SNARKs in practice

An entirely too short primer

Giacomo Fenzi

EPFL

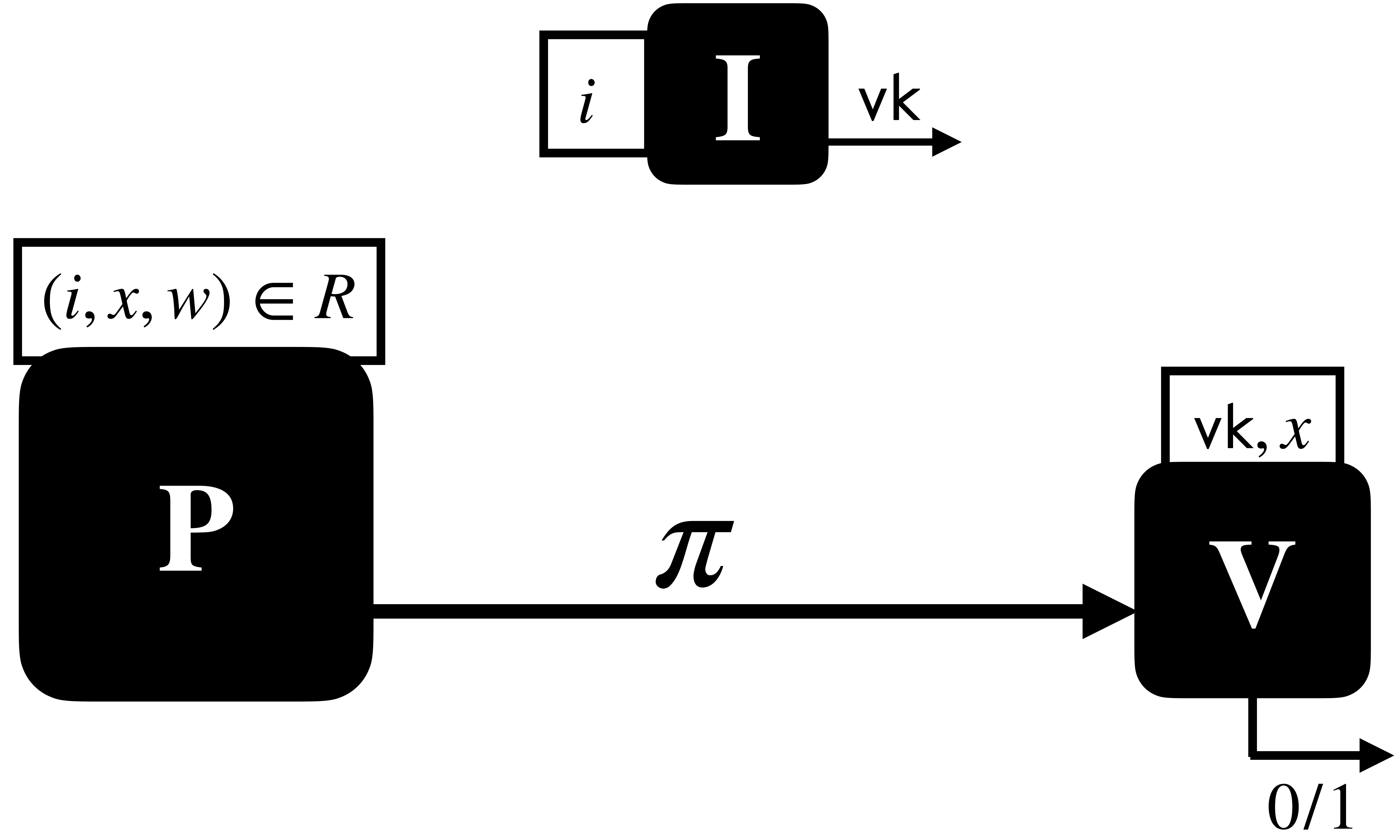
Preliminaries

SNARKs

Succinct Non-interactive Arguments of Knowledge

Want to show “knowledge” of w s.t. $(i, x, w) \in \mathscr{R}$

e.g. $\mathscr{R} := \{(C, x, w) : C(w) = x\}$



Today: we focus on

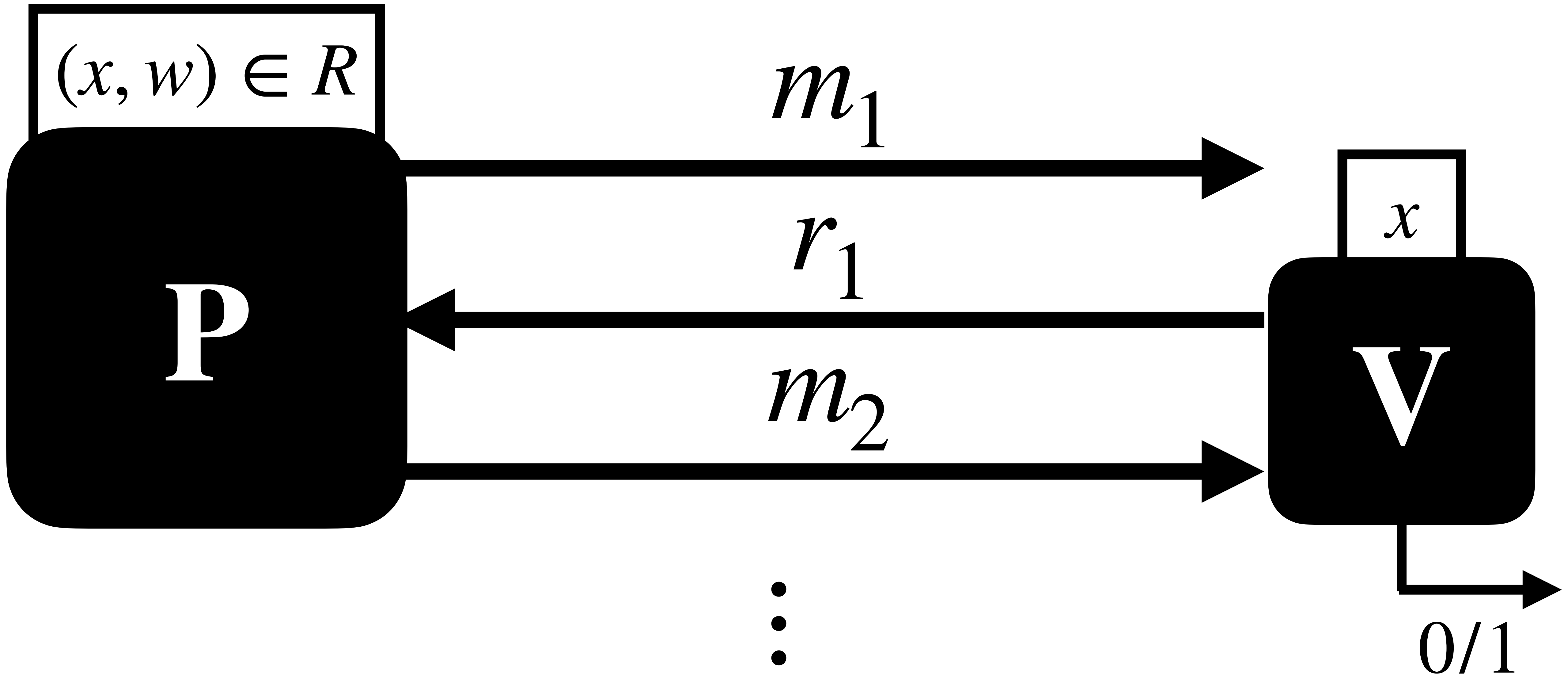
SNARKs from **functional commitments**

Almost all **practical** SNARKs follow this recipe

Security will be ultimately in **idealized models**

Interactive succinct arguments

A stepping stone to SNARGs



If number of rounds is **superconstant** regular soundness/knowledge soundness are not sufficient

IARG has **<u>state-restoration</u> soundness/knowledge soundness** $\Longrightarrow$ ARG is **sound/knowledge sound** in the ROM with $\kappa_{\mathsf{ARG}} \leq \kappa_{\mathsf{SR}}$

How to build succinct interactive arguments?

Some approaches…	**Proof string**	**Query class**	**Answer**
PCP+VC [Kilian92] **IOP+VC** [BCS16,CDGS23]	$\Pi \in \Sigma^{\ell}$	point queries $\mathbf{Q}_{\text{point}}$	$\beta = \Pi[\alpha]$ for $\alpha \in [\ell]$
LPCP+LC [LM19]	$\Pi \in \mathbb{F}^{\ell}$	linear queries $\mathbf{Q}_{\text{lin}}$	$\beta = \sum_{i \in [\ell]} \Pi[i] \cdot \alpha[i]$ for $\alpha \in \mathbb{F}^{\ell}$
PIOP+PC [CHM+20,BFS20]	$\Pi \in \mathbb{F}[X]^{\leq \text{D}}$	evaluation queries on polynomials $\mathbf{Q}_{\text{poly}}$	$\beta = \sum_{i \in [\ell]} \Pi[i] \cdot \alpha^{i-1}$ for $\alpha \in \mathbb{F}$
PIOP*+PC* [GWC19]	$\Pi \in (\mathbb{F}[X]^{\leq \text{D}})^{m+n}$ $= (f_1, \ldots, f_m, g_1, \ldots, g_n)$	evaluation queries on structured polys $\mathbf{Q}_{\text{poly}*}$	$\beta = \sum_{k \in [n]} h_k(f_1(\alpha), \cdots, f_m(\alpha)) \cdot g_k(\alpha)$

and more: Bulletproofs (and other sumcheck-based arguments), linear-only encodings [BCIOP13, GGPR13, Groth16], …

How to build succinct interactive arguments?

	Proof string	Query class	Answer
FIOP+FC	$\Pi \in \Sigma^\ell$	$\mathbf{Q} = \{\alpha : \Sigma^\ell \to \mathbb{D}\}$	$\beta = \alpha(\Pi) \in \mathbb{D}$ for $\alpha \in \mathbf{Q}$

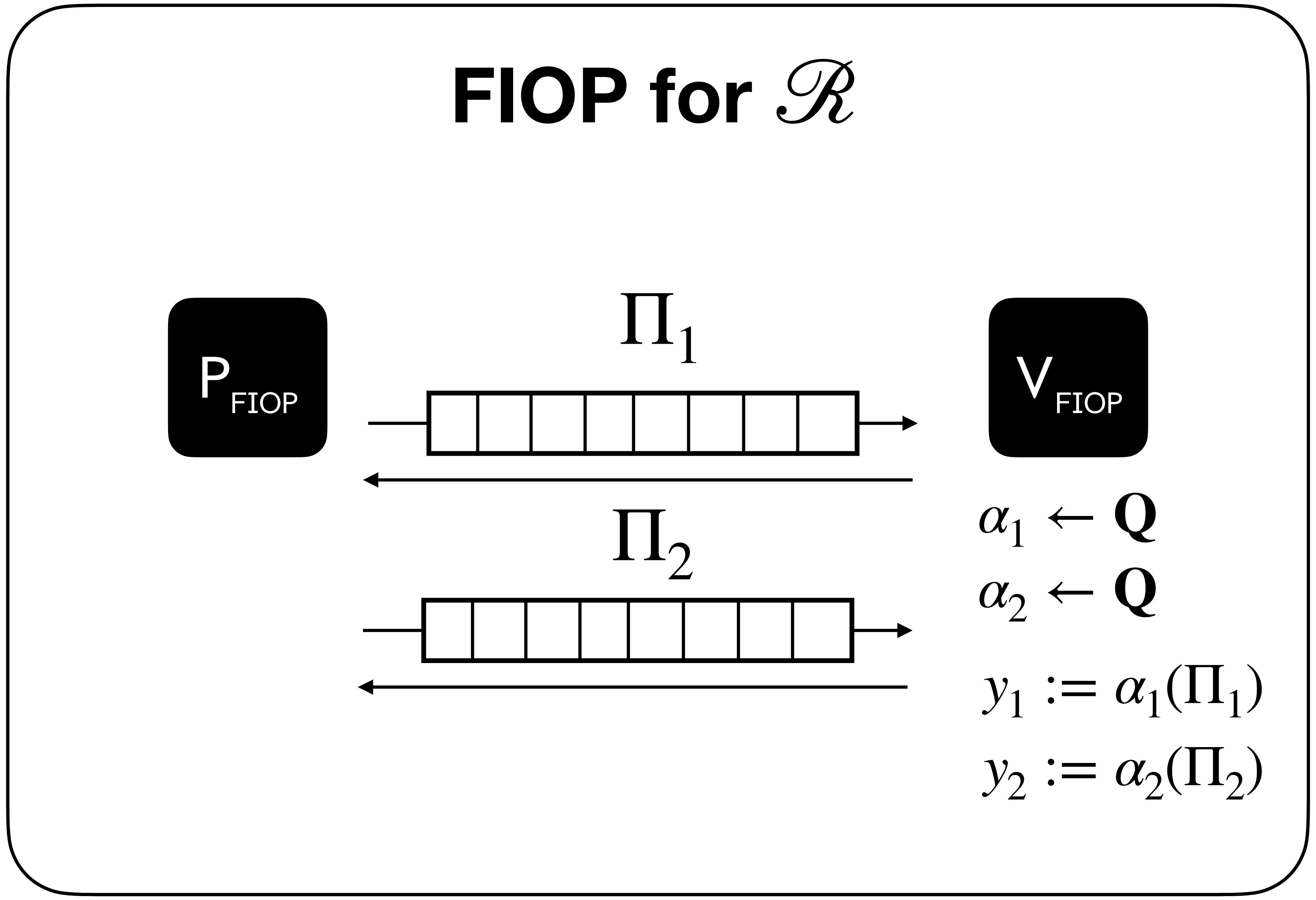
Functional IOP is purely an information theoretical construction.

+ ROM

Functional Commitments are where cryptography goes

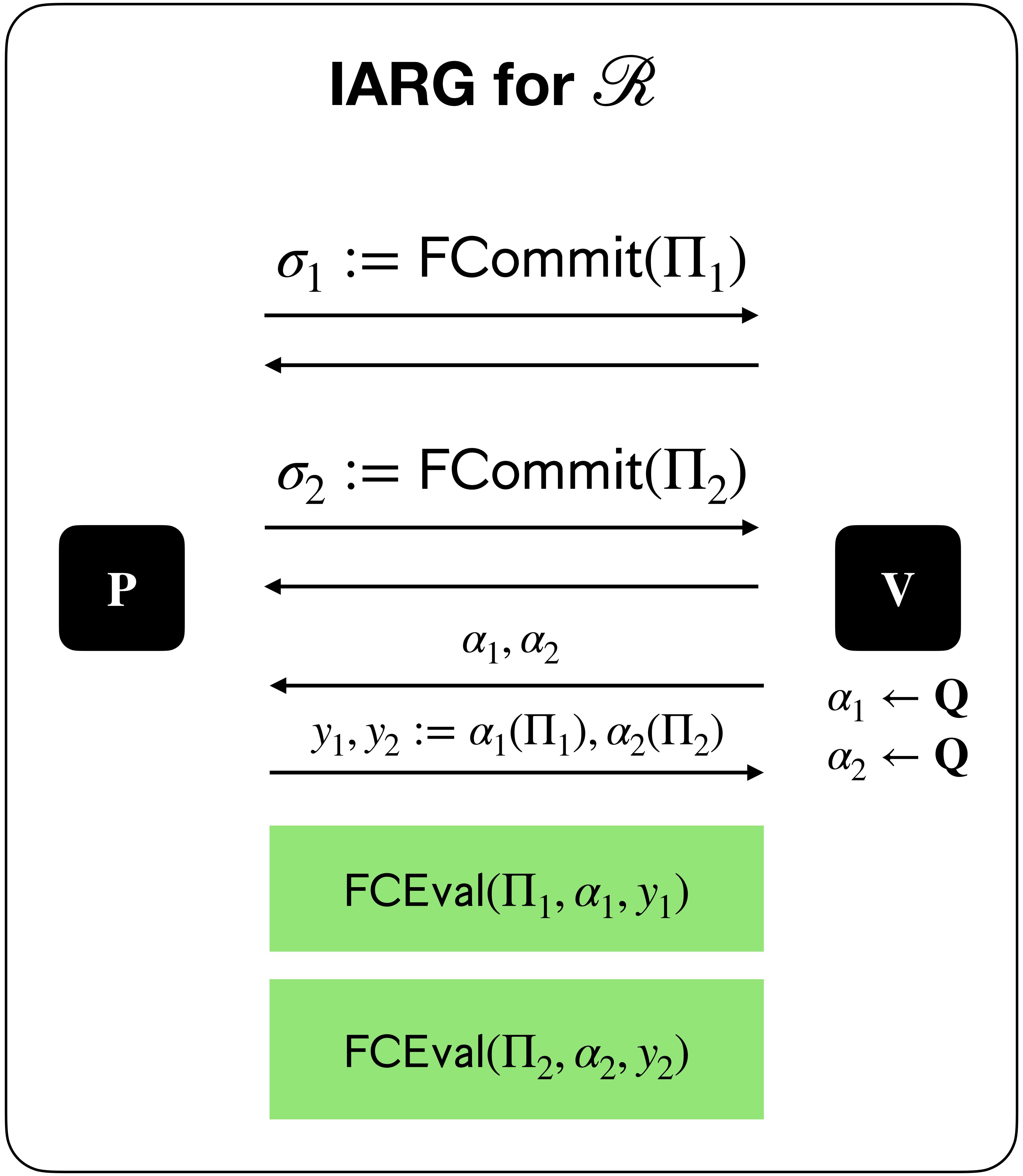
Non-interactive arguments in: ROM + Crypto[FC]

FIOP + FC



Prover sends **proof strings** Σ^ℓ

Verifier can ask **functional queries** $\alpha \in \mathbf{Q}$
where $\alpha \colon \Sigma^\ell \to \mathbb{D}$



Concrete security

Let $\mathcal{S}$ be a scheme which takes a security parameter λ
Typically, theoretical cryptography cares about **asymptotic security**

$$\forall \mathcal{A} \in \mathsf{poly}(\lambda),\ \Pr\left[\ \mathcal{A} \text{ breaks } \mathcal{S}(\lambda)\ \right] \leq \mathrm{negl}(\lambda)$$

Holds for every $\lambda > \lambda_0$ which will depend on the adversary resources.
If λ_0 is too large **no guarantees** in practice

For practical deployment, we want **concrete security**

$$\forall \mathcal{A} \in \mathsf{Adv}(R),\ \Pr\left[\mathcal{A} \text{ breaks } \mathcal{S}(\lambda)\right] \leq \epsilon(\lambda, R)$$

For every adversary with some resources

Modelling the resources R of the adversary allows to choose λ to get meaningful guarantees

Many additional considerations in this:
extractor running time, average vs worst case security

Theorem 22.1.1

Let PCP *be a PCP for a relation* $\mathcal{R}$ *with knowledge soundness error* κ_{PCP} *(see Definition 19.1.3). For every security parameter* $\lambda \in \mathbb{N}$ *and privacy parameter* $s \in \mathbb{N}$, $\mathsf{NARG} := \mathbf{Micali}[\mathsf{PCP}, \lambda, s]$ *in Construction 21.1.1 is a non-interactive argument for* $\mathcal{R}$ *with knowledge soundness error* κ_{ARG} *(see Definition 7.1.5) such that*

$$\kappa_{\mathrm{ARG}}(\lambda, t, n) \leqslant (t+1) \cdot \kappa_{\mathrm{PCP}}(n) + \kappa_{\mathrm{MT}}^{\mathrm{m}}(\lambda, t, \mathsf{l}, t+1, 1)\,.$$

Above $\kappa_{\mathrm{MT}}^{\mathrm{m}}$ *is the Merkle commitment multi-extraction error from Lemma 18.5.6, and* $\kappa_{\mathrm{MT}}^{\mathrm{m}}(\lambda, t, \mathsf{l}, t+1, 1) \leqslant 2 \cdot \frac{t^2}{2^\lambda}$ *if* $6 \cdot 1 \cdot (\log \mathsf{l} + 1) \leqslant t$.

Metrics that we care about

Lattice based? They seem to be promising to get best-of-both-worlds arguments, but more research is needed

Having fixed λ to achieve the required security guarantees, we generally care about the following three metrics:

We usually aim for $\lambda = 128$

Argument size

Relevant to minimize bandwidth

Elliptic curve based arguments:
argument size < 1 KiB

PIOP+PC paradigm

Prover time

Verifier time

Relevant to reduce the cost of deploying arguments

Hash based arguments (for instances of size 2^{26}):

- Argument string contains 160KiB
- proof generates in ~1s on a laptop
- can be verified in ~600µs

IOP+VC paradigm

Security properties
[CGKY25]

This error is **not** used in practice!
The recipe yields (in general) a secure interactive arguments.
In practice **tighter** bounds are obtained by analyzing the resulting SNARK

ARG = Funky[FIOP, FC] for a query class $\mathbf{Q}$

FIOP is state-restoration (knowledge) sound

+ FC is state-restoration function binding

$\Rightarrow$ ARG = Funky[FIOP, FC] is state-restoration (knowledge) sound

$$\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(\mathsf{k}, \ell, t) \leq \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(\mathsf{k}, \ell) + \epsilon^{\mathsf{SR}}_{\mathsf{FC}}(t \cdot \mathsf{k} \cdot \mathsf{N} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}(\ell, \mathsf{q}, \mathsf{N})$$

State-restoration (knowledge) soundness of FIOP

State-restoration function binding of FC

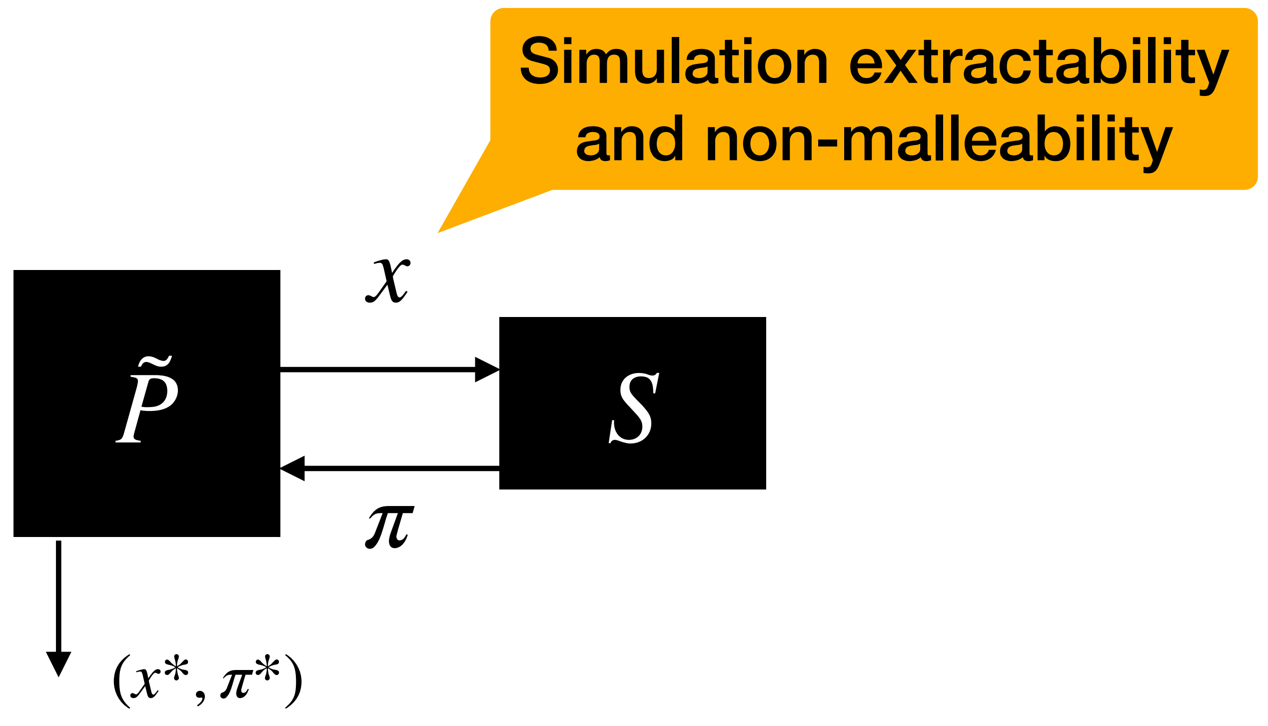
Depends only $\mathbf{Q}$

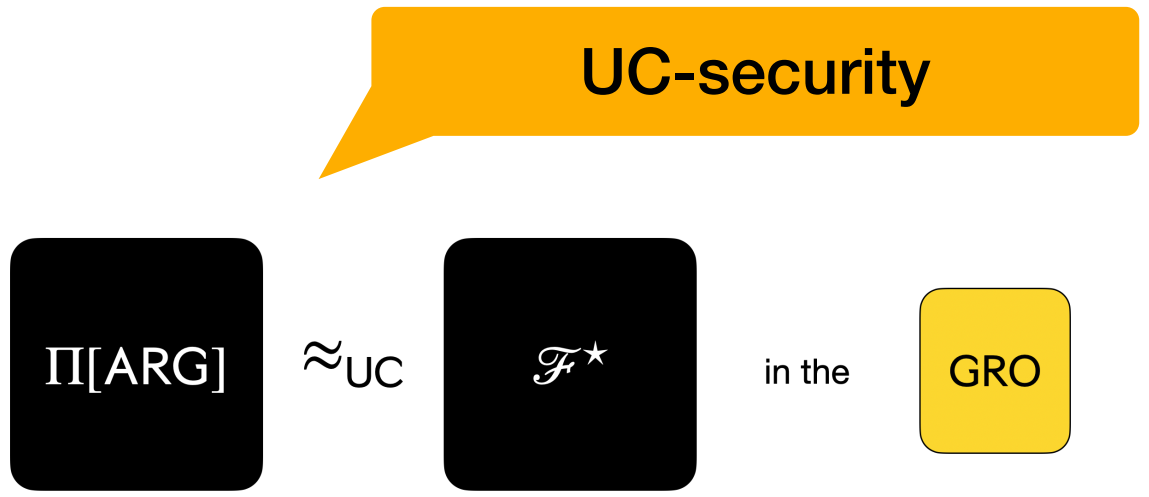
Things I will not talk about

Zero knowledge

Most practical deployed systems
do not guarantee zero-knowledge

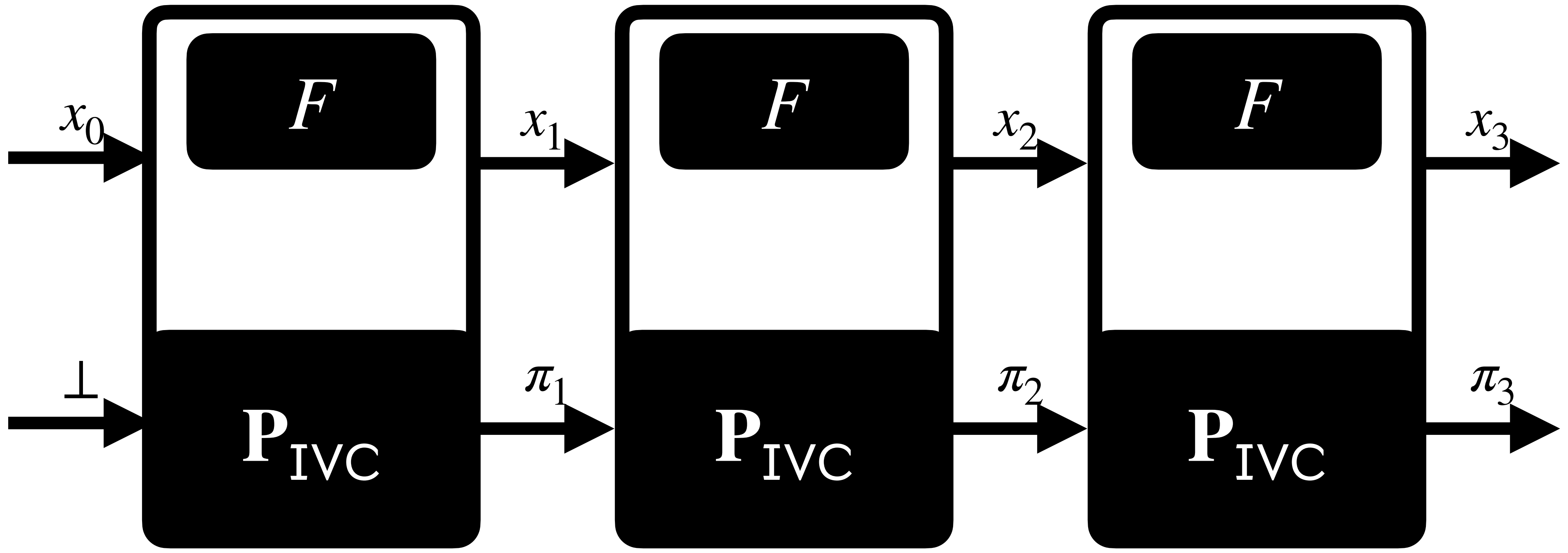
They still brand themselves as zk…





Stronger security properties

Recursion



Recursion is **extremely practical**.
Most concrete SNARKs use one or more layers of recursion.

Accumulation and **folding** schemes new exciting directions to exploit recursion.

Deployed systems only heuristically secure, problematic (especially in light of [KRS25])

Minimizing argument size

PIOP + PC

	Proof string	Query class	Answer
PIOP+PC	$\hat{f} \in \mathbb{F}^{\leq D}[X]$	point queries $\mathbf{Q}_{\text{point}}$	$\beta = \sum_{i \in [\ell]} \Pi[i] \cdot \alpha^{i-1}$ for $\alpha \in \mathbb{F}$

[KZG10]

Univariate PCS with $O(1)$ proof size

Assume bilinear pairings $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$

Other types of pairings are typically used

Setup(d)

Choose G s.t. $\mathbb{G} = \langle g \rangle$

Sample $\alpha \leftarrow \mathbb{F}$

Output:

pk $= (g, g^{\alpha}, \ldots, g^{\alpha^{d-1}})$

vk $= (g, g^{\alpha})$

Prove(pk, $z, y, \hat{f}(X)$)

Compute $\hat{w}(X) = \dfrac{\hat{f}(X) - y}{X - z}$

Output $\sigma_w = g^{\hat{w}(\alpha)}$

Commit(pk, $\hat{f}(X)$)

Output $\sigma_f = g^{\hat{f}(\alpha)}$

Can be computed with an MSM from pk

Verify(vk, σ_f, z, y, σ_w)

Check $e(\sigma_f, g) = e(\sigma_w, g^{\alpha - z}) \cdot e(g, g)^y$

Also, very good batching properties! Can batch many openings of distinct polynomials

Proof size and verification speed are **constant number of group elements and operations**

Concretely two curves are used:

On BN254 (≈ 100 bits of security)

$|\sigma_f| + |\sigma_w| = 64$B

On BLS12-381 (≈ 128 bits of security)

Of the curve, not of KZG!

$|\sigma_f| + |\sigma_w| = 96$B

Relies on a **private-coin** setup.

If α is public there is **no** security

Security proven in AGM
or under SDH-type assumptions

Univariate sumcheck [BCRSVW19]

Lemma

Let $H \subseteq \mathbb{F}$ be a multiplicative subgroup and let $\hat{f} \in \mathbb{F}^{<d}[X]$:

$$\sum_{a \in H} \hat{f}(a) = \gamma \text{ if and only if } \begin{matrix} \exists \hat{g} \in \mathbb{F}^{<d-|H|}[X] \\ \exists \hat{h} \in \mathbb{F}^{<|H|}[X] \end{matrix} \; : \; \hat{f}(X) = \hat{g}(X) \cdot V_H(X) + \hat{h}(X)$$

Lincheck PIOP:

Show that $\forall i \in H : (\mathbf{M} \cdot \hat{f})(i) = \hat{g}(i)$

$$\hat{r}(z) \cdot \hat{g}(z) - (\mathbf{M}^\top \cdot \hat{r})(z) \cdot \hat{f}(z) = \hat{p}(z) \cdot V_H(z) + z \cdot \hat{q}(z)$$

Putting it all together

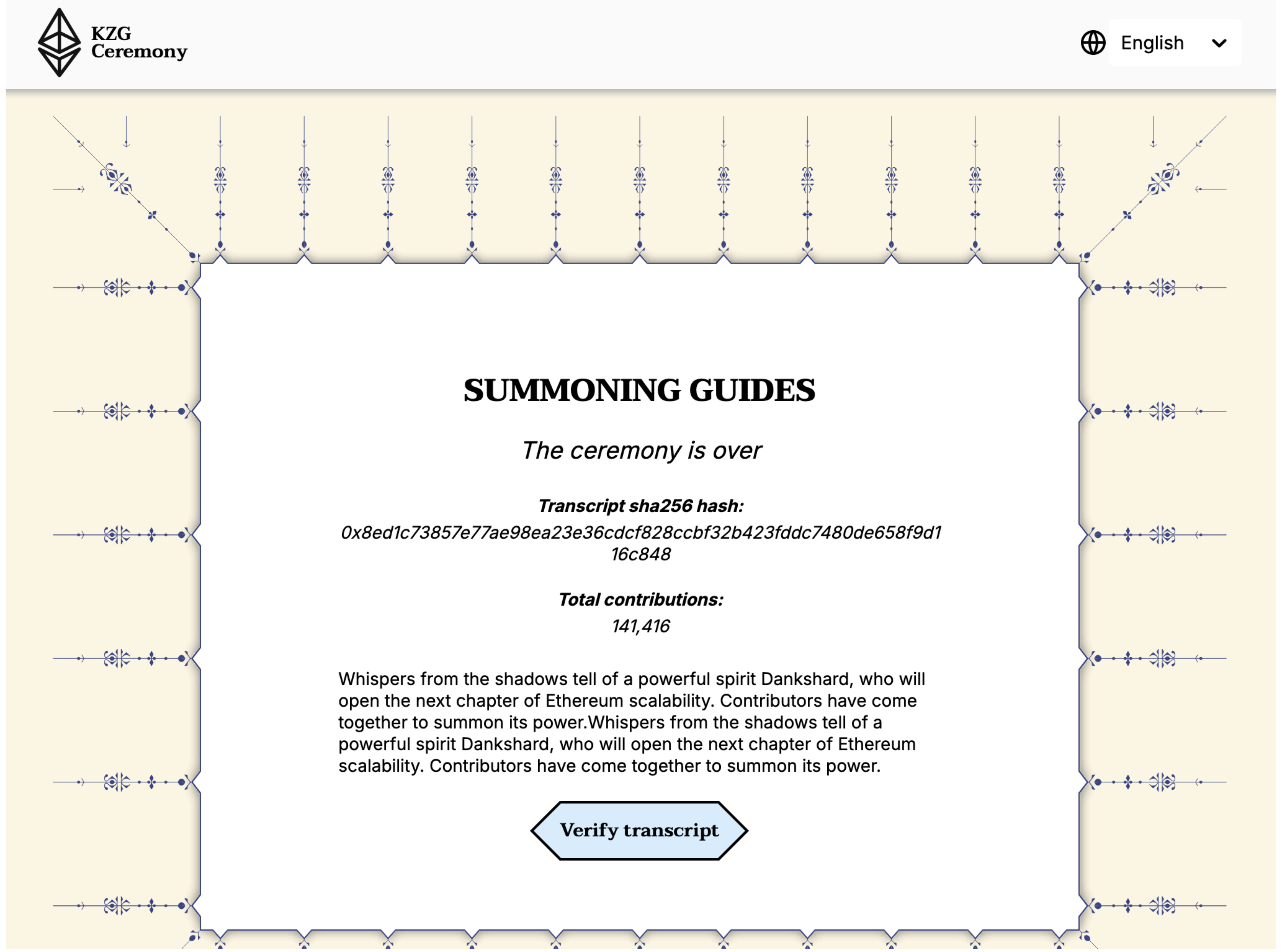
Used extensively in PIOP such as Aurora, Marlin, Fractal, Plonk, fflonk, Turboplonk, Ultraplonk, Honk, …

Argument string is a (small) **constant number** of group elements

Example:

Marlin PIOP with KZG on BL12-381 is 880B

Ceremonies



Follow a 1-out-N security model

Logistically quite hard to setup but can be done

KZG (and derived arguments) have **universal setup**:

Once done, the same pk, vk can be used for **many** different circuits being proven (as long as the setup is large enough)

Open Qs

Concretely efficient argument/polynomial commitment without private coin setup and argument size < 5 KiB

Concretely efficient argument/polynomial commitment with post-quantum security and argument size < 20 KiB

Even smaller proof sizes

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

Jens Groth**

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

Does not follow our recipe, but a similar one based on linear-PCPs

Groth16: Most widely **deployed** succinct argument

Secure in the AGM, argument string is $3 \cdot \mathbb{G}$.

Over BLS12-381 this is 192 B.

Polymath: Groth16 Is Not The Limit*
June 8, 2024

Helger Lipmaa

Over BLS12-381: 176 B

GARUDA and PARI: Faster and Smaller SNARKs via Equifficient Polynomial Commitments

Michel Dellepere, michel@provable.com, Provable
Pratyush Mishra, prat@upenn.edu, UPenn
Alireza Shirzad, alrshir@upenn.edu, UPenn

Over BLS12-381: 160 B

Recent improvement:

Achieve smaller proof sizes using a combination of AGM + ROM

Stronger private coin setup requirements: the setup must be done once <u>per circuit</u>

Open Qs

Can we get even **smaller SNARKs?**

Get argument of comparable size with **universal setup**

Minimizing prover & verifier cost

IOP + VC

	Proof string	Query class	Answer
IOP+VC	$\Pi \in \Sigma^{\ell}$	point queries $\mathbf{Q}_{\text{point}}$	$\beta = \Pi[\alpha]$ for $\alpha \in [\ell]$

Something quite magical happens here:
Merkle trees are **non-interactive** vector commitments in the **pure** ROM:

<u>Can construct succinct arguments in the pure ROM</u>

Simple cryptography, work shifted to the IOP

Open Qs

Are Merkle tree optimal as VC in the ROM?

More concretely efficient post-quantum vector commitment schemes

Hash-based SNARKs

In practice

Instantiating random oracle gives amazing SNARKs:

- Transparent setup (choice of hash)
- Highly efficient implementations (no public-key crypto)
- Plausibly post-quantum secure (secure in QROM)

Used to secure billions of dollars in real-world blockchains:

Committing to 2^{26} field elements on a laptop can be done in ≈ 800ms

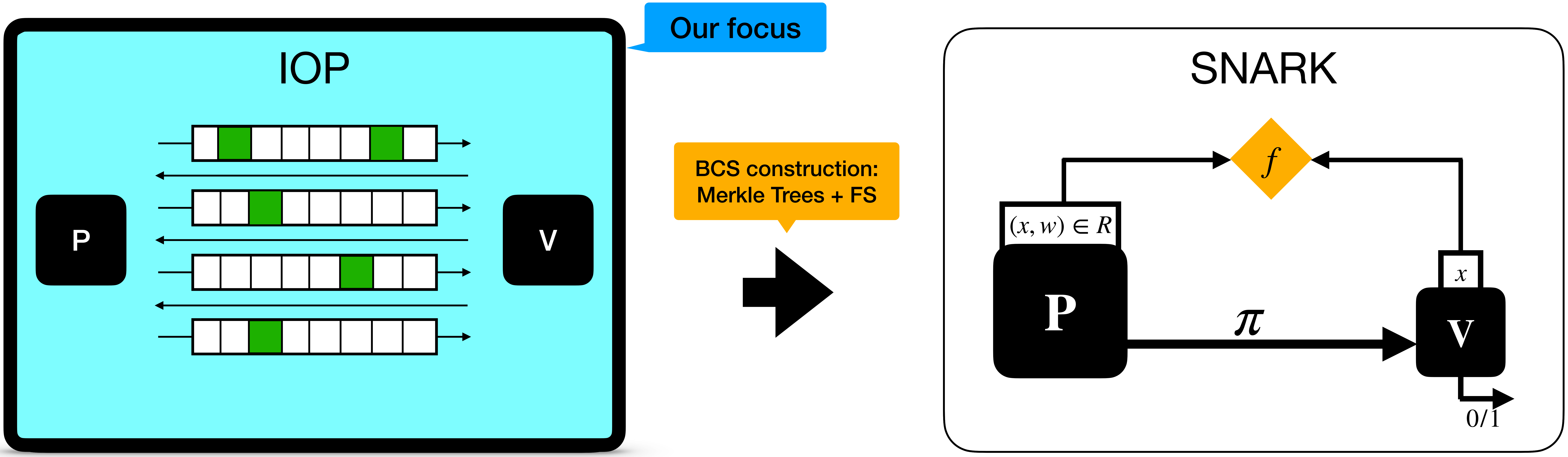
Committing to 2^{26} field elements using an MSM is on the order of ≈ 3m

Not an entirely fair comparison, but gives some intuition

And many more…

IOP-based SNARKs

[BCS16] Construction



Proof length l $\approx O(n)$ — Large, think 2^{24}

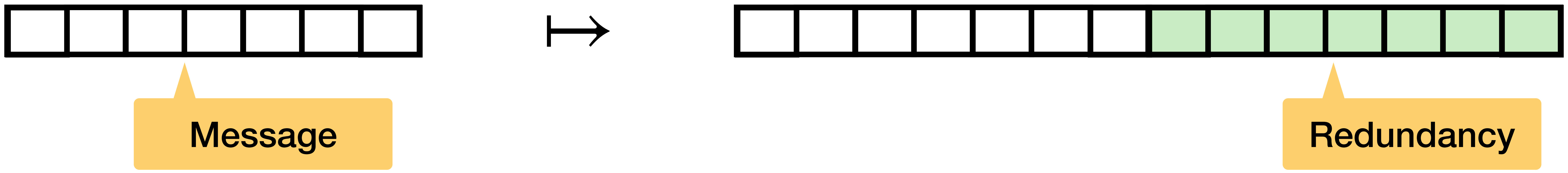
Queries q $\approx O(\log n)$ — Small, think ~400

Argument size $O(\lambda \cdot \mathrm{q} \cdot \log \mathrm{l})$ — Small, tens of KiB

Linear codes $\mathscr{C}: \mathbb{F}^k \to \mathbb{F}^n, \mathscr{C}(\mathbf{v}) = \mathbf{G}_{\mathscr{C}} \cdot \mathbf{v}$

Injective

Generator matrix



Parameters of interest

Translates to verifier efficiency

Hamming distance

Minimum distance: $\delta_{\mathscr{C}} := \min_{u,u' \in \mathscr{C}} \Delta(u, u')$

Rate: $\rho = \dfrac{k}{n}$ and **encoding time:** $\mathrm{enc}_{\mathscr{C}}$

Translates to prover ro queries. Usually $O(1)$

Translates to prover time

It's all about the code

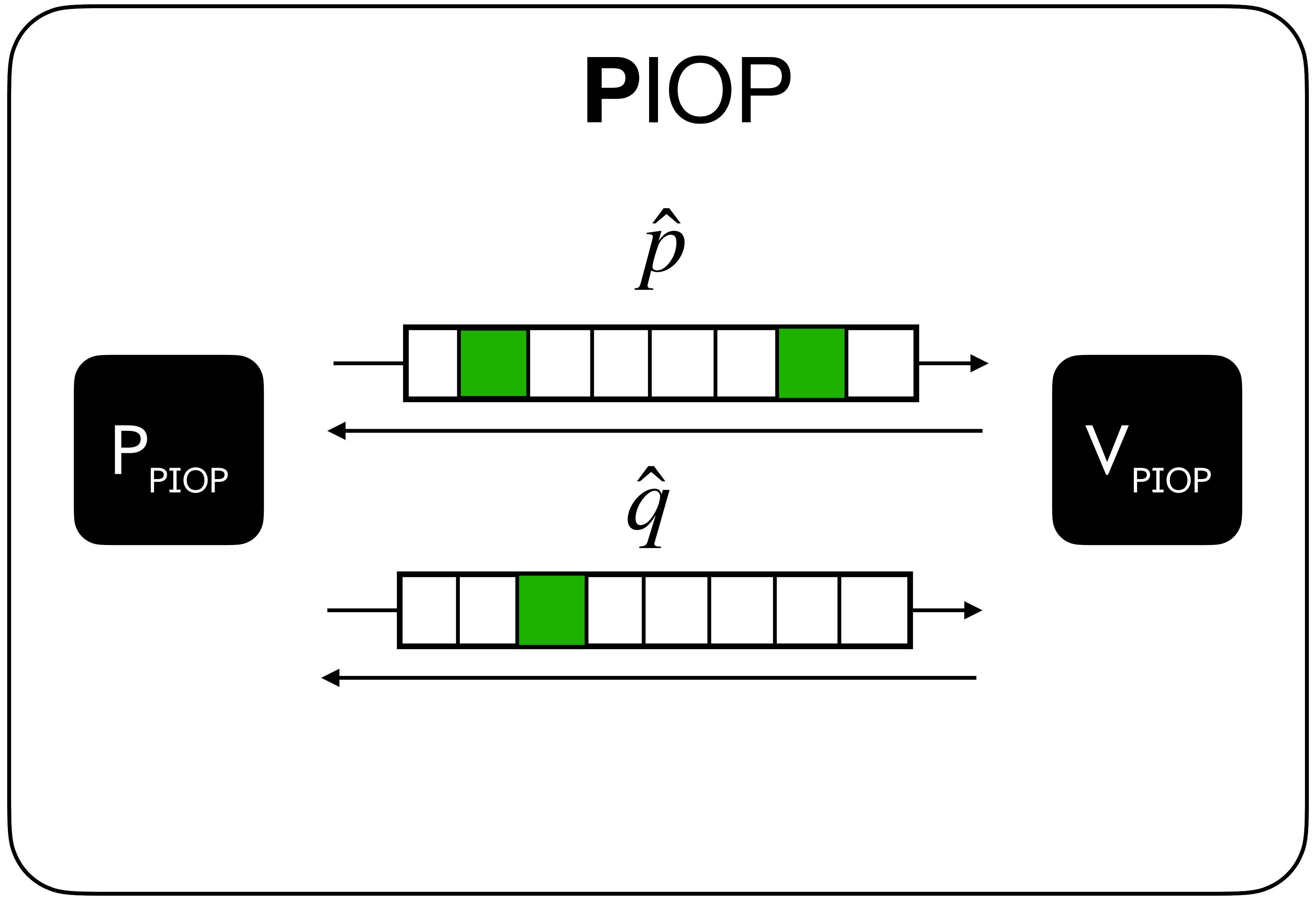
Error correcting codes will be used to redundantly encode witnesses of NP relation, and then prove claims about them.

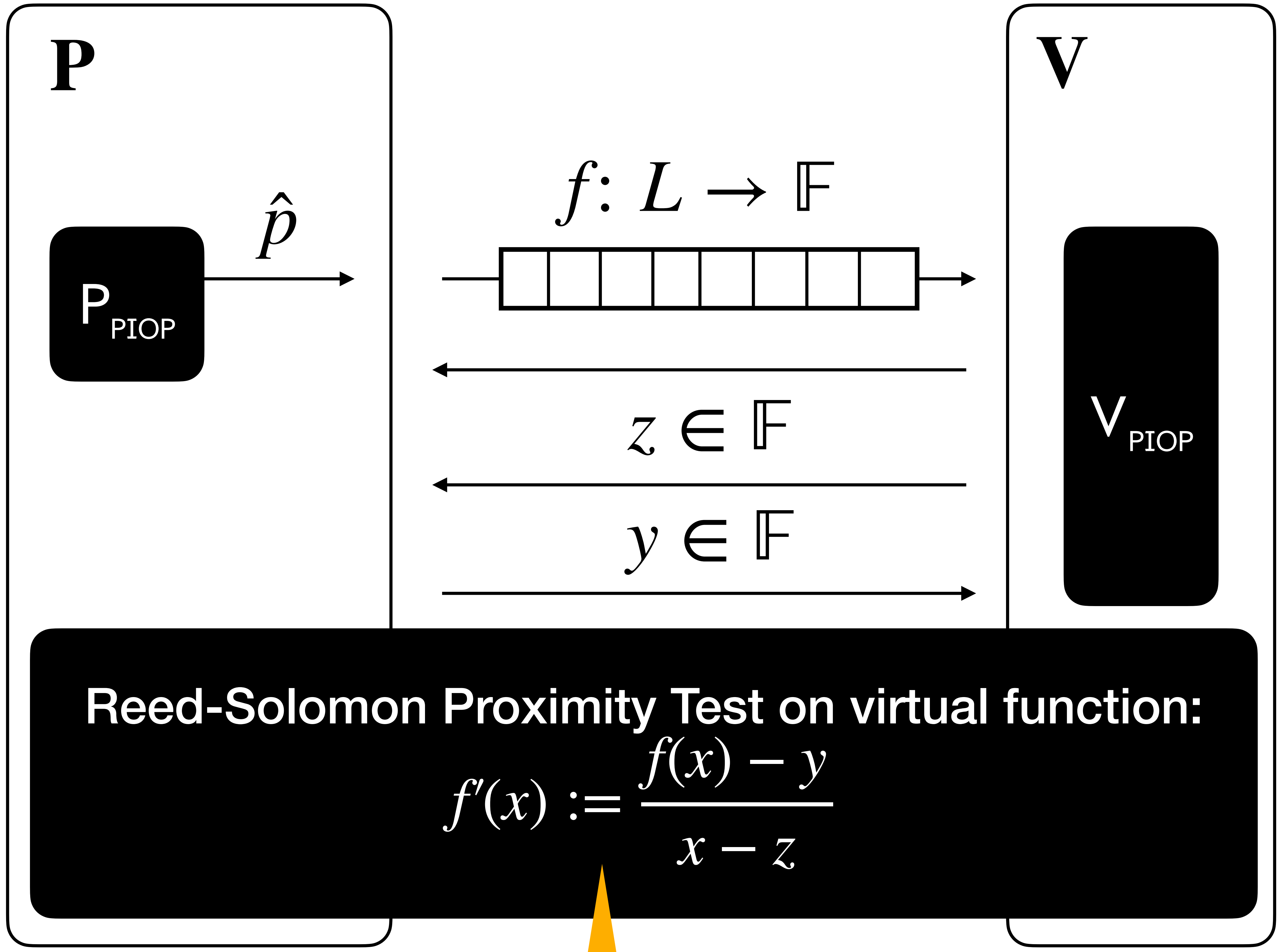
Modern IOP (and accumulation schemes) almost entirely inherit their efficiency property from the choice of code used

Constructing IOPs

Traditionally

Strategy: use Reed-Solomon codes as "redundant" encoding. Use a proximity test to check claims on encoded oracles.





The efficiency of the IOP is almost entirely determined by the proximity test

IOP of Proximity to RS codes

Convenience

$$\mathrm{RS}[n, m, \rho] := \left\{ \text{Evaluations of polynomials of degree } < 2^m \text{ on a domain } L \subseteq \mathbb{F} \text{ of size } n.\ \rho := \frac{2^m}{n} \right\}$$

Rate of the code

Completeness

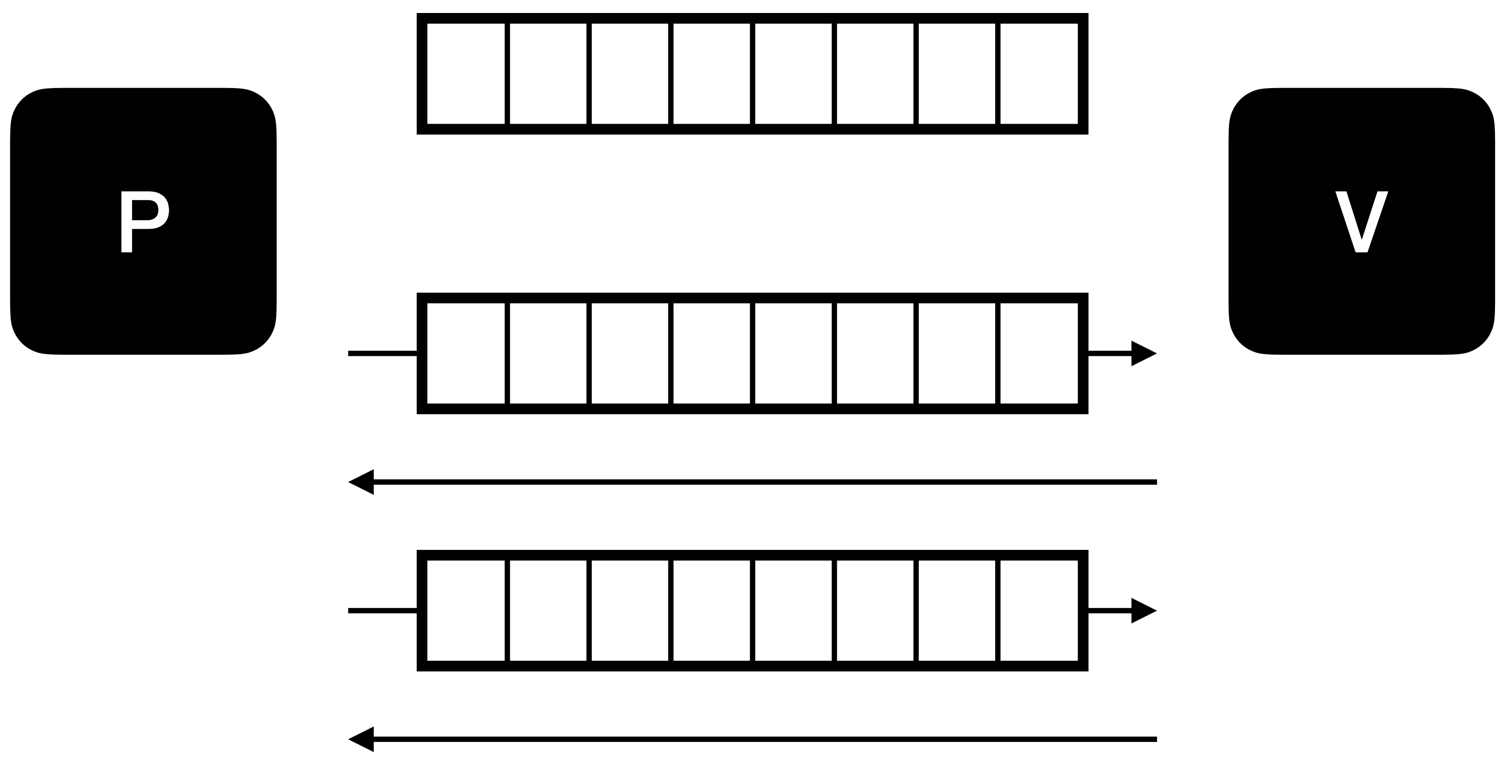
If $f \in \mathrm{RS}[n, m, \rho]$ then $\mathbf{V}$ accepts

Soundness

If $\Delta(f, \mathrm{RS}[n, m, \rho]) > \delta$ then w.h.p. $\mathbf{V}$ rejects

IOPP for RS

$f : L \to \mathbb{F}$



Goal: minimize queries to f and other proof oracles.

Constrained RS tests

What we are running:

Reed-Solomon Proximity Test on virtual function:

$$f'(x) := \frac{f(x) - y}{x - z}$$

What we really want to show:

I have a polynomial $\hat{f}$ and a commitment to (an encoding of it) f such that

$$\hat{f}(z) = y$$

Break it down as:

Test for **constrained** encoding

Quotient $f'(x) := \dfrac{f(x) - y}{x - z}$ **+** Reed—Solomon proximity test for f'

Can the proximity test **directly** enforce the constraint?

Yes! IOPP for **constrained codes**

Constrained codes

Idea: enrich proximity tests by enforcing constraints on the messages encoded by the word being tested.

$$\mathscr{C}[\hat{P}, \beta, \eta] = \{f \in \mathscr{C} : \hat{P}(\beta, \mathscr{C}^{-1}(f)) = \eta\}$$

Constraint and parameters

Value of constraint

Note:

$\Delta(f, \mathscr{C}[\hat{P}, \beta, \eta]) > \delta$ means that:

For every $u \in \Lambda(\mathscr{C}, f, \delta)$, the **message** $w := \mathscr{C}^{-1}(u)$ does not satisfy the constraint $\hat{P}(\beta, w) = \eta$

We will be interested in one particular type of constraints

Sumcheck constraints

$$\mathscr{C}[\hat{v}, \eta] = \left\{ f \in \mathscr{C} \ : \ \sum_{\mathbf{b} \in \{0,1\}^m} \hat{f}(\mathbf{b}) \cdot \hat{v}(\mathbf{b}) = \eta \right\}$$

Very powerful! If $\hat{v}(\mathbf{b}) = \mathrm{eq}(\mathbf{b}, \mathbf{z})$ this captures evaluation claims of **multilinear** polynomials at $\mathbf{z} \in \mathbb{F}^m$

Multilinear sumcheck

[LFNK92]

Open Qs

Is there a constant round sumcheck protocol with linear prover time?

$$\sum_{\mathbf{b}\in\{0,1\}^m} \hat{v}(\mathbf{b}) \cdot \hat{f}(\mathbf{b}) = \eta$$

Compute:

$$\hat{h}_1(X) = \sum_{\mathbf{b}\in\{0,1\}^{m-1}} \hat{v}(X, \mathbf{b}) \cdot \hat{f}(X, \mathbf{b})$$

$$\hat{h}_2(X) = \sum_{\mathbf{b}\in\{0,1\}^{m-2}} \hat{v}(\alpha_1, X, \mathbf{b}) \cdot \hat{f}(\alpha_1, X, \mathbf{b})$$

…

$$\mathrm{P} \quad \begin{array}{c} \xrightarrow{\hat{h}_1} \\ \xleftarrow{\alpha_1} \\ \vdots \\ \xrightarrow{\hat{h}_m} \\ \xleftarrow{\alpha_m} \end{array} \quad \mathrm{V}$$

Check that:

$$\hat{h}_1(0) + \hat{h}_1(1) = \eta$$

$$\hat{h}_2(0) + \hat{h}_2(1) = \hat{h}(\alpha_1),$$

…

$$\hat{f}(\alpha) \cdot \hat{v}(\alpha) = \hat{h}_m(\alpha_m)$$

Extremely powerful!

Only requires m rounds

Prover can be implemented in $O(2^m)$ field operations

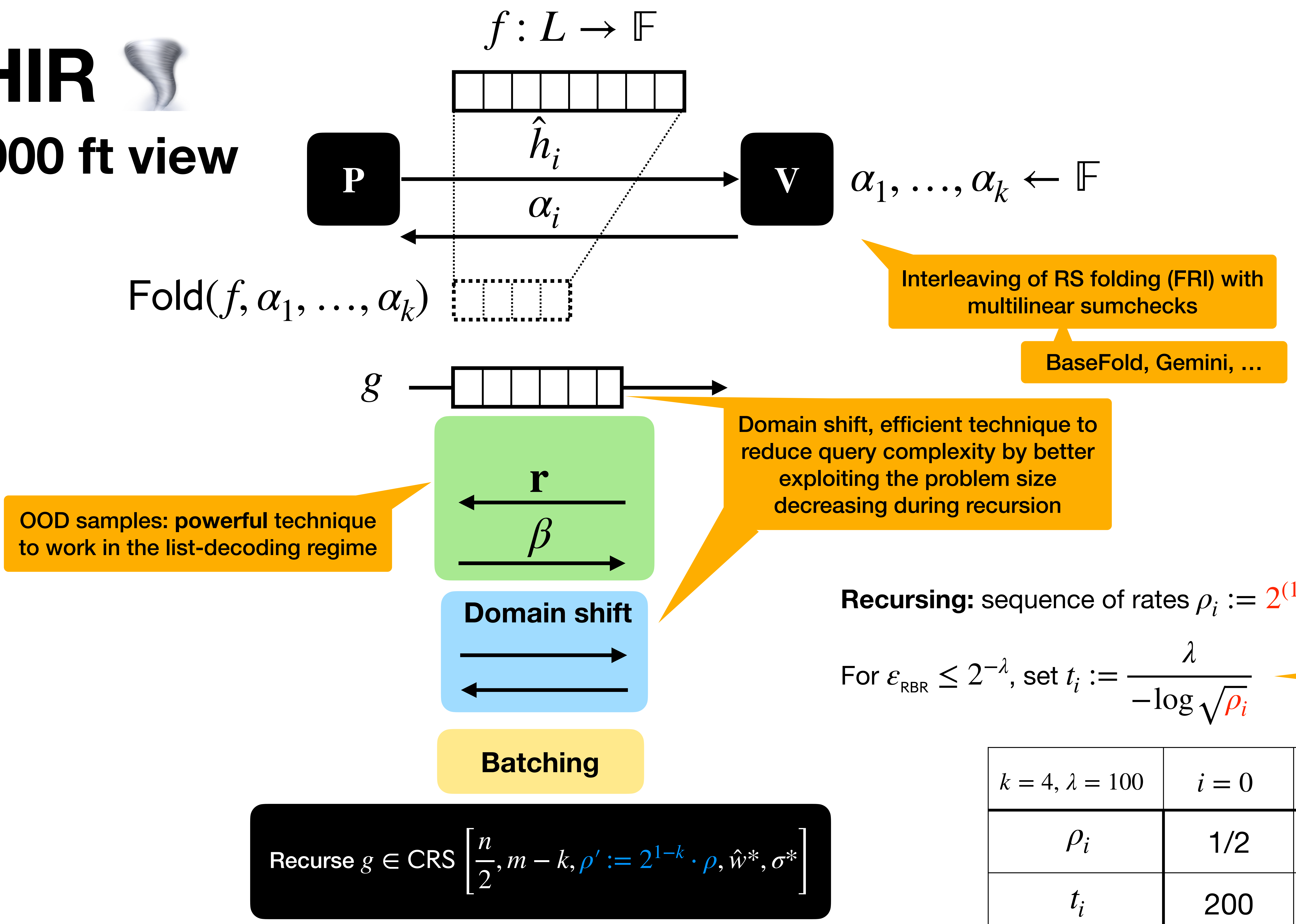
Verifier runs in time $O(m + |\hat{f}| + |\hat{v}|)$

This makes sense when argument size is not a bottleneck

Recent trend in succinct arguments is to use **multilinear PIOPs instead of univariate because of multilinear sumcheck**

WHIR 🌪️

A 1000 ft view



Recursing: sequence of rates $\rho_i := 2^{(1-k) \cdot i} \cdot \rho$.

For $\varepsilon_{\text{RBR}} \leq 2^{-\lambda}$, set $t_i := \dfrac{\lambda}{-\log \sqrt{\rho_i}}$

Fewer each round!

$k = 4, \lambda = 100$	$i = 0$	$i = 1$	$i = 2$
ρ_i	1/2	1/16	1/128
t_i	200	50	29

...

Comparison with prior work

	Queries	Verifier Time	Alphabet
BaseFold	$q_{\mathsf{BF}} = O(\lambda \cdot m)$	$O(q_{\mathsf{BF}})$	$\mathbb{F}^2$
FRI	$q_{\mathsf{FRI}} = O\left(\frac{\lambda}{k} \cdot m\right)$	$O(q_{\mathsf{FRI}} \cdot 2^k)$	$\mathbb{F}^{2^k}$
STIR	$q_{\mathsf{STIR}} = O\left(\frac{\lambda}{k} \cdot \log m\right)$	$O(q_{\mathsf{STIR}} \cdot 2^k + \lambda^2 \cdot 2^k)$	$\mathbb{F}^{2^k}$
WHIR	$q_{\mathsf{WHIR}} = O\left(\frac{\lambda}{k} \cdot \log m\right)$	$O(q_{\mathsf{WHIR}} \cdot (2^k + m))$	$\mathbb{F}^{2^k}$

When $k \approx \log m$
$q_{\mathsf{WHIR}} = O(\lambda)$
OPTIMAL

When $k \approx \log m$
$O(q_{\mathsf{WHIR}} \cdot |\Sigma|)$
OPTIMAL

When $k \approx \log m$
$\Sigma = \mathbb{F}^m$
Improving is an open question

Super fast verifier

- The WHIR verifier typically runs in a few hundred **MICRO**-seconds.
- Other verifiers require several **MILLI**-seconds (and more).
- While maintaining **state-of-the-art** prover time & argument size

Verifier time (ms)	Brakedown	Ligero	Greyhound	Hyrax	Trusted setup Pairing-based		Transparent setup Hash-based	
					PST	**KZG**	**WHIR[1/2]**	**WHIR[1/16]**
$\lambda = 100$	3500	733	-	100	7.81	2.42	0.61	0.29
$\lambda = 128$	3680	750	130	151	9.92	3.66	1.4	0.6

WHIR[ρ] denotes WHIR with rate ρ

Comparison with FRI and STIR

Note: prover time graph is now outdated due to new optimizations discovered

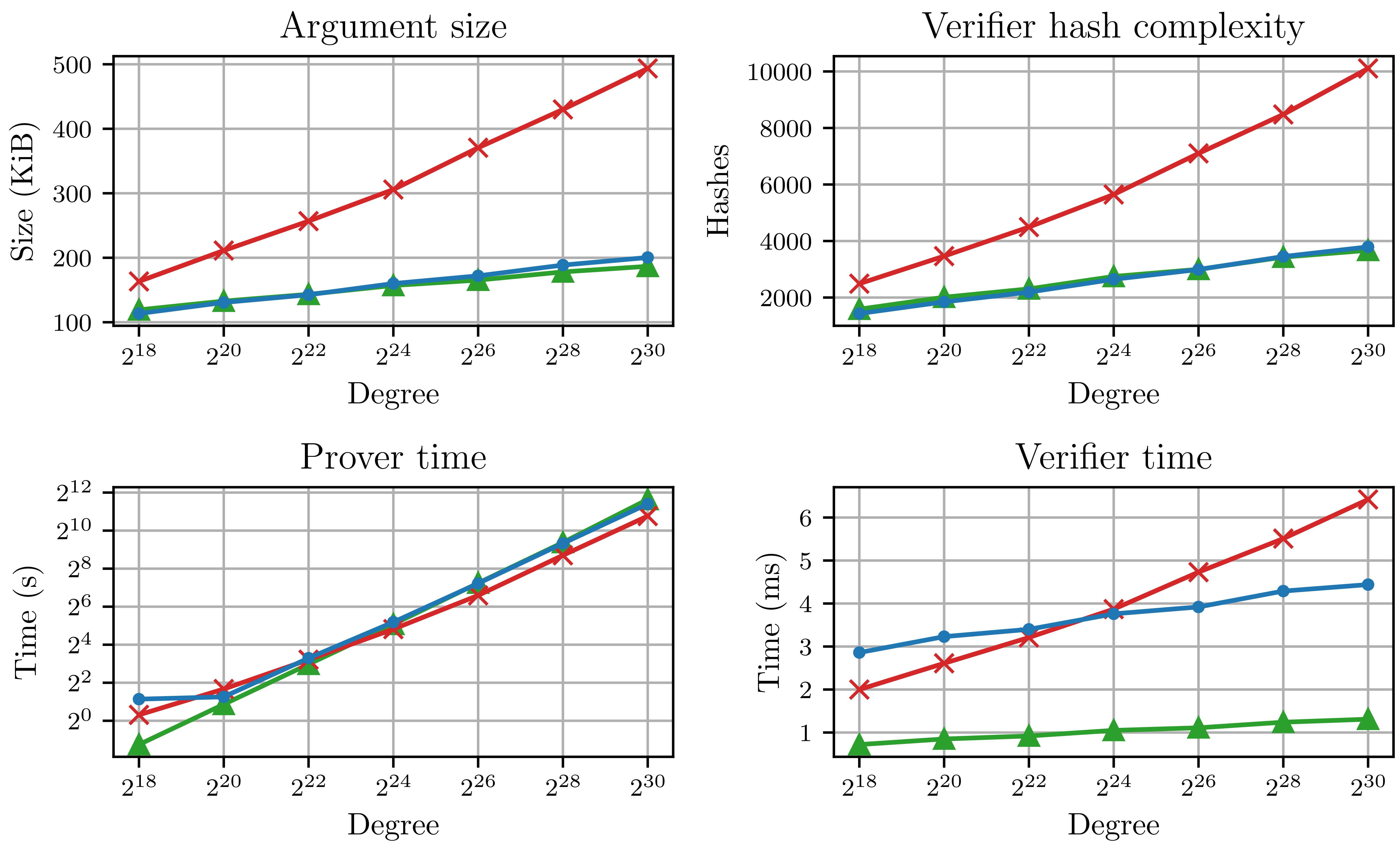
128-bits security level.

$\lambda = 106 + 22$ bits of PoW + "list-decoding" assumptions.

2^24 coeffs rate = 1/4	FRI	WHIR
Size (KiB)	177	110
Verifier time	2.4ms	700μs

2^30 coeffs rate = 1/2	FRI	WHIR
Size (KiB)	494	187
Verifier time	4.4ms	1.3ms

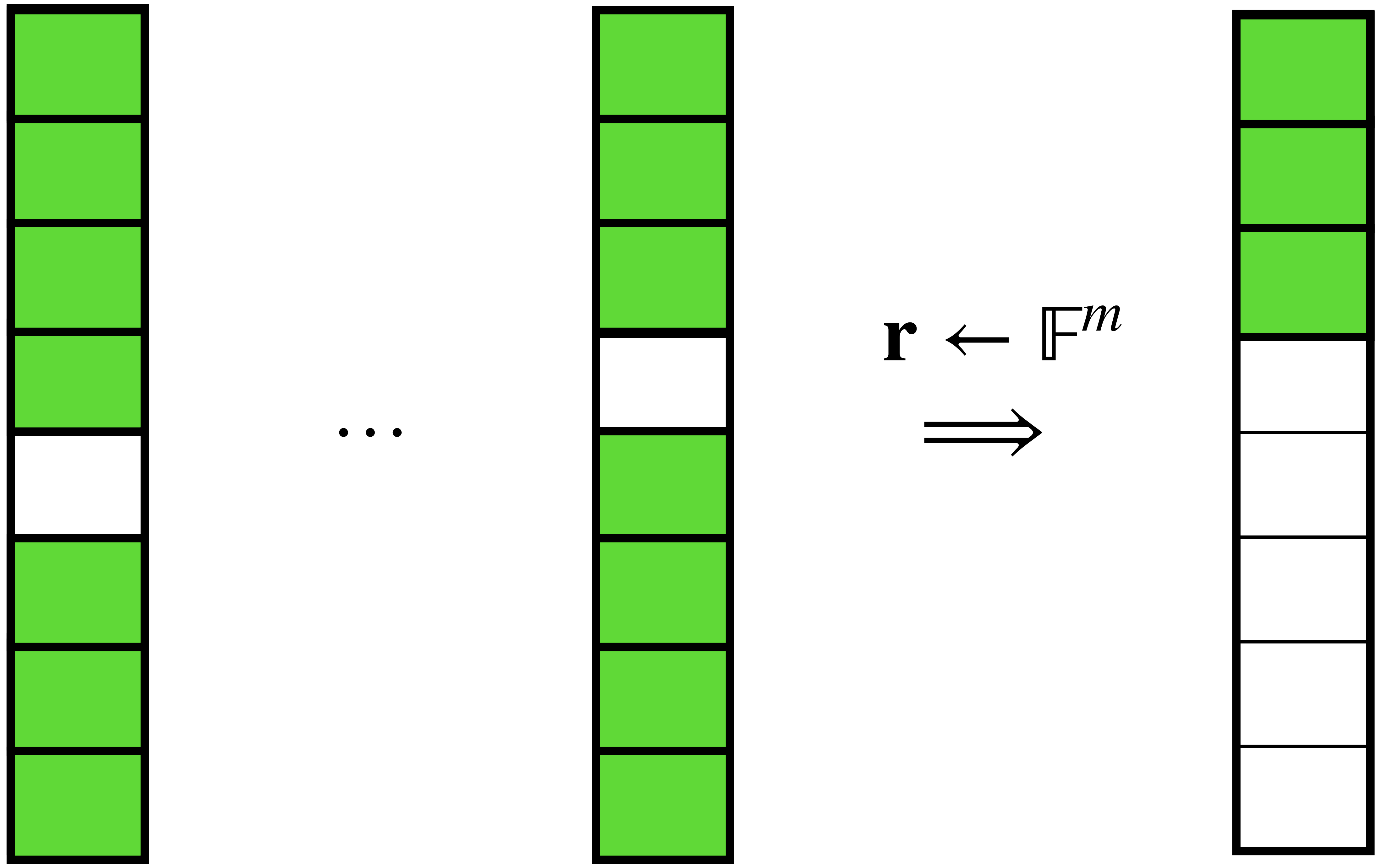
Rate of the code $\rho = 1/2$



FRI, STIR, WHIR

Mutual correlated agreement

Security analysis relies on properties of random combinations of linear error correcting codes



f_1 $\quad$ f_m $\quad$ $f^* := \sum_i r_i f_i$

Huge Open Q

Correlated agreement for RS codes up to distance $\delta > 1 - \sqrt{\rho}$

if w.h.p. $\Delta(f^*, \mathscr{C}) \leq \delta$:

Agreement: then $\Delta(f_i, \mathscr{C}) \leq \delta$.

Correlated agreement: then $f_1, \ldots, f_m$ agree with $\mathscr{C}$ on the same "stripe"

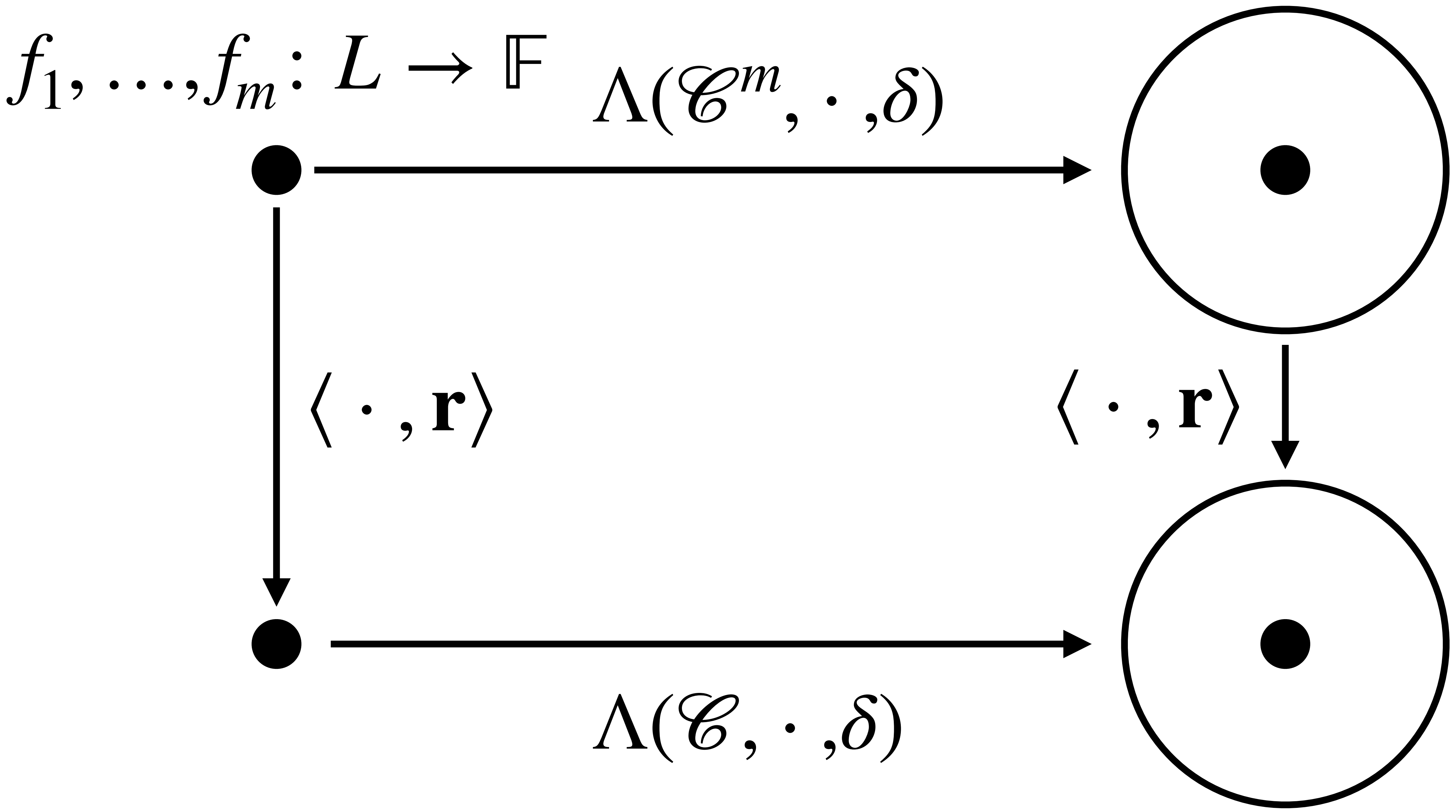
Mutual correlated agreement: the stripe in which $f_1, \ldots, f_m$ agree with $\mathscr{C}$ is the same on which f^* does:

"*No new correlated domains appear*"

Folding and lists commute

Implied by mutual correlated agreement

$\Lambda(\mathscr{C}, f, \delta)$ is the list of codewords of $\mathscr{C}$ that are δ-close to f



Stronger than what is required for FRI/STIR's soundness

Lemma

w.h.p. over $\mathbf{r}$:

$$\Lambda(\mathscr{C}, \langle \mathbf{f}, \mathbf{r} \rangle, \delta) = \left\{ \langle \mathbf{u}, \mathbf{r} \rangle : \mathbf{u} \in \Lambda(\mathscr{C}^m, \mathbf{f}, \delta) \right\}$$

Lemma

w.h.p. over $\mathbf{r}$:

$$\Lambda(\mathscr{C}, \mathsf{Fold}(f, \alpha), \delta) = \left\{ \mathsf{Fold}(u, \alpha) : u \in \Lambda(\mathscr{C}, f, \delta) \right\}$$

Recent results show that mutual correlated agreement holds up to 1.5 Johnson for general linear codes!

We show correlated agreement implies mutual correlated agreement in *unique decoding.*

Conclusion

Open questions

In this talk

Open Qs

Can we get even **smaller** SNARKs?

Get argument of comparable size with **universal setup**

Open Qs

Are Merkle tree optimal as VC in the ROM?

More concretely efficient post-quantum vector commitment schemes

Open Qs

Is there a constant round sumcheck protocol with linear prover time?

Open Qs

Concretely efficient argument/polynomial commitment **without private coin setup** and argument size < 5 KiB

Concretely efficient argument/polynomial commitment with **post-quantum** security and argument size < 20 KiB

Open Qs

IOPP with optimal query complexity and verifier time in the constant alphabet regime.

Huge Open Q

Correlated agreement for RS codes up to distance $\delta > 1 - \sqrt{\rho}$

Open questions

Other things that are super interesting

Open Qs

Lattice-based polynomial commitments/arguments with: polylogarithmic verification & argument size < 50 KiB

Open Qs

Concretely efficient constructions of relativizing arguments in the AROM or similar idealized models.

Open Qs

Linear time accumulation schemes with constant round complexity

Open Qs

Is there an analogue of parallel repetition that amplifies state-restoration soundness?

Open Qs

UC-security for recursive proof composition (in GROM or AROM)

Open Qs

Linear time accumulation schemes over small fields

Lots of schemes!

Awh	Arc	Aurora	BaseFold
Binius	Brakedown	Bulletproofs	CycleFold
Dory	DEEP-FRI	FRI	Fractal
Geppetto	Gemini	Groth16	Greyhound
Halo	Halo2	HyperNova	HyperPlonk
Hyrax	Jolt	Kilonova	KZG
Lasso	LatticeFold	Libra	Ligero
Lova	Mangrove	Marlin	MIRAGE
Mova	Nova	Origami	Orion
Pickles	Pinocchio	Plonk	Plonky2
ProtoGalaxy	ProtoStar	Reverie	Sangria
Shout	SLAP	Sonic	Spartan
SPARKs	Spice	STARK	STIR
SuperNova	Supersonic	TinySpartan	Twist
Twinkle	Virgo	WARP	WHIR

ChatGPT generated, so most likely not complete

- **Accumulation scheme**
- **Polynomial commitment scheme**
- **Interactive oracle proof of proximity**
- **Memory argument/lookup**
- **Full SNARK**

Thanks for listening!

Extra slides

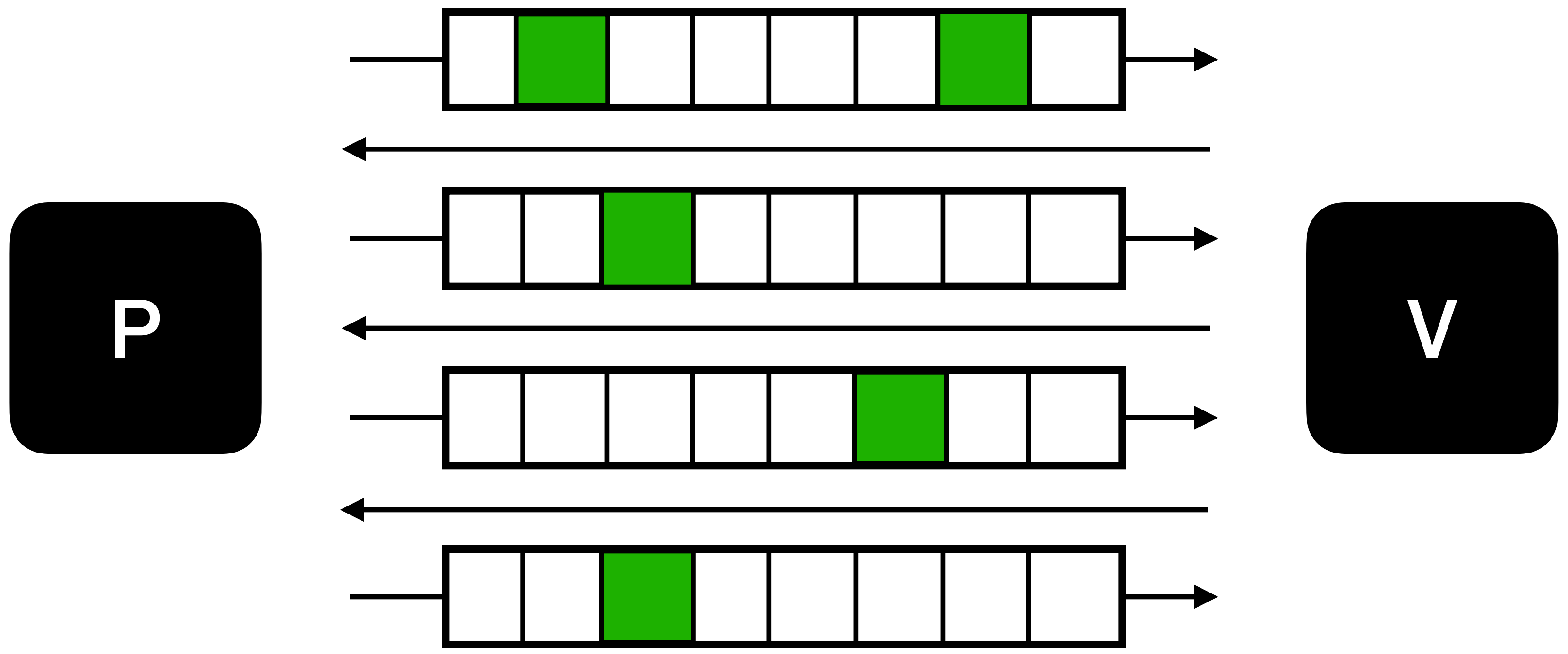
PIOP + PCS

The modular way™

Plenty of choices available: Aurora, Fractal, Spartan…

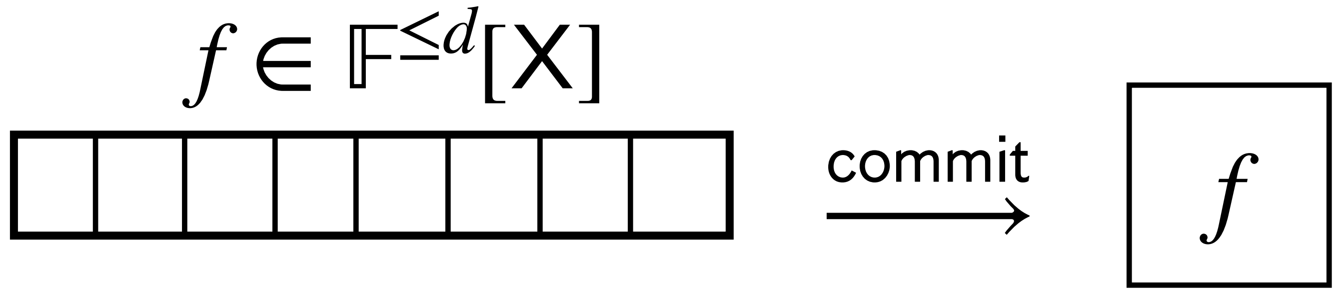
We focus on this!

PIOP



+

PCS



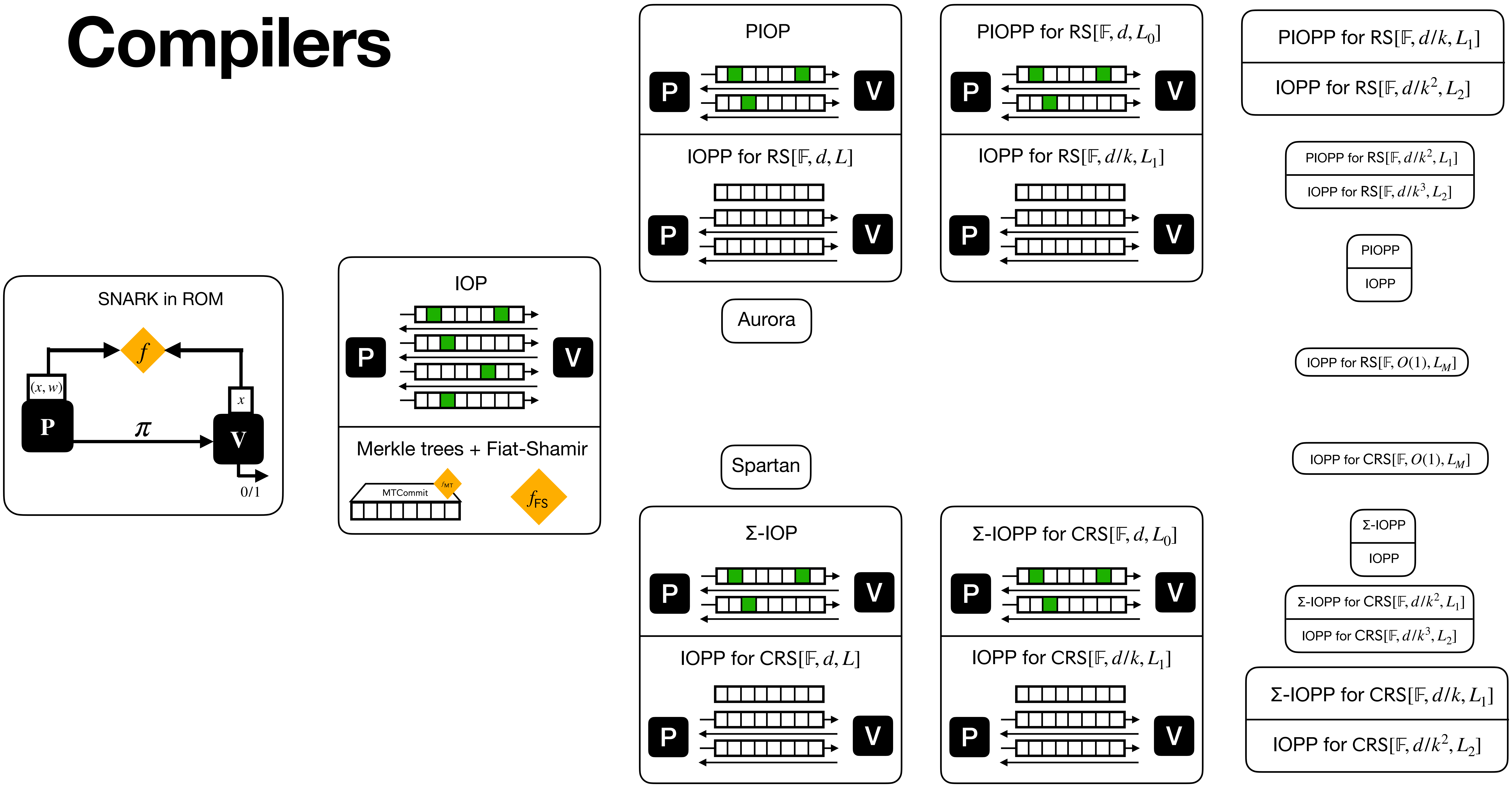
Later, can prove that:

$$f(x) = y, \text{ for } x, y \in \mathbb{F}$$

- Oracles are polynomials
- Security is information-theoretical
- Proof length is $\Omega(n)$ (not succinct)
- Verifiers are very efficient

- Cryptography goes here!
- Computational security
- We can achieve succinctness

Compilers

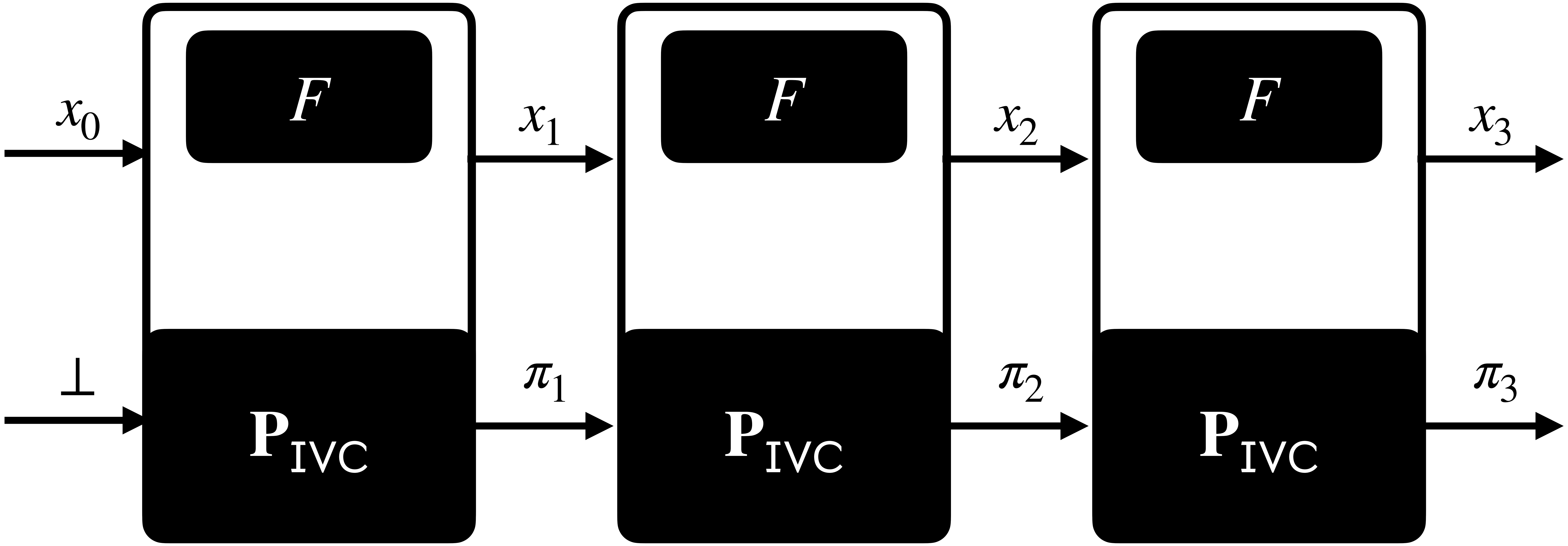

SNARK in ROM
f
(x, w)
x
P
π
V
0/1
IOP
P
V
Merkle trees + Fiat-Shamir
MTCommit
f_{MT}
f_{FS}
PIOP
IOPP for RS$[\mathbb{F}, d, L]$
Aurora
Spartan
Σ-IOP
IOPP for CRS$[\mathbb{F}, d, L]$
PIOPP for RS$[\mathbb{F}, d, L_0]$
IOPP for RS$[\mathbb{F}, d/k, L_1]$
Σ-IOPP for CRS$[\mathbb{F}, d, L_0]$
IOPP for CRS$[\mathbb{F}, d/k, L_1]$
PIOPP for RS$[\mathbb{F}, d/k, L_1]$
IOPP for RS$[\mathbb{F}, d/k^2, L_2]$
PIOPP for RS$[\mathbb{F}, d/k^2, L_1]$
IOPP for RS$[\mathbb{F}, d/k^3, L_2]$
PIOPP
IOPP
IOPP for RS$[\mathbb{F}, O(1), L_M]$
IOPP for CRS$[\mathbb{F}, O(1), L_M]$
Σ-IOPP
IOPP
Σ-IOPP for CRS$[\mathbb{F}, d/k^2, L_1]$
IOPP for CRS$[\mathbb{F}, d/k^3, L_2]$
Σ-IOPP for CRS$[\mathbb{F}, d/k, L_1]$
IOPP for CRS$[\mathbb{F}, d/k^2, L_2]$

Accumulation

Incrementally Verifiable Computation (IVC)

To prove $x_T = F^T(x_0)$, prove $\exists x_1, \ldots, x_{T-1}$ such that $\forall i \in [T], x_i = F(x_{i-1})$.

E.g. signature aggregation:
$F((\sigma_i, pk_i), b_i) := b_i \wedge \mathsf{SigVfy}(\mathsf{st}, pk_i, \sigma_i)$



$\mathbf{V}_{\mathrm{IVC}}(x_{i-1}, x_i, \pi_i)$ checks that π_i attests the whole computation!

$\mathbf{P}_{\mathrm{IVC}}$ costs independent from T ✅

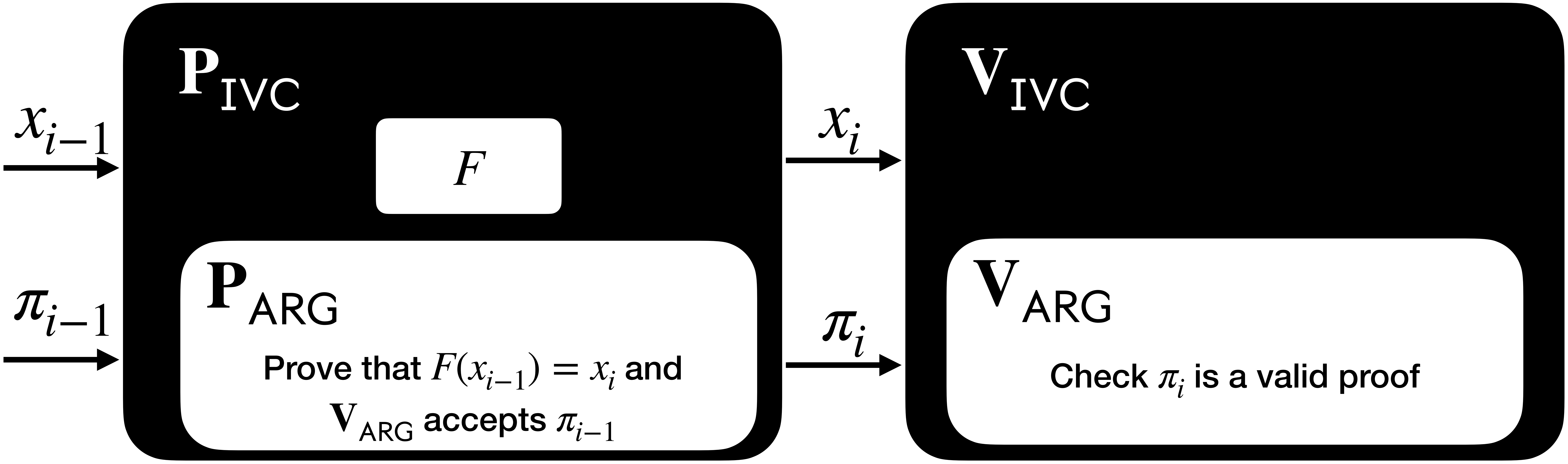
IVC can be generalized to **Proof-Carrying-Data** (PCD).
PCD considers a directed acyclic graph instead of a line.
PCD in practice is preferable to IVC, as it enables reducing the prover's latency.

Wonderful! How do I get IVC?

IVC from SNARKs

Recursive proof composition

(*) more complex than this, needs preprocessing



PQ SNARK
$\implies$ PQ IVC ✅

$|\pi|$ independent from T ✅

Cheap verification ✅

Memory costs
independent from T ✅

Cost of $\mathbf{P}_{\text{IVC}} \approx |F| + |\mathbf{V}_{\text{ARG}}|$
Concretely: $|\mathbf{V}_{\text{ARG}}| \approx 2^{20}$ constraints
i.e. recursive overhead is quite large
Good starting point, but can be improved!

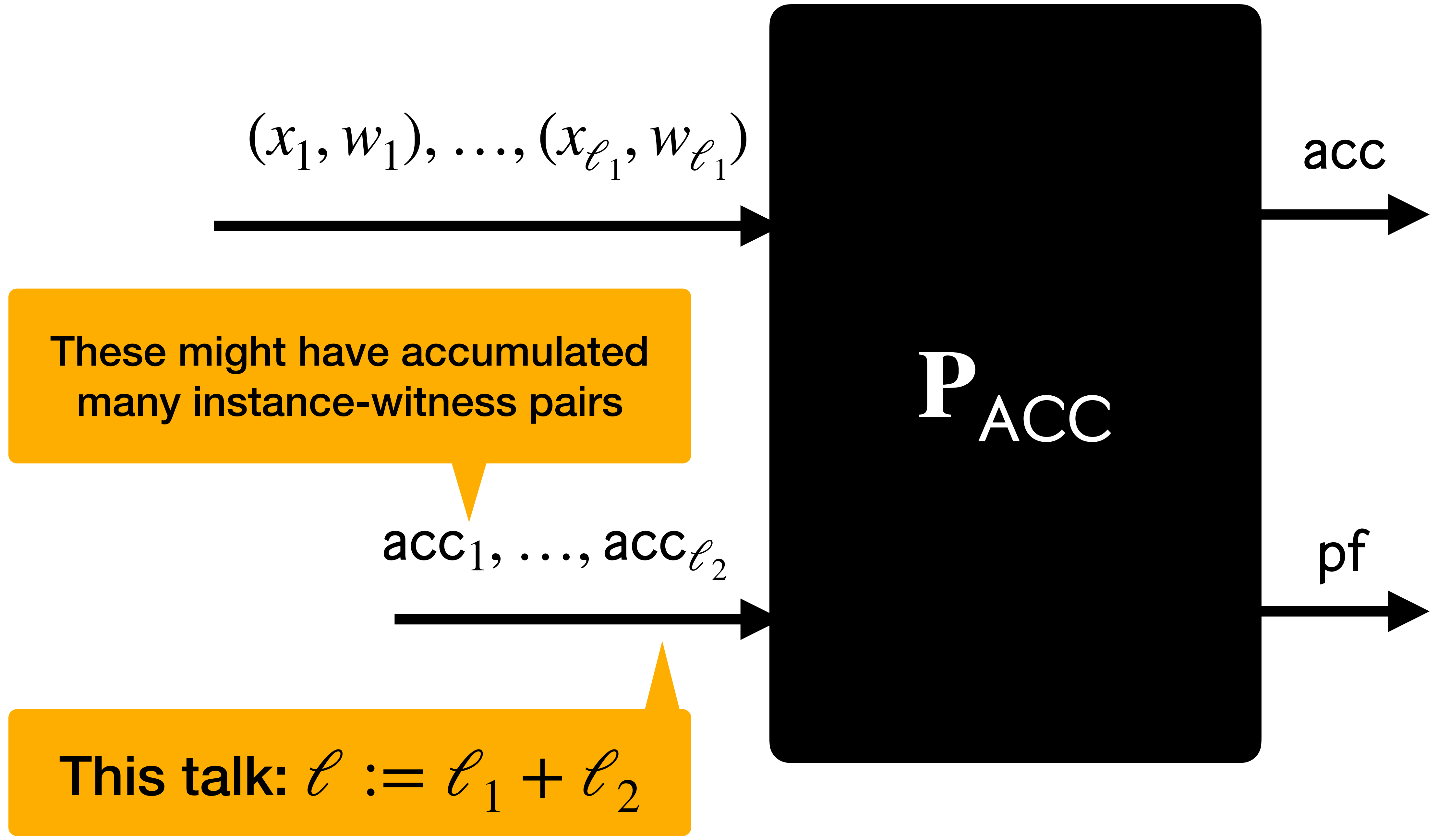
Accumulation Schemes

A lightweight tool for batching

Any ARG yields ACC with $|\mathbf{V}_{\mathsf{ACC}}| \approx |\mathbf{V}_{\mathsf{ARG}}(\ell_1)|$.
We can do (significantly) better!

Enables batching many checks $(x_i, w_i) \in_? \mathscr{R}$ into an accumulator acc.

- $\mathbf{V}_{\mathsf{ACC}}$ verifies that adding the inputs into acc was done correctly
- $\mathbf{D}_{\mathsf{ACC}}$ decides whether acc is valid.



If:

a) $\mathbf{V}_{\mathsf{ACC}}((x_i)_i, (\mathsf{acc}_j)_j, \mathsf{acc}, \mathsf{pf}) = 1$

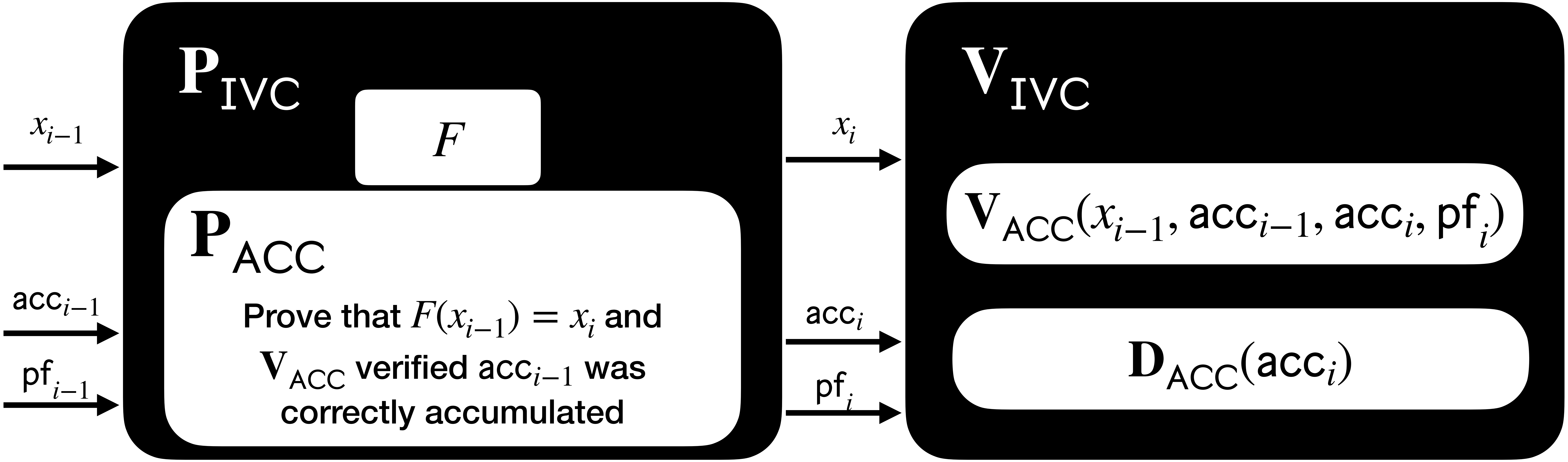
b) $\mathbf{D}_{\mathsf{ACC}}(\mathsf{acc}) = 1$

Then w.h.p:

$$\forall i \in [\ell_1] : \ (x_i, w_i) \in \mathscr{R}$$
$$\forall j \in [\ell_2] : \ \mathbf{D}_{\mathsf{ACC}}(\mathsf{acc}_j) = 1$$

IVC from accumulation

(*) actually we need a more refined notion: "split" accumulation schemes



PQ Accumulation $\implies$ PQ IVC ✅

Memory costs independent from T ✅

$|\pi|$ independent from T ✅

Cost of $\mathbf{P}_{\mathsf{IVC}} \approx |F| + |\mathbf{V}_{\mathsf{ACC}}|$ ✅

$\ll |\mathbf{V}_{\mathsf{ARG}}|$

Not succinct

Cost of $\mathbf{V}_{\mathsf{IVC}} \approx |\mathbf{V}_{\mathsf{ACC}}| + |\mathbf{D}_{\mathsf{ACC}}|$

Wrap with a final SNARK $\implies$ succinct verification ✅

One more thing…

Accumulation schemes are broadly useful for integrity in distributed systems with repeated computations.

Verifiable Virtual Machines (VVMs)

SP1

RISC ZERO

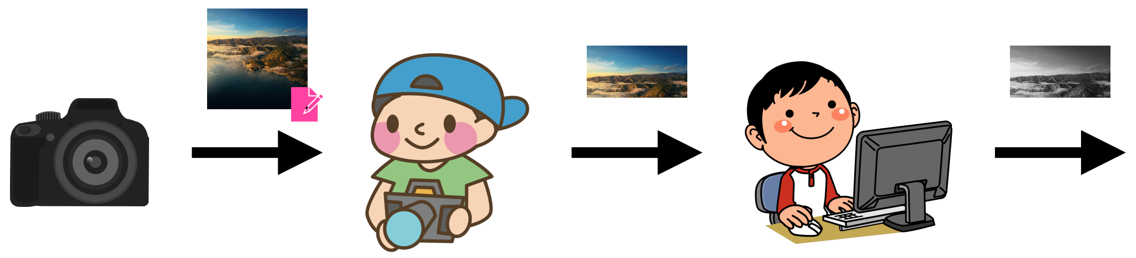
NEXUS

OpenVM

+ At least 25 more…

Digital provenance

VIMz: Private Proofs of Image Manipulation using Folding-based zkSNARKs*
Stefan Dziembowski Shahriar Ebrahimi Parisa Hassanizadeh



Eva: Efficient Privacy-Preserving Proof of Authenticity for Lossily Encoded Videos
Chengru Zhang[1], Xiao Yang[2], David Oswald[2], Mark Ryan[2], and Philipp Jovanovic[3]

Consensus

Breaking the $O(\sqrt{n})$-Bit Barrier: Byzantine Agreement with Polylog Bits Per Party
Elette Boyle* Ran Cohen† Aarushi Goel‡

And more…

Reef: Fast Succinct Non-Interactive Zero-Knowledge Regex Proofs
Sebastian Angel* Eleftherios Ioannidis* Elizabeth Margolin* Srinath Setty† Jess Woods*
*University of Pennsylvania †Microsoft Research

ALPACA: Anonymous Blocklisting with Constant-Sized Updatable Proofs
Jiwon Kim, University of Michigan; Abhiram Kothapalli, University of California, Berkeley; Orestis Chardouvelis, Carnegie Mellon University; Riad S. Wahby, Carnegie Mellon University; Paul Grubbs, University of Michigan

Mangrove: A Scalable Framework for Folding-based SNARKs
Wilson Nguyen Trisha Datta Binyi Chen Nirvan Tyagi Dan Boneh

Accumulation schemes:

Group-based

Nova, Supernova, Hypernova, Protostar, Protogalaxy, NeutronNova, KZHFold, …

Must use 256-bit fields, accumulation time super-linear, cycles of curves required for recursion, not pq

Lattice-based

Latticefold, Lova, Latticefold+, Neo

Very promising, accumulation costs super-linear, plausibly pq some field flexibility

Hash-based

Awh, ARC, WARP

Accumulation costs can be linear, plausibly pq, full field flexibility

NP-relation of choice

$$\mathscr{R}_{\mathsf{R1CS}}(\mathbb{F}) = \left\{ (i, x, w) : \begin{array}{c} i = (\mathbf{A}, \mathbf{B}, \mathbf{C}, M, N, k) \\ x \in \mathbb{F}^{N-k} \\ w \in \mathbb{F}^{k} \\ \mathbf{A}\begin{pmatrix} x \\ w \end{pmatrix} \odot \mathbf{B}\begin{pmatrix} x \\ w \end{pmatrix} = \mathbf{C}\begin{pmatrix} x \\ w \end{pmatrix} \end{array} \right\}$$

Hadamard product in $\mathbb{F}^m$

Typically $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are matrices with at most $O(M)$ non-zero entries

Modern SNARK design often focuses on **richer** relations such as Plonk, CCS, GR1CS, PESAT, AIR, ecc

The richer relation allows modeling complex circuits with smaller constraints, overall speeding the system