How to Prove False Statements: Practical Attacks on Fiat-Shamir

Ron Rothblum

Succinct

TECHNION

Joint work with Dmitry Khovratovich and Lev Soukhanov

The Fiat-Shamir Transform [FS86]

How To Prove Yourself:
Practical Solutions to Identification
and Signature Problems

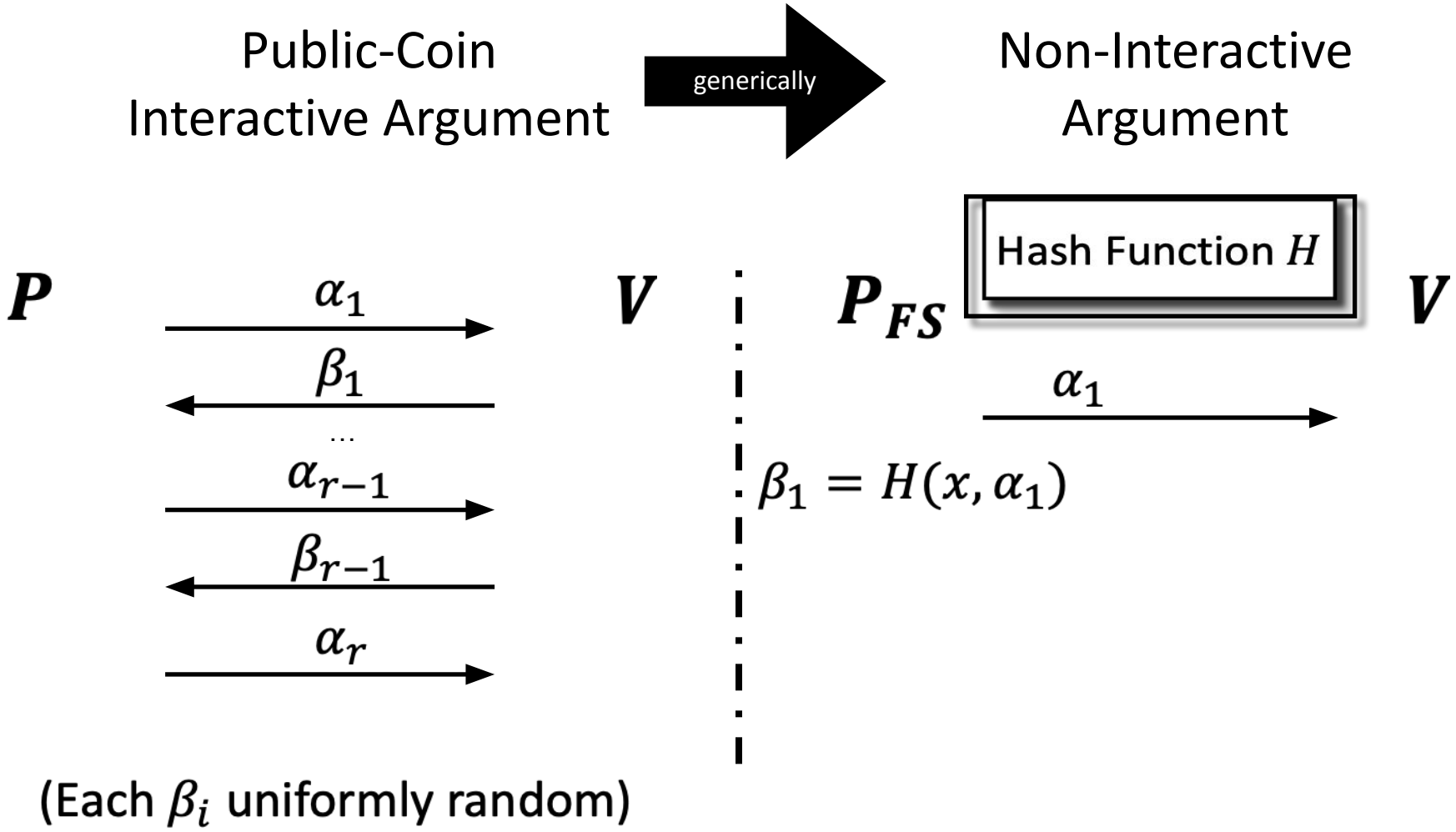
Amos Fiat and Adi Shamir
Department of Applied Mathematics
The Weizmann Institute of Science
Rehovot 76100, Israel

In a nutshell: Awesome technique for turning (public-coin) interactive protocols to be non-interactive.

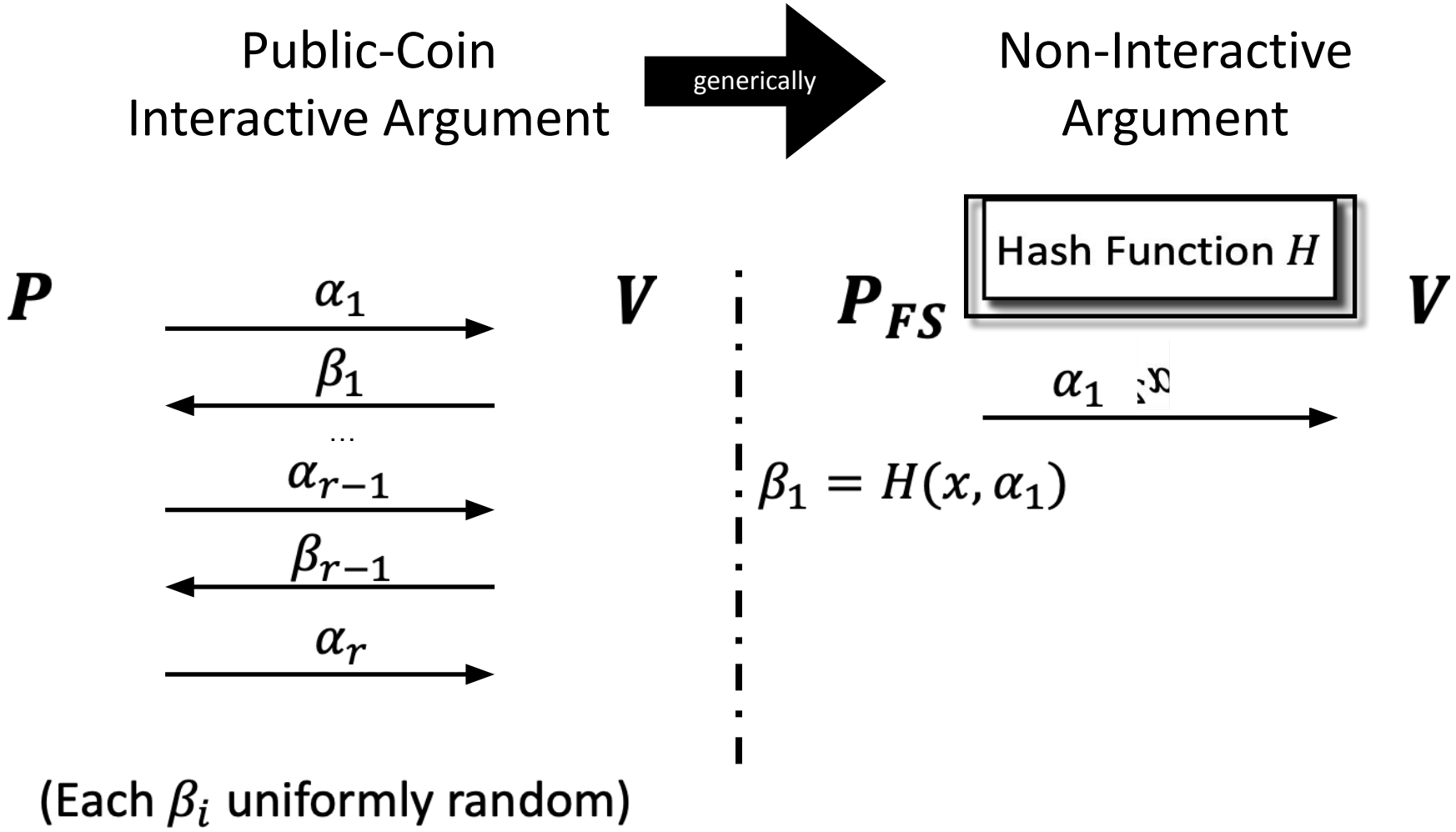
Critical component in most SNARK constructions.

* Original goal was transforming ID schemes into signature schemes.

The Fiat-Shamir Transform



The Fiat-Shamir Transform

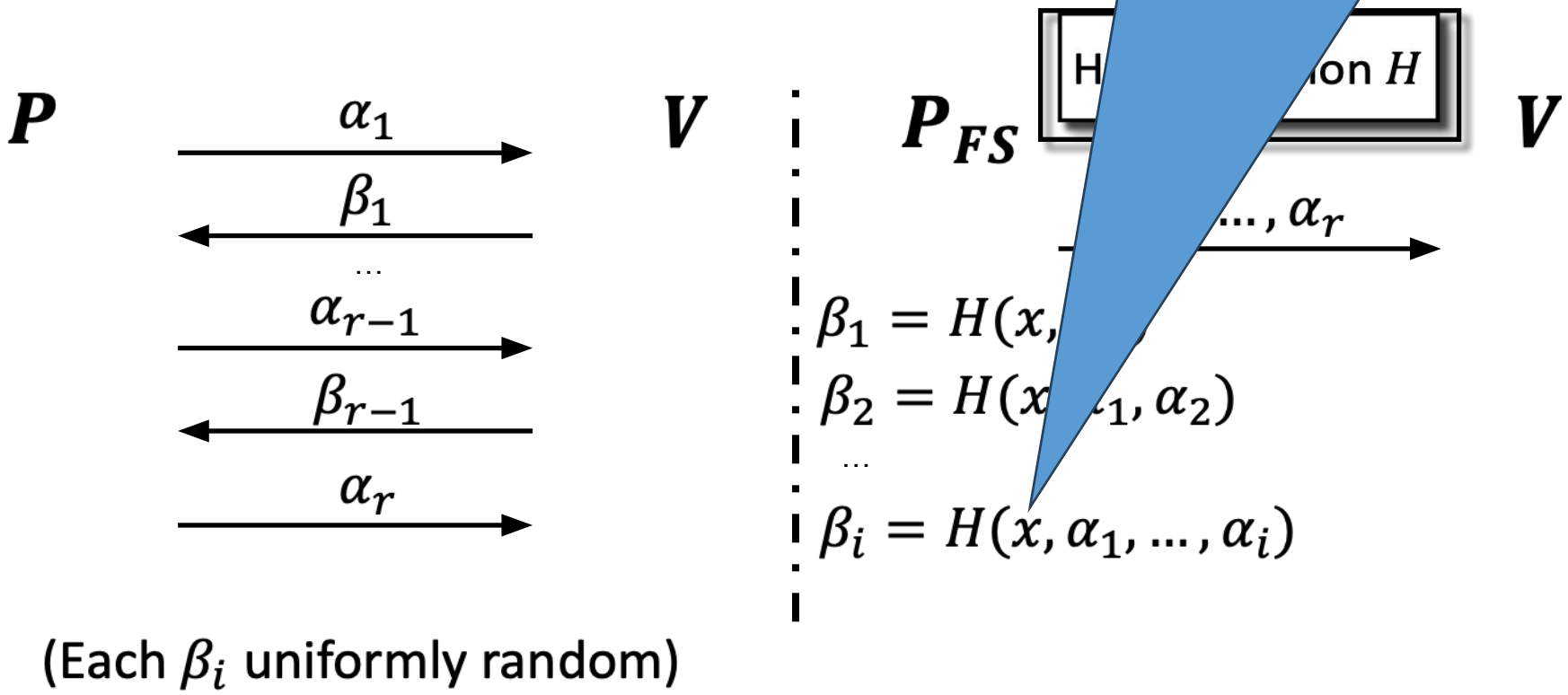


The Fiat-Shamir

Public-Coin
Interactive Argument

generically

Notorious for implementation bugs
[BPW12,HLPT20,DMWG23,Tha23]



(Each β_i uniformly random)

The Fiat-Shamir Transform

Extremely influential methodology.

The Fiat-Shamir Transform

Extremely influential methodology.

Powerful: We know that interaction buys a lot (sumcheck, FRI, GKR, bulletproofs,…).

The Fiat-Shamir Transform

Extremely influential methodology.

Powerful: We know that interaction buys a lot (sumcheck, FRI, GKR, bulletproofs,…).

FS makes interaction free.

The Fiat-Shamir Transform

Extremely influential methodology.

Powerful: We know that interaction buys a lot (sumcheck, FRI, GKR, bulletproofs,…).

FS makes interaction free.

Practical: Very low overhead.

The Fiat-Shamir Transform

Extremely influential methodology.

Powerful: We know that interaction buys a lot (sumcheck, FRI, GKR, bulletproofs,…).

FS makes interaction free.

Practical: Very low overhead.

Expressive: Efficient Signature, CS proofs, general purpose (zk-)SNARKs, STARKs…

Soundness?

Intuition:

Verifier's challenges seem hard to predict until previous prover messages are determined.

Analysis:

[PS96,Folklore]: FS is secure in the "random oracle model" (ROM).

Common Interpretation:

Any attack on Fiat-Shamir must be due to a weakness of the hash function.

“Random Oracle Methodology, Revisited”

[CGH04]: exhibited secure cryptographic schemes in ROM, that are broken when ROM is replaced by *any* concrete hash.

[B01,GK03]: specifically for FS.

[BBHMR19]: specifically for protocols similar to ones we commonly use (based on IOP + vector commitment/PCS).

Still, all of these results rely on some *contrived* component.

This Work

A natural, standard, and deployed protocol is insecure when FS is applied.

Specifically, can convince the verifier to accept a false statement.

A Little Bit of Background

1. (Multilinear) Polynomial Commitment Scheme (MLPCS)
2. The GKR Protocol

Polynomial Commitment Scheme (PCS)

Succinct "commit" to a large polynomial P.

Later, prove statements of the form "$P(x) = y$".

- Will not define formally what this means.
- **Today:** focus on multilinear polynomials $P: \mathbb{F}^m \to \mathbb{F}$, where $\mathbb{F}$ is sufficiently large finite field.

The GKR Protocol [GKR08]

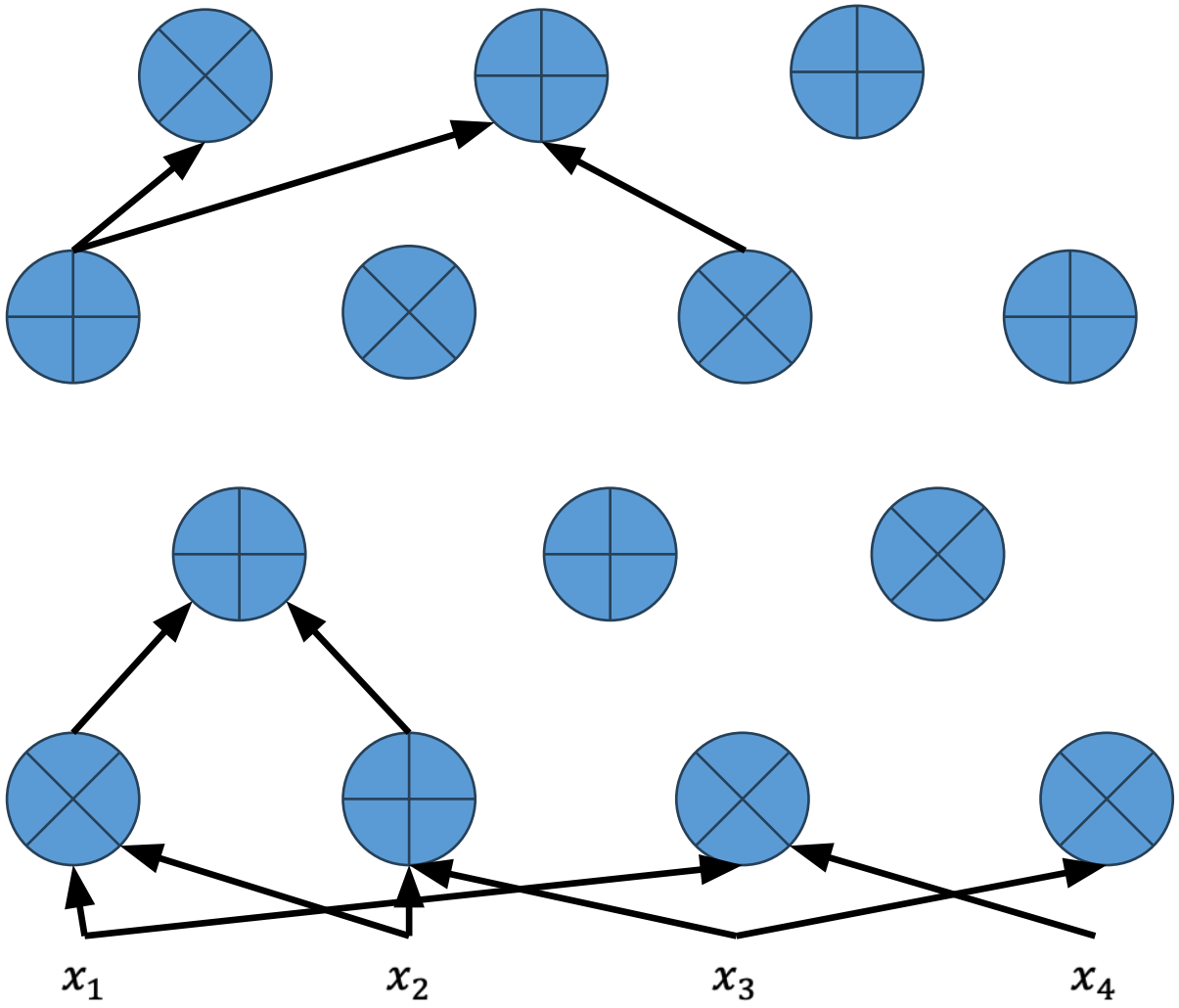
Statistically sound interactive proof for bounded depth arithmetic circuits.

Prove "$C(x) = y$" where C is a depth d arithmetic circuit over $\mathbb{F}$.

Deterministic computation – no witness.

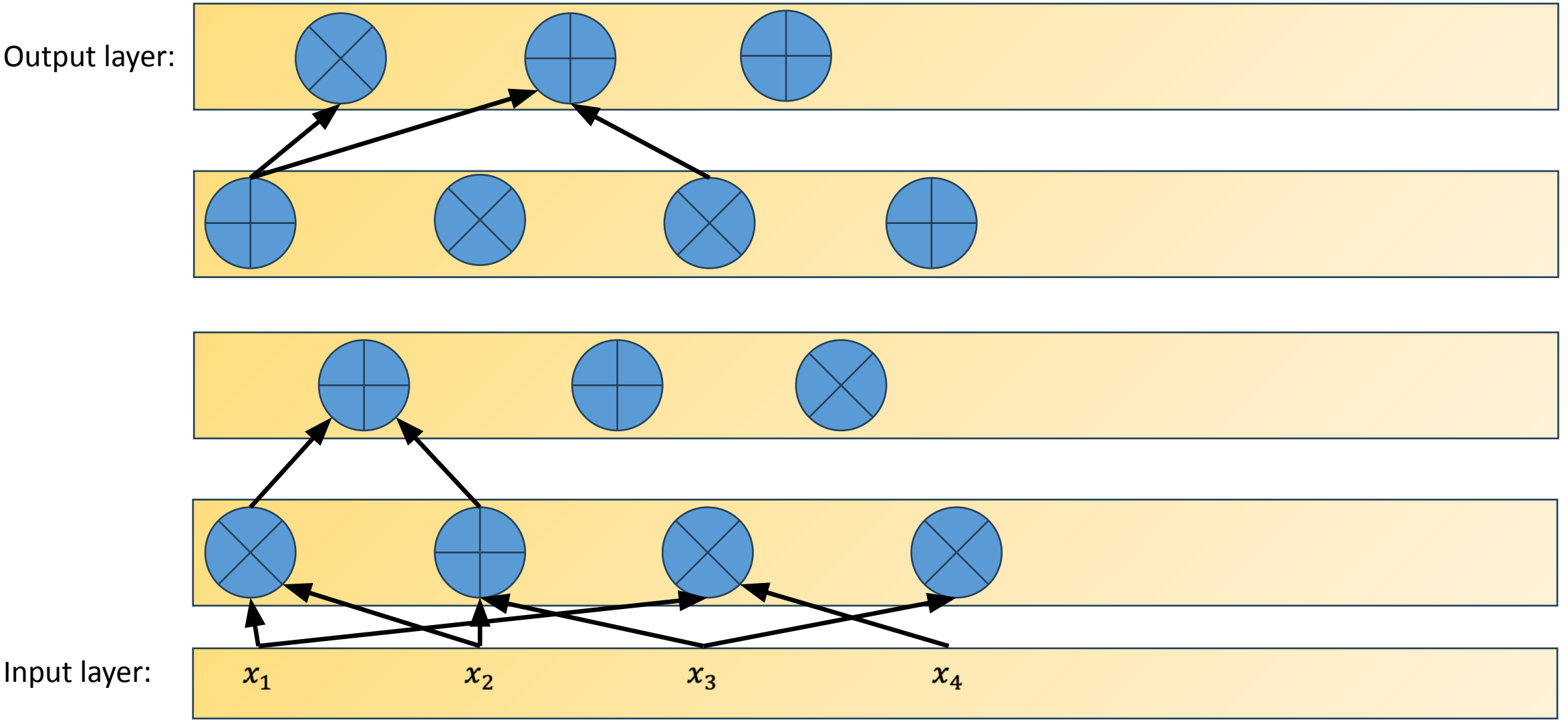
The GKR Protocol [GKR08]

Output layer:

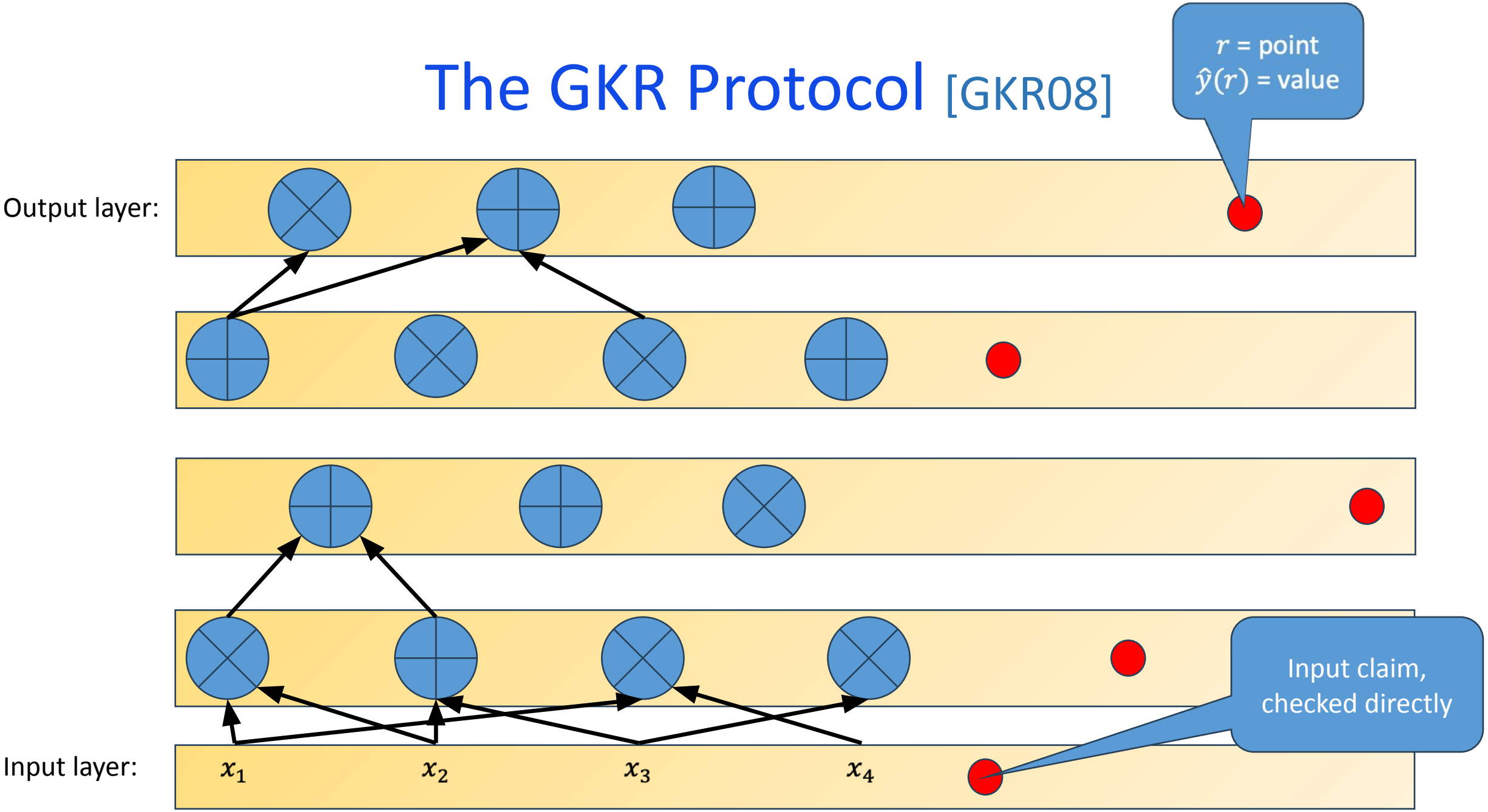


Input layer:

The GKR Protocol [GKR08]



The GKR Protocol [GKR08]



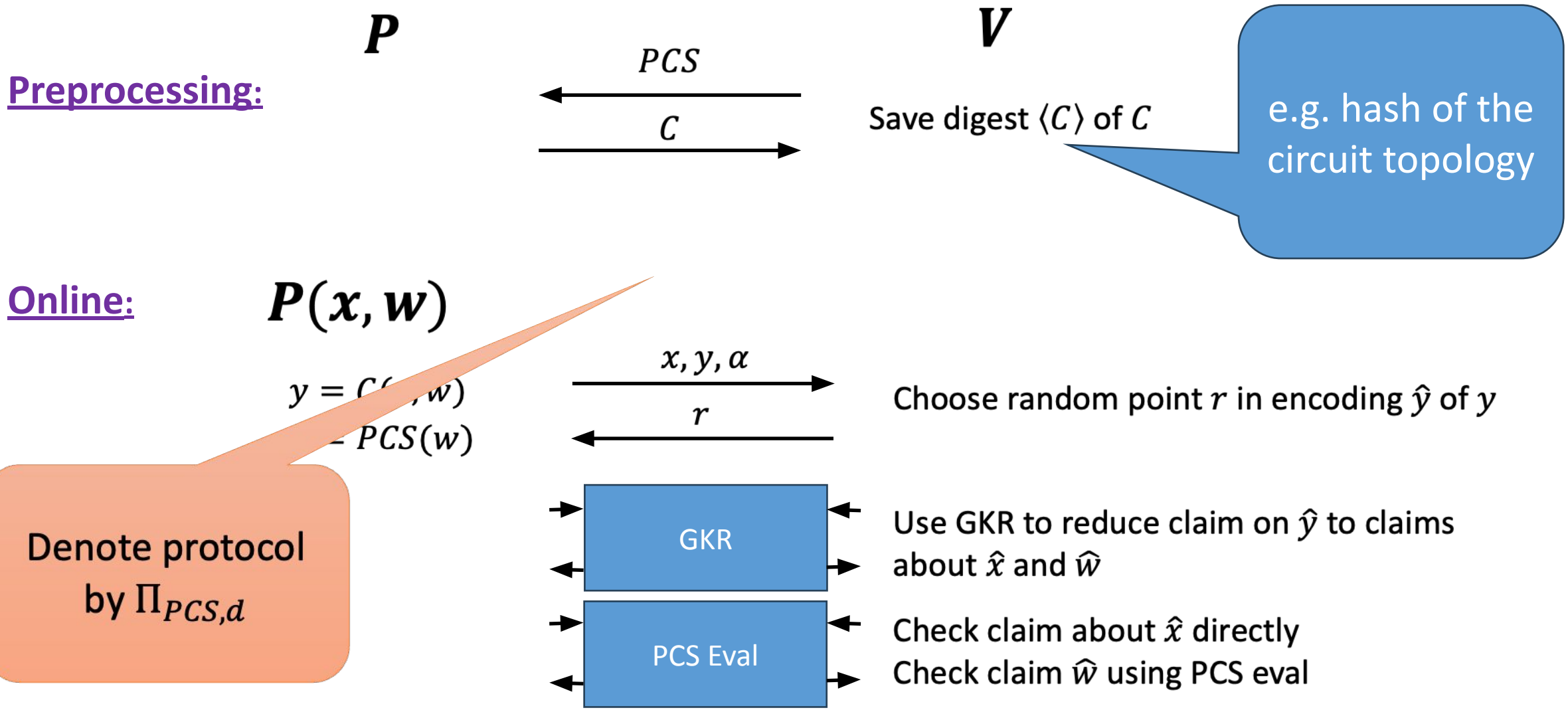
Succinct Argument from GKR

GKR by itself is only for deterministic computations.

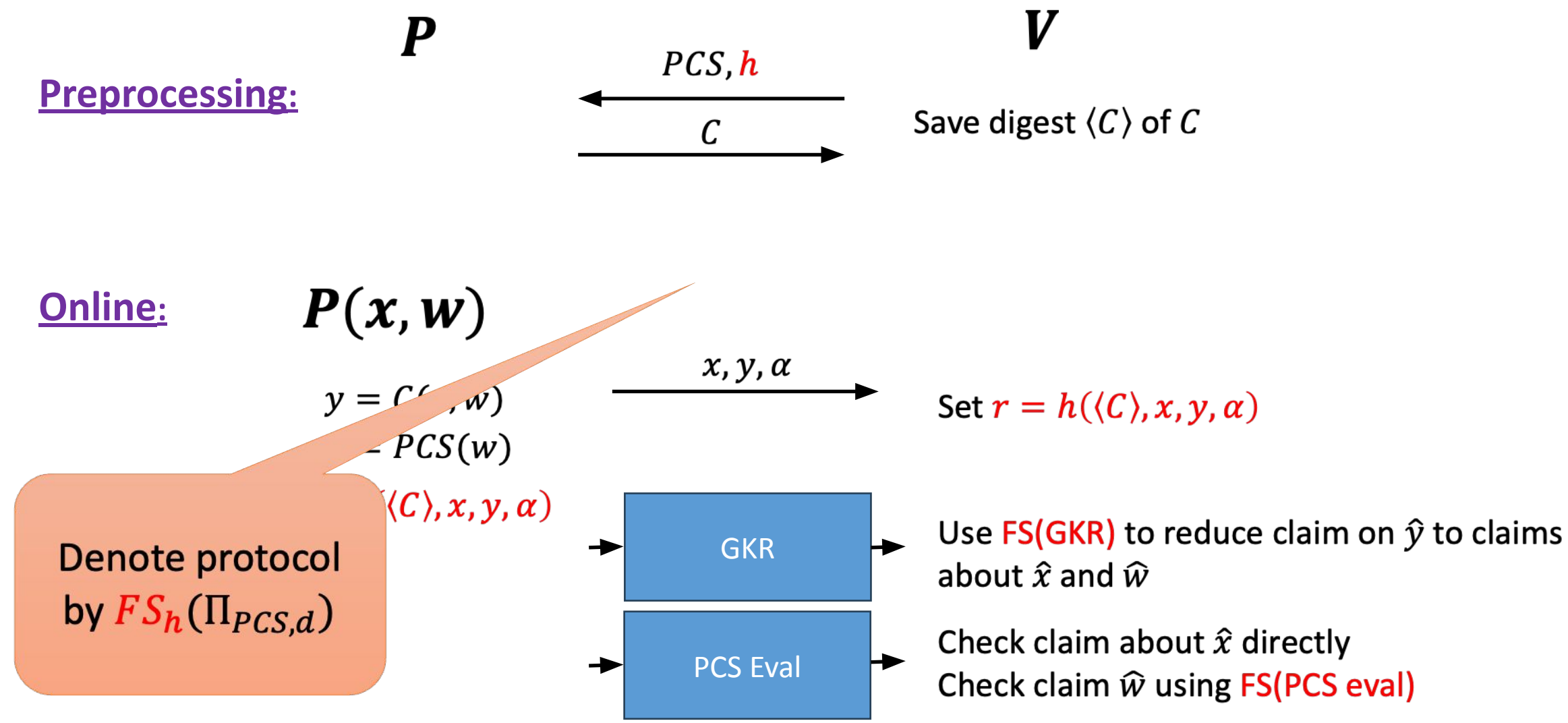
Want: prove statements such as $\exists w$ s.t. $C(x, w) = y$.

Solution: combine GKR with a (multilinear) PCS.

Succinct Argument from GKR



Applying Fiat-Shamir



Succinct Argument from GKR

Denote protocol $FS_h(\Pi_{PCS,d})$.

Used in vSQL [ZGKPP17], Hyrax [WTSTW18], Libra [XZZPS19, ZLWZSXZ21] and widely deployed in practice (e.g., Expander).

Introducing Expander: The Fastest GKR Proof System to Date

Polyhedra · Follow

Published in Polyhedra Network · 4 min read · May 1, 2024

fix fiat-shamir #184

Merged niconiconi merged 5 commits into `dev` from `zz/fix-fs` on Jan 22

Main Result 1: Proving False Statements

Thm 1: for every PCS and h for $d > depth(PCS) + depth(h) + 1$, the protocol $FS_h(\Pi_{PCS,d})$ is **not adaptively sound**.

Main Result 1: Proving False Statements

Thm 1: for every PCS and h for $d > depth(PCS) + depth(h) + 1$, the protocol $FS_h(\Pi_{PCS,d})$ is **not adaptively sound**.

Given PCS and h can generate a circuit C^*, output y^* and proof π s.t.:
1. $\forall w, C^*(w) \neq y^*$
2. verifier accepts given (C^*, y^*, π)

Main Result 1: Proving False Statements

Thm 1: for every PCS and h for $d > depth(PCS) + depth(h) + 1$, the protocol $FS_h(\Pi_{PCS,d})$ is **not adaptively sound**.

Given PCS and h can generate a circuit C^*, output y^* and proof π s.t.:
1. $\forall w, C^*(w) \neq y^*$
2. verifier accepts given (C^*, y^*, π)

Question: does C^* look contrived?

Main Result 1: Proving False Statements

Thm 1: for every PCS and h for $d > depth(PCS) + depth(h) + 1$, the protocol $FS_h(\Pi_{PCS,d})$ is **not adaptively sound**.

Given PCS and h can generate a circuit C^*, output y^* and proof π s.t.:
1. $\forall w, C^*(w) \neq y^*$
2. verifier accepts given (C^*, y^*, π)

Question: does C^* look contrived?

Answer: not really, to be discussed

Main Result 2: Backdoor Attack

Thm 2: (informal) can change **any** circuit C into a functionally equivalent circuit C^* but for which we can prove a false statement.

Main Result 2: Backdoor Attack

Thm 2: (informal) can change **any** circuit C into a functionally equivalent circuit C^* but for which we can prove a false statement.

Interpretation: attack does not depend on the functionality of the circuit, only on the specific implementation.

Main Result 3: Non-adaptive Attacks

Thm 3: (informal) can construct circuits that do not depend on the PCS and hash function h (except for their sizes), for which we can mount an attack.

Main Result 3: Non-adaptive Attacks

Thm 3: (informal) can construct circuits that do not depend on the PCS and hash function h (except for their sizes), for which we can mount an attack.

Downside: attacks are either less practical, or assume some (natural) structure of the PCS.

Main Result 3: Non-adaptive Attacks

Thm 3: (informal) can construct circuits that do not depend on the PCS and hash function h (except for their sizes), for which we can mount an attack.

Downside: attacks are either less practical, or assume some (natural) structure of the PCS.

Interpretation: insufficient that circuit does not contain "bad parts". Roughly speaking, potential for attack if circuit contains a "universal" component.

Main Result 1:
Proving a False Statement

Proving a False Statement

Idea: construct circuit C^* that **never** outputs the all-zero string y^* and yet we can make verifier accept the false statement:

$$\text{“}\exists w,\ C^*(w) = y^*\text{”}$$

Proving a False Statement

Idea: construct circuit C^* that **never** outputs the all-zero string y^* and yet we can make verifier accept the false statement:

$$\text{“}\exists w,\ C^*(w) = y^*\text{”}$$

Adaptive Attack: Circuit C^* can depend on choice of PCS and hash h.

High Level Idea

Construct a circuit that tries to predict the Fiat-Shamir hash value.

High Level Idea

Construct a circuit that tries to predict the Fiat-Shamir hash value.

$\boldsymbol{C^*(w)}$

1. …
2. Compute $\gamma = h(\langle C^* \rangle, y^*, \alpha)$
3. …

High Level Idea

Construct a circuit that tries to predict the Fiat-Shamir hash value.

$C^*(w)$

1. …
2. Compute $\gamma = h(\langle C^* \rangle, y^*, \alpha)$
3. …

Problem: circularity ☹

High Level Idea

Construct a circuit that tries to predict the Fiat-Shamir hash value.

$C^*(w)$

1. …
2. Compute $\gamma = h(\langle C^* \rangle, y^*, \alpha)$
3. …

Problem: circularity ☹

Idea: provide the circuit digest $\langle C^* \rangle$ as a witness!

The Attacking Circuit C^*

The Attacking Circuit C^*

$\underline{C^*(w)}$

The Attacking Circuit C^*

$C^*(w)$

1. Interpret w as a circuit digest ψ.

The Attacking Circuit C^*

$C^*(w)$

1. Interpret w as a circuit digest ψ.
2. Compute $\alpha = PCS(w)$

The Attacking Circuit C^*

$C^*(w)$

1. Interpret w as a circuit digest ψ.
2. Compute $\alpha = PCS(w)$
3. Compute $\gamma = h(\psi, y^*, \alpha)$

The Attacking Circuit C^*

$C^*(w)$

1. Interpret w as a circuit digest ψ.
2. Compute $\alpha = PCS(w)$
3. Compute $\gamma = h(\psi, y^*, \alpha)$
4. Output $(\gamma, \gamma - 1)$

The Attacking Circuit C^*

$C^*(w)$

1. Interpret w as a circuit digest ψ.
2. Compute $\alpha = PCS(w)$
3. Compute $\gamma = h(\psi, y^*, \alpha)$
4. Output $(\gamma, \gamma - 1)$

Claim 1: circuit never outputs $y^* = (0,0)$

The Attacking Circuit C^*

$C^*(w)$

1. Interpret w as a circuit digest ψ.
2. Compute $\alpha = PCS(w)$
3. Compute $\gamma = h(\psi, y^*, \alpha)$
4. Output $(\gamma, \gamma - 1)$

Claim 1: circuit never outputs $y^* = (0,0)$

Claim 2: can produce proof π s.t. verifier accepts blatantly false statement:

"$\exists w$ s.t. $C^*(w) = y^*$"

Proof of Claim 2

Claim 2: can produce proof π s.t. verifier accepts blatantly false statement "exists w s.t. $C^*(w) = y^*$.

Proof of Claim 2

Claim 2: can produce proof π s.t. verifier accepts blatantly false statement "exists w s.t. $C^*(w) = y^*$.

Proof:

Proof of Claim 2

Claim 2: can produce proof π s.t. verifier accepts blatantly false statement "exists w s.t. $C^*(w) = y^*$.

Proof:

Cheating Prover:

Proof of Claim 2

Claim 2: can produce proof π s.t. verifier accepts blatantly false statement "exists w s.t. $C^*(w) = y^*$.

Proof:

Cheating Prover:

1. Sends C^* as the target circuit.

Proof of Claim 2

Claim 2: can produce proof π s.t. verifier accepts blatantly false statement "exists w s.t. $C^*(w) = y^*$.

Proof:

Cheating Prover:

1. Sends C^* as the target circuit.
2. Sends $y^* = (0,0)$ as claimed output, and $\alpha = PCS(\langle C^* \rangle)$

Proof of Claim 2

The actual output of the circuit is $(\gamma, \gamma - 1)$.

Proof of Claim 2

The actual output of the circuit is $(\gamma, \gamma - 1)$.

Q: What is the multilinear extension of $(\gamma, \gamma - 1)$ at the point r?

Proof of Claim 2

The actual output of the circuit is $(\gamma, \gamma - 1)$.

Q: What is the multilinear extension of $(\gamma, \gamma - 1)$ at the point r?

A: $\gamma - r$.

Proof of Claim 2

The actual output of the circuit is $(\gamma, \gamma - 1)$.

Q: What is the multilinear extension of $(\gamma, \gamma - 1)$ at the point r?
A: $\gamma - r$.

By construction of C^*: $\gamma = h(\langle C^* \rangle, y^*, \alpha)$

Proof of Claim 2

The actual output of the circuit is $(\gamma, \gamma - 1)$.

Q: What is the multilinear extension of $(\gamma, \gamma - 1)$ at the point r?

A: $\gamma - r$.

By construction of C^*: $\gamma = h(\langle C^* \rangle, y^*, \alpha)$

By FS: $r = h(\langle C^* \rangle, y^*, \alpha)$

Proof of Claim 2

The actual output of the circuit is $(\gamma, \gamma - 1)$.

Q: What is the multilinear extension of $(\gamma, \gamma - 1)$ at the point r?
A: $\gamma - r$.

By construction of C^*: $\gamma = h(\langle C^* \rangle, y^*, \alpha)$
By FS: $r = h(\langle C^* \rangle, y^*, \alpha)$

Conclusion: multilinear extension of real output at point r is 0.

Proof of Claim 2

The **fake** output for the circuit is $(0,0)$.

Proof of Claim 2

The **fake** output for the circuit is $(0,0)$.

Q: What is the multilinear extension of $(0,0)$ at the point r?

Proof of Claim 2

The **fake** output for the circuit is $(0,0)$.

Q: What is the multilinear extension of $(0,0)$ at the point r?

A: 0

Proof of Claim 2

The **fake** output for the circuit is $(0,0)$.

Q: What is the multilinear extension of $(0,0)$ at the point r?

A: 0

Conclusion: multilinear extensions of fake and real outputs agree on the point r!

Proof of Claim 2

Started off with the **false** claim that the circuit outputs $(0,0)$.

Proof of Claim 2

Started off with the **false** claim that the circuit outputs $(0,0)$.

Reduced to the **correct** claim that the encoding of the input at the point r is 0!

Proof of Claim 2

Started off with the **false** claim that the circuit outputs $(0,0)$.

Reduced to the **correct** claim that the encoding of the input at the point r is 0!

At this point can just run the honest GKR prover strategy.

Attacking Fiat-Shamir

Preprocessing:

P^* $\quad$ V

PCS, h (→)

C^* (←)

Save digest $\langle C^* \rangle$

Online:

$y^* = (0,0)$

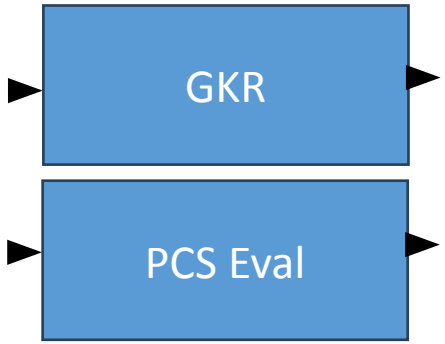
$\alpha = PCS(w)$

y^*, α (→)

Set $r = h(\langle C^* \rangle, y^*, \alpha)$

Now the claim has become true!

Run honest strategies to prove that the MLE of real output at point r is 0



Is C^* Contrived?

Is C^* Contrived?

First glance: seems weird, circuit evaluates the FS hash and PCS.

Is C^* Contrived?

First glance: seems weird, circuit evaluates the FS hash and PCS.

Second glance: actually, these are circuits we really care about!

- Proving that a x is a hash of a string satisfying some property.
- Recursion circuit computes FS hash function.

Reclaiming Soundness?

Theory literature: rich line of work [CCR16, KRR17, CCRR18, CCHLRRW19, BKM20, HLR21, JKKZ21, CJJ21, CGJJZ23, CT24, CRT25] establishing security based on standard assumptions such as LWE, DDH, etc.

Challenge:

- limited scope (partial non-determinism)
- make it practical!

Mitigations

A key source for our attack is that the circuit being proved is powerful enough to compute the hash.

Idea: increase the depth of the FS hash to be deeper than circuit.

deployed in practice in Expander.

[AY25]: increase the size of the hash via a PoW.

Security shown in a new idealized model.

Model is natural but requires further scrutiny.

Summary

Random oracle idealization can be problematic in practice!

Open questions:

- Pretty easy to avoid the attack, but evidence for security is limited.
- Strongly encouraged to try to break the suggested mitigations!
- Specific challenge: break FS when circuit is more shallow than the hash function.
- Break specific circuits: recursion? lookup arguments?
- Diagnolization attacks on GGM? [Dent2002]