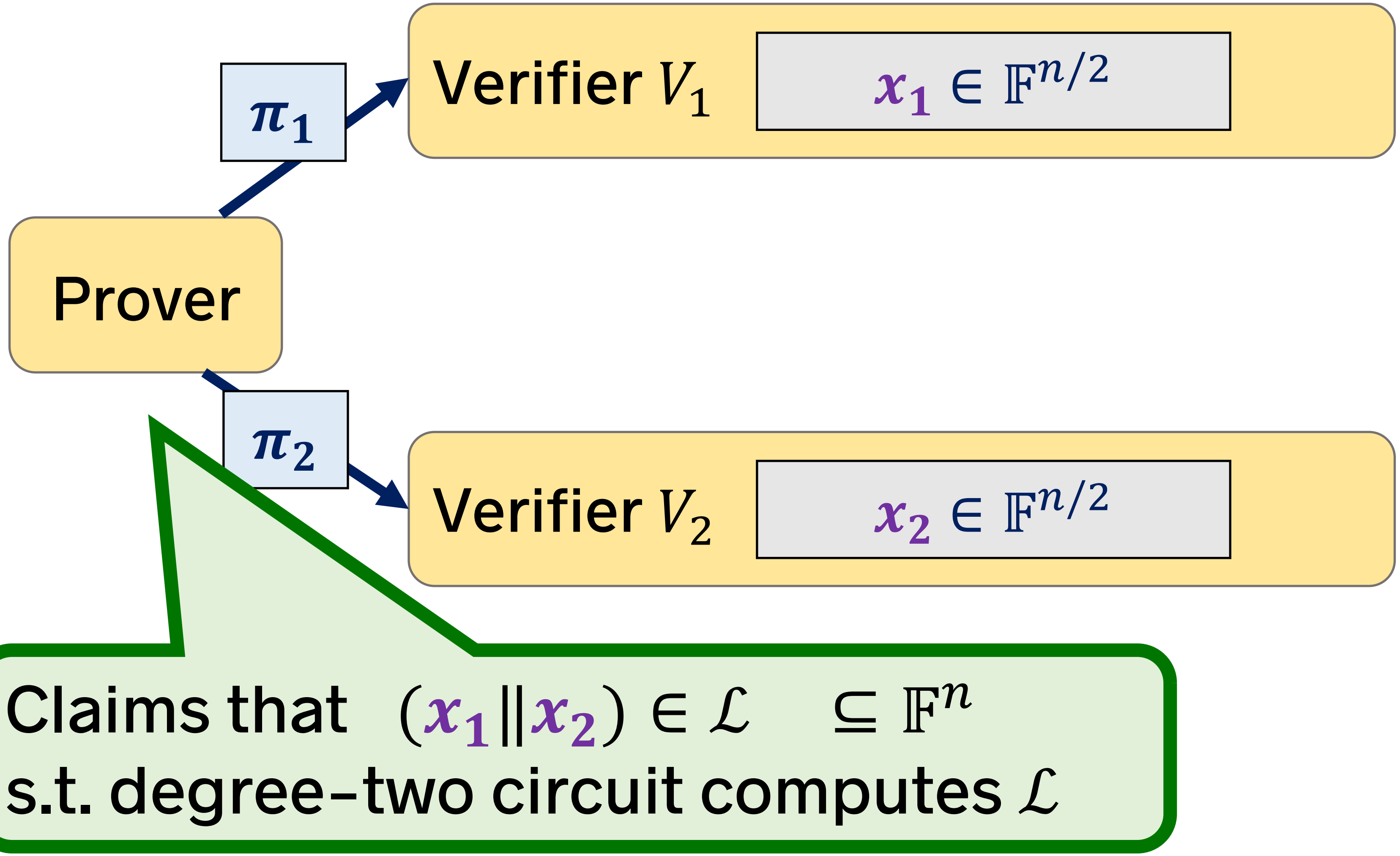
Verifier V_1 $\boldsymbol{x_1} \in \mathbb{F}^{n/2}$

Prover

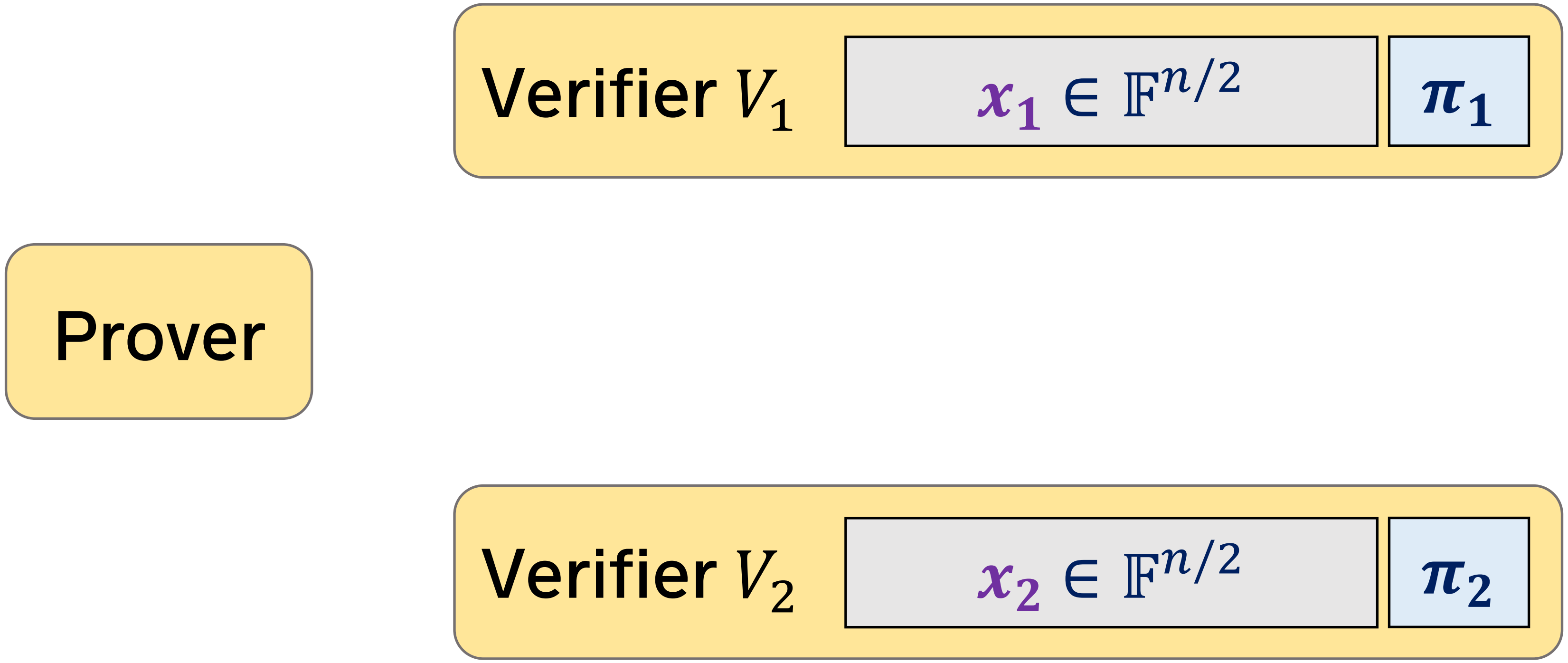
Verifier V_2 $\boldsymbol{x_2} \in \mathbb{F}^{n/2}$

Claims that $(\boldsymbol{x_1} \| \boldsymbol{x_2}) \in \mathcal{L} \subseteq \mathbb{F}^n$
s.t. degree-two circuit computes $\mathcal{L}$

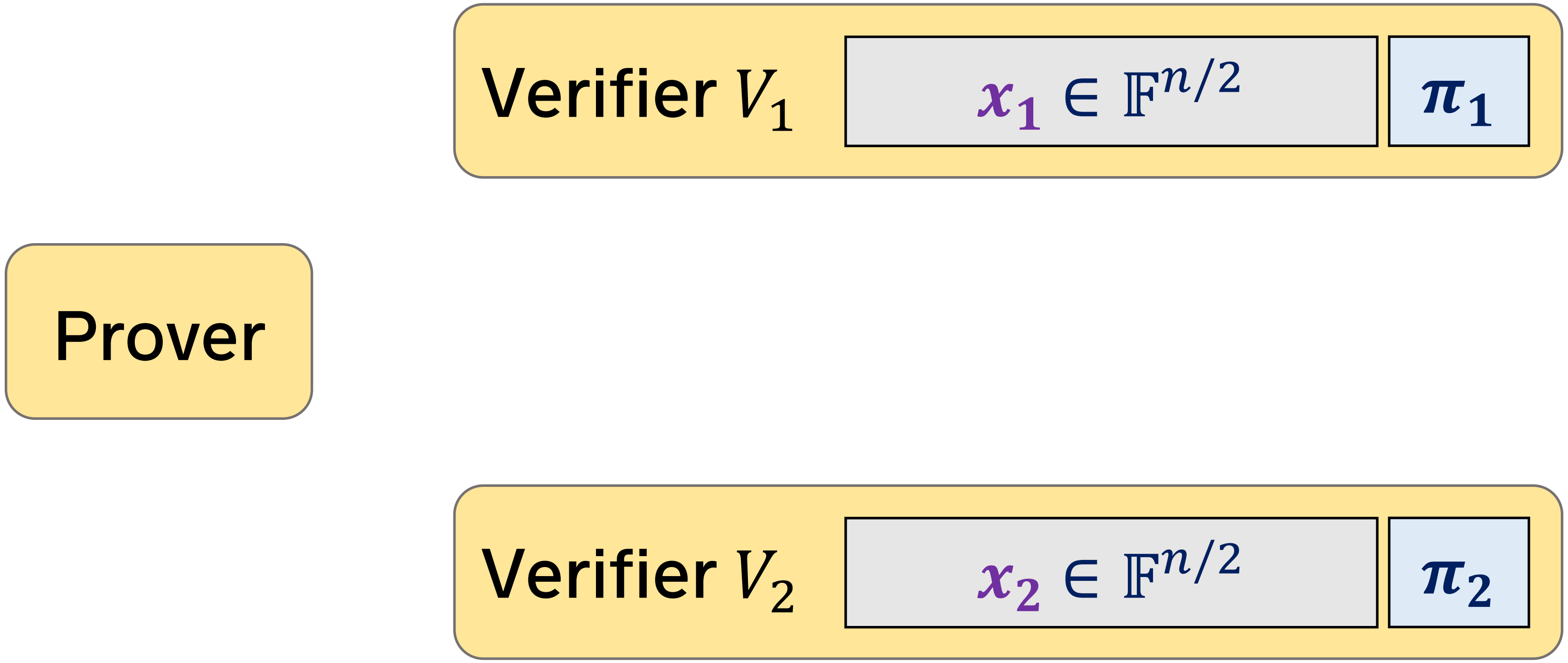
Short proofs for degree–two circuits



Short proofs for degree-two circuits



Short proofs for degree–two circuits

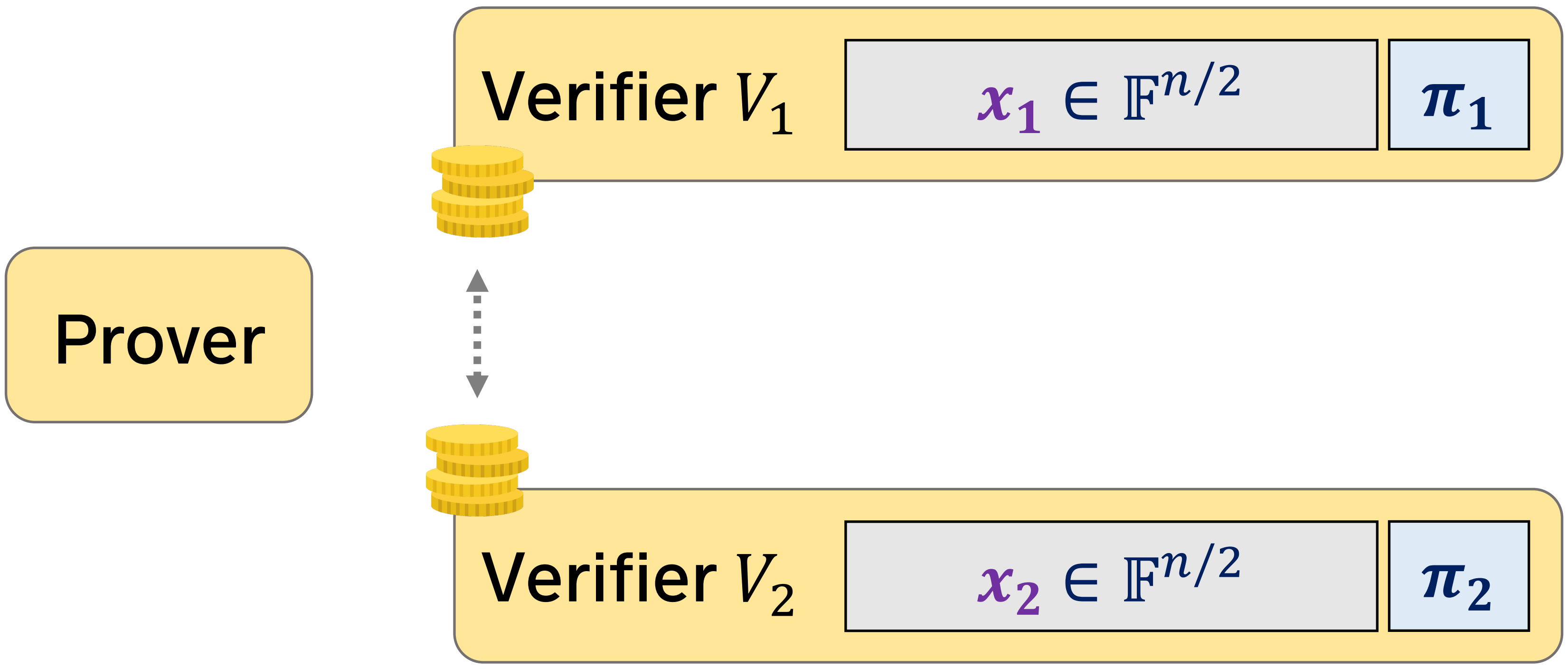


To check proof:

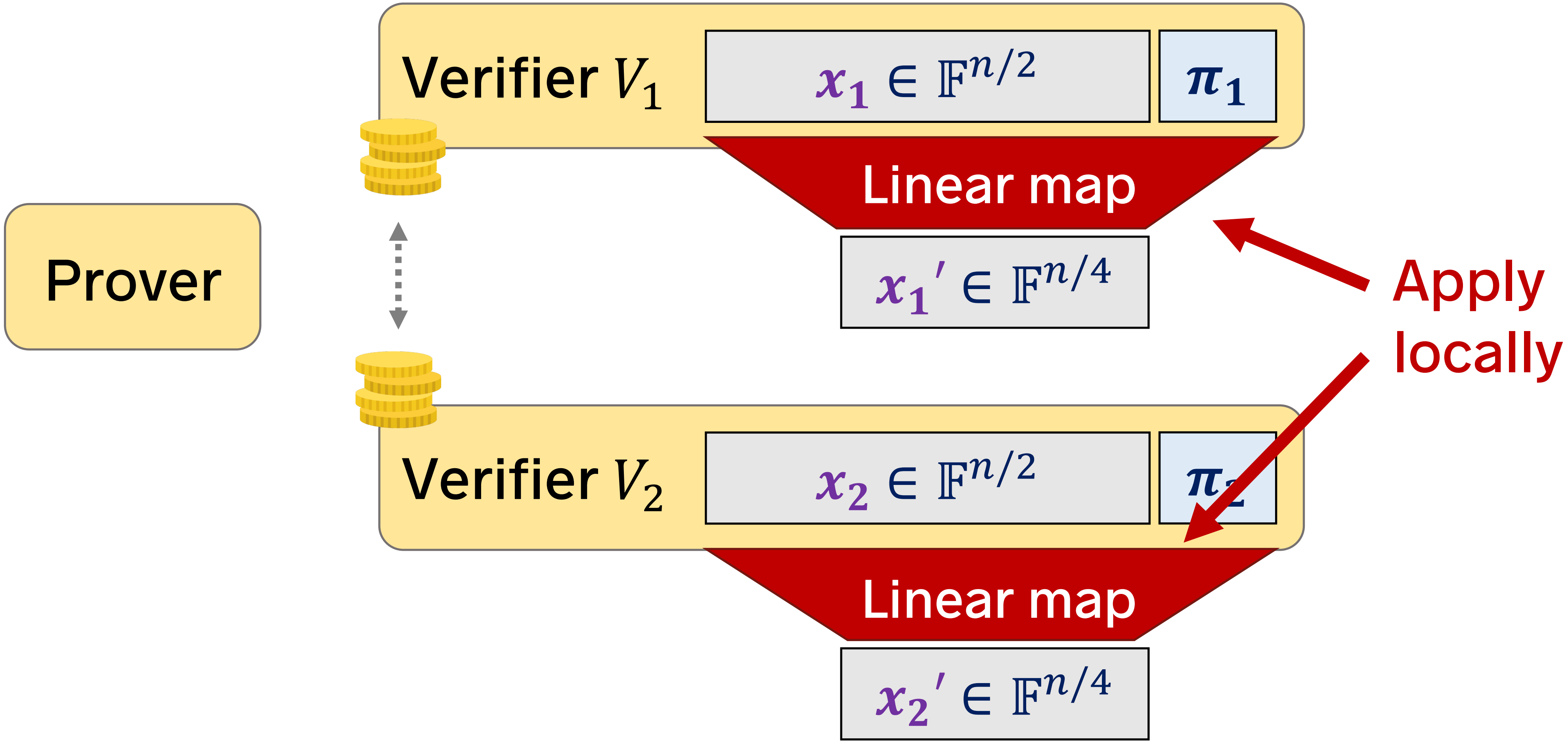
(1) apply a randomized linear map to (x, π) and

(2) evaluate a degree–two circuit on result.

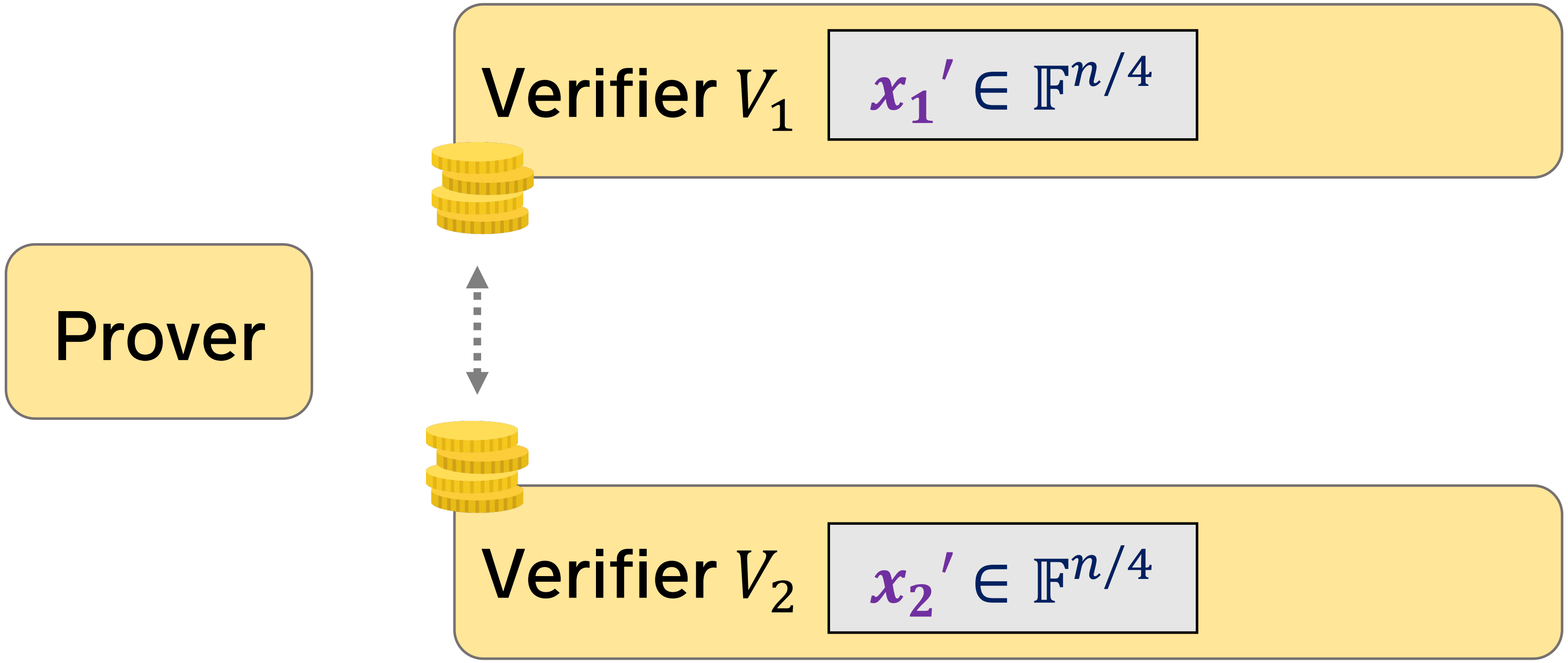
Short proofs for degree-two circuits



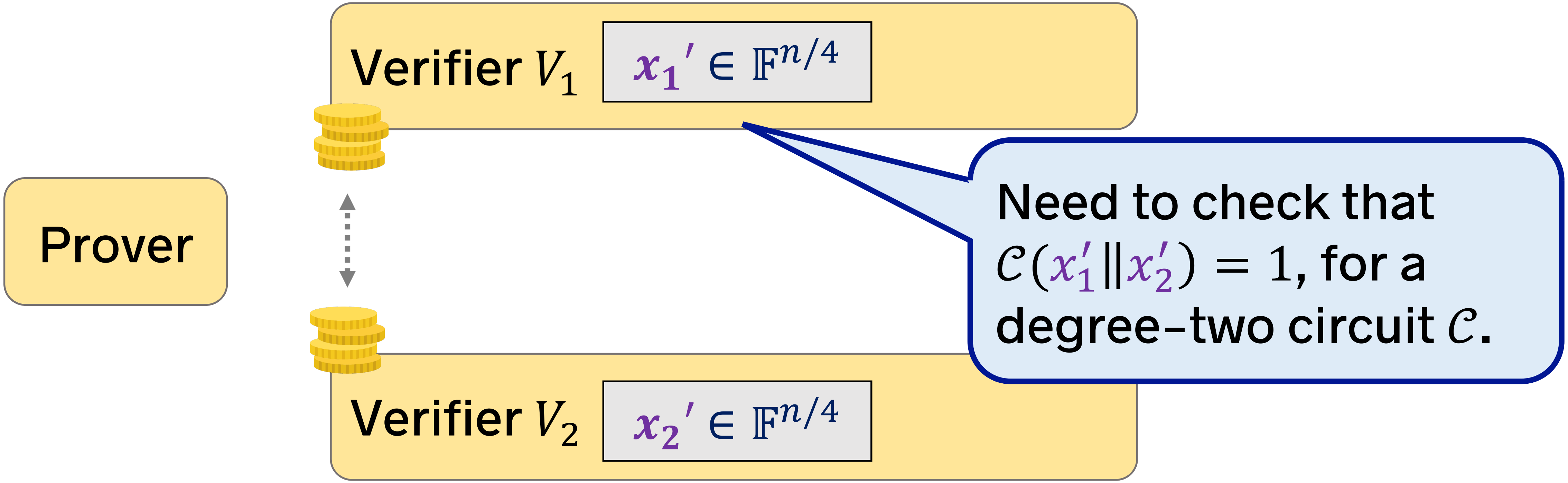
Short proofs for degree-two circuits



Short proofs for degree-two circuits



Short proofs for degree-two circuits



Short proofs for degree-two circuits

Verifier V_1 $\boldsymbol{x_1}' \in \mathbb{F}^{n/4}$

Prover

Verifier V_2 $\boldsymbol{x_2}' \in \mathbb{F}^{n/4}$

Need to check that $\mathcal{C}(x_1' \| x_2') = 1$, for a degree-two circuit $\mathcal{C}$.

- Send coins to prover.
- Invoke proof system recursively.

This talk

- ZK proofs on distributed data
- **Fully linear PCPs**
- Application: Three-party computation

This talk

- ZK proofs on distributed data
- Fully linear PCPs
- **Application: Three-party computation**

Our results: Application to MPC

Theorem. For any arithmetic circuit $\mathcal{C}$ over field $\mathbb{F}$, there is a secure three-party protocol for computing $\mathcal{C}$ that

- Tolerates **one malicious party**
- Is computationally secure with abort (assuming only PRGs)
- Has amortized communication **1** element of $\mathbb{F}$ per party per gate.

	Over $\mathbb{Z}_2$	Large fields
State of the art	7 [ABFLLNOWW17], ...	2 [CGHIKLN18], ...
This work	1	1

Our results: Application to MPC

Theorem. For any arithmetic circuit $\mathcal{C}$ over field $\mathbb{F}$, there is a secure three-party protocol for computing $\mathcal{C}$ that

- Tolerates **one malicious party**
- Is computationally secure with abort (assuming only PRGs)
- Has amortized communication **1** element of $\mathbb{F}$ per party per gate.

	Over $\mathbb{Z}_2$	Large fields
State of the art	**7** [ABFLLNOWW17], …	**2** [CGHIKLN18], …
This work	**1**	**1**

Our results: Application to MPC

Theorem. For any arithmetic circuit $\mathcal{C}$ over field $\mathbb{F}$, there is a secure three-party protocol for computing $\mathcal{C}$ that

- Tolerates **one malicious party**
- Is computationally secure with abort (assuming only PRGs)
- Has amortized communication **1** element of $\mathbb{F}$ per party per gate.

	Over $\mathbb{Z}_2$
State of the art	**7** [ABFLLNOWW17],
This work	**1**

Matches cost of best 3PC with semi-honest security [AFLNO16]

Our results: Application to MPC

Theorem. For any arithmetic circuit $\mathcal{C}$ over field $\mathbb{F}$, there is a secure three-party protocol for computing $\mathcal{C}$ that

- Tolerates **one malicious party**
- Is computationally secure with abort (assuming only PRGs)
- Has amortized communication **1** element of $\mathbb{F}$ per party per gate.

	Over $\mathbb{Z}_2$	Large fields
State of the art	**7** [ABFLLNOWW17], ...	**2** [CGHIKLN18], ...
This work	**1**	**1**

We use a semi-honest MPC protocol Φ that has two extra properties…

I. **Protocol reveals nothing until the last message.**
- Holds even if some parties are malicious.
- Malicious behavior at last message can only cause abort.

II. **Checkable by a degree-two relation.**

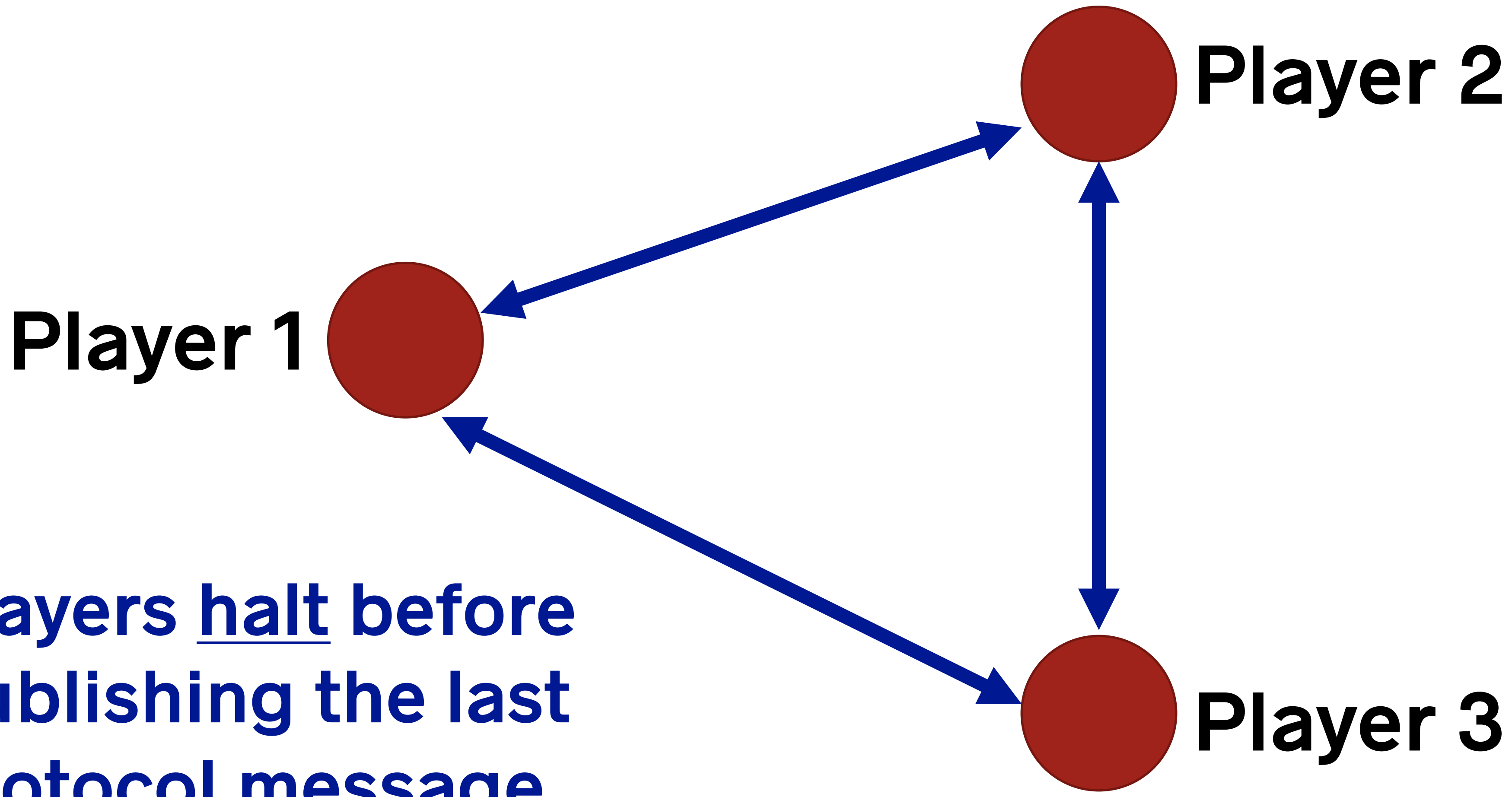
Each of player i's messages is a degree-two function of:

1. player i's input and
2. the messages that player i has received so far.

Can instantiate with existing protocols: [AFLNO16], [KKW18], …

Overview of 3PC our protocol

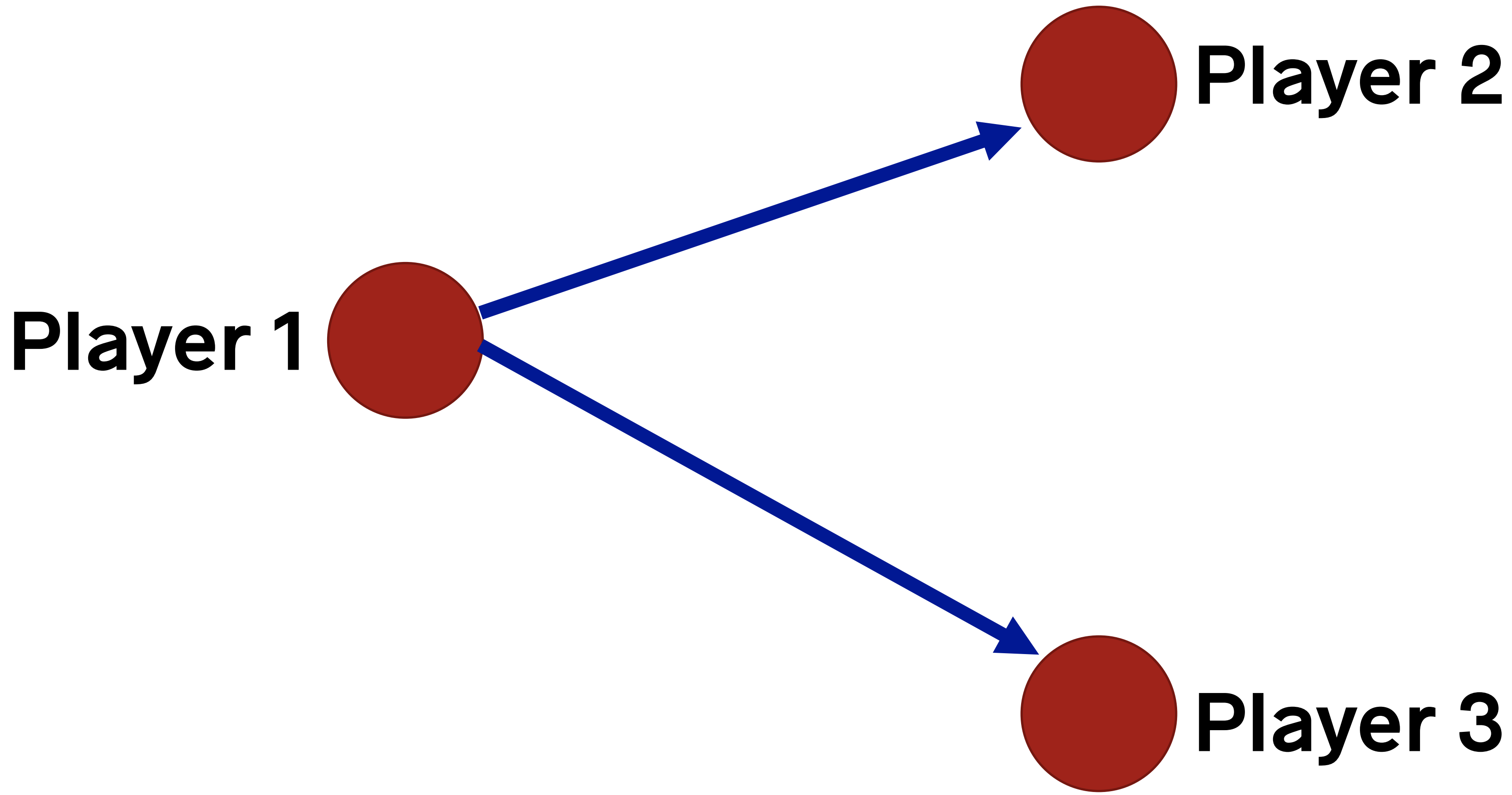
1. Run semi-honest MPC protocol Φ



Players halt before publishing the last protocol message

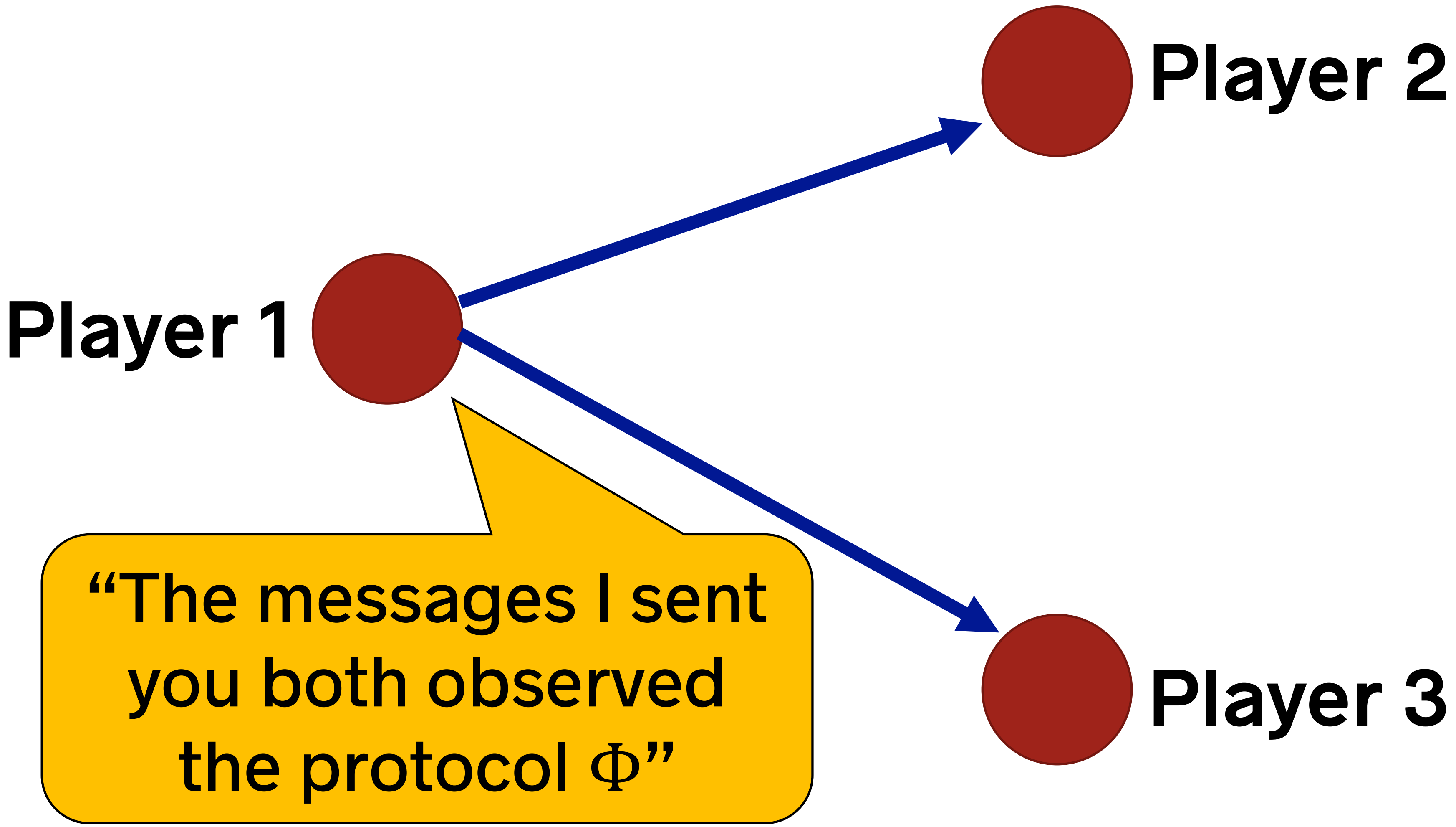
Overview of 3PC our protocol

2. Prove that messages complied with Φ



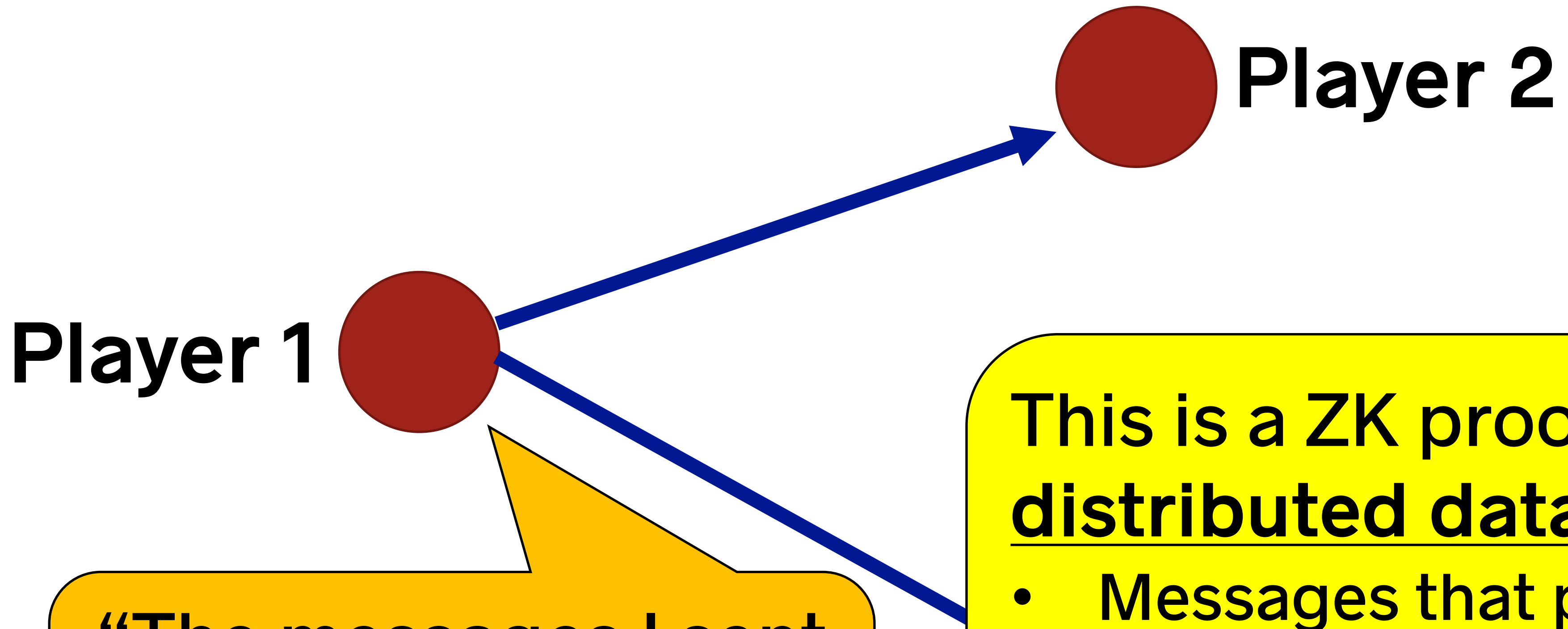
Overview of 3PC our protocol

2. Prove that messages complied with Φ



Overview of 3PC our protocol

2. Prove that messages complied with Φ



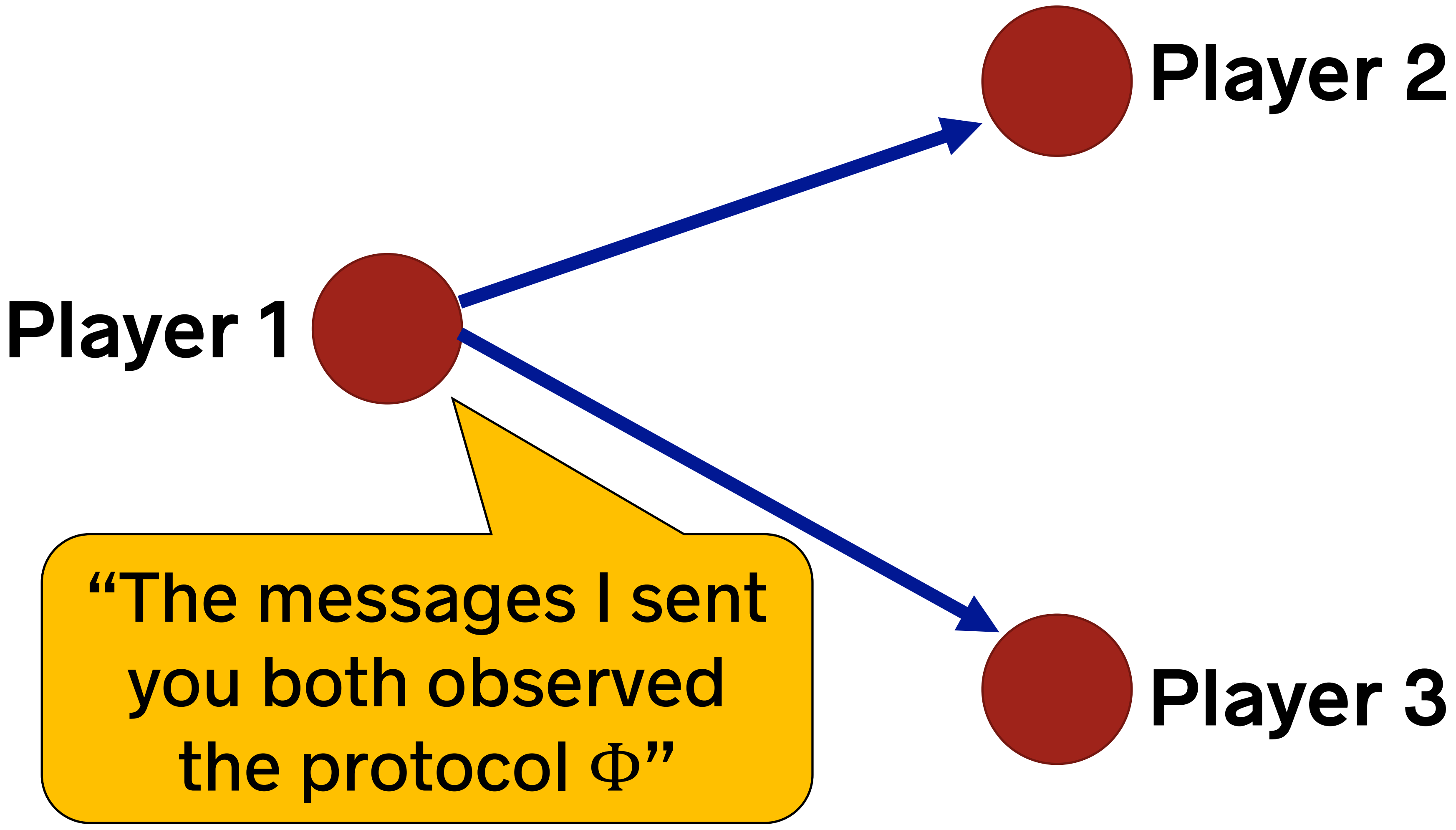
"The messages I sent you both observed the protocol Φ"

This is a ZK proof on **distributed data:**

- Messages that player 1 sent are split across players 2 and 3
- The language is recognized by a **degree-two** circuit

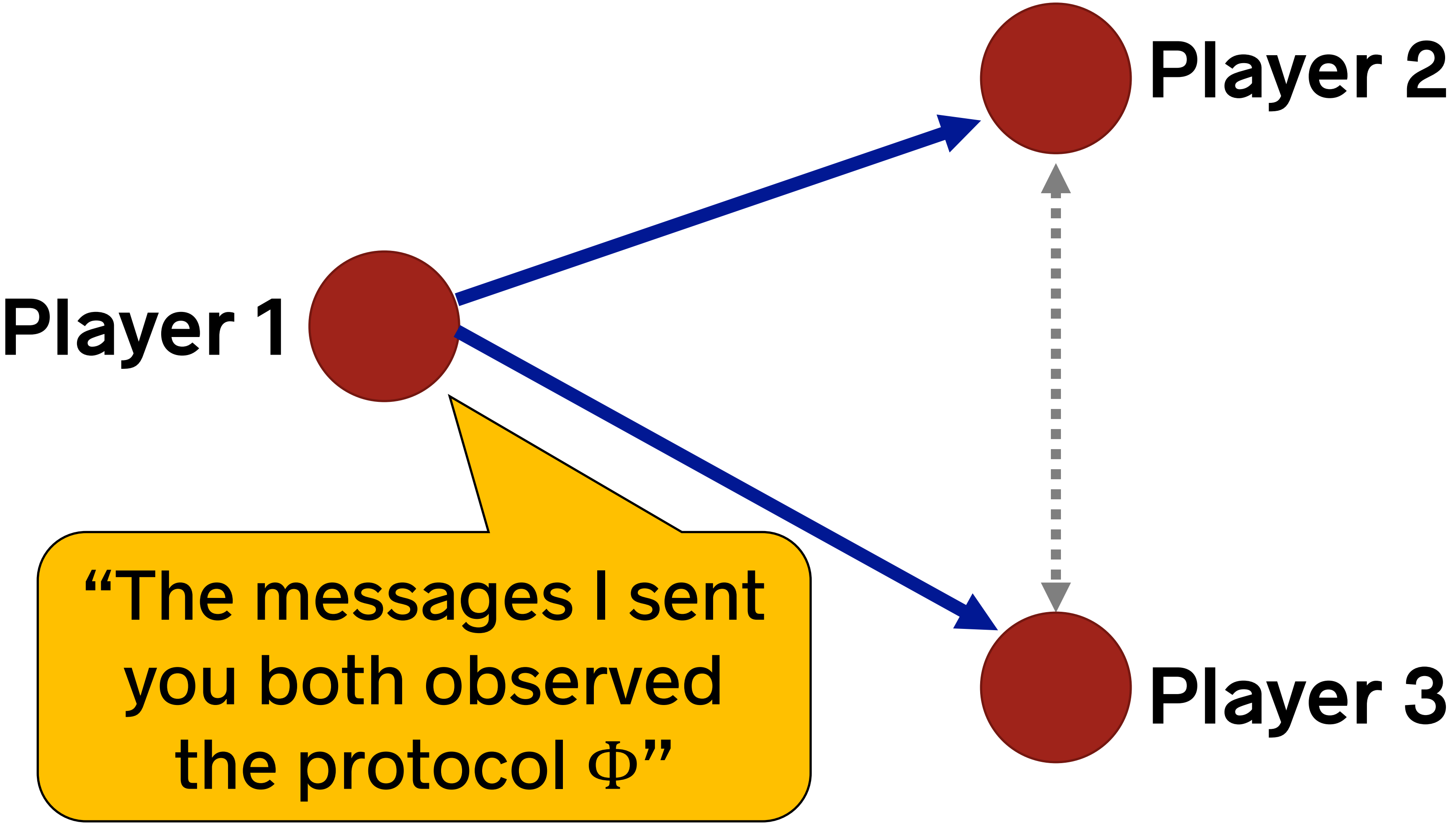
Overview of 3PC our protocol

2. **Prove** that messages complied with Φ



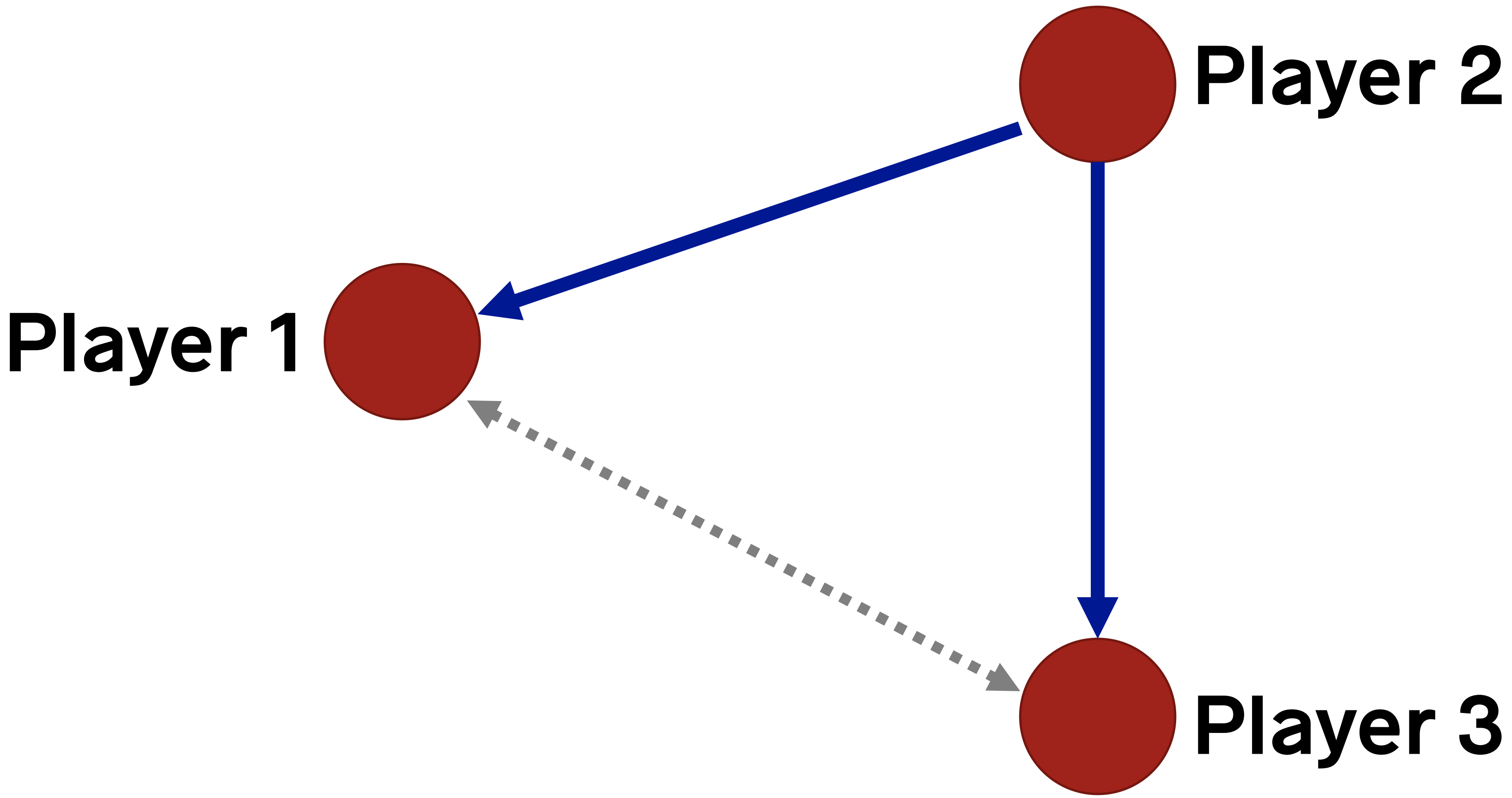
Overview of 3PC our protocol

2. Prove that messages complied with Φ



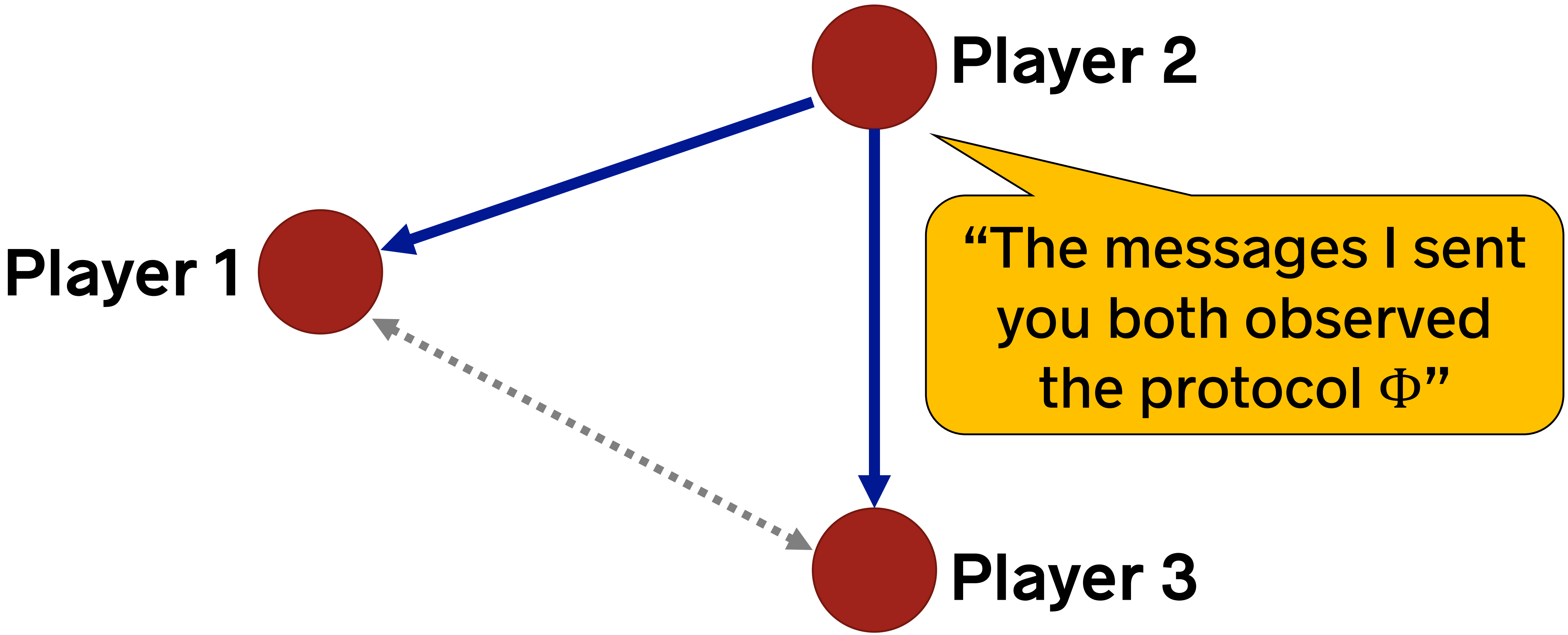
Overview of 3PC our protocol

2. Prove that messages complied with Φ



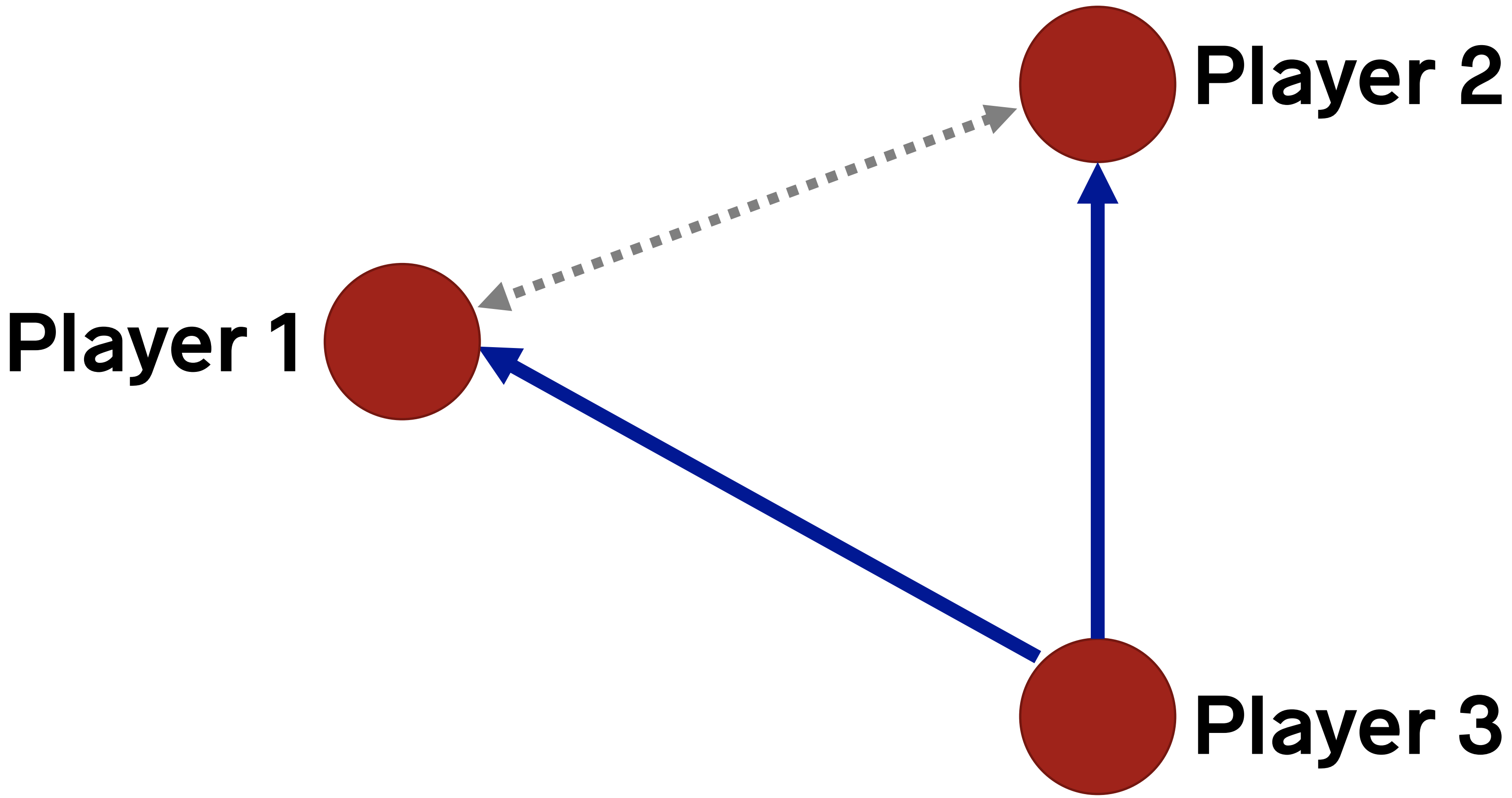
Overview of 3PC our protocol

2. Prove that messages complied with Φ



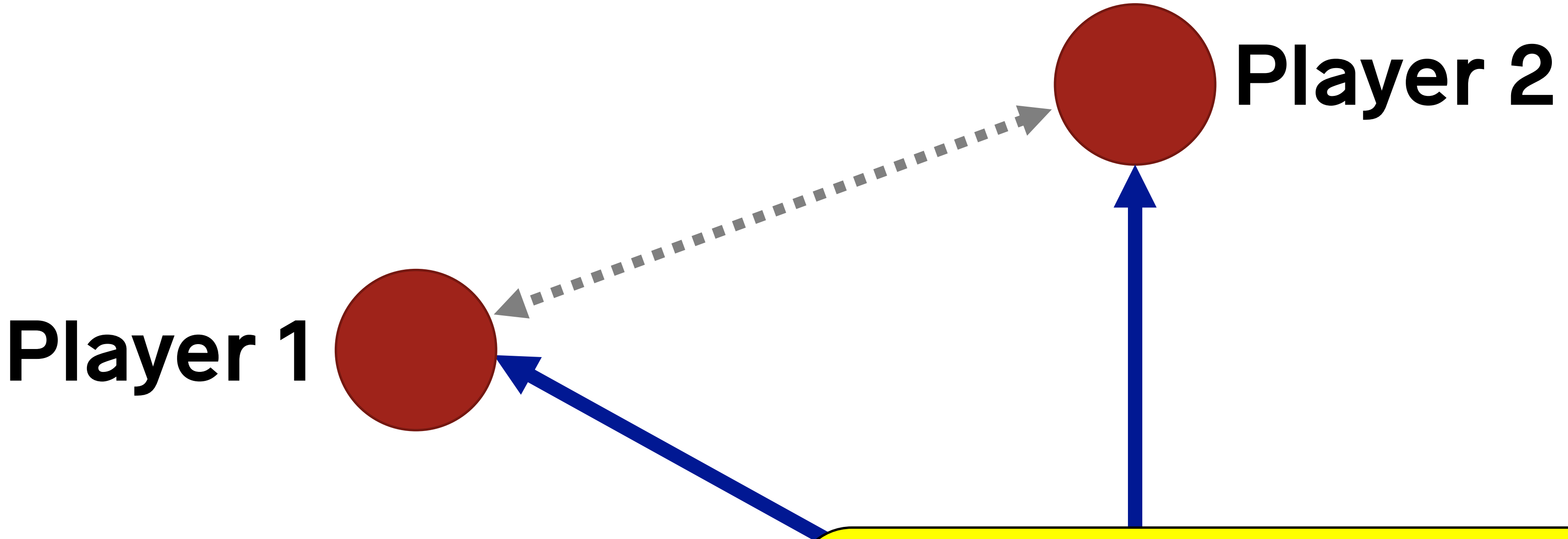
Overview of 3PC our protocol

2. Prove that messages complied with Φ



Overview of 3PC our protocol

2. Prove that messages complied with Φ

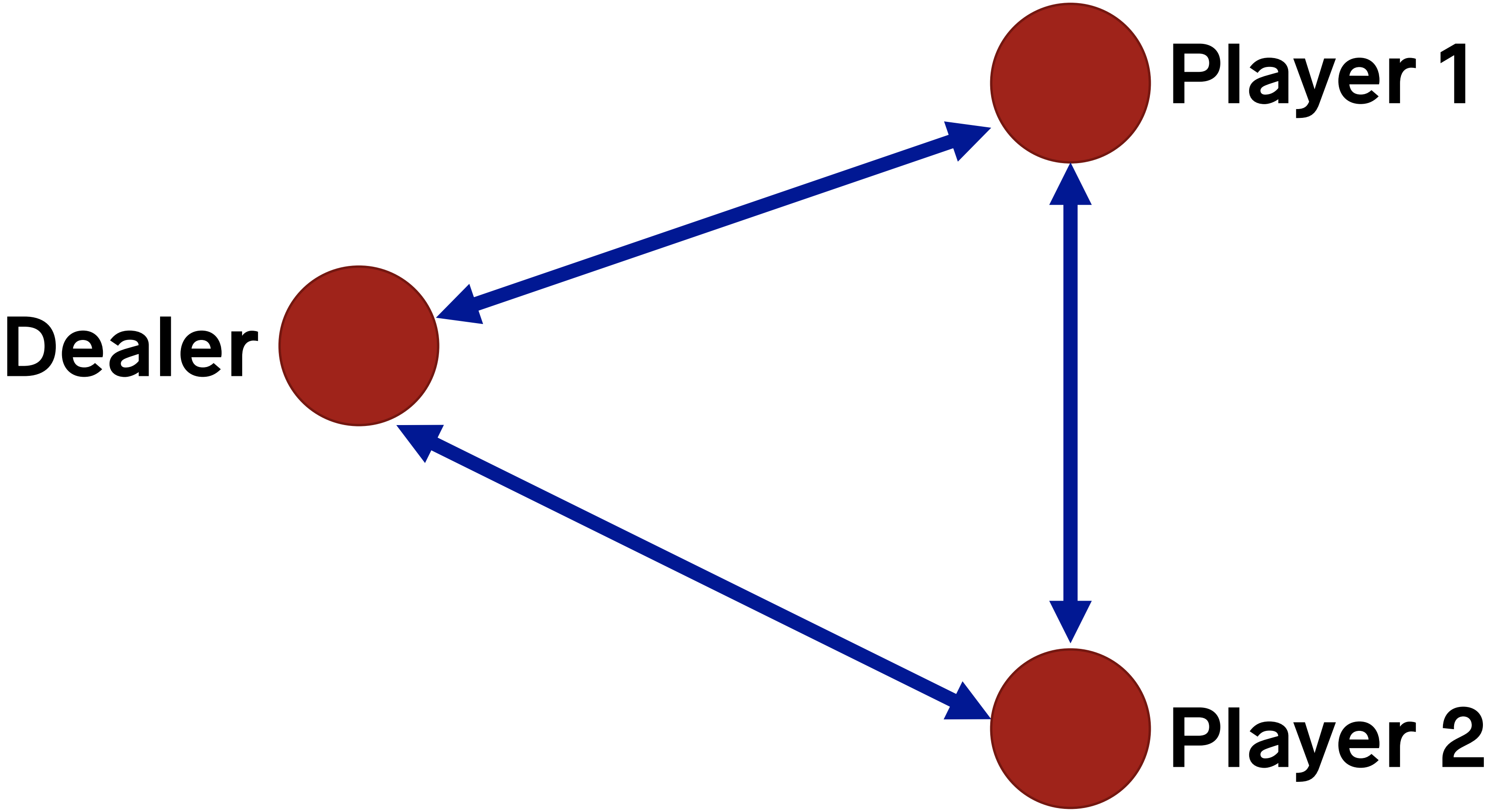


Communication:
$O(\log|\mathcal{C}|)$ per player

Possible with our new ZK proofs on distributed data for degree-two relations

Overview of 3PC our protocol

3. Reveal last message to recover output



Summary of our three-party protocol

Communication cost per player (field elements)

Messages from Φ	$\lvert\mathcal{C}\rvert + o(\lvert\mathcal{C}\rvert)$
Proofs	$O(\log \lvert\mathcal{C}\rvert)$
TOTAL	$\lvert\mathcal{C}\rvert + o(\lvert\mathcal{C}\rvert)$
...per gate:	$\mathbf{1 + \mathit{o}(1)}$

Generalizations: [See paper]

- to $O(1)$-parties with honest majority
- to arbitrary rings $\mathbb{Z}_{2^k}$

Comparison to GMW compiler [GMW87]

Like GMW, our compiler converts:

Semi-honest Φ $\rightarrow$ Malicious-secure Φ

Differences:

- GMW uses "message-by-message" ZK proofs.
  We use one big (but sublinear-size) proof at the end.

- GMW requires assumptions/commitments.
  Our compiler is information theoretically secure.

- GMW requires that all players see all messages (**broadcast channel**).
  With distributed ZK, can use **point-to-point channels**.

Summary: ZK proofs on distributed data

- One prover, <u>multiple</u> verifiers, each with different input
  - Protocol hides verifiers' inputs from each other
- Proofs are information theoretic and lightweight
- **New key tool:** Fully linear proof systems
  - Can unify with sum-check-based proofs? [GKR08], [CTY11], [T16], ...
- **Applications:** MPC, privacy-preserving systems, ...
  - Also to other models of distributed proof? [KOS18], [NPY18], ...

Dan Boneh, Elette Boyle, <u>Henry Corrigan-Gibbs</u>, Niv Gilboa, Yuval Ishai
https://eprint.iacr.org/2019/188