STIR 🥣

Reed-Solomon Proximity Testing with Fewer Queries

Gal Arnon

מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

Giacomo Fenzi

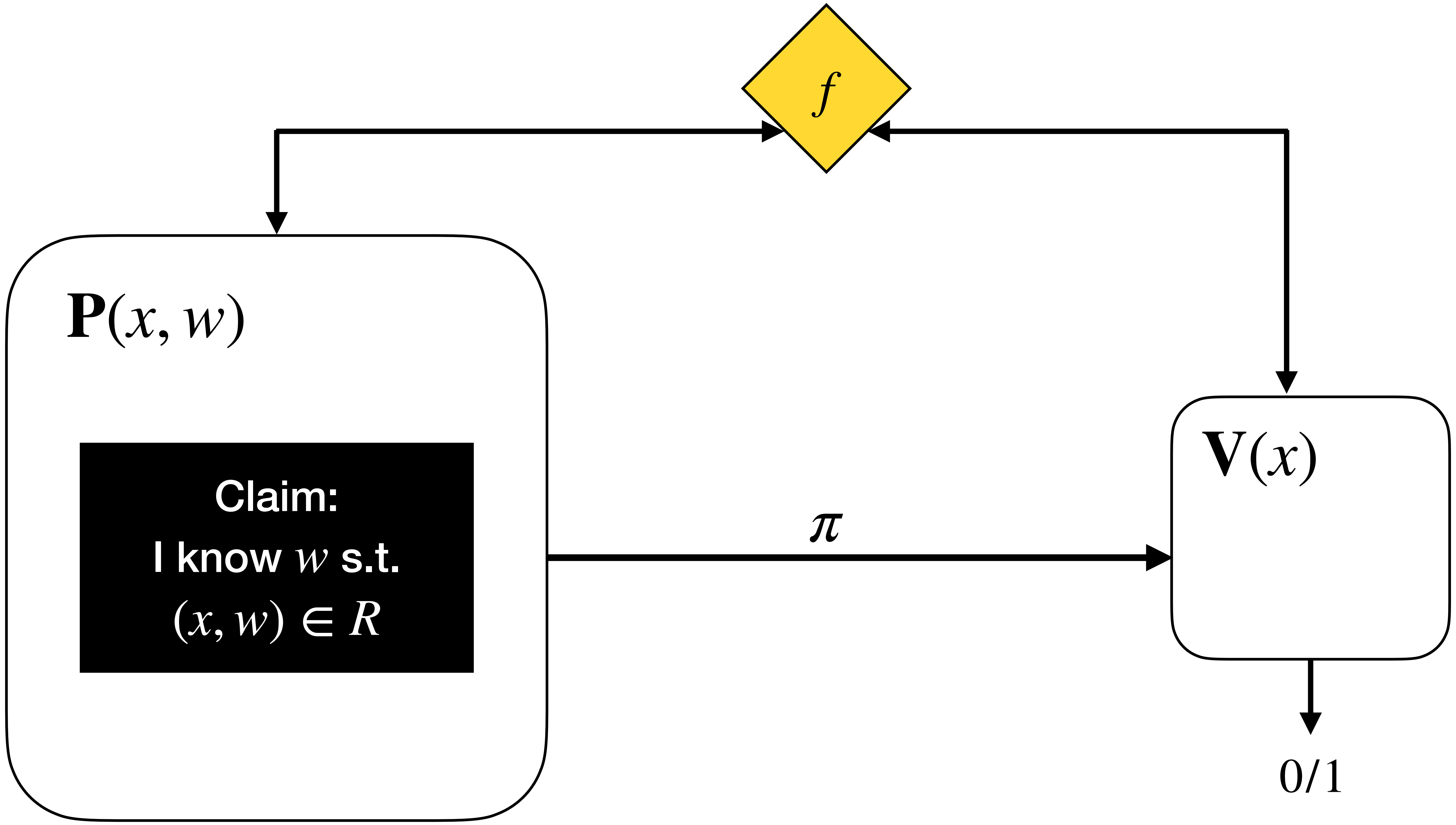
EPFL

Alessandro Chiesa

EPFL

Eylon Yogev

SNARKs in the ROM



- Succinct
  - $|\pi| \ll |w|$
- Non-interactive
- Argument of Knowledge
  - Straightline extractor

SNARKs in Practice

SNARKs in Practice

- ROM instantiated using a cryptographic hash function

SNARKs in Practice

- ROM instantiated using a cryptographic hash function
- Very fast and efficient instantiations

SNARKs in Practice

- ROM instantiated using a cryptographic hash function
- Very fast and efficient instantiations
- Used to secure billions of dollars in real-world blockchains

SNARKs in Practice

- ROM instantiated using a cryptographic hash function
- Very fast and efficient instantiations
- Used to secure billions of dollars in real-world blockchains

Rollups: STARKWARE polygon zkSync

zkVMs: Succinct RISC ZERO **And more…**

BCS Transformation

BCS Transformation

IOP

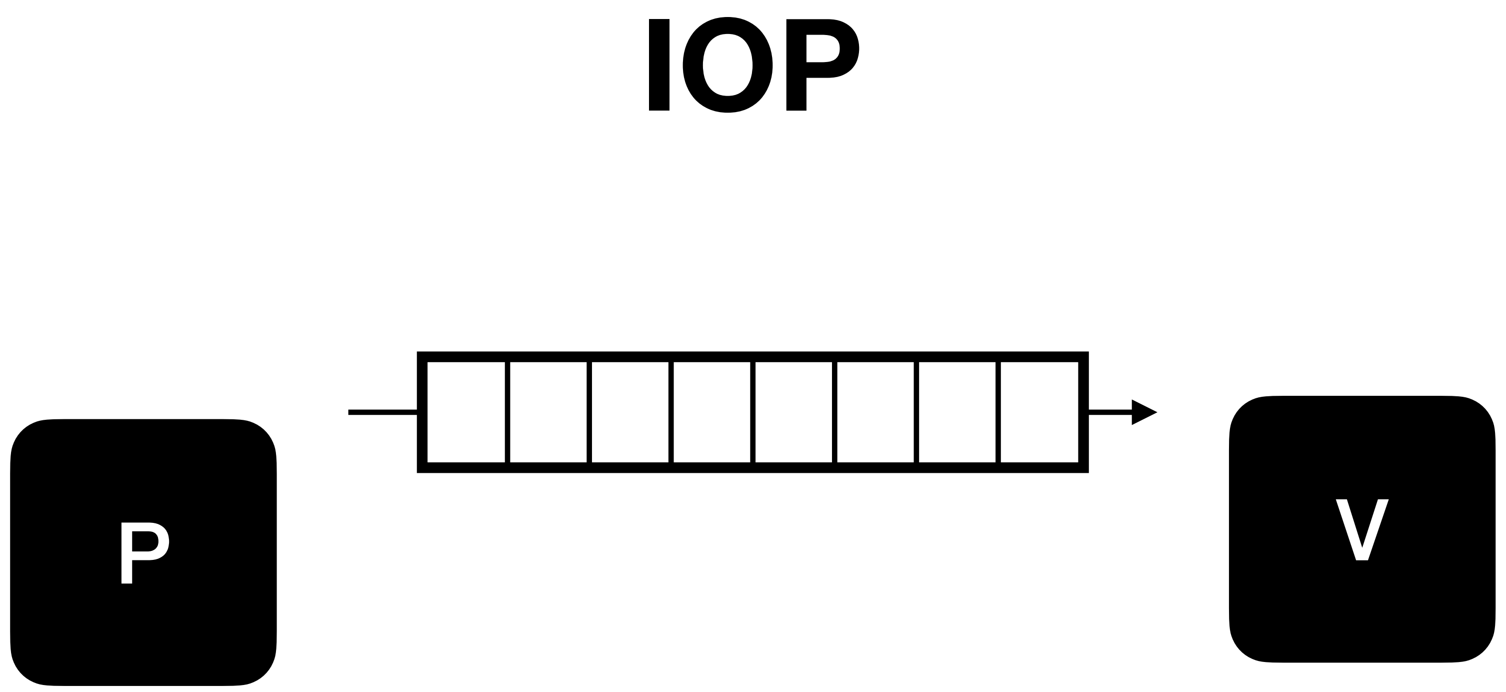
BCS Transformation

IOP

P

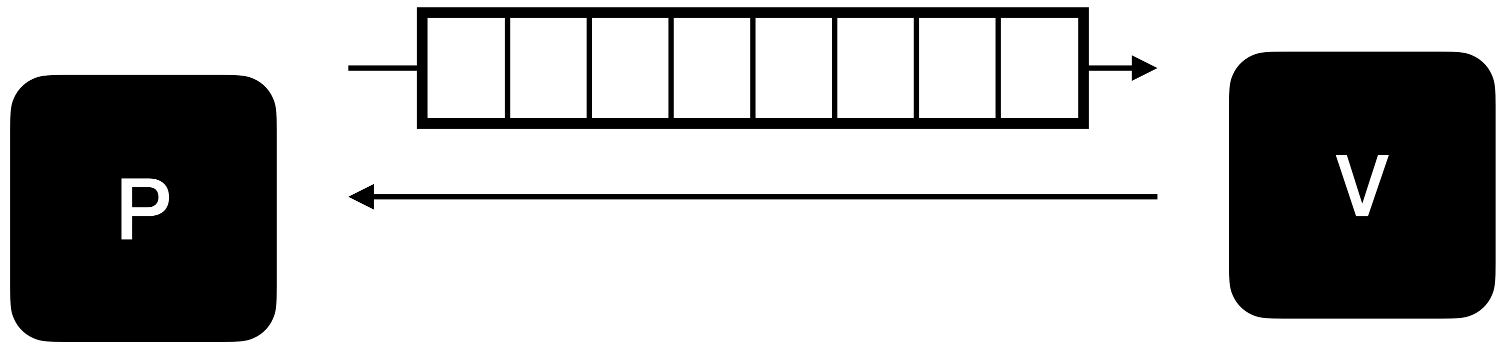
V

BCS Transformation



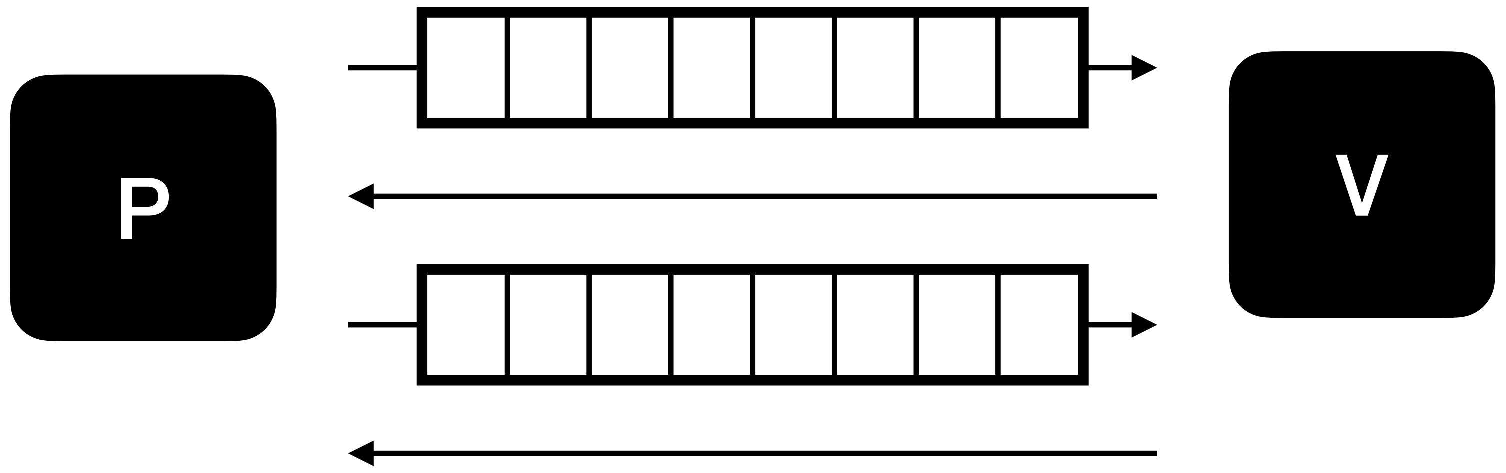
BCS Transformation

IOP



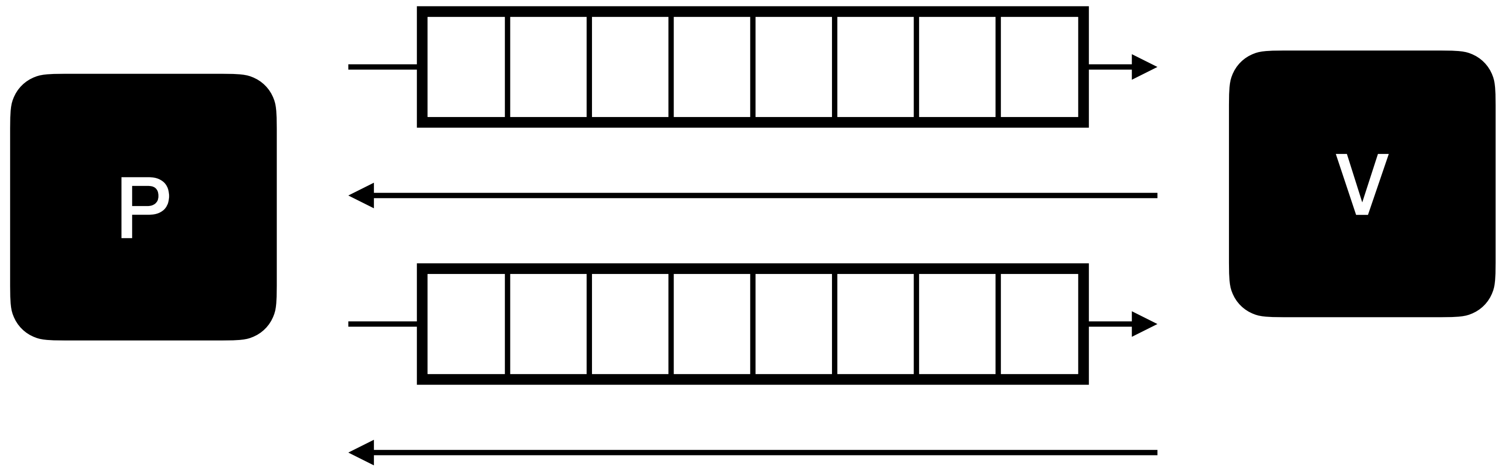
BCS Transformation

IOP



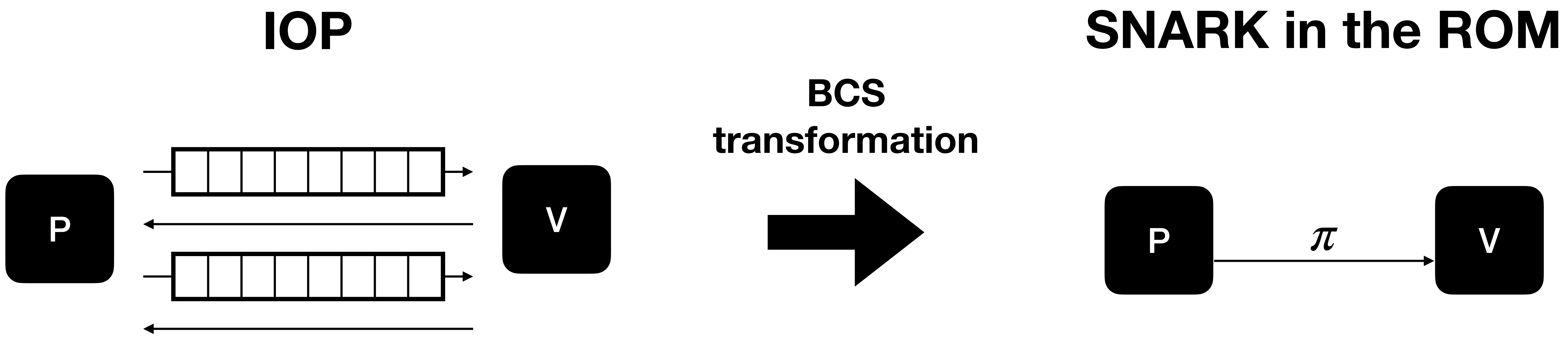
BCS Transformation

IOP



BCS transformation

BCS Transformation

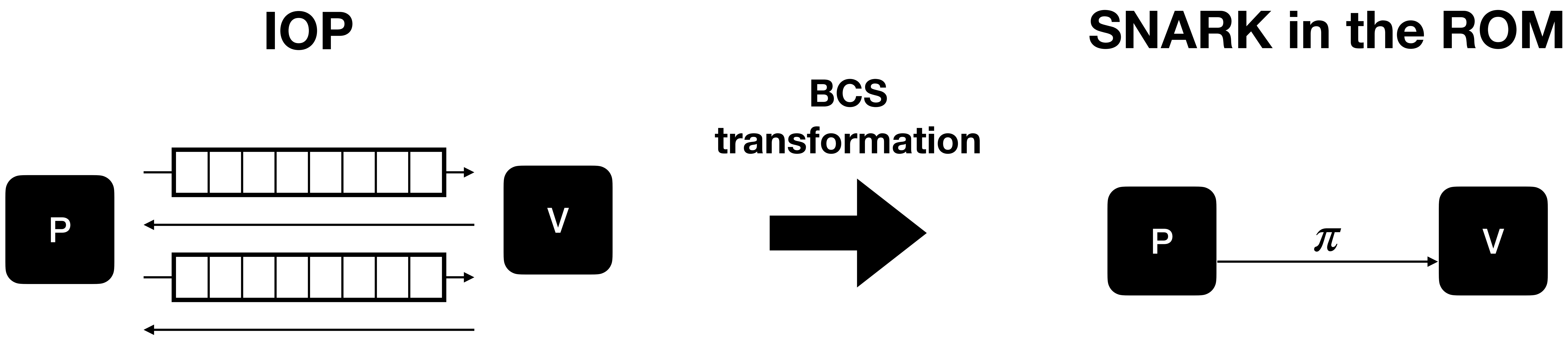


IOP

BCS transformation

SNARK in the ROM

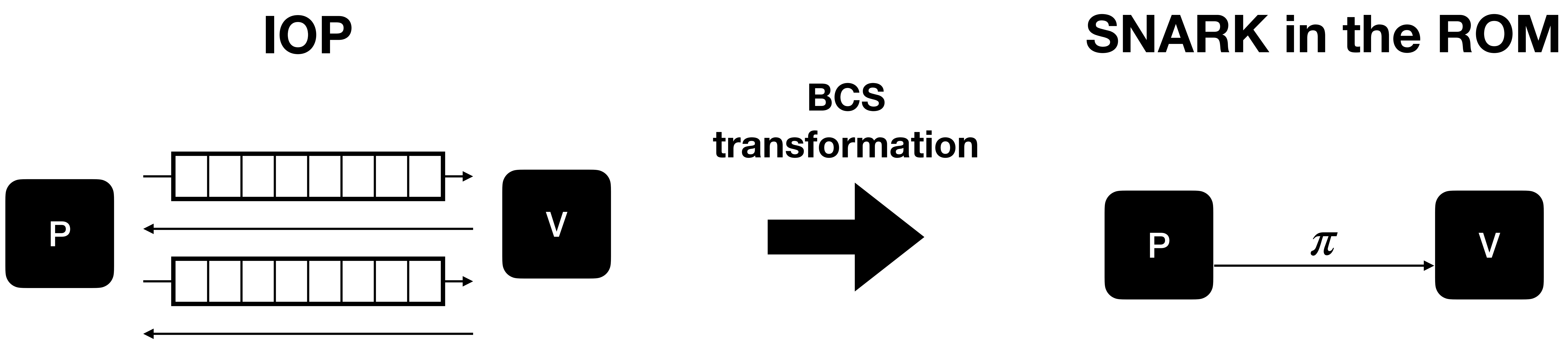
BCS Transformation



Proof length: l

Queries: q

BCS Transformation


IOP

BCS transformation

SNARK in the ROM

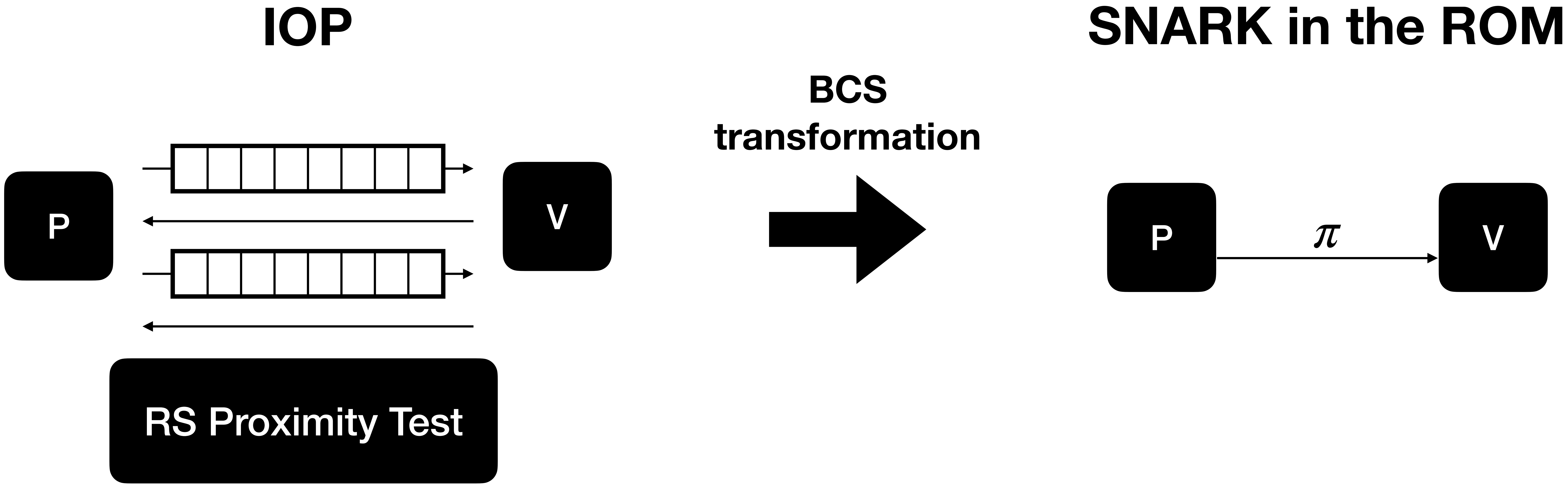
Proof length: l

Queries: q

Verifier hashes: $O(\mathsf{q} \cdot \log \mathsf{l})$

Argument size: $O(\lambda \cdot \mathsf{q} \cdot \log \mathsf{l})$

BCS Transformation



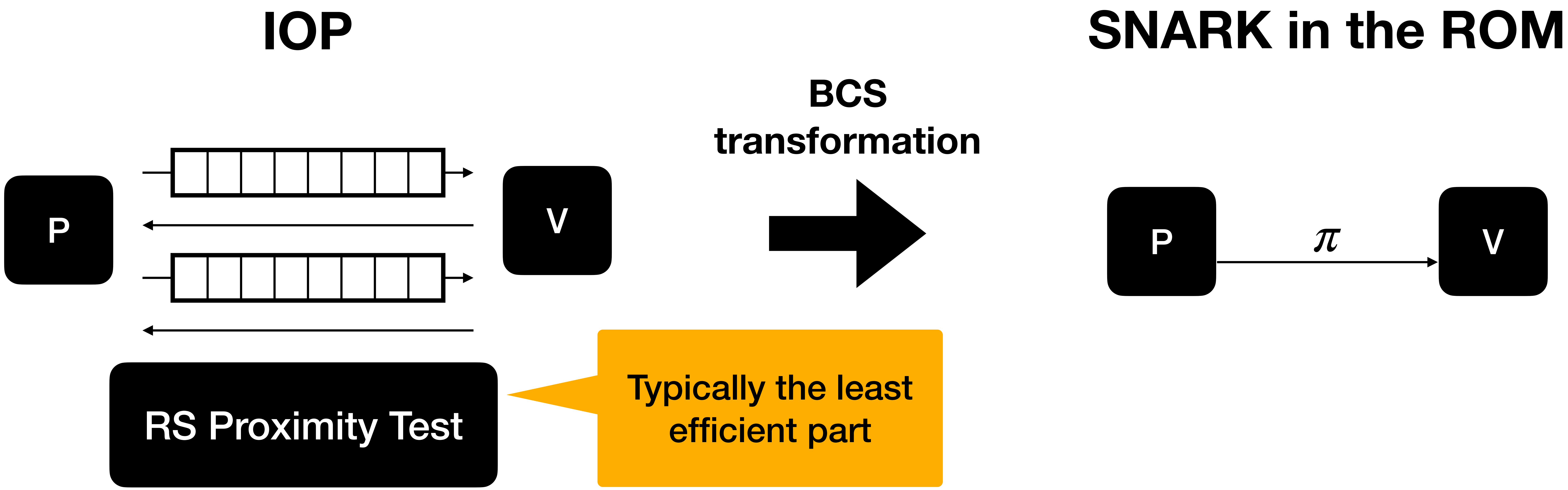
Proof length: l

Queries: q

Verifier hashes: $O(\mathsf{q} \cdot \log \mathsf{l})$

Argument size: $O(\lambda \cdot \mathsf{q} \cdot \log \mathsf{l})$

BCS Transformation



Proof length: l

Queries: q

Verifier hashes: $O(\mathsf{q} \cdot \log \mathsf{l})$

Argument size: $O(\lambda \cdot \mathsf{q} \cdot \log \mathsf{l})$

RS Codes and IOPPs

RS Codes and IOPPs

$\mathsf{RS}[\mathbb{F}, L, d]$

RS Codes and IOPPs

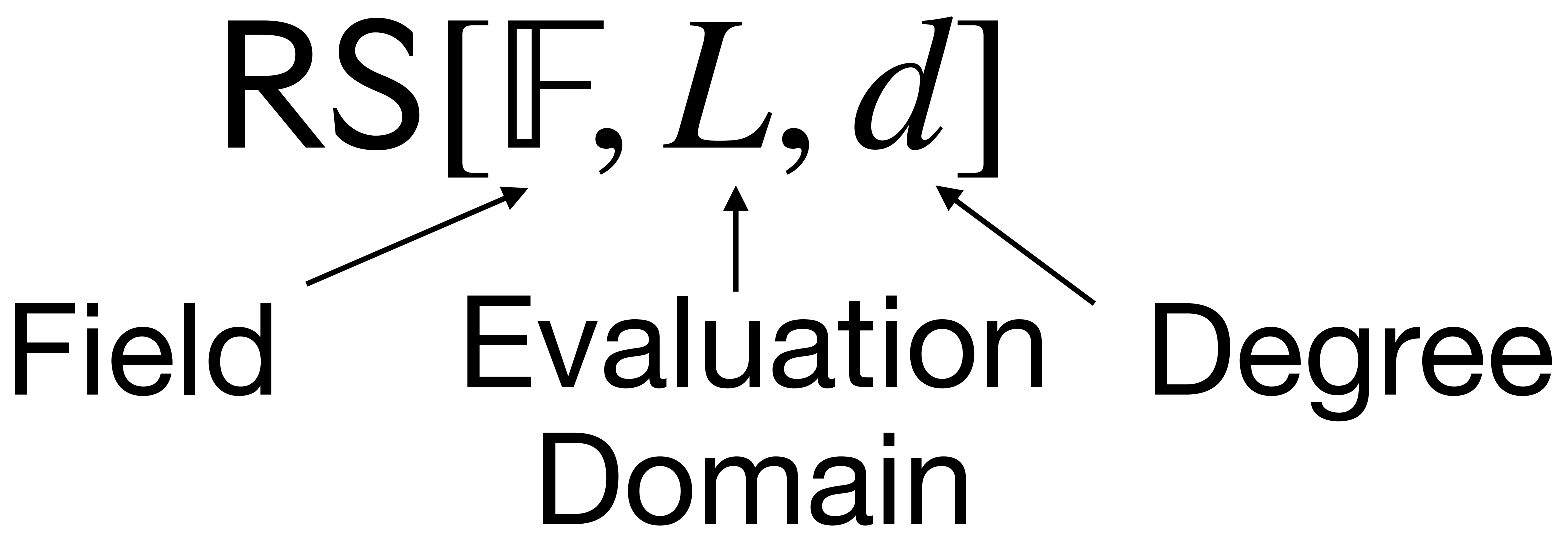
$$\mathsf{RS}[\mathbb{F}, L, d]$$

Field

RS Codes and IOPPs

$$\mathrm{RS}[\underset{\substack{\nearrow\\ \text{Field}}}{\mathbb{F}}, \underset{\substack{\uparrow\\ \text{Evaluation}\\ \text{Domain}}}{L}, d]$$

RS Codes and IOPPs



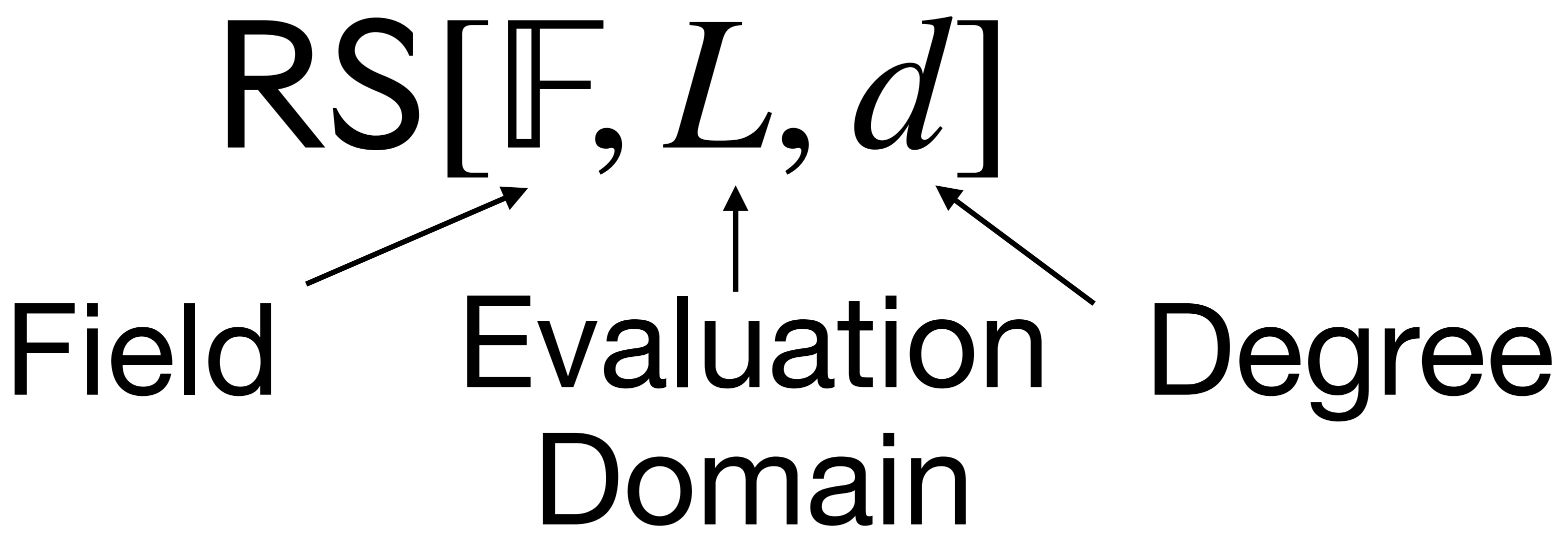
$$\mathrm{RS}[\mathbb{F}, L, d]$$

Field

Evaluation Domain

Degree

RS Codes and IOPPs



$$\mathsf{RS}[\mathbb{F}, L, d]$$

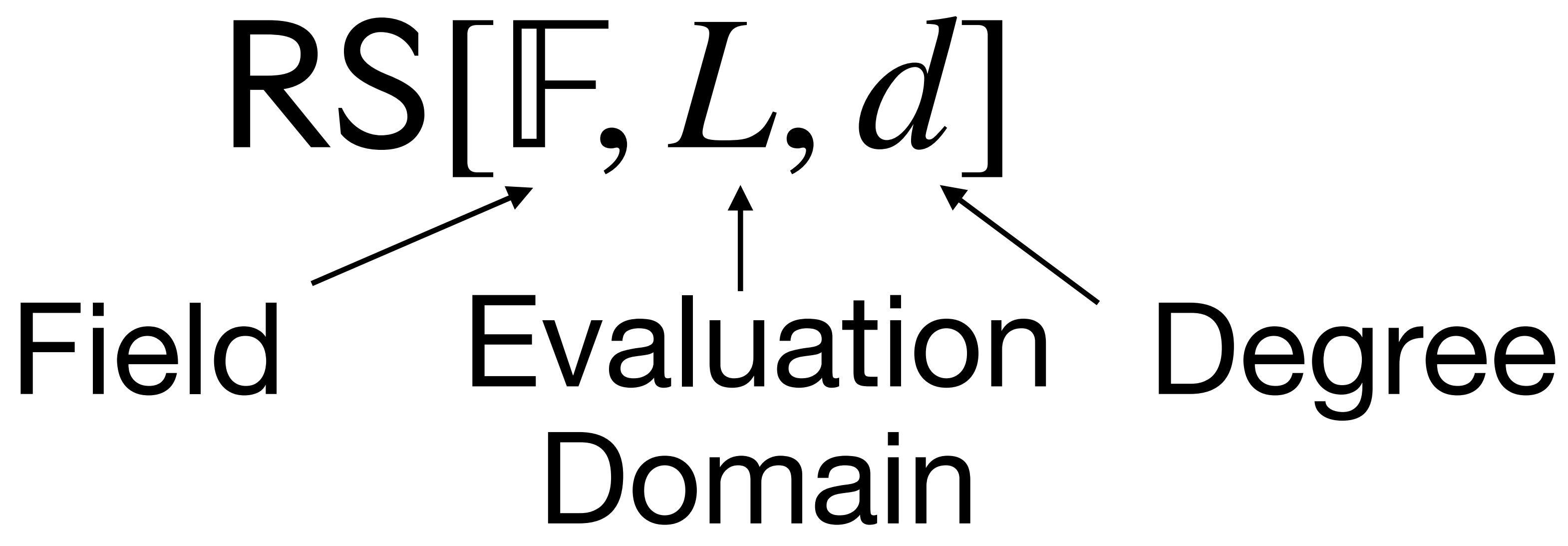
Field

Evaluation Domain

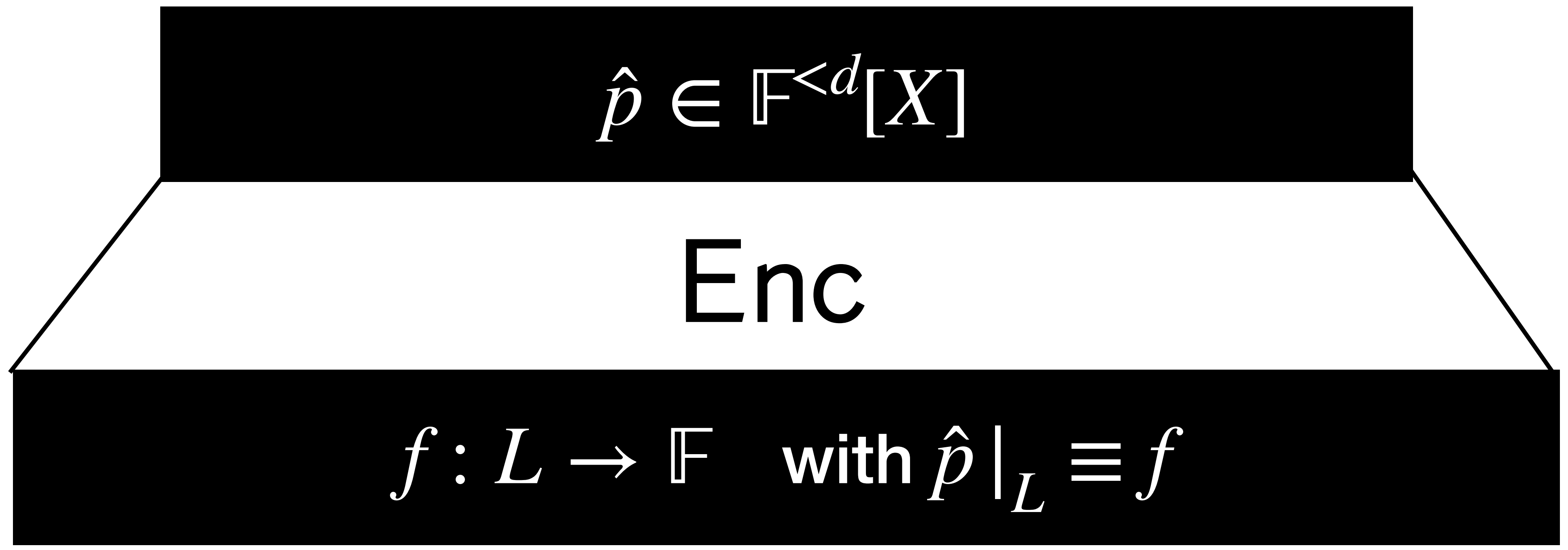
Degree

$\hat{p} \in \mathbb{F}^{<d}[X]$

RS Codes and IOPPs



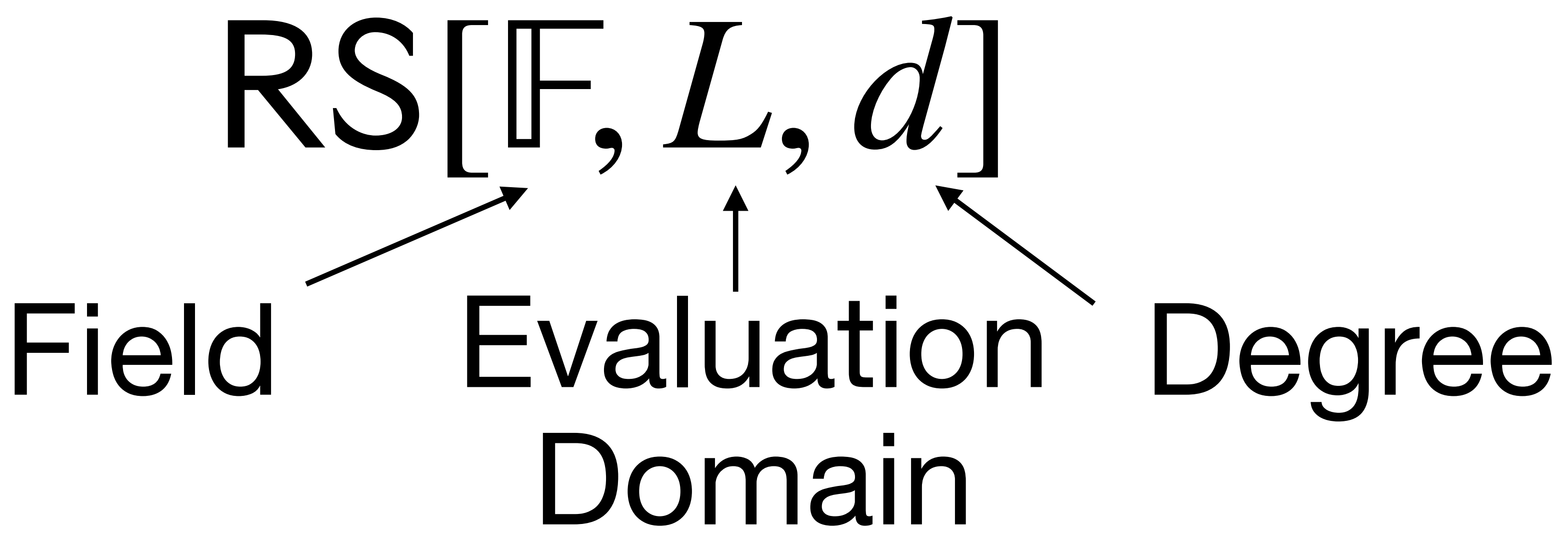
$\mathsf{RS}[\mathbb{F}, L, d]$



$\hat{p} \in \mathbb{F}^{<d}[X]$

$f : L \to \mathbb{F}$ with $\hat{p}\,|_L \equiv f$

RS Codes and IOPPs

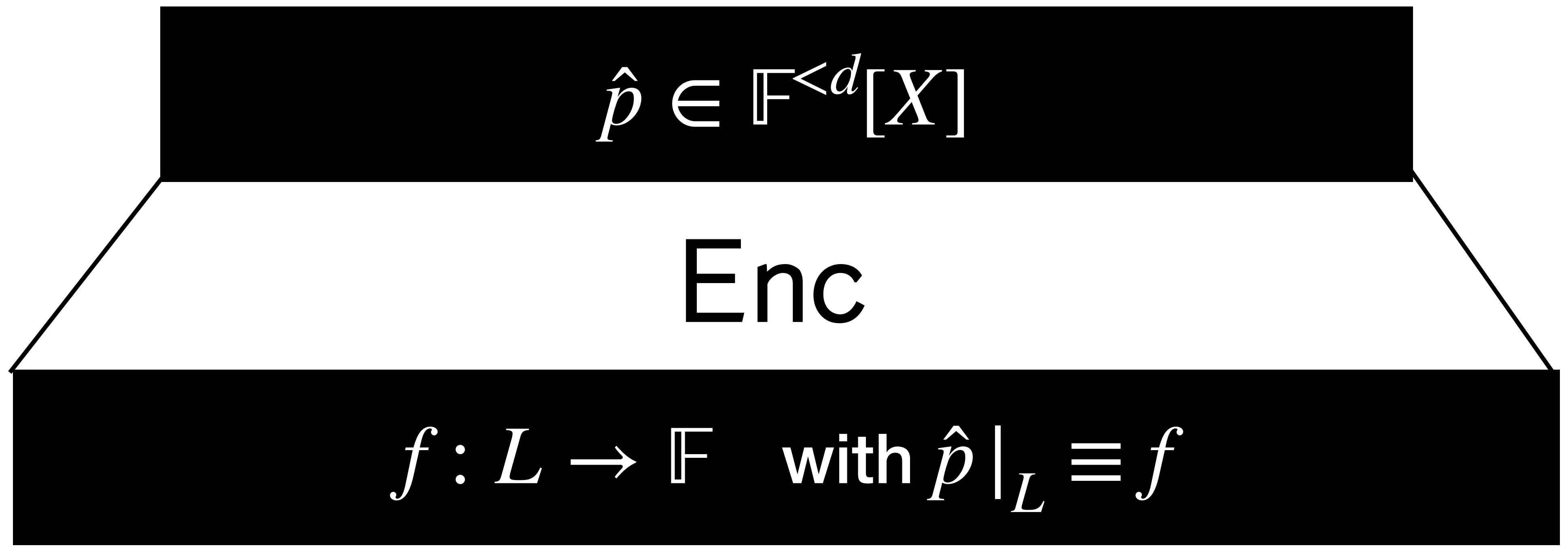


$\mathsf{RS}[\mathbb{F}, L, d]$

Field

Evaluation Domain

Degree



$\hat{p} \in \mathbb{F}^{<d}[X]$

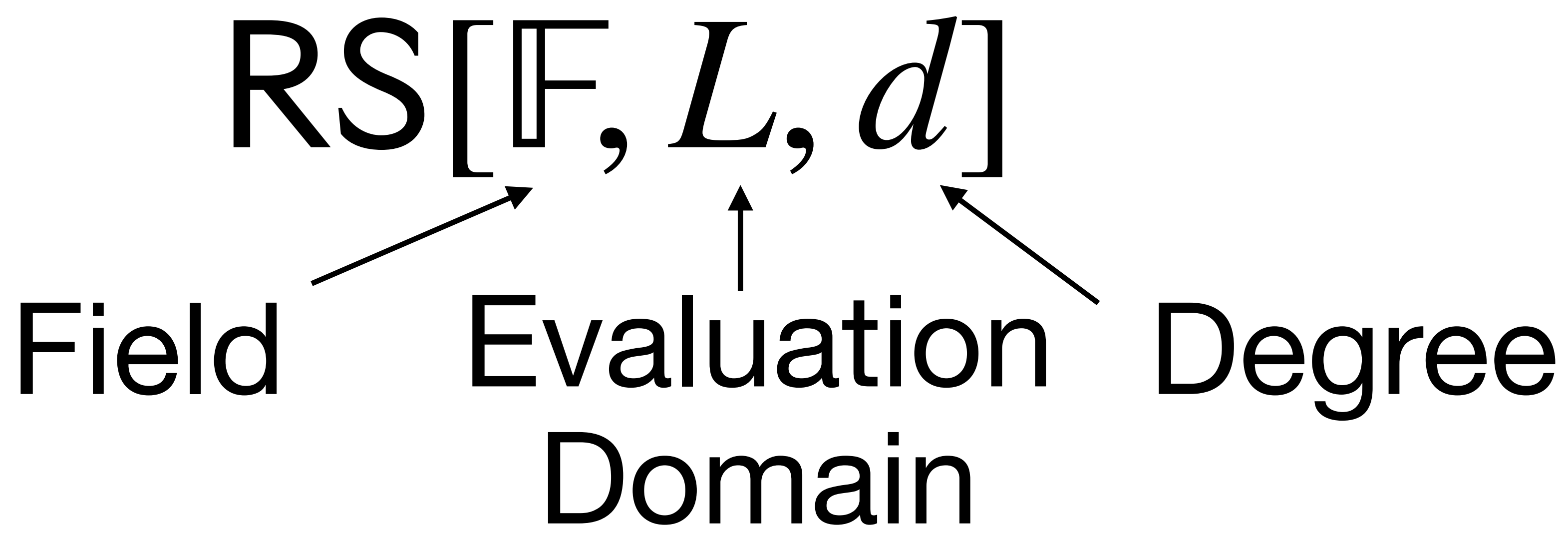
Enc

$f : L \to \mathbb{F}$ with $\hat{p}|_L \equiv f$

Rate: $\rho = d/|L|$, think $\rho = 1/4$

L smooth $\implies$ Enc is an FFT

RS Codes and IOPPs

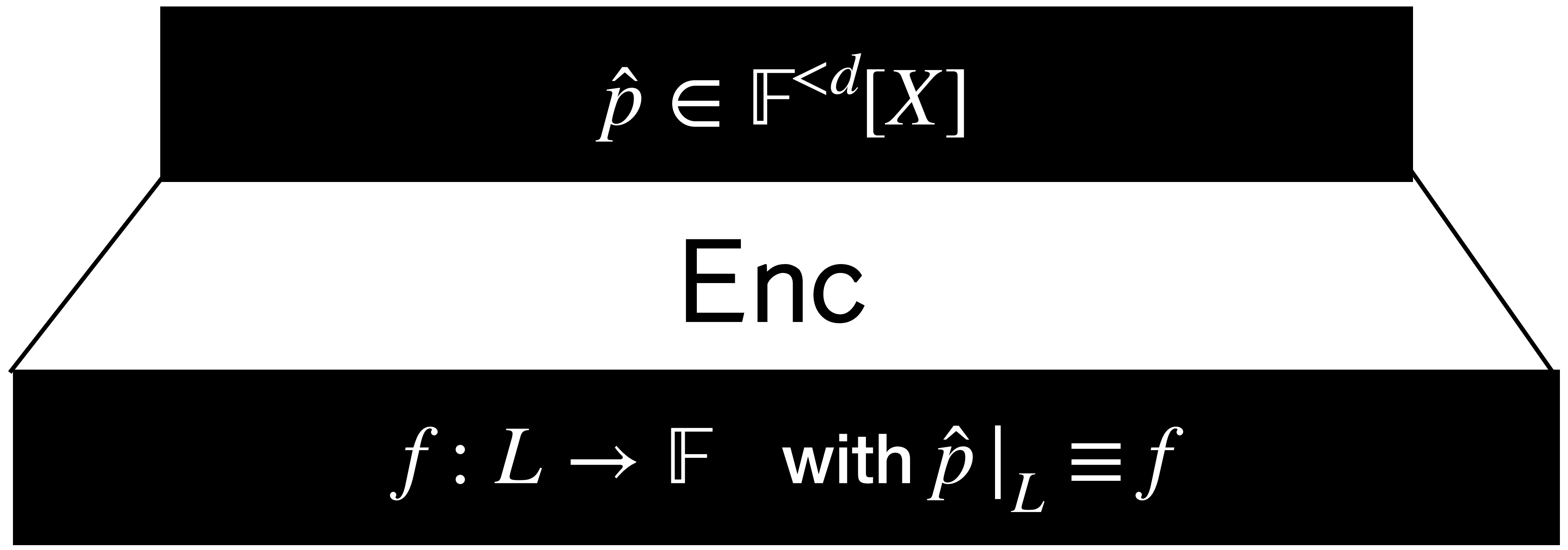


$\mathsf{RS}[\mathbb{F}, L, d]$

Field

Evaluation Domain

Degree



$\hat{p} \in \mathbb{F}^{<d}[X]$

Enc

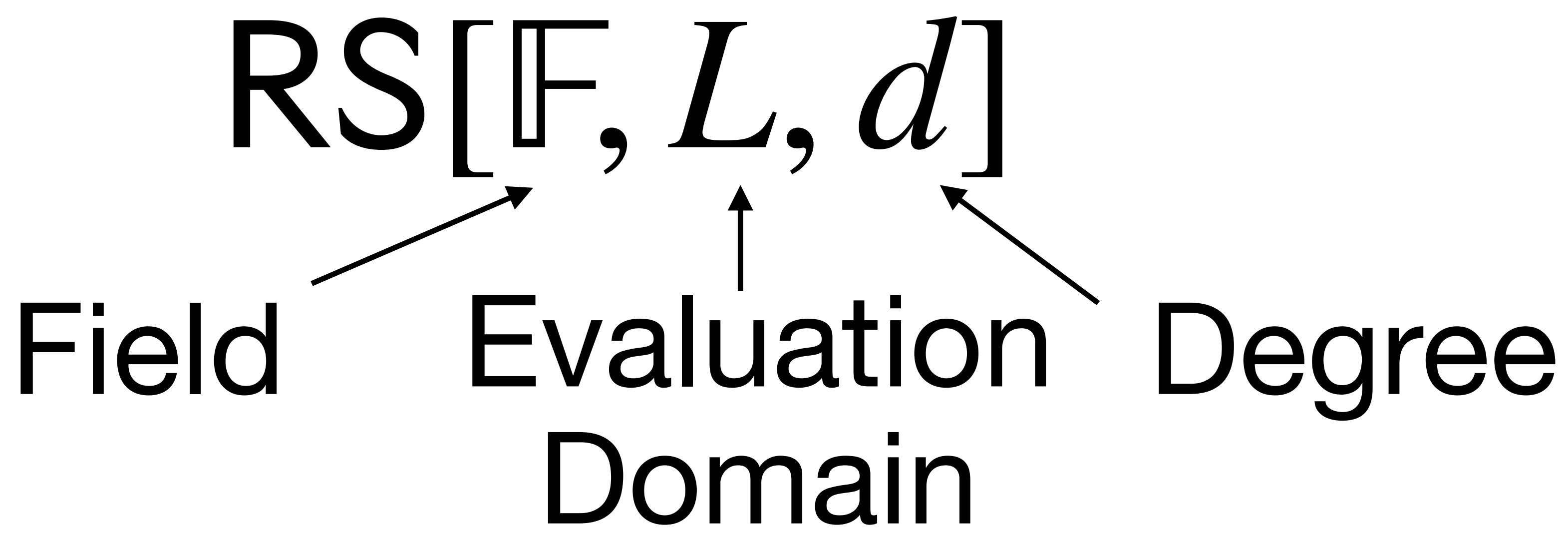
$f : L \to \mathbb{F}$ with $\hat{p}\,|_L \equiv f$

Rate: $\rho = d/|L|$, think $\rho = 1/4$

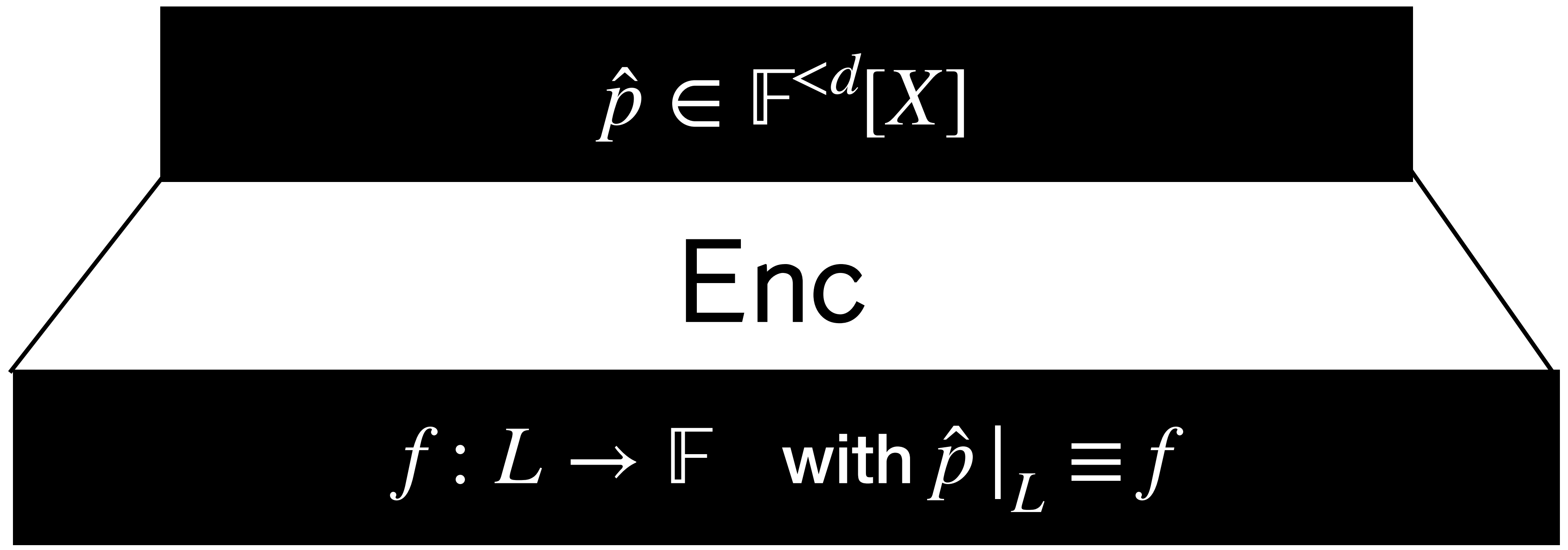
L smooth $\implies$ Enc is an FFT

IOPP for RS

RS Codes and IOPPs



$\mathsf{RS}[\mathbb{F}, L, d]$



$\hat{p} \in \mathbb{F}^{<d}[X]$

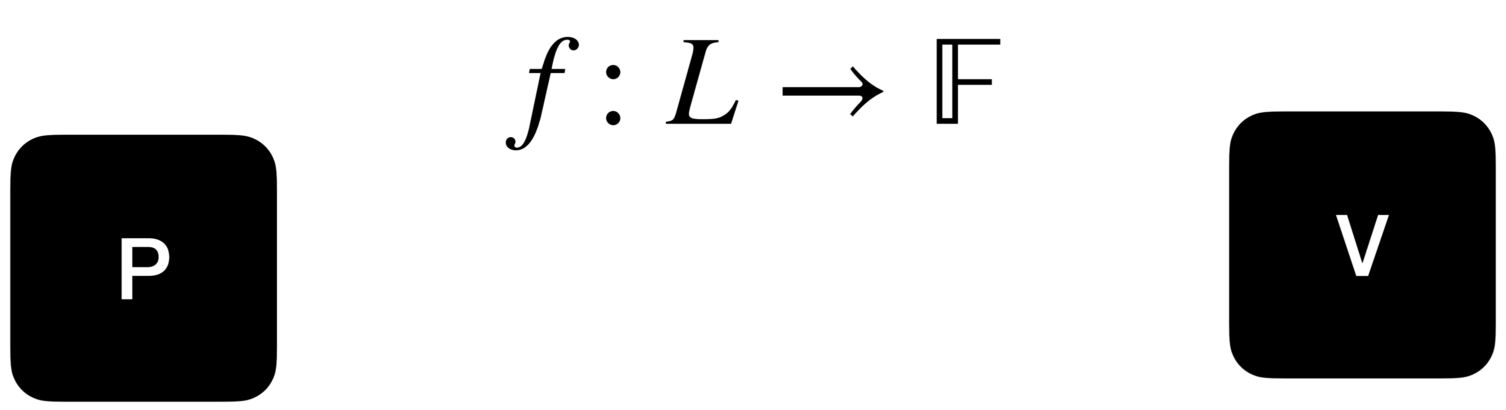
Enc

$f : L \to \mathbb{F}$ with $\hat{p}\,|_L \equiv f$

Rate: $\rho = d/|L|$, think $\rho = 1/4$

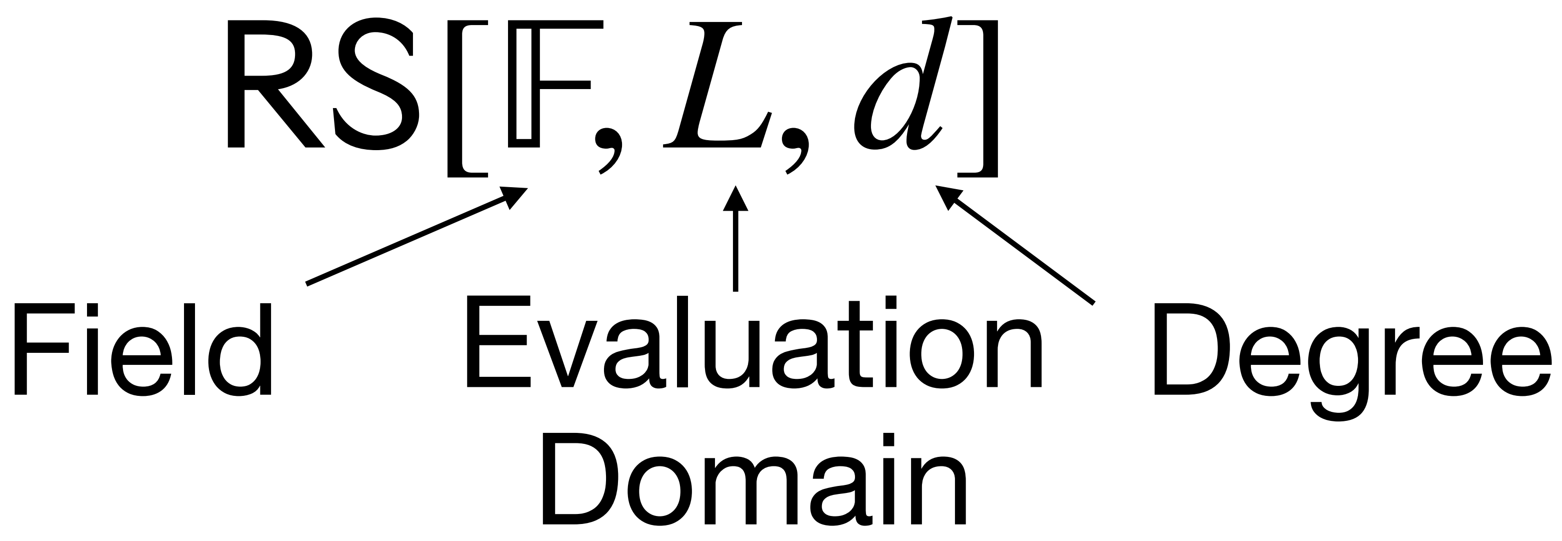
L smooth $\Longrightarrow$ Enc is an FFT

IOPP for RS



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

RS Codes and IOPPs

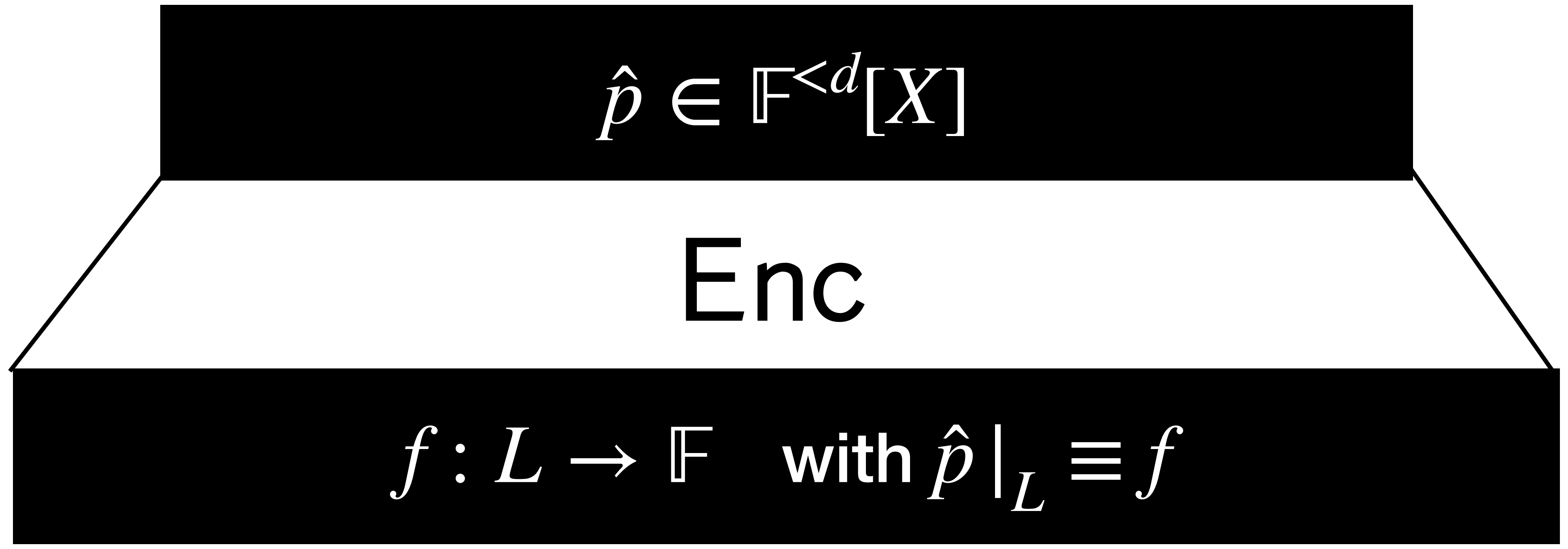


$\mathsf{RS}[\mathbb{F}, L, d]$

Field

Evaluation Domain

Degree



$\hat{p} \in \mathbb{F}^{<d}[X]$

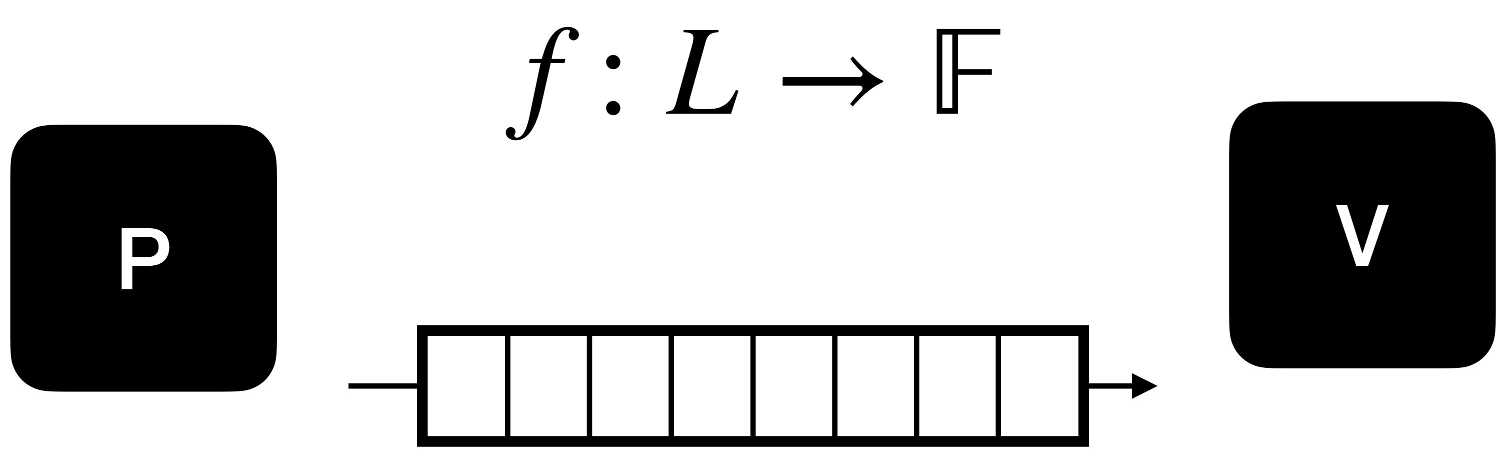
Enc

$f : L \to \mathbb{F}$ with $\hat{p}\,|_L \equiv f$

Rate: $\rho = d/|L|$, think $\rho = 1/4$

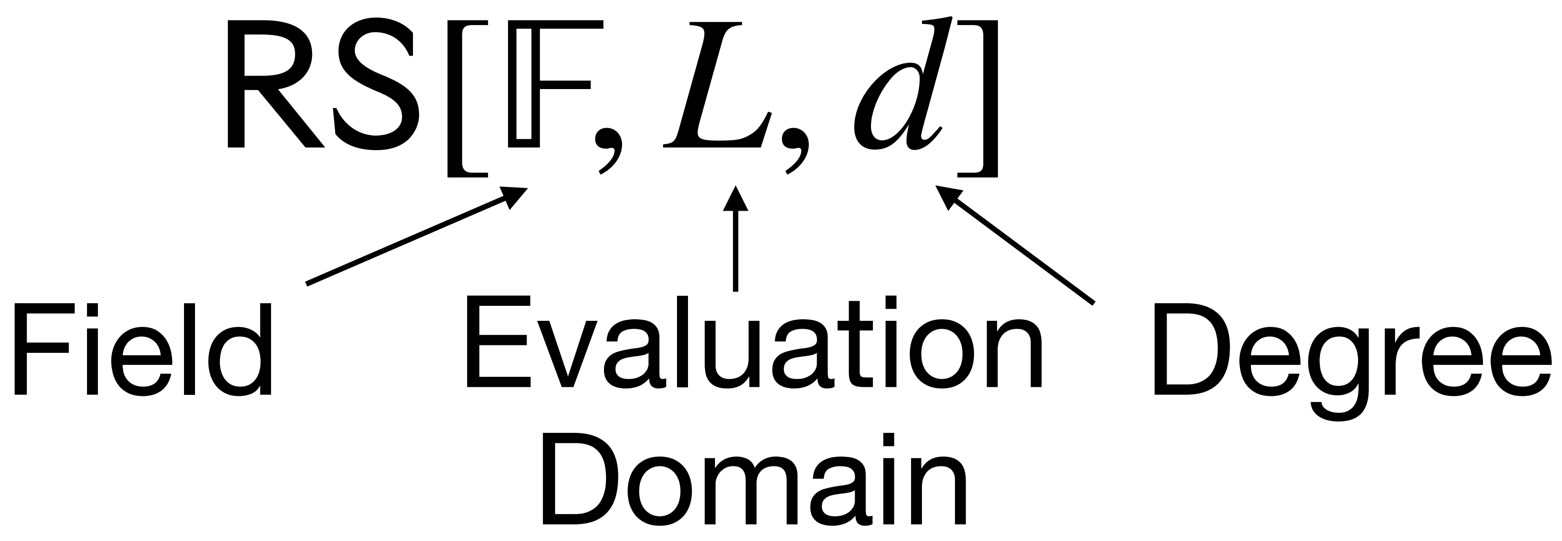
L smooth $\implies$ Enc is an FFT

IOPP for RS

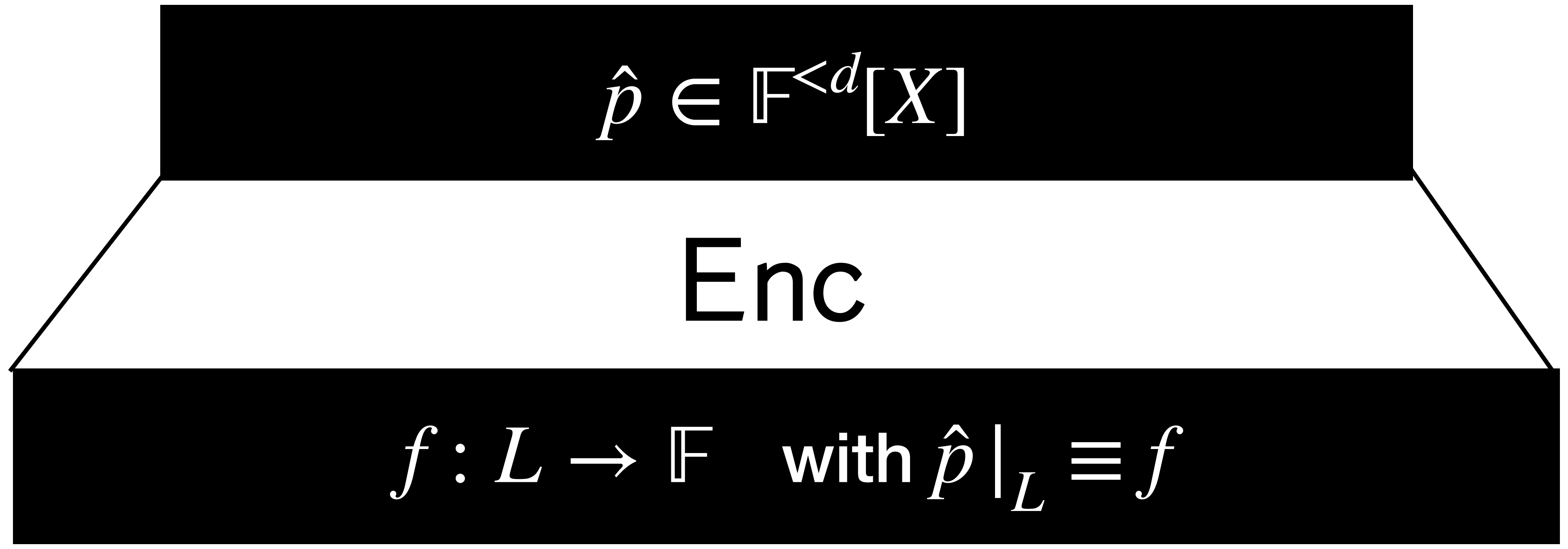


$f : L \to \mathbb{F}$

RS Codes and IOPPs



$\mathsf{RS}[\mathbb{F}, L, d]$



$\hat{p} \in \mathbb{F}^{<d}[X]$

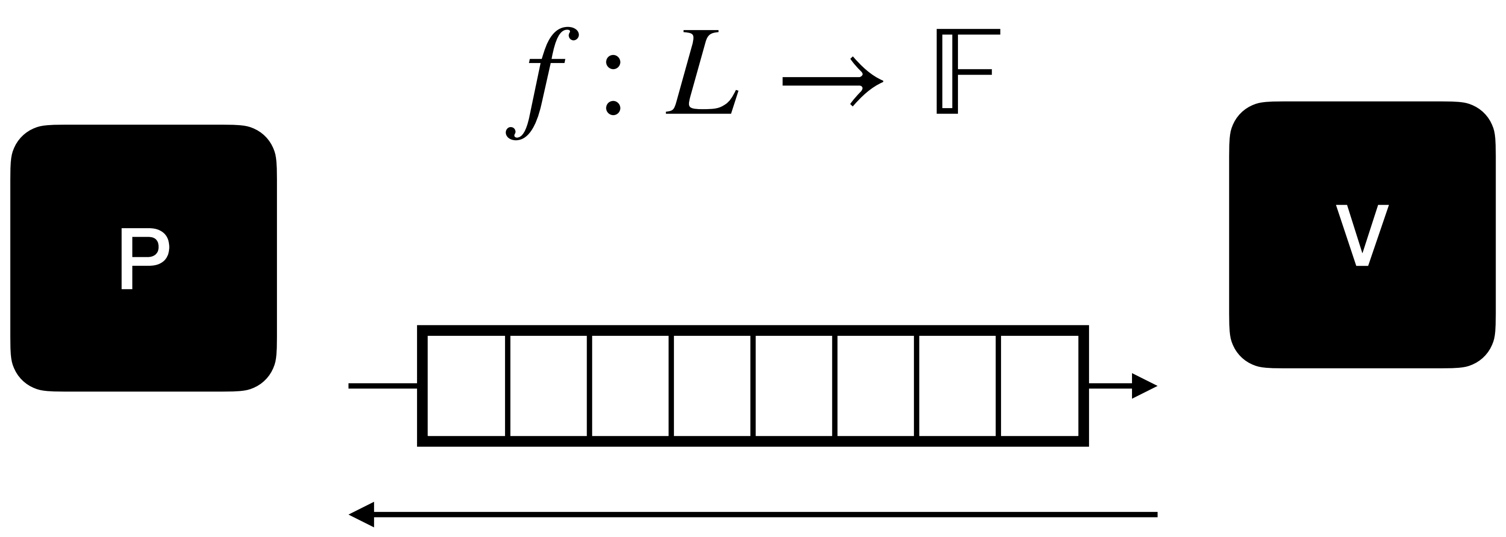
Enc

$f : L \to \mathbb{F}$ with $\hat{p}\,|_L \equiv f$

Rate: $\rho = d/|L|$, think $\rho = 1/4$

L smooth $\implies$ Enc is an FFT

IOPP for RS



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

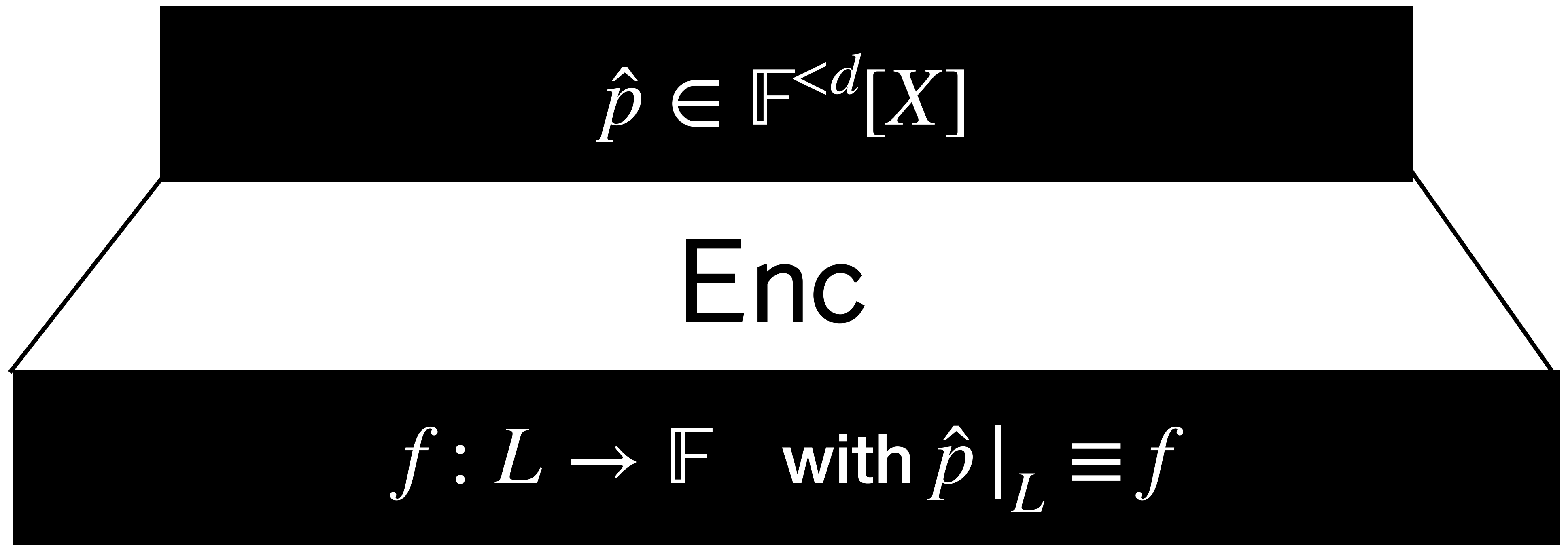
RS Codes and IOPPs

$$\mathrm{RS}[\mathbb{F}, L, d]$$

Field

Evaluation Domain

Degree



$\hat{p} \in \mathbb{F}^{<d}[X]$

Enc

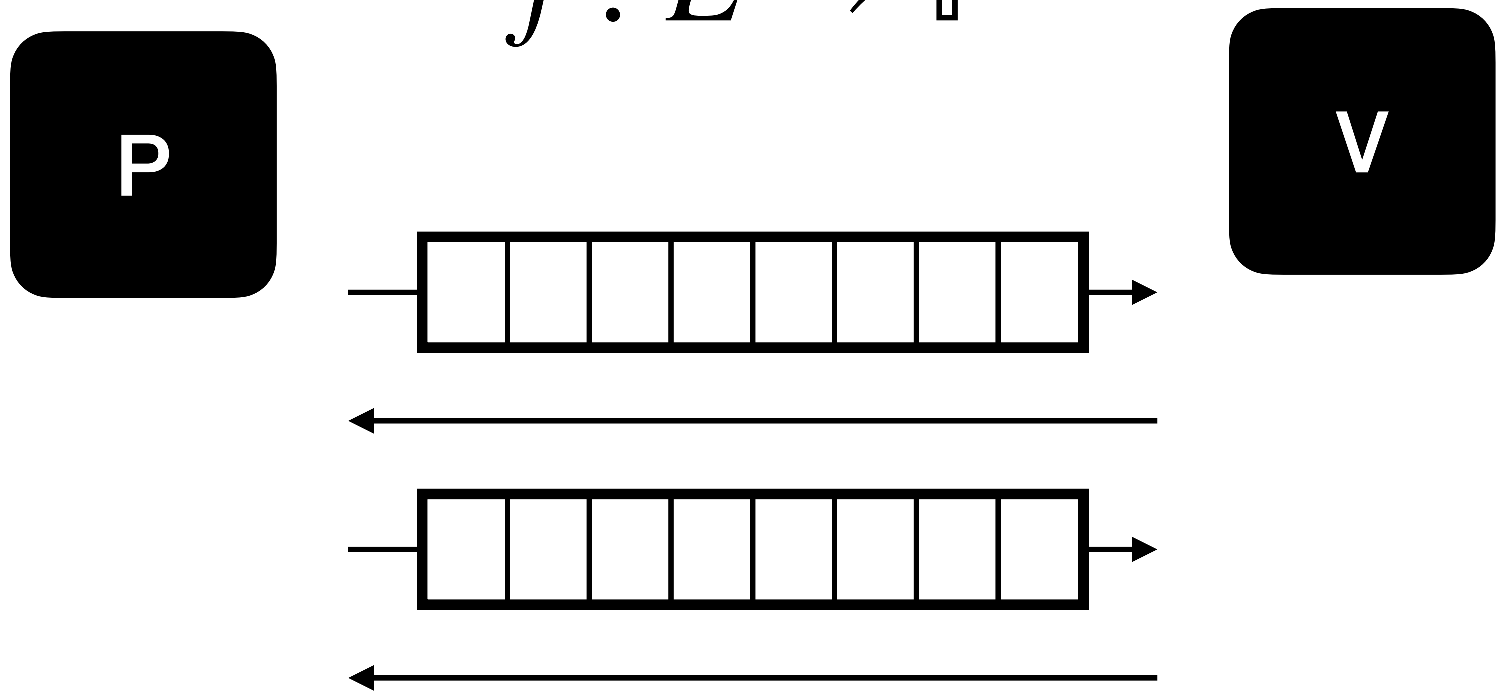
$f : L \to \mathbb{F}$ with $\hat{p}|_L \equiv f$

Rate: $\rho = d/|L|$, think $\rho = 1/4$

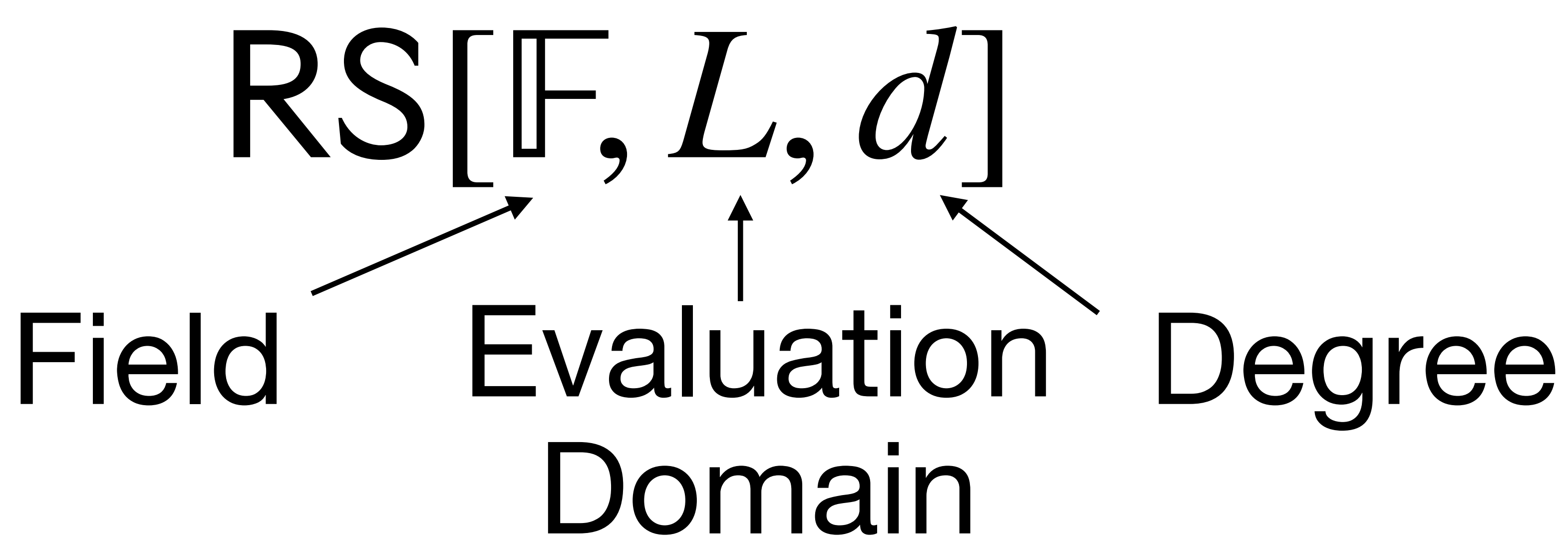
L smooth $\implies$ Enc is an FFT

IOPP for RS

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



RS Codes and IOPPs

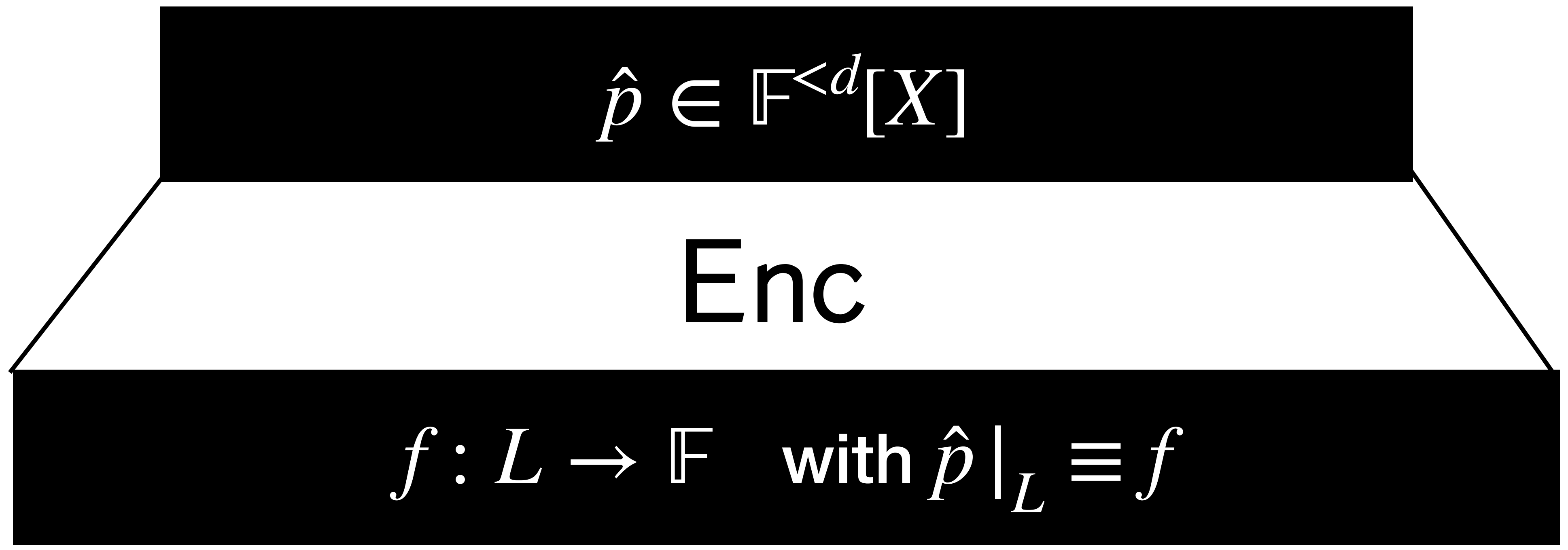


$\mathsf{RS}[\mathbb{F}, L, d]$

Field

Evaluation Domain

Degree



$\hat{p} \in \mathbb{F}^{<d}[X]$

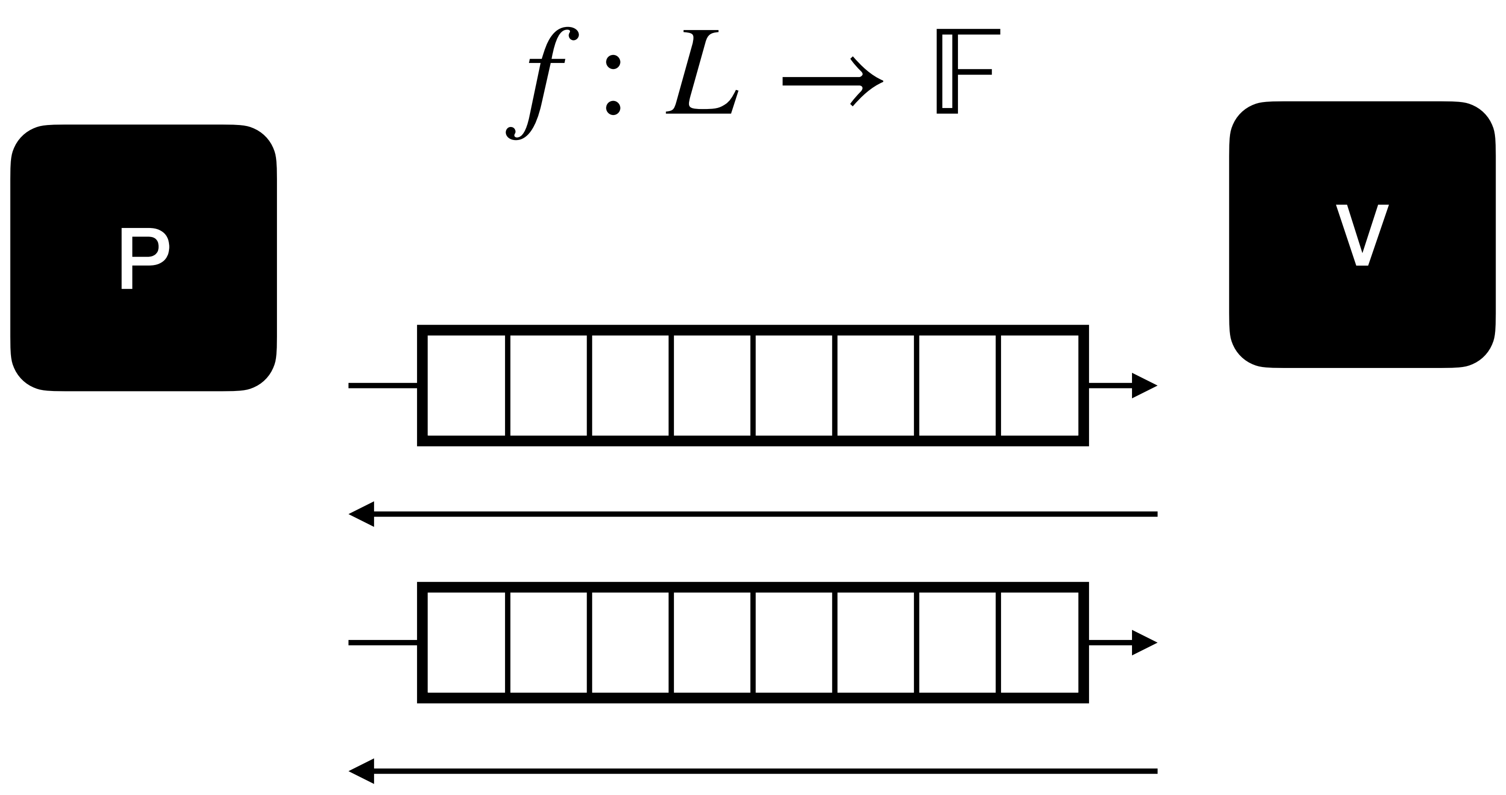
Enc

$f : L \to \mathbb{F}$ with $\hat{p}\,|_L \equiv f$

Rate: $\rho = d/|L|$, think $\rho = 1/4$

L smooth $\implies$ Enc is an FFT

IOPP for RS



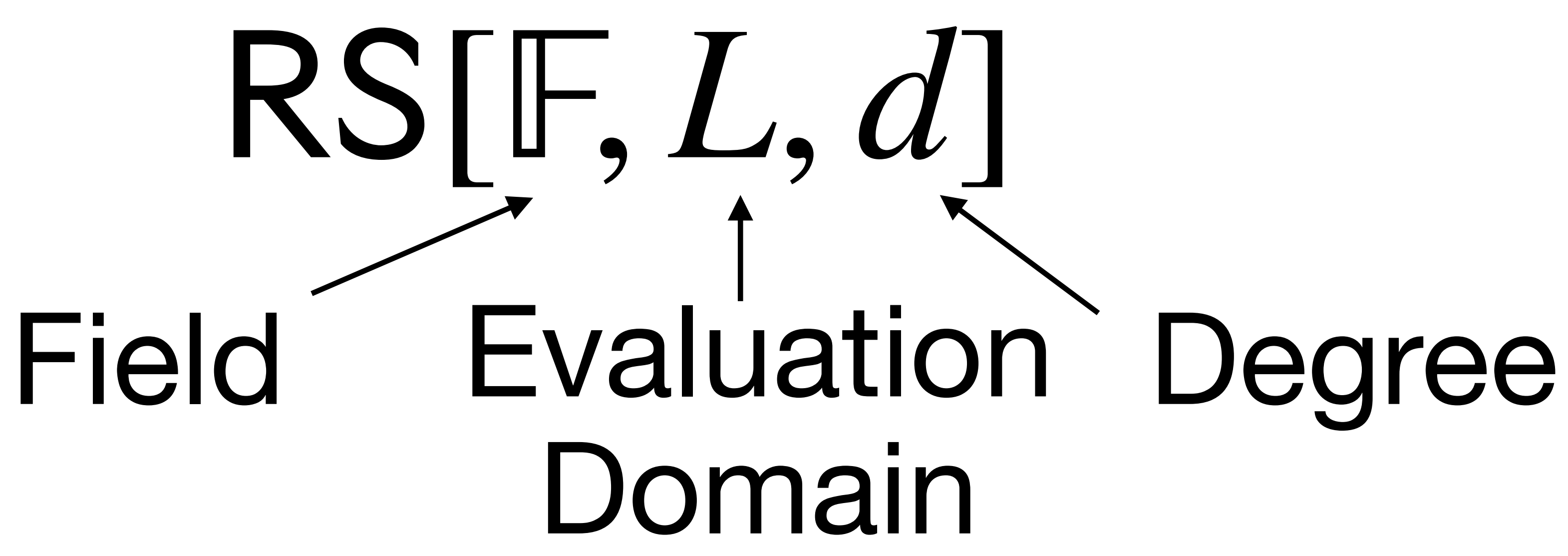
$f : L \to \mathbb{F}$

P

V

- $f \in \mathsf{RS}[\mathbb{F}, L, d] \implies \mathbf{V}$ accepts

RS Codes and IOPPs

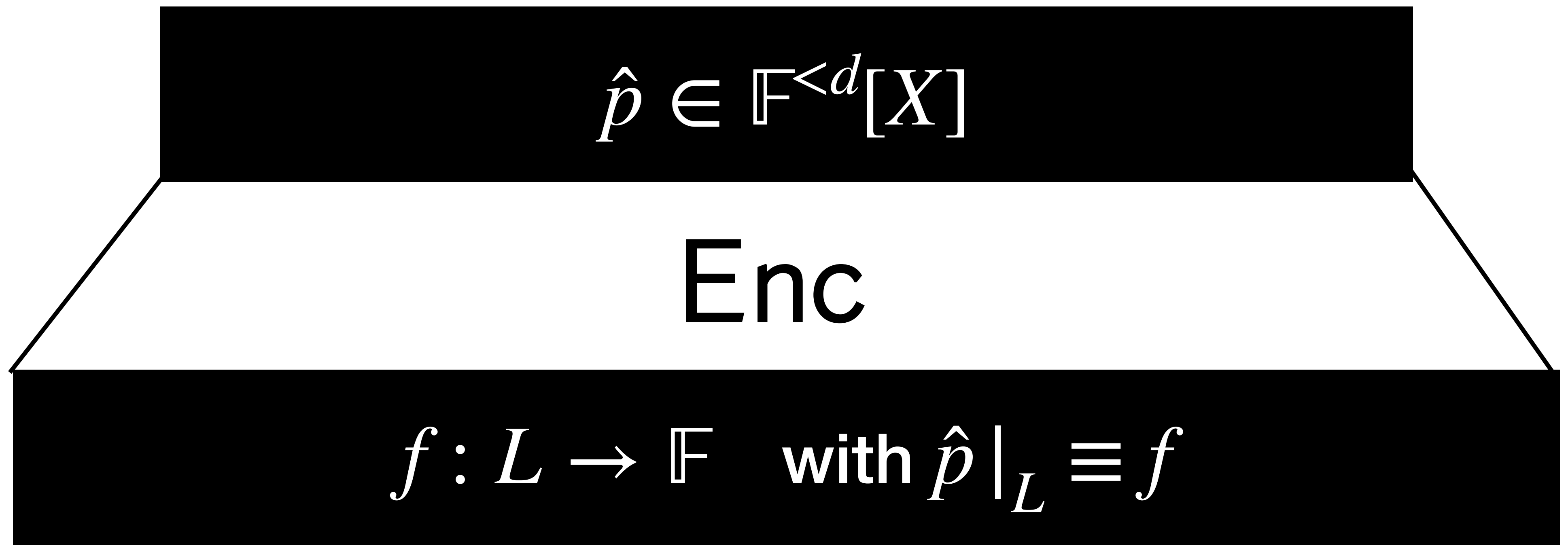


$\mathsf{RS}[\mathbb{F}, L, d]$

Field

Evaluation Domain

Degree



$\hat{p} \in \mathbb{F}^{<d}[X]$

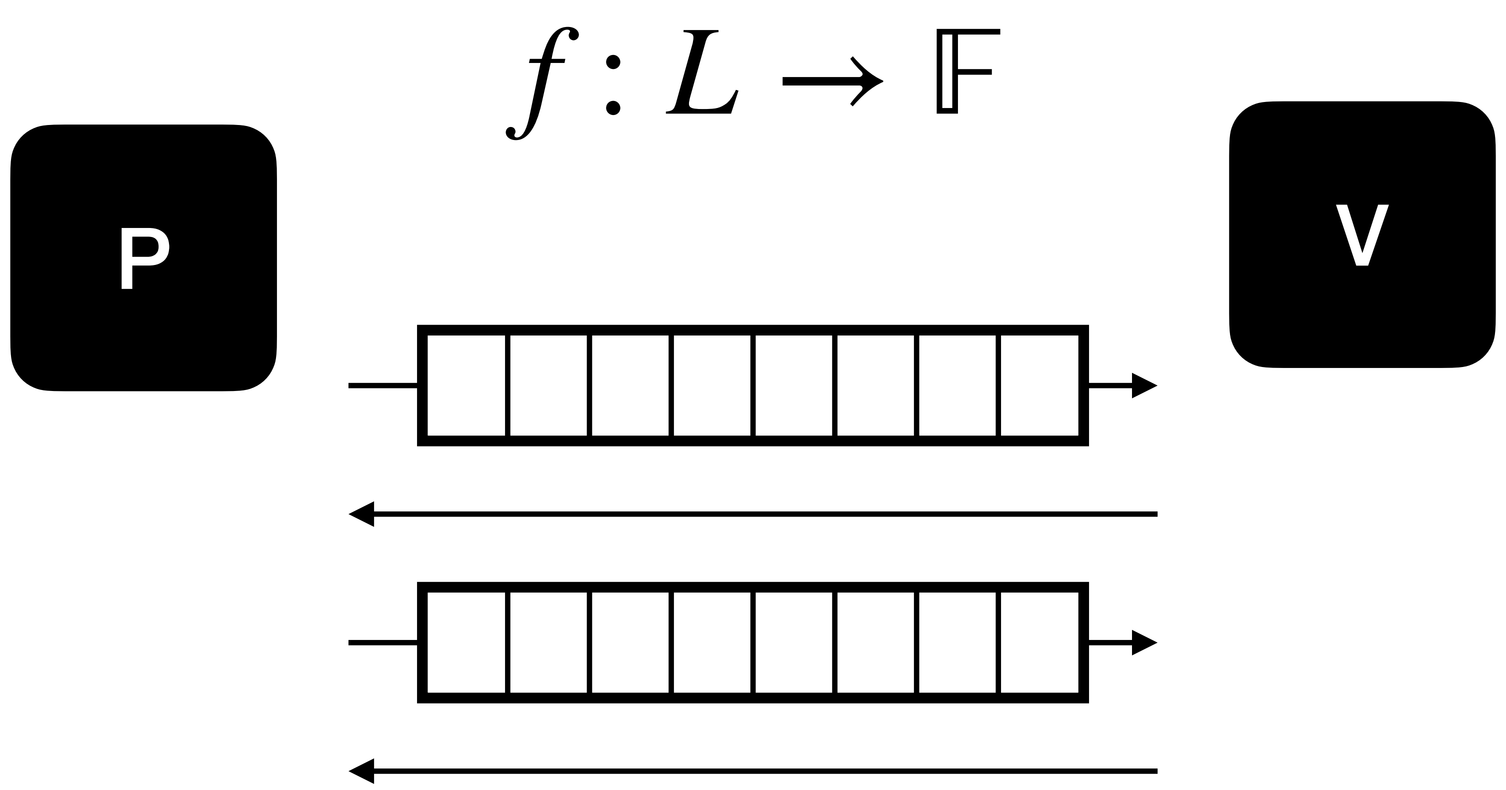
Enc

$f : L \to \mathbb{F}$ with $\hat{p}\,|_L \equiv f$

Rate: $\rho = d/|L|$, think $\rho = 1/4$

L smooth $\implies$ Enc is an FFT

IOPP for RS



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

- $f \in \mathsf{RS}[\mathbb{F}, L, d] \implies \mathbf{V}$ accepts
- f is δ-far from $\mathsf{RS}[\mathbb{F}, L, d] \implies \mathbf{V}$ rejects w.h.p.

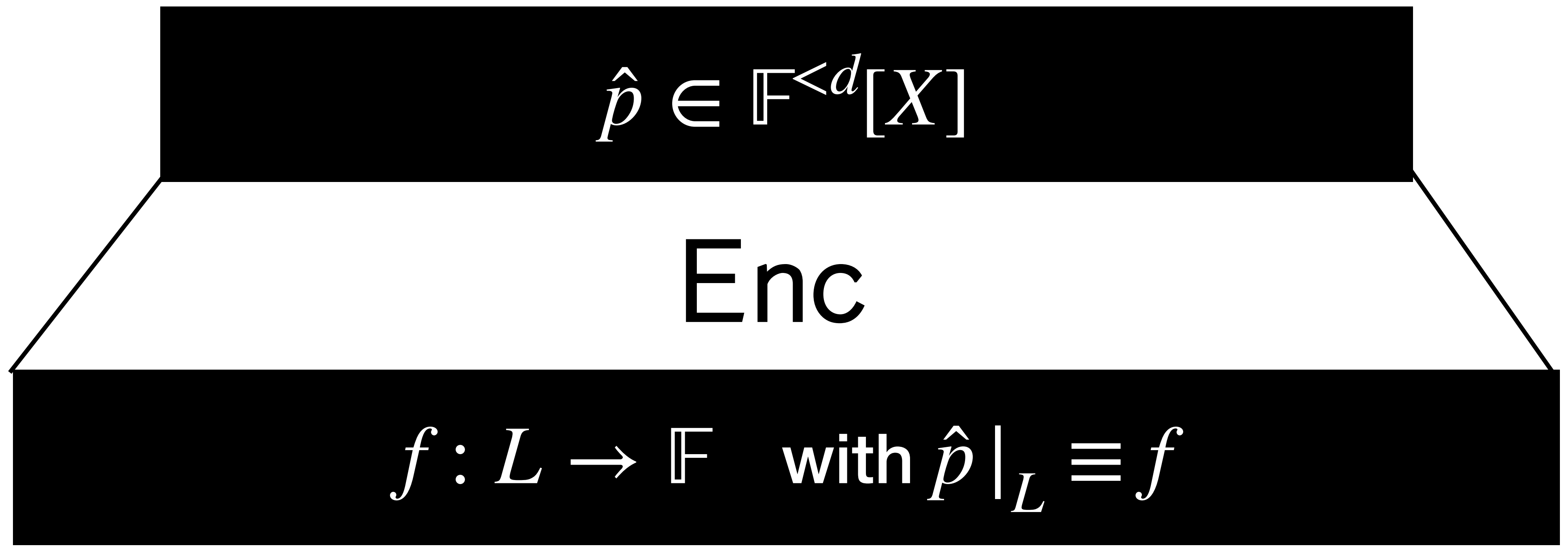
RS Codes and IOPPs

$\mathsf{RS}[\mathbb{F}, L, d]$

Field

Evaluation Domain

Degree

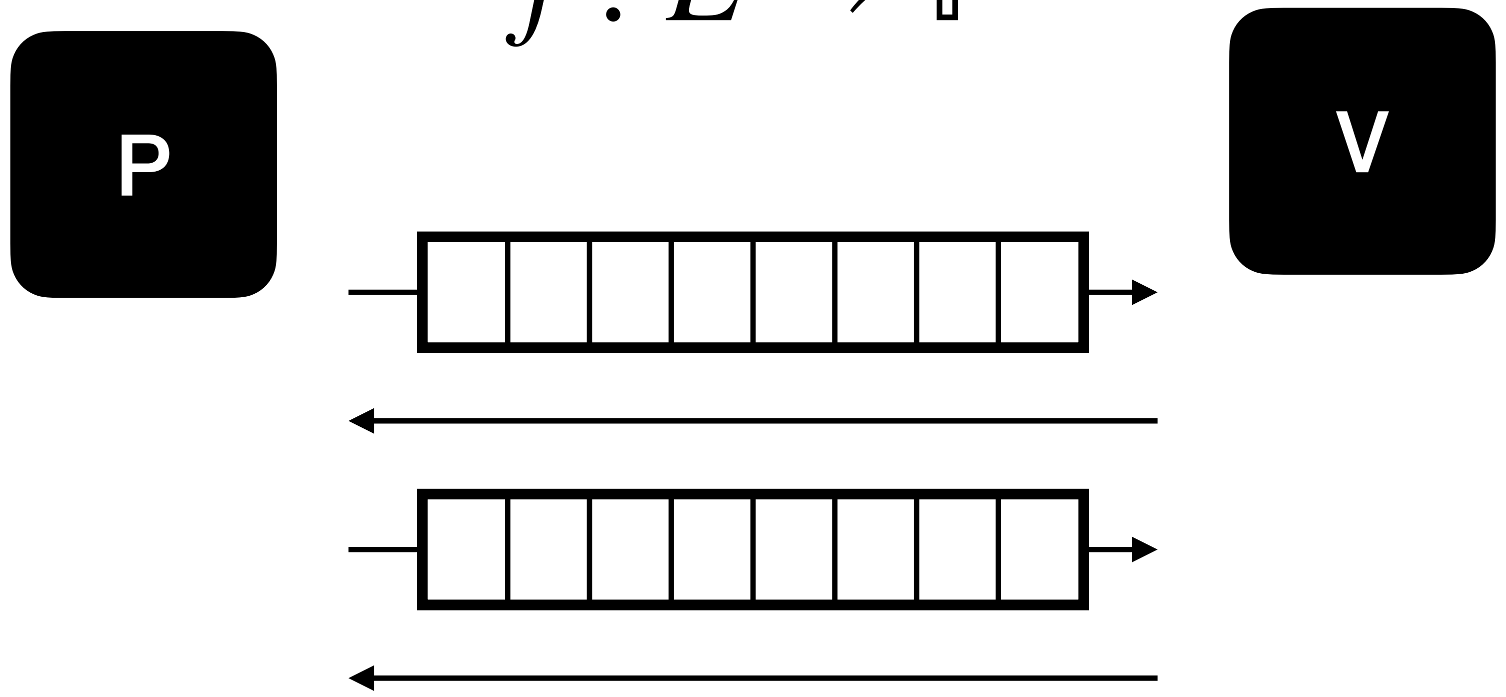

$\hat{p} \in \mathbb{F}^{<d}[X]$
Enc
$f : L \to \mathbb{F}$ with $\hat{p}\,|_L \equiv f$

Rate: $\rho = d/|L|$, think $\rho = 1/4$

L smooth $\implies$ Enc is an FFT

IOPP for RS

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



- $f \in \mathsf{RS}[\mathbb{F}, L, d] \implies$ **V** accepts
- f is δ-far from $\mathsf{RS}[\mathbb{F}, L, d] \implies$ **V** rejects w.h.p.

V makes “few queries” to f and proof oracles

Our results

STIR 🥣: An IOPP for RS

Rounds: $O(\log d)$

Proof length: $O(|L|)$

STIR 🥣: An IOPP for RS

Rounds: $O(\log d)$

Proof length: $O(|L|)$

Queries: $O\left(\lambda \cdot \log\left(\frac{\log d}{-\log\sqrt{\rho}}\right) + \log d\right)$ for $\delta = 1 - \sqrt{\rho}$

round-by-round

(To get λ-bits of security, **without conjecture**)

STIR 🥣: An IOPP for RS

Rounds: $O(\log d)$

Proof length: $O(|L|)$

Queries: $O\left(\lambda \cdot \log\left(\frac{\log d}{-\log\sqrt{\rho}}\right) + \log d\right)$ for $\delta = 1 - \sqrt{\rho}$

round-by-round

(To get λ-bits of security, **without conjecture**)

FRI: $O\left(\lambda \cdot \frac{\log d}{-\log\sqrt{\rho}}\right)$

Implementation

Implementation

- Rust 🦀 implementation, available at WizardOfMenlo/stir

Implementation

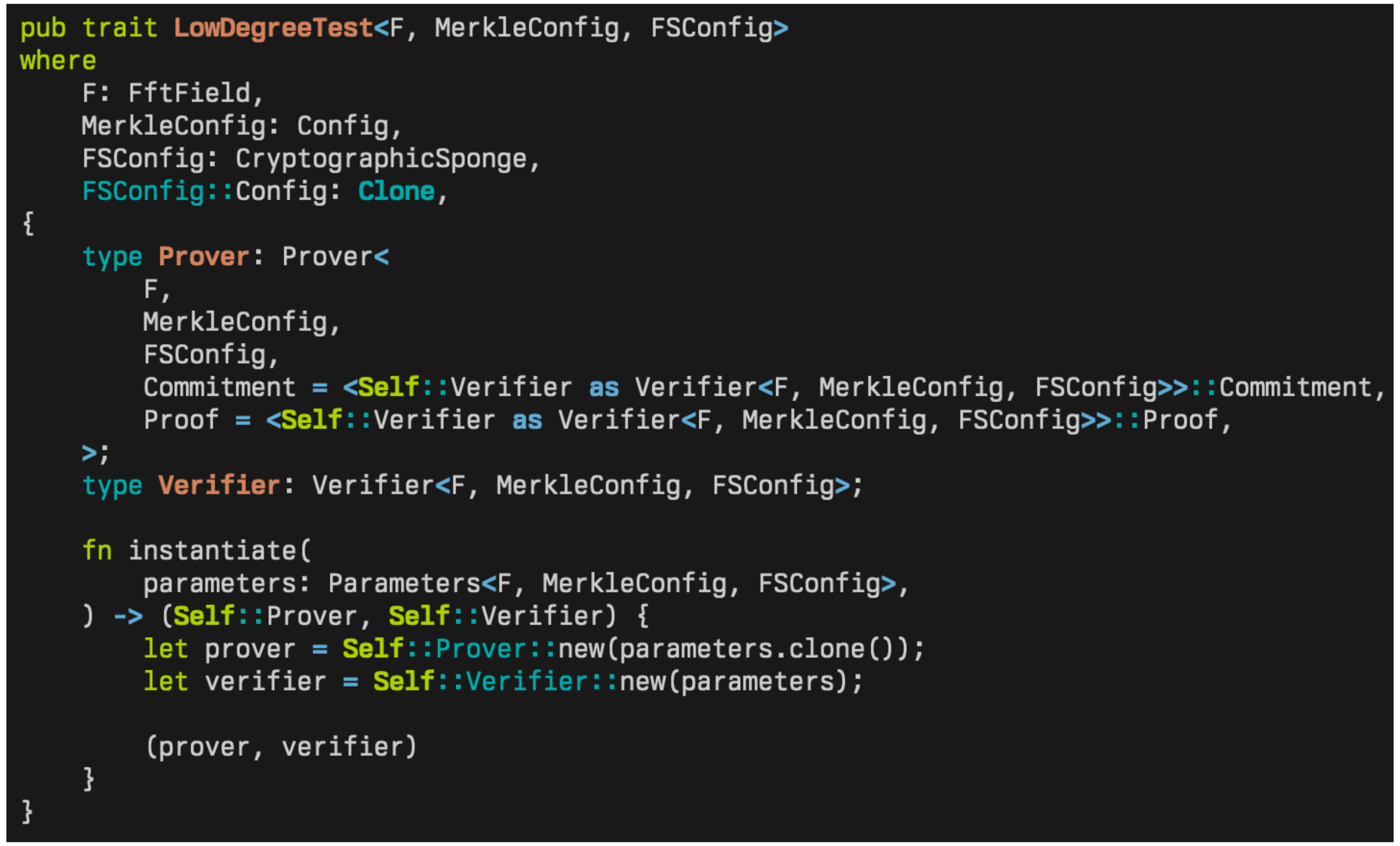
- Rust 🦀 implementation, available at WizardOfMenlo/stir
- Arkworks as backend, 192-bit field for benchmarks, reasonably optimized

Implementation

- Rust 🦀 implementation, available at WizardOfMenlo/stir
- Arkworks as backend, 192-bit field for benchmarks, reasonably optimized
- Implemented both FRI and STIR

Implementation

- Rust 🦀 implementation, available at WizardOfMenlo/stir
- Arkworks as backend, 192-bit field for benchmarks, reasonably optimized
- Implemented both FRI and STIR
- Decently well-written (for academia! 📚)



```
pub trait LowDegreeTest<F, MerkleConfig, FSConfig>
where
    F: FftField,
    MerkleConfig: Config,
    FSConfig: CryptographicSponge,
    FSConfig::Config: Clone,
{
    type Prover: Prover<
        F,
        MerkleConfig,
        FSConfig,
        Commitment = <Self::Verifier as Verifier<F, MerkleConfig, FSConfig>>::Commitment,
        Proof = <Self::Verifier as Verifier<F, MerkleConfig, FSConfig>>::Proof,
    >;
    type Verifier: Verifier<F, MerkleConfig, FSConfig>;

    fn instantiate(
        parameters: Parameters<F, MerkleConfig, FSConfig>,
    ) -> (Self::Prover, Self::Verifier) {
        let prover = Self::Prover::new(parameters.clone());
        let verifier = Self::Verifier::new(parameters);

        (prover, verifier)
    }
}
```

Comparison to FRI

Comparison to FRI

- **Drop-in** replacement of FRI

Comparison to FRI

FRI: $O\left(\lambda \cdot \frac{\log d}{-\log \sqrt{\rho}}\right)$

STIR: $O\left(\lambda \cdot \log\left(\frac{\log d}{-\log \sqrt{\rho}}\right) + \log d\right)$

- **Drop-in** replacement of FRI
- **Fewer** queries leads to:

Comparison to FRI

FRI: $O\left(\lambda \cdot \frac{\log d}{-\log \sqrt{\rho}}\right)$

STIR: $O\left(\lambda \cdot \log\left(\frac{\log d}{-\log \sqrt{\rho}}\right) + \log d\right)$

- **Drop-in** replacement of FRI
- **Fewer** queries leads to:
  - Fewer authentication paths $\Longrightarrow$ **smaller** argument size

Comparison to FRI

FRI: $O\left(\lambda \cdot \frac{\log d}{-\log \sqrt{\rho}}\right)$

STIR: $O\left(\lambda \cdot \log\left(\frac{\log d}{-\log \sqrt{\rho}}\right) + \log d\right)$

- **Drop-in** replacement of FRI
- **Fewer** queries leads to:
  - Fewer authentication paths $\Longrightarrow$ **smaller** argument size
  - Smaller verifier hash-complexity $\Longrightarrow$ **faster** and more **recursion-friendly**!

Comparison to FRI

FRI: $O\left(\lambda \cdot \frac{\log d}{-\log \sqrt{\rho}}\right)$

STIR: $O\left(\lambda \cdot \log\left(\frac{\log d}{-\log \sqrt{\rho}}\right) + \log d\right)$

- **Drop-in** replacement of FRI
- **Fewer** queries leads to:
  - Fewer authentication paths $\Longrightarrow$ **smaller** argument size
  - Smaller verifier hash-complexity $\Longrightarrow$ **faster** and more **recursion-friendly**!
- Rough query comparison:

Comparison to FRI

FRI: $O\left(\lambda \cdot \frac{\log d}{-\log \sqrt{\rho}}\right)$

STIR: $O\left(\lambda \cdot \log\left(\frac{\log d}{-\log \sqrt{\rho}}\right) + \log d\right)$

- **Drop-in** replacement of FRI
- **Fewer** queries leads to:
  - Fewer authentication paths $\implies$ **smaller** argument size
  - Smaller verifier hash-complexity $\implies$ **faster** and more **recursion-friendly**!
- Rough query comparison:
  - Example: $d = 2^{20}, \rho = 1/4$, targeting 100-bits of security

Comparison to FRI

FRI: $O\left(\lambda \cdot \frac{\log d}{-\log \sqrt{\rho}}\right)$

STIR: $O\left(\lambda \cdot \log\left(\frac{\log d}{-\log \sqrt{\rho}}\right) + \log d\right)$

- **Drop-in** replacement of FRI
- **Fewer** queries leads to:
  - Fewer authentication paths $\Longrightarrow$ **smaller** argument size
  - Smaller verifier hash-complexity $\Longrightarrow$ **faster** and more **recursion-friendly**!
- Rough query comparison:
  - Example: $d = 2^{20}$, $\rho = 1/4$, targeting 100-bits of security

can increase by PoW

Comparison to FRI

FRI: $O\left(\lambda \cdot \frac{\log d}{-\log\sqrt{\rho}}\right)$

STIR: $O\left(\lambda \cdot \log\left(\frac{\log d}{-\log\sqrt{\rho}}\right) + \log d\right)$

- **Drop-in** replacement of FRI
- **Fewer** queries leads to:
  - Fewer authentication paths $\Longrightarrow$ **smaller** argument size
  - Smaller verifier hash-complexity $\Longrightarrow$ **faster** and more **recursion-friendly**!
- Rough query comparison:
  - Example: $d = 2^{20}$, $\rho = 1/4$, targeting 100-bits of security
  - FRI: ~ **400** queries vs STIR: ~ **200** queries

can increase by PoW

Comparison to FRI

FRI: $O\left(\lambda \cdot \frac{\log d}{-\log \sqrt{\rho}}\right)$

STIR: $O\left(\lambda \cdot \log\left(\frac{\log d}{-\log \sqrt{\rho}}\right) + \log d\right)$

- **Drop-in** replacement of FRI
- **Fewer** queries leads to:
  - Fewer authentication paths $\Longrightarrow$ **smaller** argument size
  - Smaller verifier hash-complexity $\Longrightarrow$ **faster** and more **recursion-friendly**!
- Rough query comparison:
  - Example: $d = 2^{20}$, $\rho = 1/4$, targeting 100-bits of security
  - FRI: ~ **400** queries vs STIR: ~ **200** queries

can increase by PoW

- Similar prover runtime (bottleneck is initial function evaluation)

Comparison to FRI

Comparison to FRI

- Better **argument size** and **verifier hash complexity** across **all params**!
- Larger improvements when degree and rate increase

Comparison to FRI

Assuming conjecture

128 bits of security, 22 by PoW

- Better **argument size** and **verifier hash complexity** across **all params**!
- Larger improvements when degree and rate increase

$d = 2^{24}, \rho = 1/4$	**FRI**	**STIR**
Size (KiB)	177	107
Hashes	3.5k	1.8k

Comparison to FRI

Assuming conjecture

128 bits of security, 22 by PoW

- Better **argument size** and **verifier hash complexity** across **all params**!
- Larger improvements when degree and rate increase

$d = 2^{24}, \rho = 1/4$	FRI	STIR
Size (KiB)	177	107
Hashes	3.5k	1.8k

$d = 2^{30}, \rho = 1/2$	FRI	STIR
Size (KiB)	494	200
Hashes	10k	3.8k

Comparison to FRI

Assuming conjecture

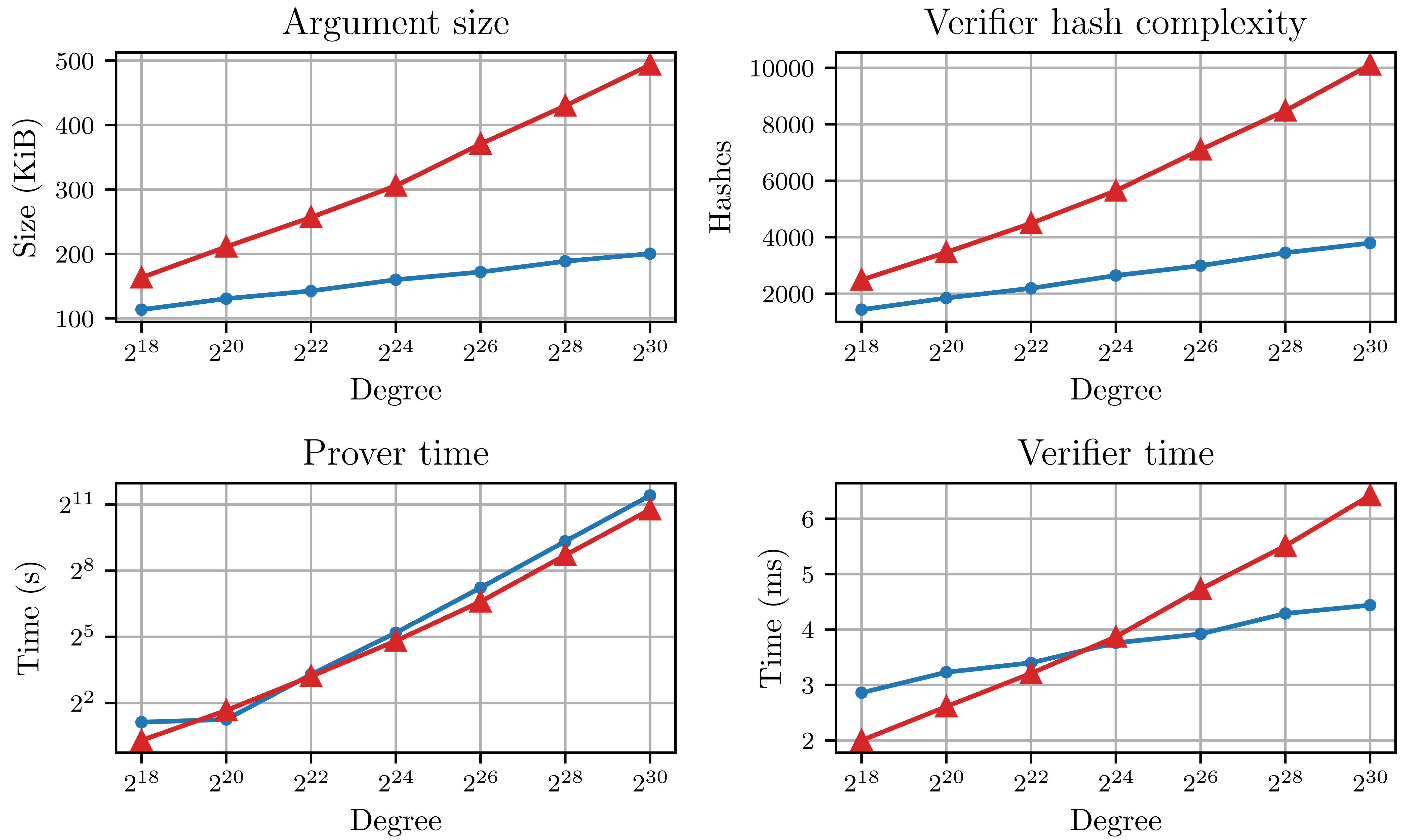
128 bits of security, 22 by PoW

- Better **argument size** and **verifier hash complexity** across **all params**!
- Larger improvements when degree and rate increase

$d = 2^{24}, \rho = 1/4$	**FRI**	**STIR**
Size (KiB)	177	107
Hashes	3.5k	1.8k

$d = 2^{30}, \rho = 1/2$	**FRI**	**STIR**
Size (KiB)	494	200
Hashes	10k	3.8k

$\rho = 1/2$



What about the conjecture?

FRI and STIR benefit in roughly the same way

- Conjecture on list-decoding up to distance $1-\rho$ (instead of $1-\sqrt{\rho}$)
- STIR queries: $$O\left(\lambda \cdot \log\left(\frac{\log d}{-\log \rho}\right) + \log d\right)$$
- FRI queries: $$O\left(\lambda \cdot \frac{\log d}{-\log \rho}\right)$$

In both, for $\delta = 1-\rho$,

reduces queries by ~2x

Techniques

Folding

Reduce $\mathsf{RS}[\mathbb{F}, L, d]$ **to** $\mathsf{RS}[\mathbb{F}, L^k, d/k]$

Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

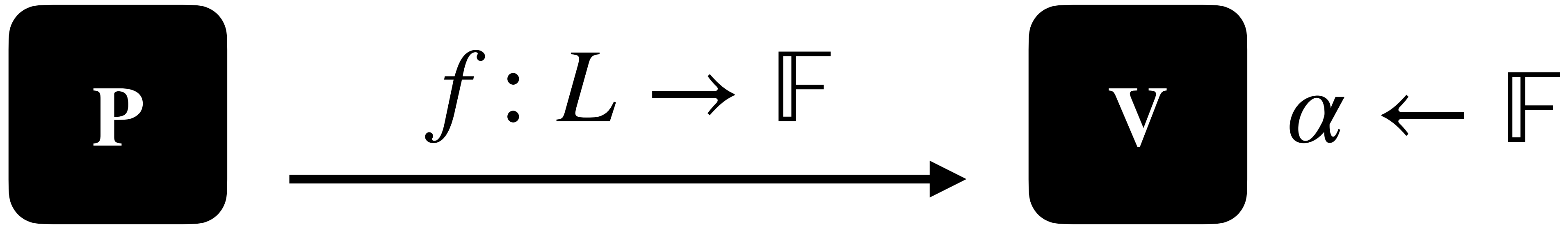
Reduce $\mathsf{RS}[\mathbb{F}, L, d]$ **to** $\mathsf{RS}[\mathbb{F}, L^k, d/k]$

Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

Reduce $\mathsf{RS}[\mathbb{F}, L, d]$ **to** $\mathsf{RS}[\mathbb{F}, L^k, d/k]$

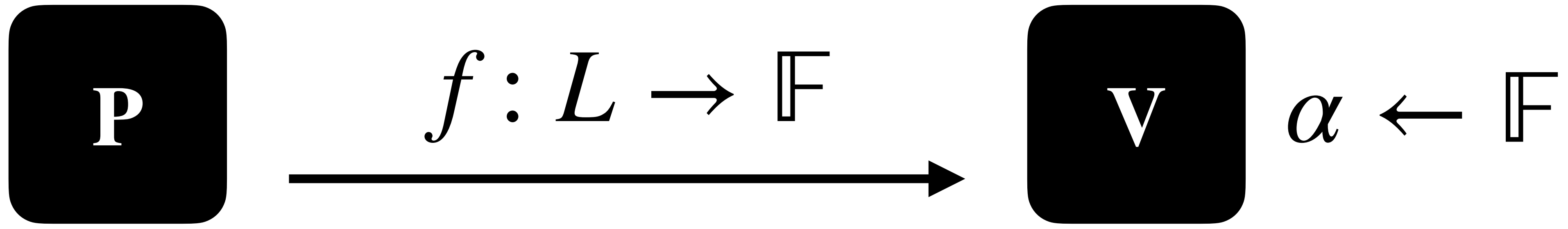


Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

Reduce $\mathsf{RS}[\mathbb{F}, L, d]$ **to** $\mathsf{RS}[\mathbb{F}, L^k, d/k]$



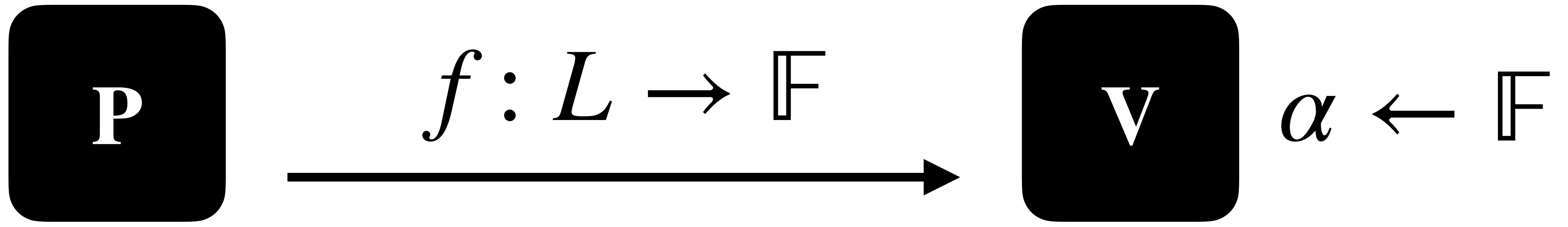
Selecting α defines a function $\mathsf{Fold}(f, k, \alpha)$

Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

Reduce $\mathsf{RS}[\mathbb{F}, L, d]$ **to** $\mathsf{RS}[\mathbb{F}, L^k, d/k]$



P

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

V

$\alpha \leftarrow \mathbb{F}$

Selecting α defines a function $\mathsf{Fold}(f, k, \alpha)$

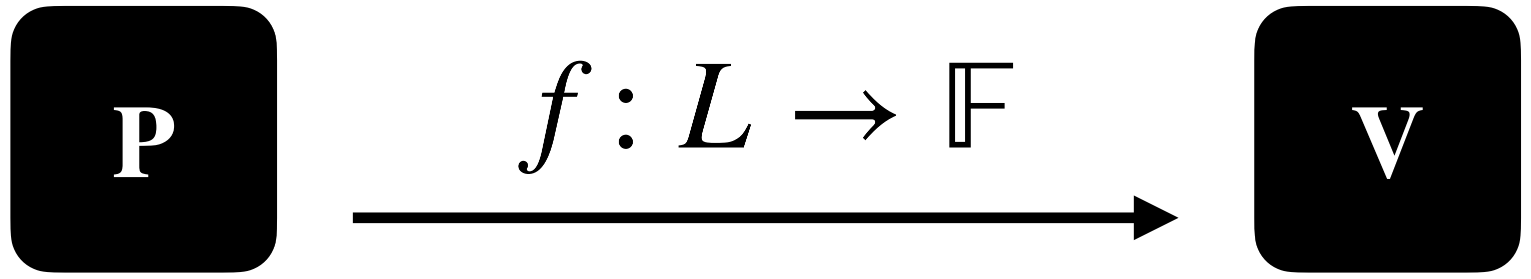
Local

Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

Reduce RS$[\mathbb{F}, L, d]$ **to** RS$[\mathbb{F}, L^k, d/k]$



$\alpha \leftarrow \mathbb{F}$

Selecting α defines a function Fold(f, k, α)

Local

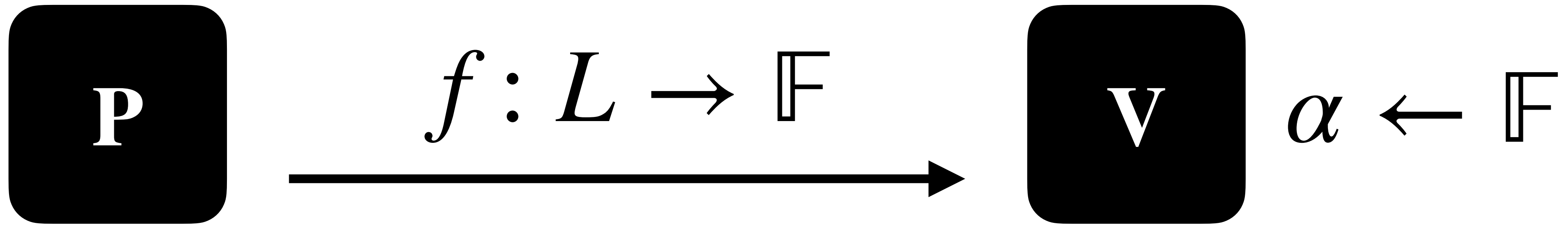
By querying f at k locations, $\mathbf{V}$ can compute Fold(f, k, α) at $z \in L^k$

Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

Reduce $\mathsf{RS}[\mathbb{F}, L, d]$ **to** $\mathsf{RS}[\mathbb{F}, L^k, d/k]$

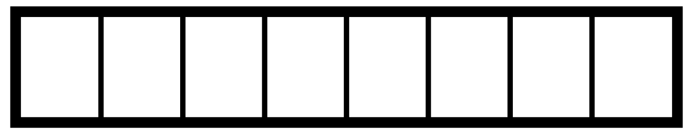


P $\xrightarrow{f : L \to \mathbb{F}}$ **V** $\alpha \leftarrow \mathbb{F}$

Selecting α defines a function $\mathsf{Fold}(f, k, \alpha)$

Local

By querying f at k locations, $\mathbf{V}$ can compute $\mathsf{Fold}(f, k, \alpha)$ at $z \in L^k$



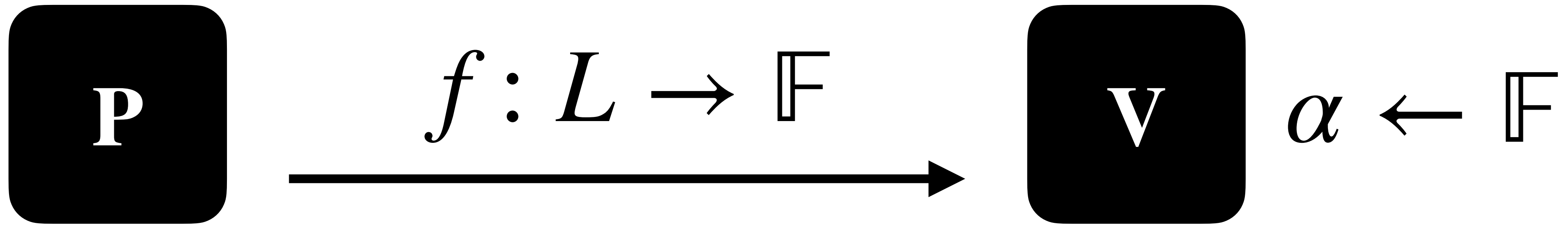
$\mathsf{Fold}(f, k, \alpha)$

Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

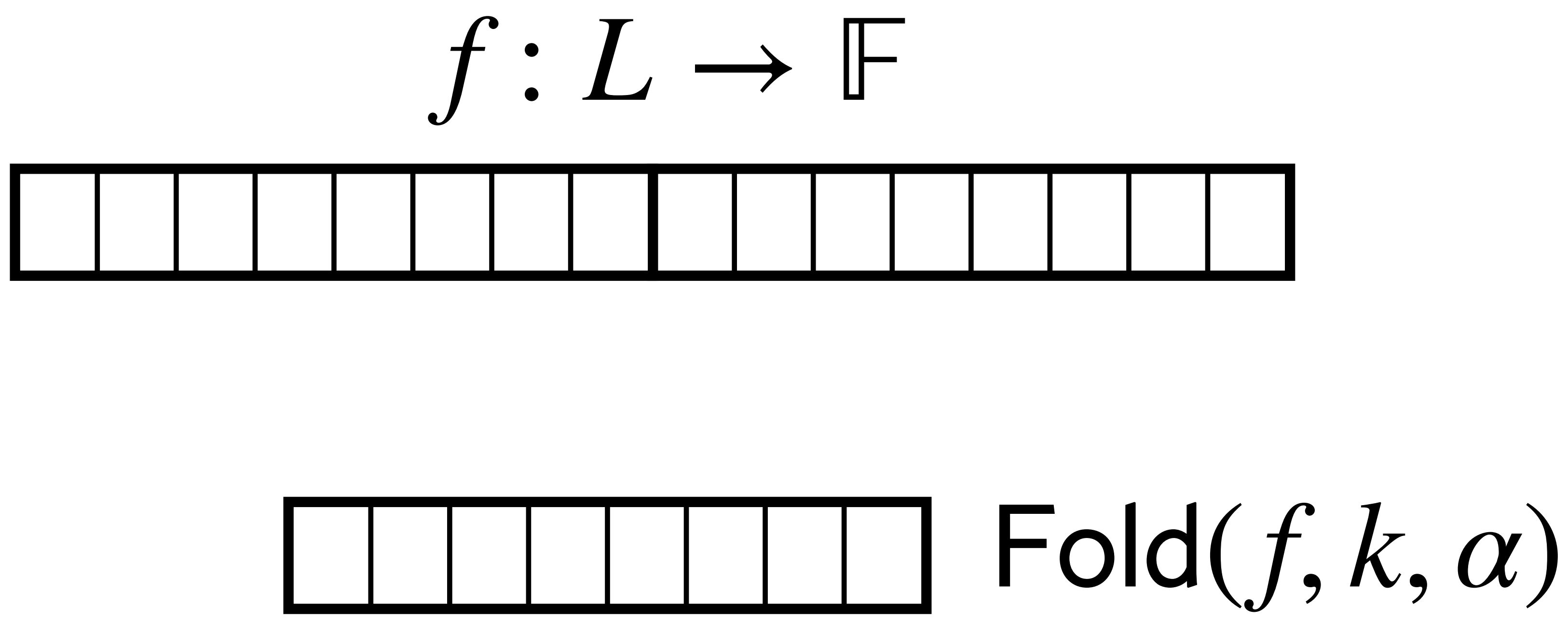
Reduce RS$[\mathbb{F}, L, d]$ **to** RS$[\mathbb{F}, L^k, d/k]$



Selecting α defines a function Fold(f, k, α)

Local

By querying f at k locations, $\mathbf{V}$ can compute Fold(f, k, α) at $z \in L^k$

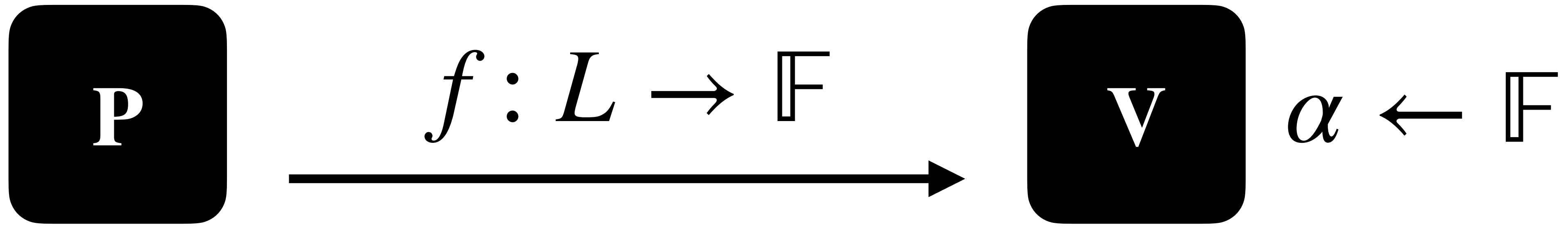


Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

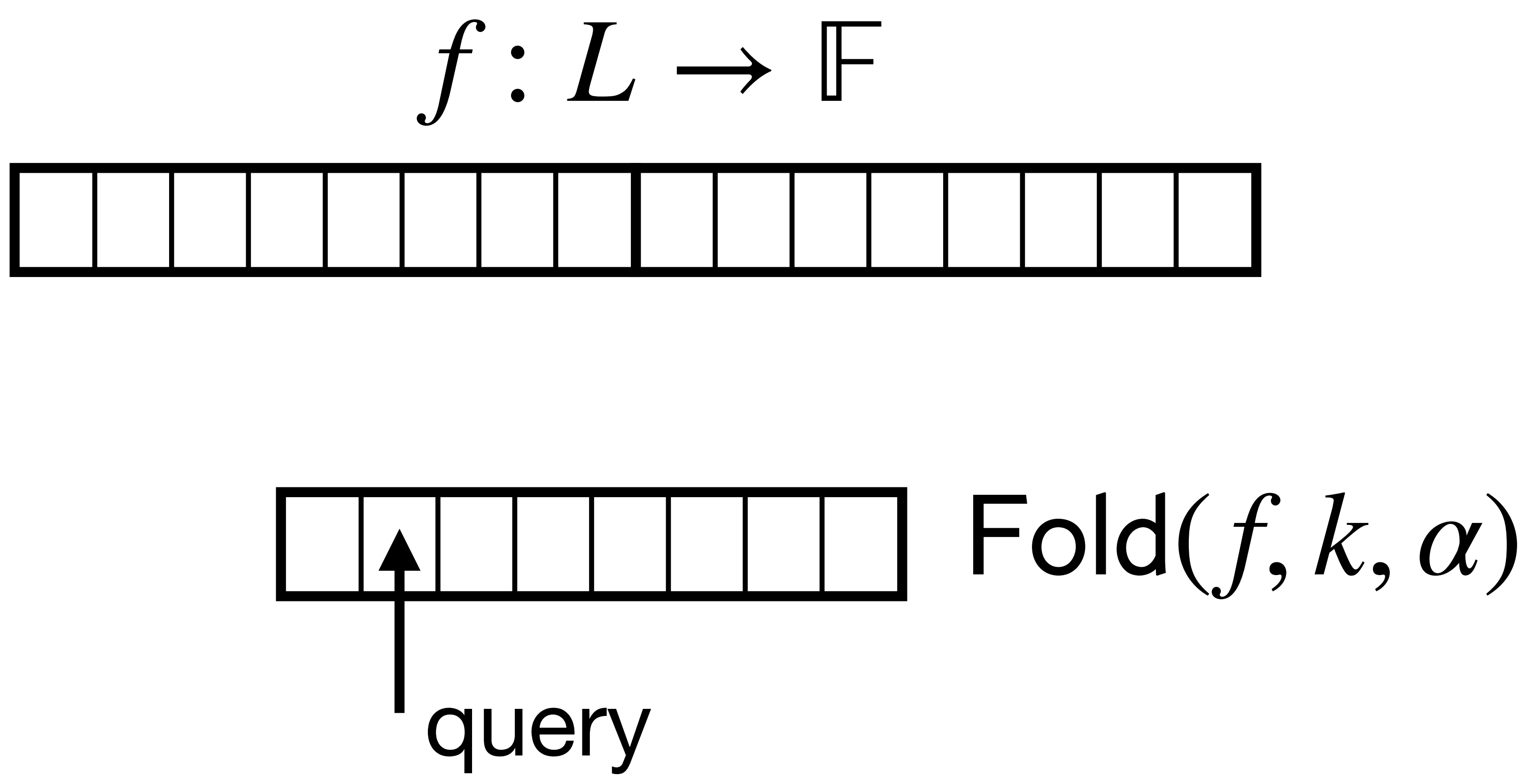
Reduce RS$[\mathbb{F}, L, d]$ **to** RS$[\mathbb{F}, L^k, d/k]$



Selecting α defines a function Fold(f, k, α)

Local

By querying f at k locations, $\mathbf{V}$ can compute Fold(f, k, α) at $z \in L^k$

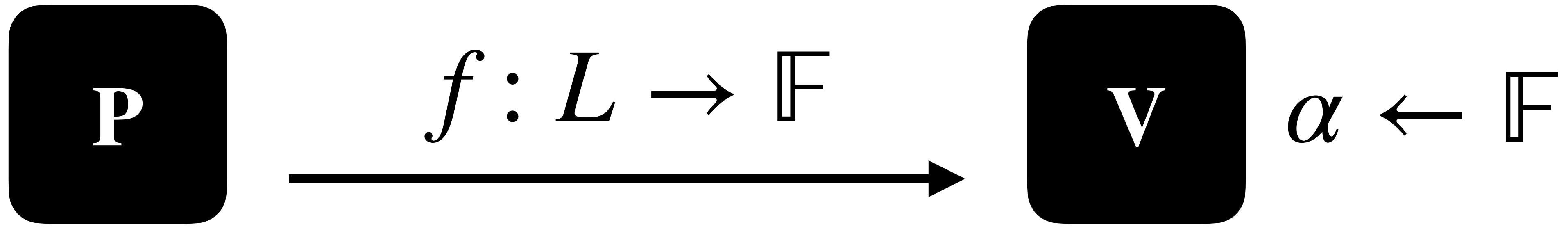


Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

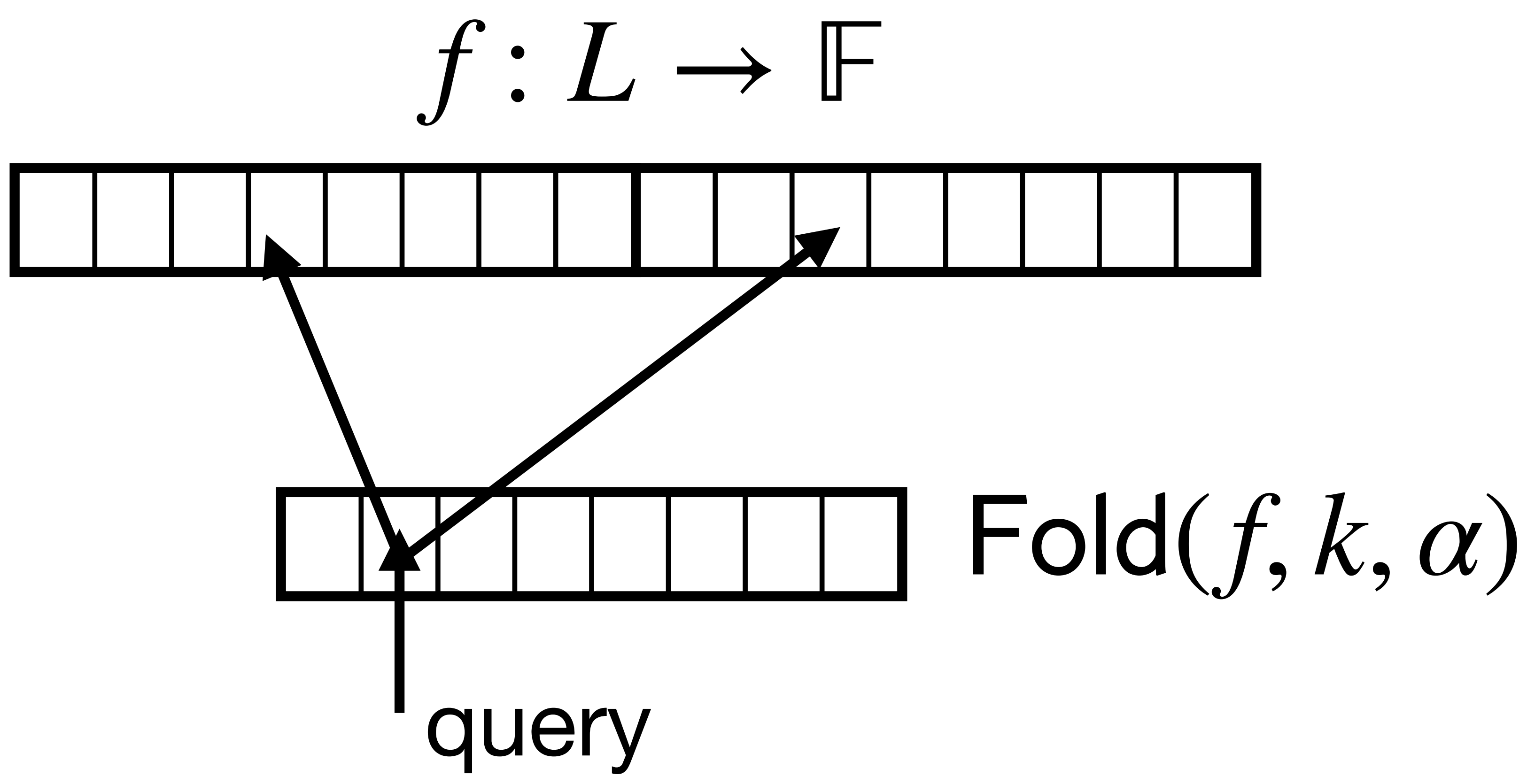
Reduce $\mathsf{RS}[\mathbb{F}, L, d]$ **to** $\mathsf{RS}[\mathbb{F}, L^k, d/k]$



Selecting α defines a function $\mathsf{Fold}(f, k, \alpha)$

Local

By querying f at k locations, $\mathbf{V}$ can compute $\mathsf{Fold}(f, k, \alpha)$ at $z \in L^k$

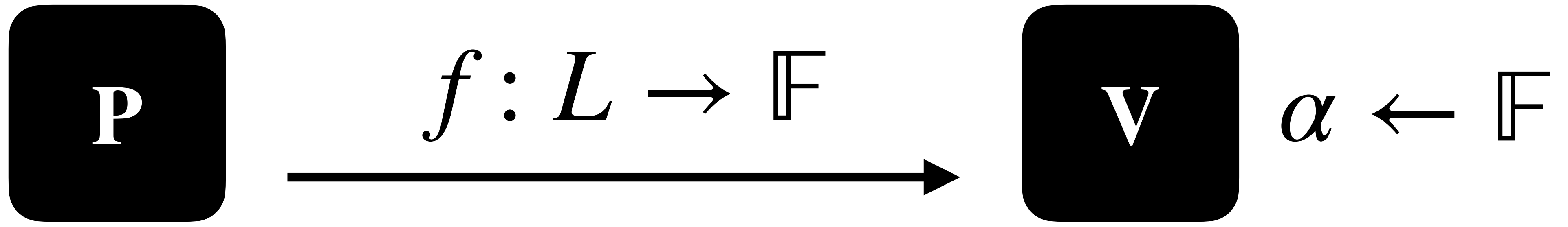


Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

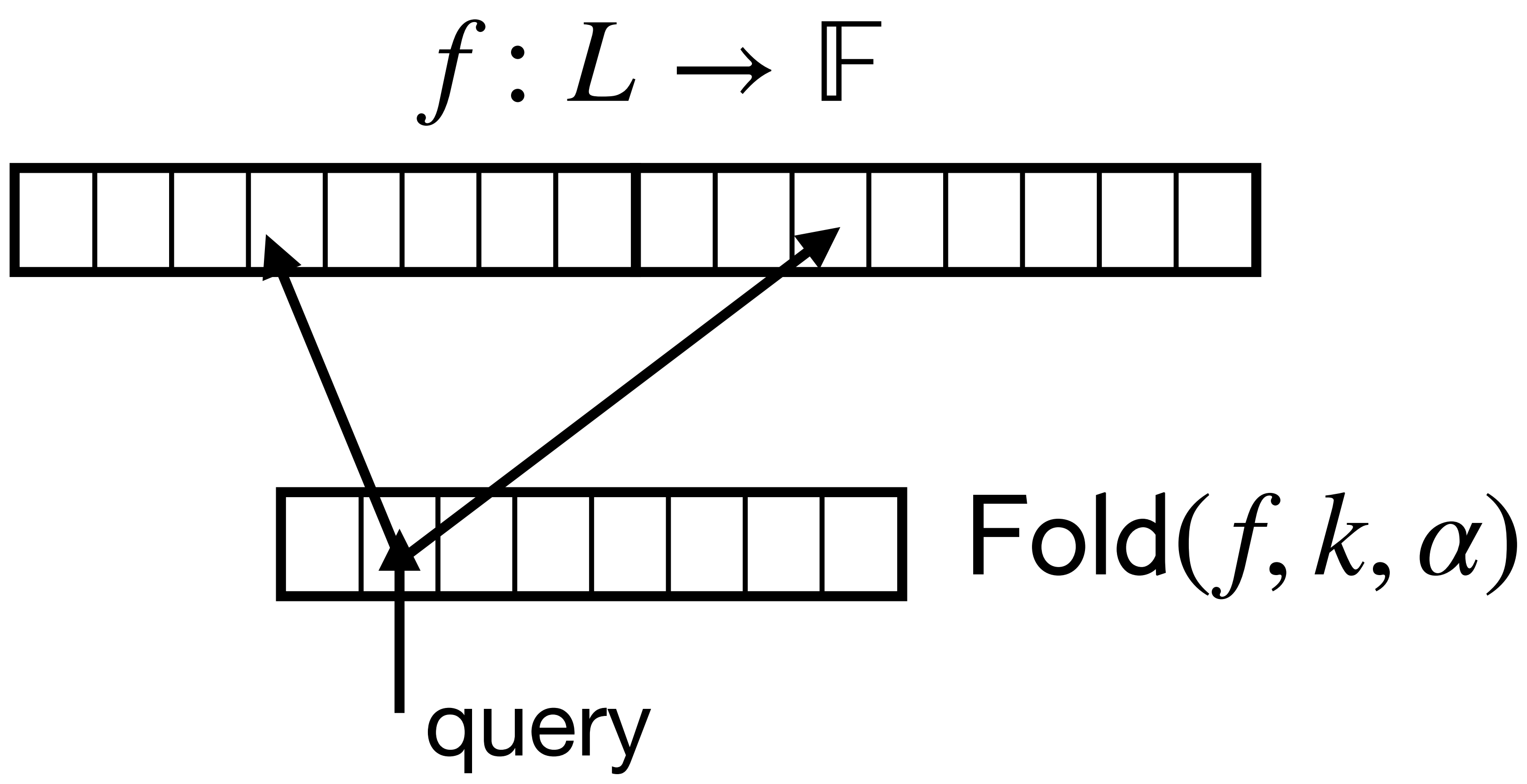
Reduce RS$[\mathbb{F}, L, d]$ **to** RS$[\mathbb{F}, L^k, d/k]$



Selecting α defines a function Fold(f, k, α)

Local

By querying f at k locations, $\mathbf{V}$ can compute Fold(f, k, α) at $z \in L^k$



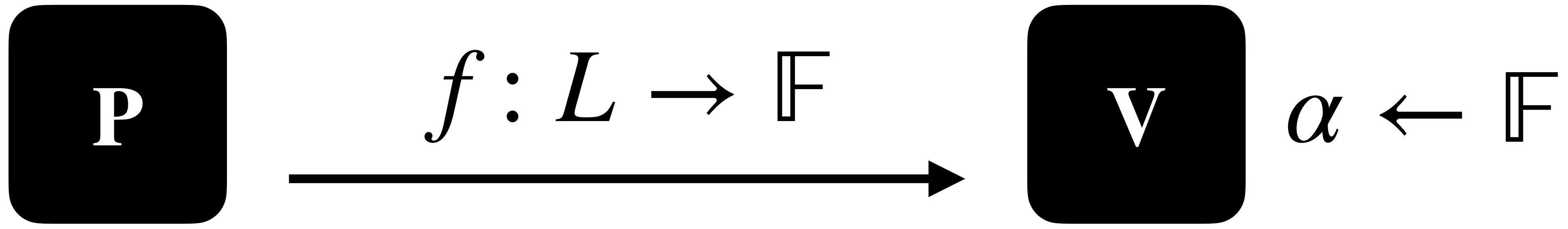
Distance Preserving

Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

Reduce RS$[\mathbb{F}, L, d]$ **to** RS$[\mathbb{F}, L^k, d/k]$

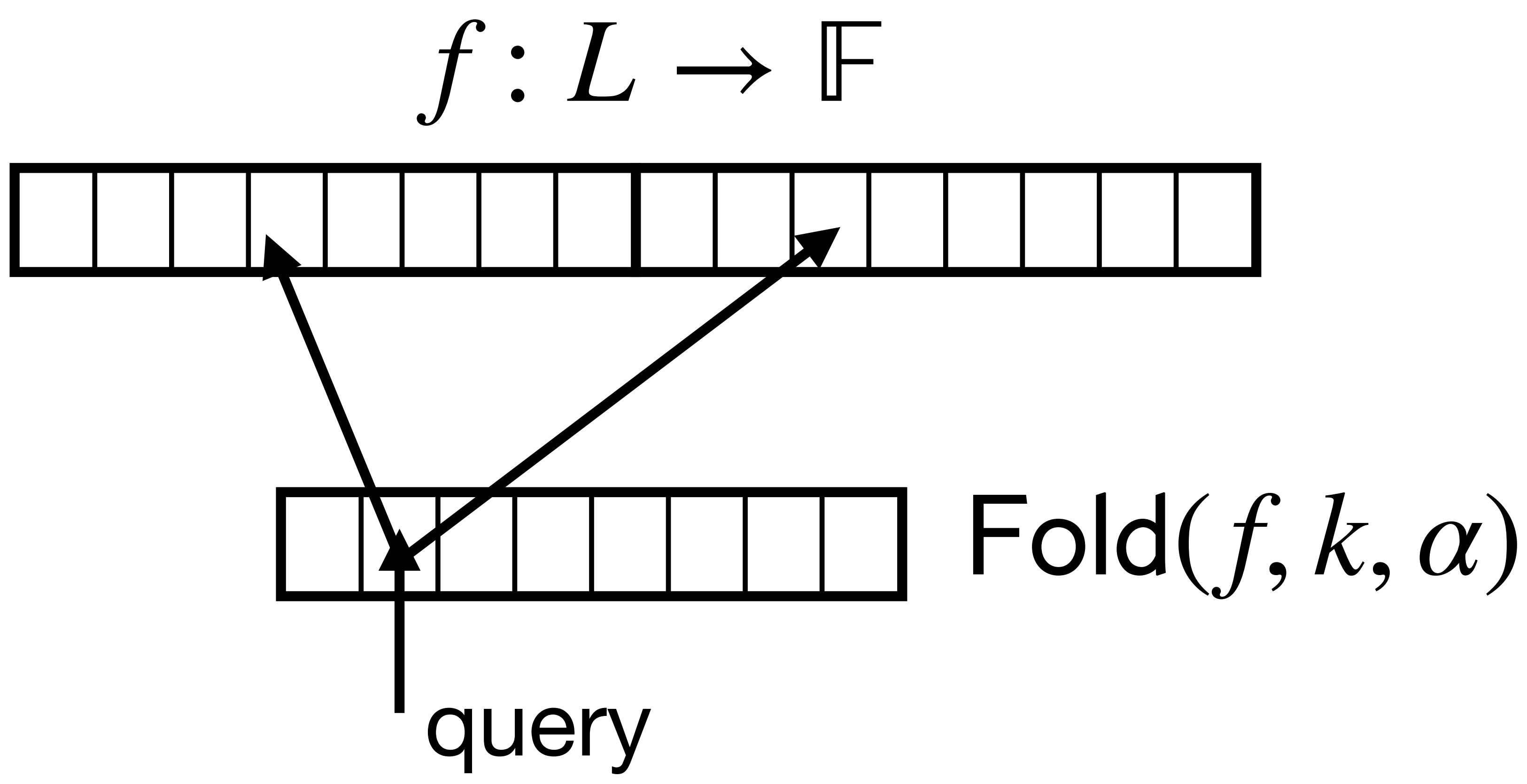


P $\xrightarrow{f : L \to \mathbb{F}}$ **V** $\alpha \leftarrow \mathbb{F}$

Selecting α defines a function Fold(f, k, α)

Local

By querying f at k locations, $\mathbf{V}$ can compute Fold(f, k, α) at $z \in L^k$

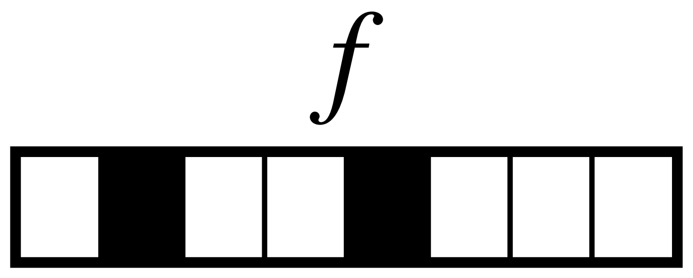


$f : L \to \mathbb{F}$

Fold(f, k, α)

query

Distance Preserving



f

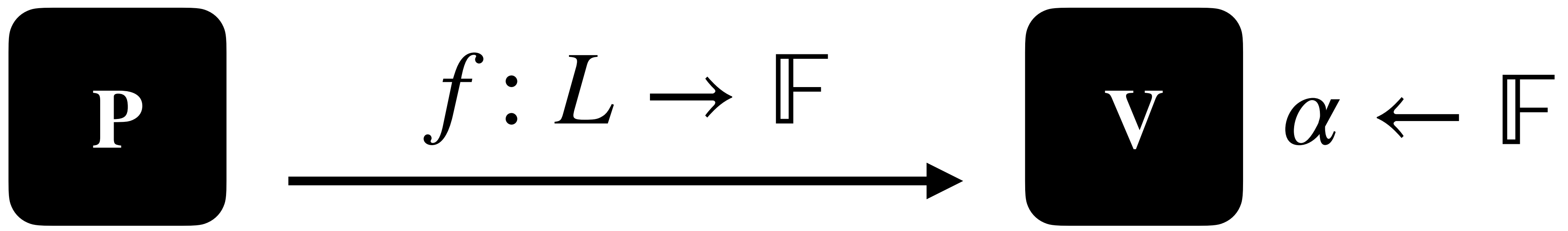
f is δ-far from RS

Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

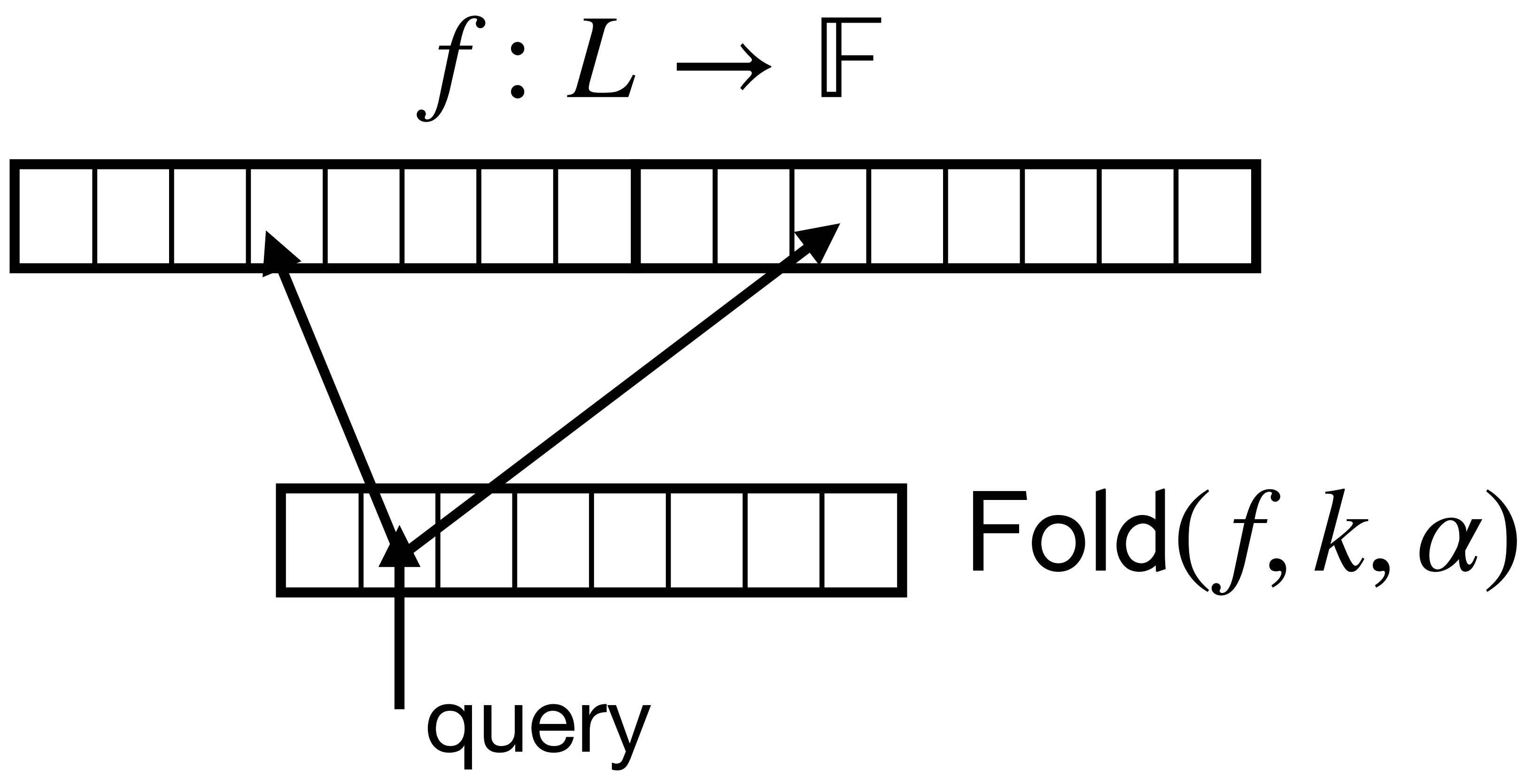
Reduce $\mathrm{RS}[\mathbb{F}, L, d]$ **to** $\mathrm{RS}[\mathbb{F}, L^k, d/k]$



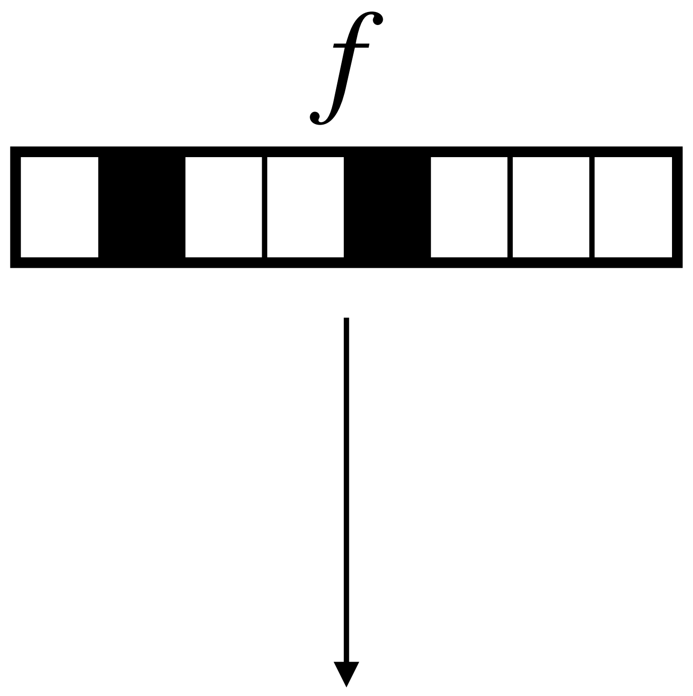
Selecting α defines a function $\mathrm{Fold}(f, k, \alpha)$

Local

By querying f at k locations, $\mathbf{V}$ can compute $\mathrm{Fold}(f, k, \alpha)$ at $z \in L^k$



Distance Preserving



f is δ-far from RS

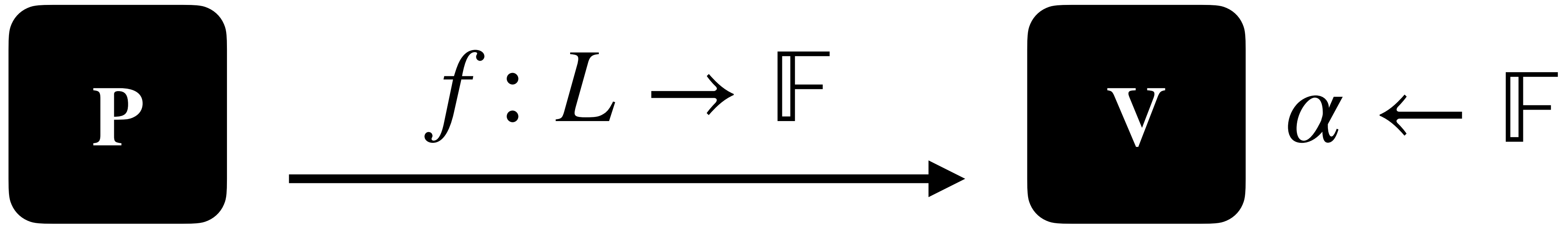
w.h.p. over α

Folding

$L^k = \{x^k : x \in L\}$

$|L^k| = |L|/k$

Reduce RS$[\mathbb{F}, L, d]$ **to** RS$[\mathbb{F}, L^k, d/k]$



P

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

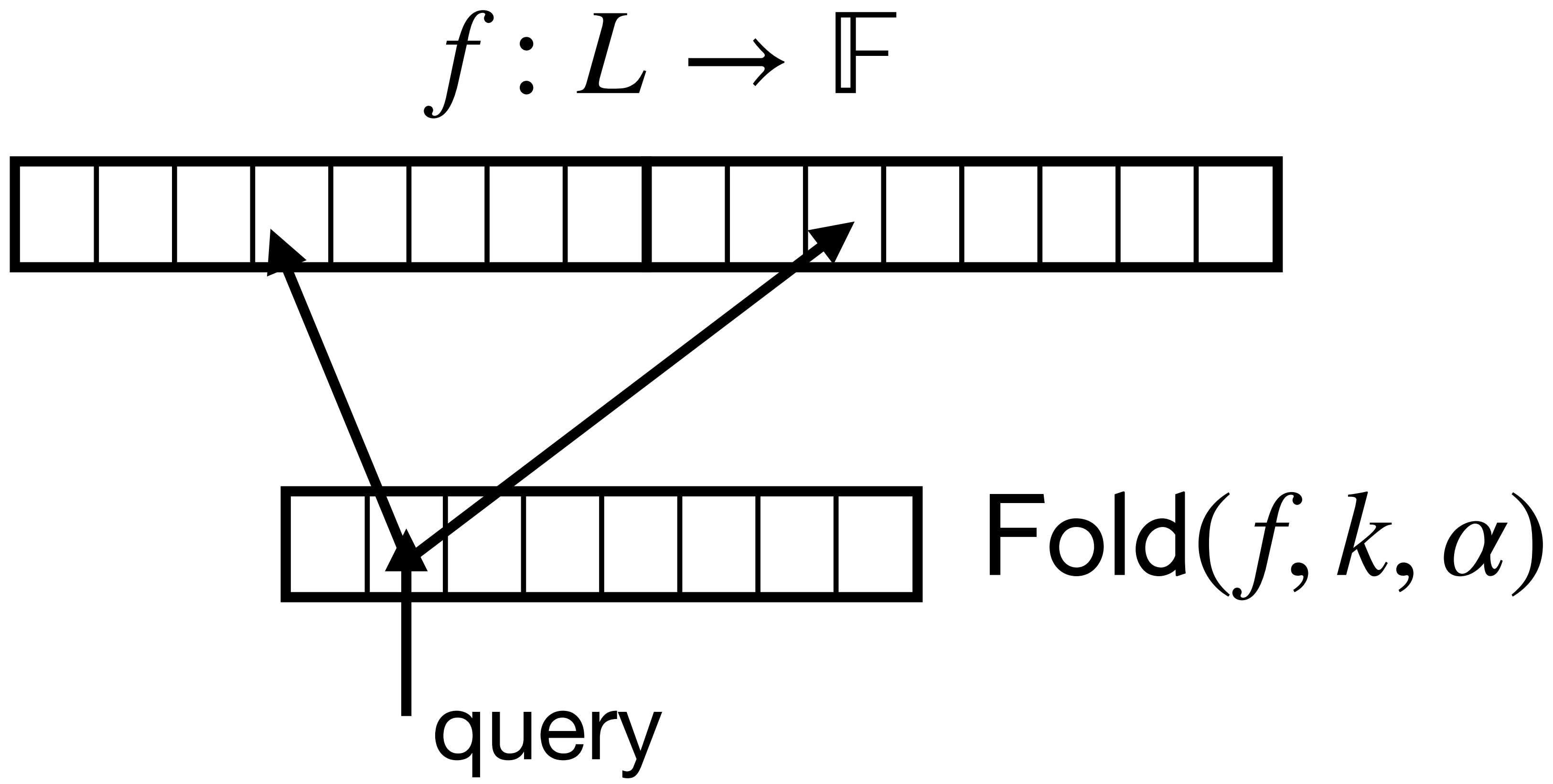
V

$\alpha \leftarrow \mathbb{F}$

Selecting α defines a function Fold(f, k, α)

Local

By querying f at k locations, $\mathbf{V}$ can compute Fold(f, k, α) at $z \in L^k$

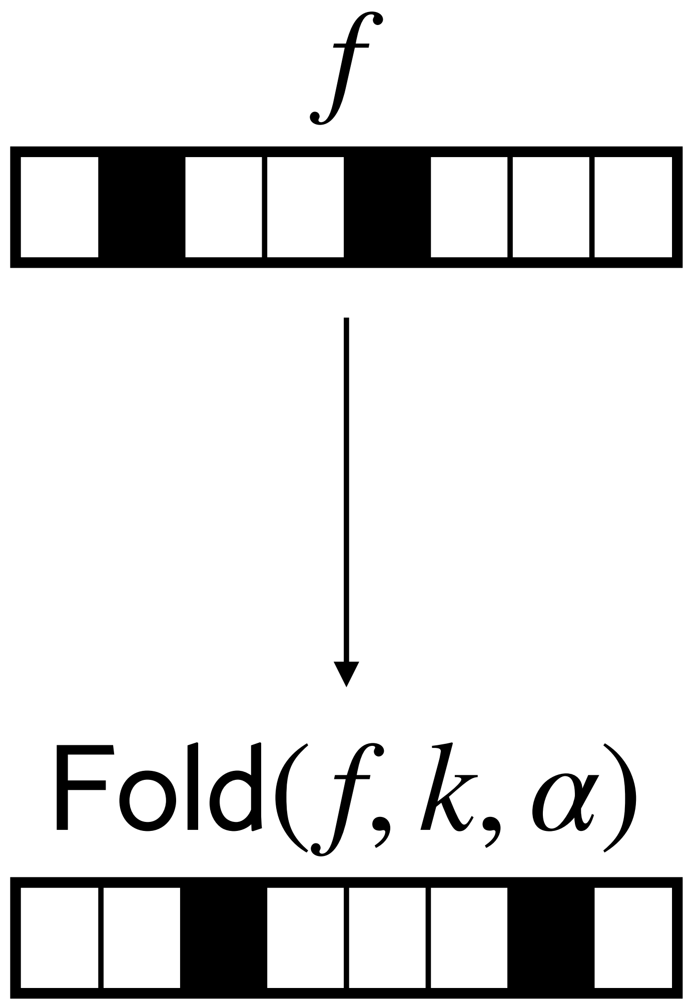


$f : L \to \mathbb{F}$

Fold(f, k, α)

query

Distance Preserving



f

Fold(f, k, α)

f is δ-far from RS

w.h.p. over α

Fold is δ-far from RS

FRI Protocol

FRI Protocol

$$f_0 : L \to \mathbb{F}$$

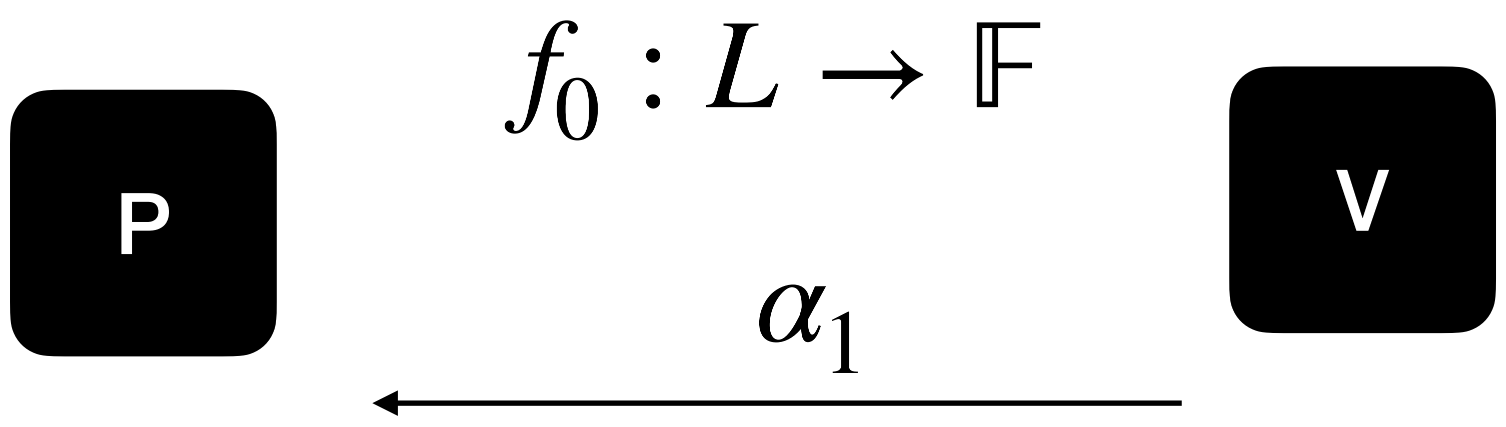
FRI Protocol

P

$f_0 : L \to \mathbb{F}$

V

FRI Protocol



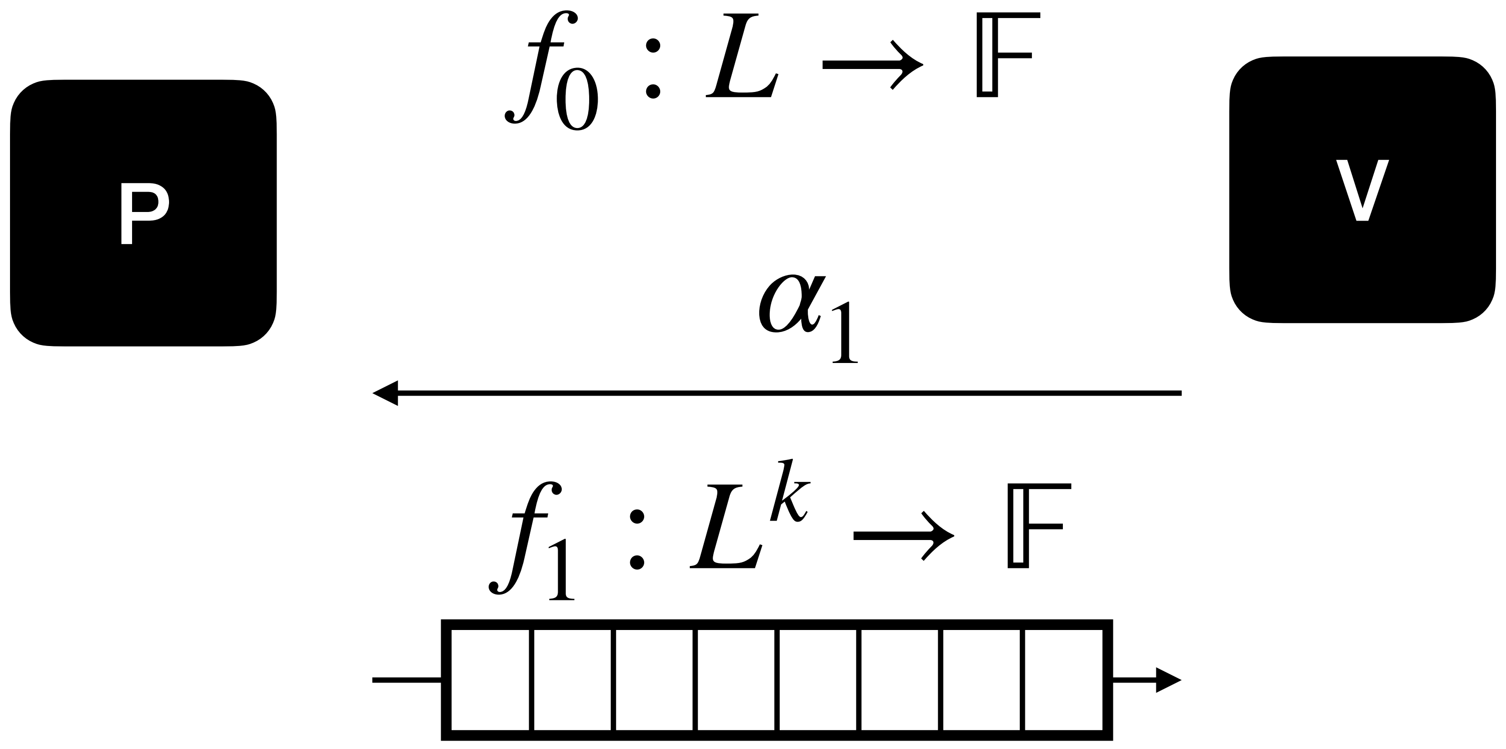
P

$f_0 : L \to \mathbb{F}$

V

α_1

FRI Protocol

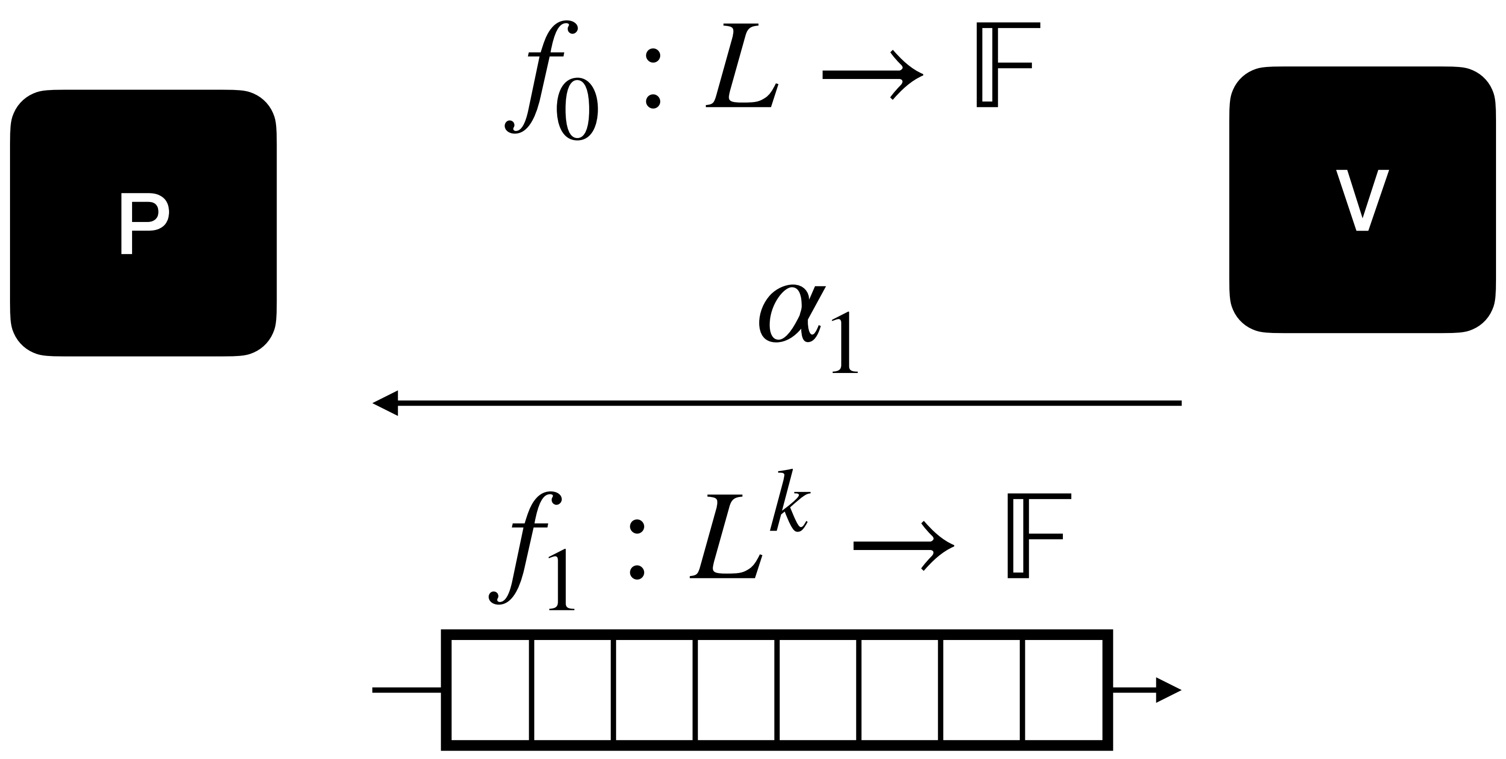


$f_0 : L \to \mathbb{F}$

α_1

$f_1 : L^k \to \mathbb{F}$

FRI Protocol



P

V

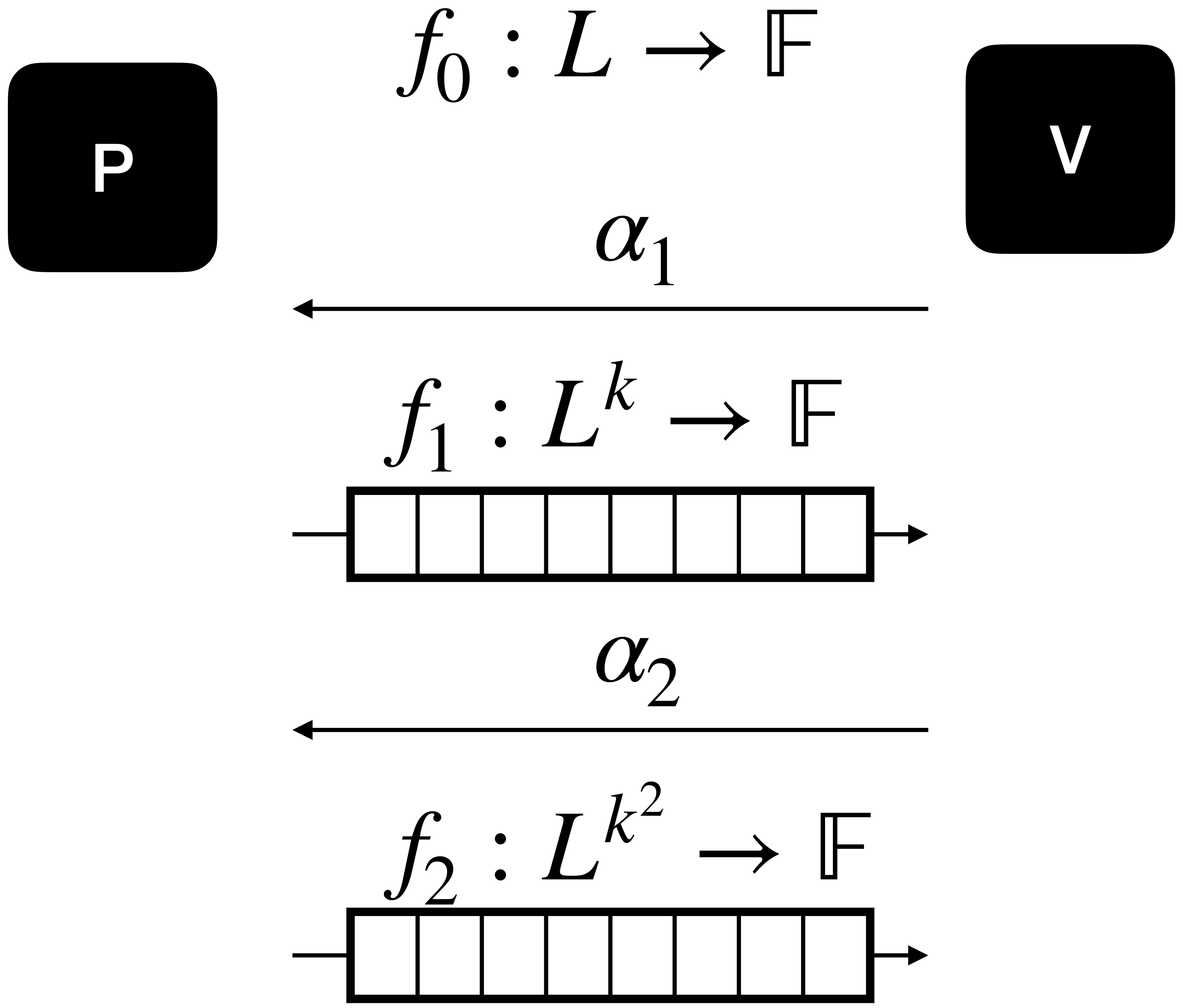
$f_0 : L \to \mathbb{F}$

α_1

$f_1 : L^k \to \mathbb{F}$

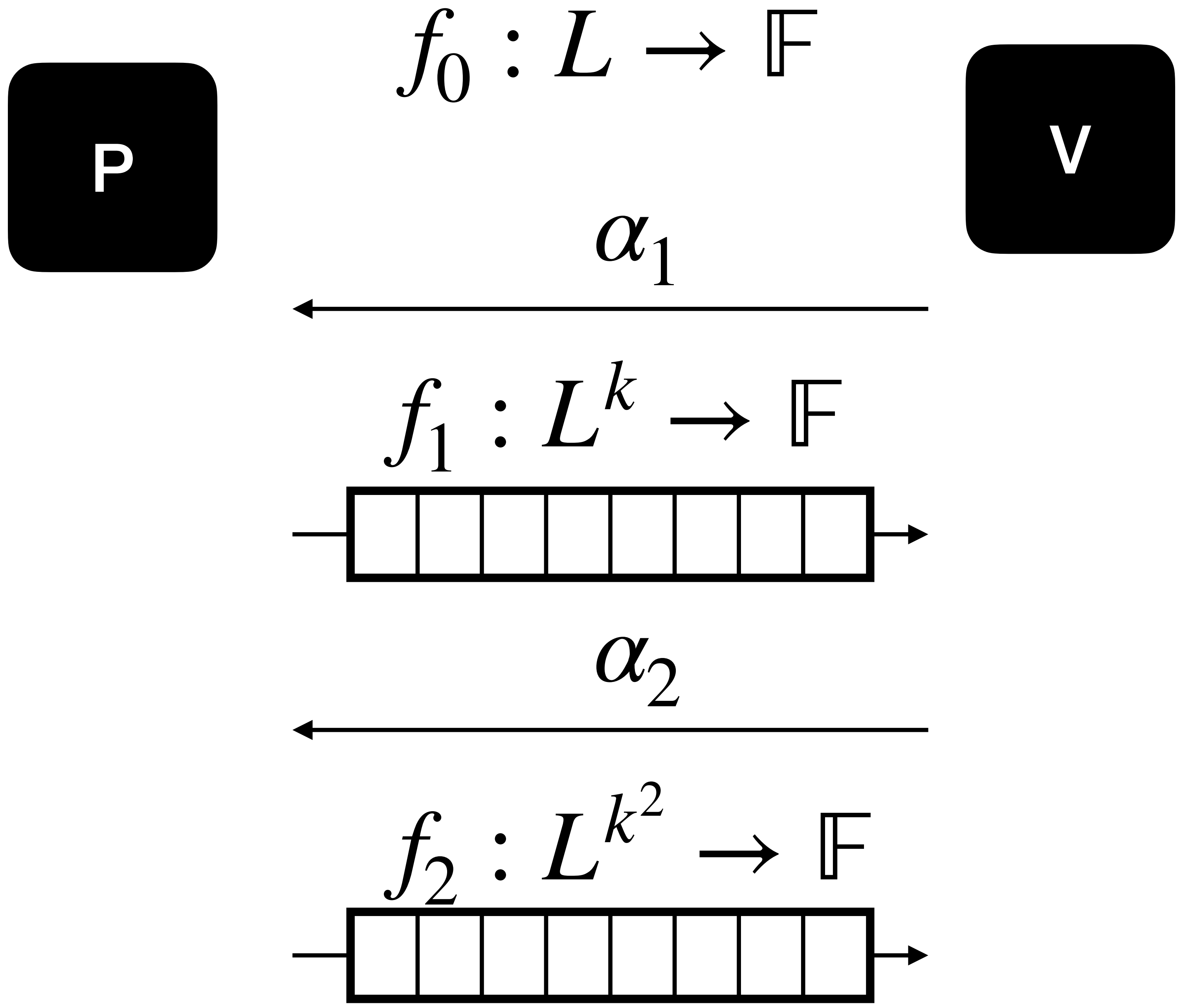
$f_1 =$ Fold(f_0, k, α) suppose to be in RS$[\mathbb{F}, L^k, d/k]$

FRI Protocol



$f_1 = \mathsf{Fold}(f_0, k, \alpha)$ suppose to be in $\mathsf{RS}[\mathbb{F}, L^k, d/k]$

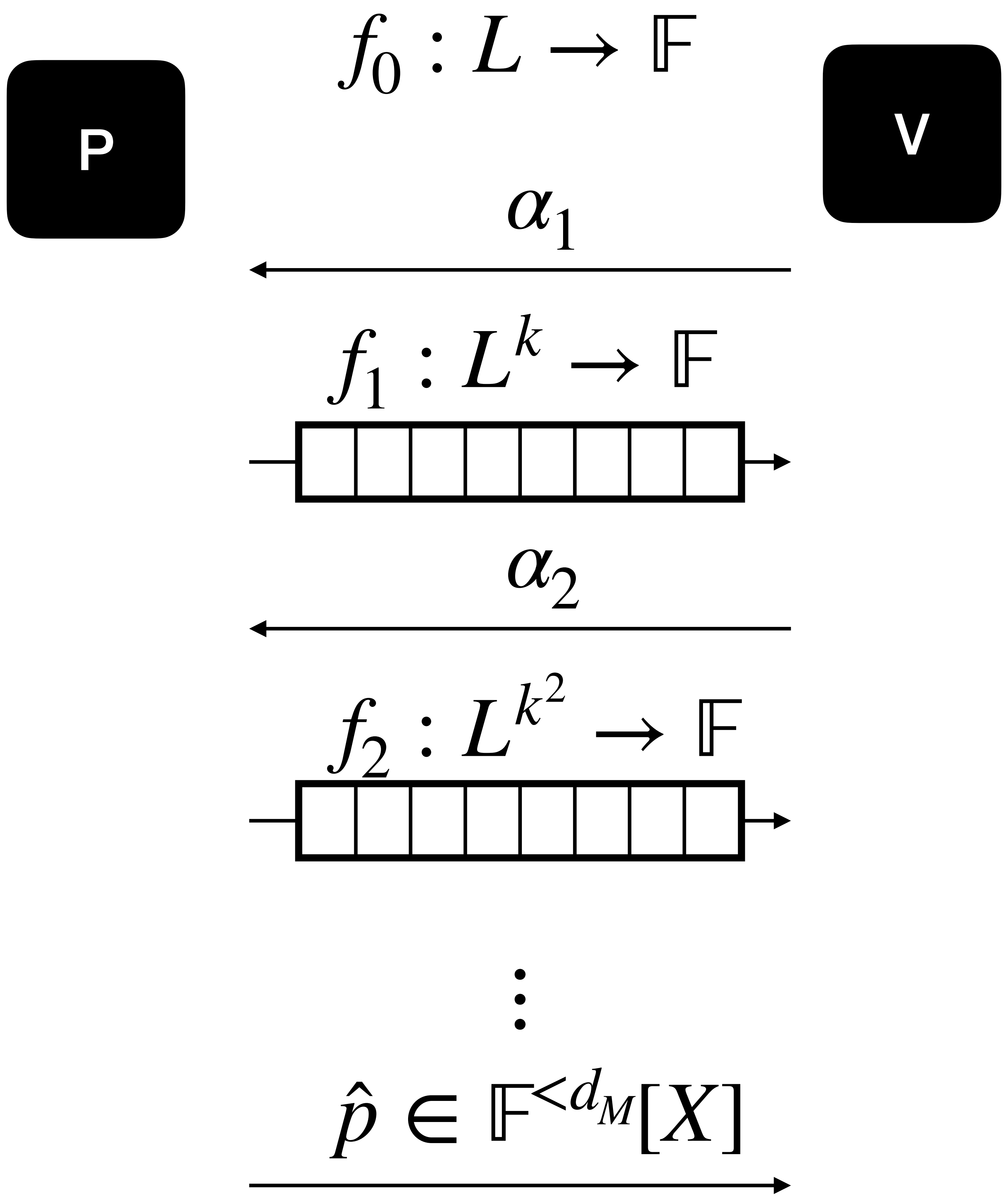
FRI Protocol



$f_1 = \text{Fold}(f_0, k, \alpha)$ suppose to be in $\text{RS}[\mathbb{F}, L^k, d/k]$

$f_i = \text{Fold}(f_{i-1}, k, \alpha_{i-1})$ suppose to be in $\text{RS}[\mathbb{F}, L^{k^i}, d/k^i]$

FRI Protocol



P

V

$f_0 : L \to \mathbb{F}$

α_1

$f_1 : L^k \to \mathbb{F}$

α_2

$f_2 : L^{k^2} \to \mathbb{F}$

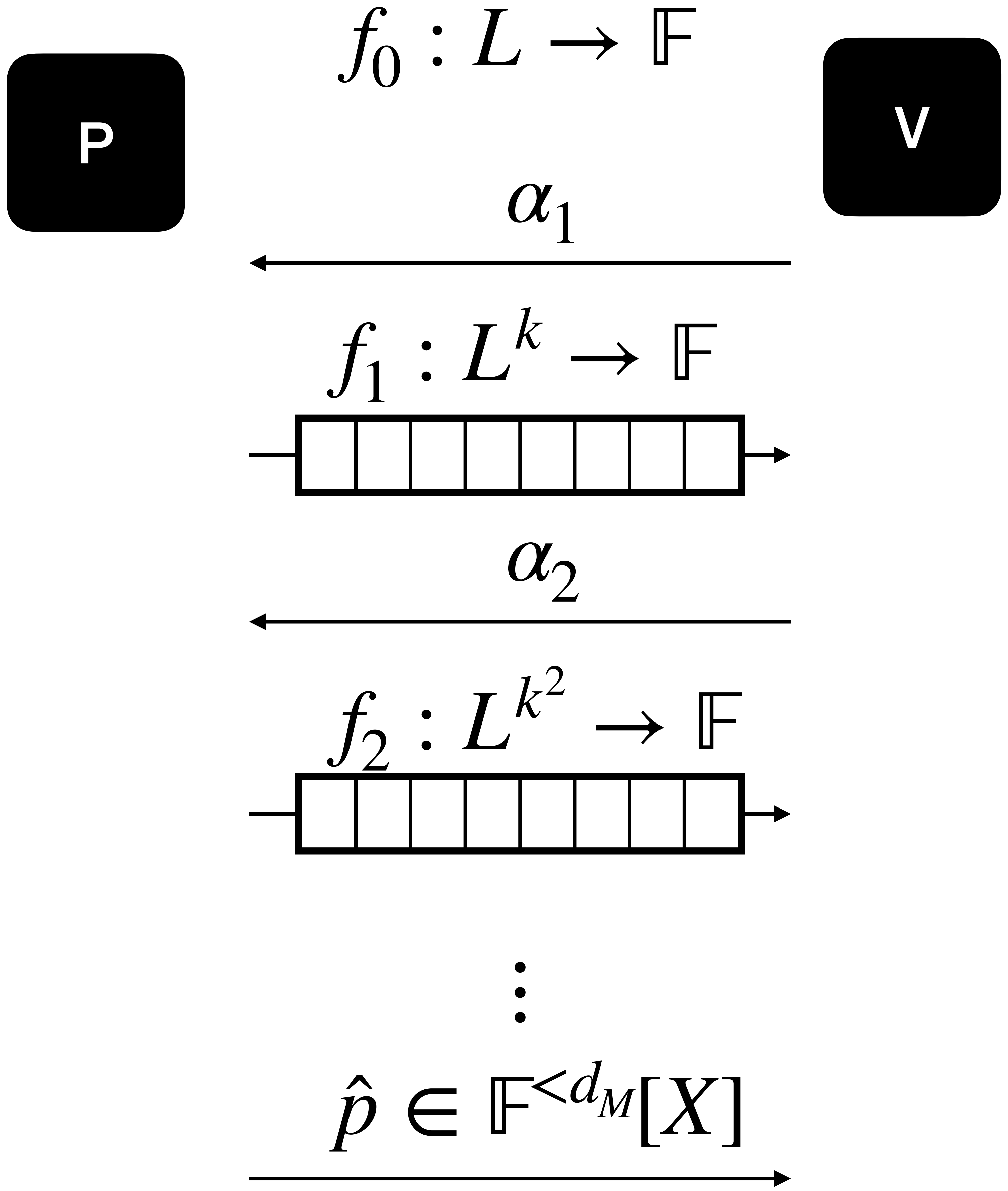
$\vdots$

$\hat{p} \in \mathbb{F}^{<d_M}[X]$

$f_1 =$ Fold(f_0, k, α) suppose to be in RS$[\mathbb{F}, L^k, d/k]$

$f_i =$ Fold$(f_{i-1}, k, \alpha_{i-1})$ suppose to be in RS$[\mathbb{F}, L^{k^i}, d/k^i]$

FRI Protocol



P

V

$f_0 : L \to \mathbb{F}$

α_1

$f_1 : L^k \to \mathbb{F}$

α_2

$f_2 : L^{k^2} \to \mathbb{F}$

$\vdots$

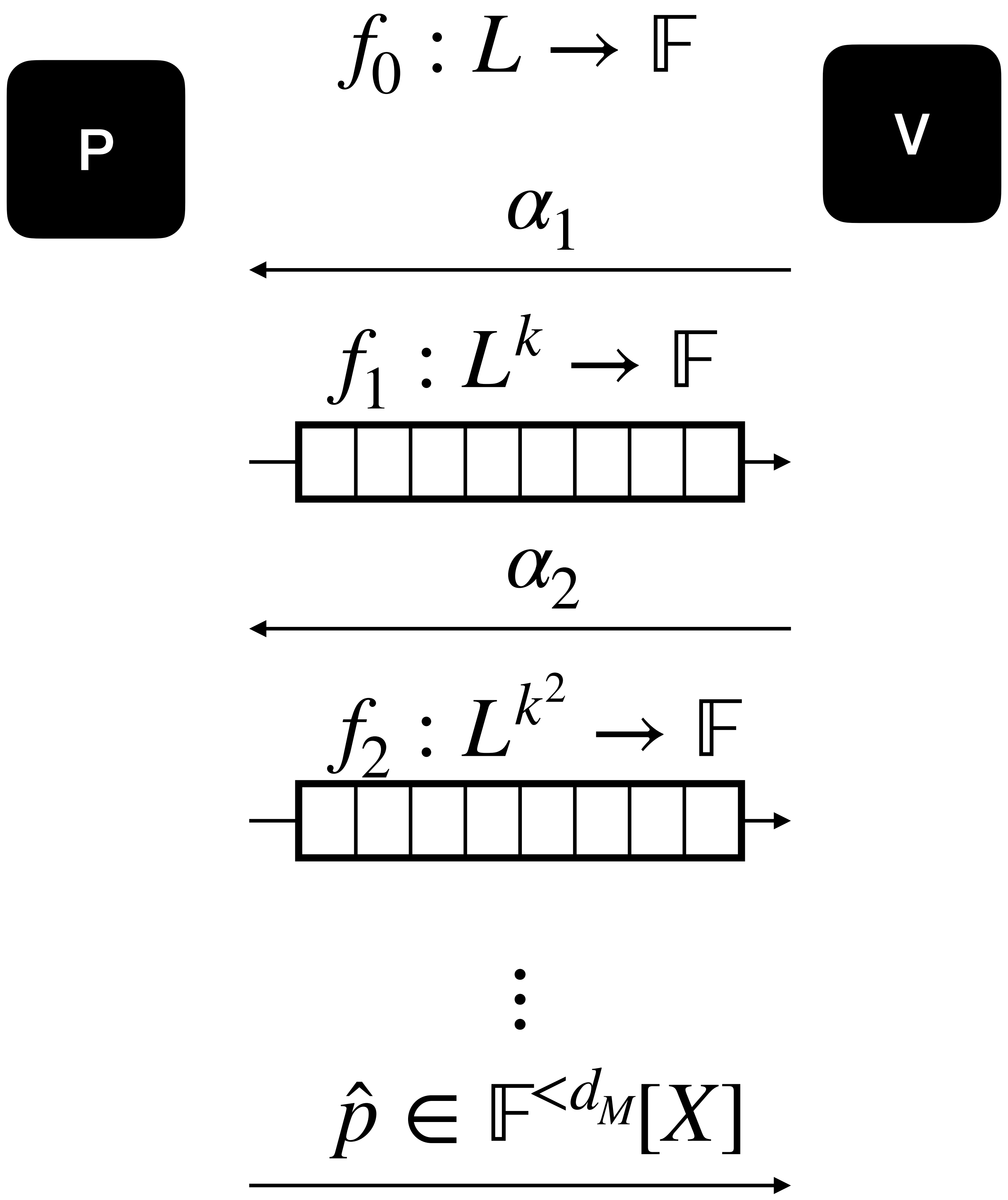
$\hat{p} \in \mathbb{F}^{<d_M}[X]$

$f_1 = \text{Fold}(f_0, k, \alpha)$ suppose to be in $\text{RS}[\mathbb{F}, L^k, d/k]$

$f_i = \text{Fold}(f_{i-1}, k, \alpha_{i-1})$ suppose to be in $\text{RS}[\mathbb{F}, L^{k^i}, d/k^i]$

Check consistency between functions using t queries

FRI Protocol



P

V

$f_0 : L \to \mathbb{F}$

α_1

$f_1 : L^k \to \mathbb{F}$

α_2

$f_2 : L^{k^2} \to \mathbb{F}$

$\vdots$

$\hat{p} \in \mathbb{F}^{<d_M}[X]$

$f_1 =$ Fold(f_0, k, α) suppose to be in RS$[\mathbb{F}, L^k, d/k]$

$f_i =$ Fold$(f_{i-1}, k, \alpha_{i-1})$ suppose to be in RS$[\mathbb{F}, L^{k^i}, d/k^i]$

Check consistency between functions using t queries

Soundness error: $\rho^{\frac{t}{2}}$

STIR: Shift-To-Improve-Rate

STIR: Shift-To-Improve-Rate

- Main idea to reduce:

  $$\text{RS}[\mathbb{F}, L, d] \quad \Rightarrow \quad \text{RS}[\mathbb{F}, L^2, d/k]$$

  where $|L^2| = |L|/2$.

STIR: Shift-To-Improve-Rate

- Main idea to reduce:

$$\mathsf{RS}[\mathbb{F}, L, d] \quad \Rightarrow \quad \mathsf{RS}[\mathbb{F}, L^2, d/k]$$

  where $|L^2| = |L|/2$.

- New rate: $\rho' = \dfrac{2\rho}{k}$. If $k > 2 \Rightarrow$ rate improves $\Rightarrow$ new code is easier to test

STIR: Shift-To-Improve-Rate

- Main idea to reduce:

  $$\mathsf{RS}[\mathbb{F}, L, d] \quad \Rightarrow \quad \mathsf{RS}[\mathbb{F}, L^2, d/k]$$

  where $|L^2| = |L|/2$.

- New rate: $\rho' = \frac{2\rho}{k}$. If $k > 2 \Rightarrow$ rate improves $\Rightarrow$ new code is easier to test

- Distance: preserve distance expect w.p. $\rho^{t/2}$ where t is number of queries

STIR: Shift-To-Improve-Rate

- Main idea to reduce:
  $$\mathsf{RS}[\mathbb{F}, L, d] \quad \Rightarrow \quad \mathsf{RS}[\mathbb{F}, L^2, d/k]$$
  where $|L^2| = |L|/2$.
- New rate: $\rho' = \dfrac{2\rho}{k}$. If $k > 2 \Rightarrow$ rate improves $\Rightarrow$ new code is easier to test
- Distance: preserve distance expect w.p. $\rho^{t/2}$ where t is number of queries
- Round-by-round errors:

STIR: Shift-To-Improve-Rate

- Main idea to reduce:

  $$\mathsf{RS}[\mathbb{F}, L, d] \quad \Rightarrow \quad \mathsf{RS}[\mathbb{F}, L^2, d/k]$$

  where $|L^2| = |L|/2$.

- New rate: $\rho' = \dfrac{2\rho}{k}$. If $k > 2 \Rightarrow$ rate improves $\Rightarrow$ new code is easier to test

- Distance: preserve distance expect w.p. $\rho^{t/2}$ where t is number of queries

- Round-by-round errors: $\rho_0^{\frac{t_1}{2}}, \ldots, \rho_M^{\frac{t_M}{2}}$

STIR: Shift-To-Improve-Rate

- Main idea to reduce:

  $$\mathsf{RS}[\mathbb{F}, L, d] \quad \Rightarrow \quad \mathsf{RS}[\mathbb{F}, L^2, d/k]$$

  where $|L^2| = |L|/2$.

- New rate: $\rho' = \dfrac{2\rho}{k}$. If $k > 2 \Rightarrow$ rate improves $\Rightarrow$ new code is easier to test

- Distance: preserve distance expect w.p. $\rho^{t/2}$ where t is number of queries

- Round-by-round errors: $\rho_0^{\frac{t_1}{2}}, \ldots, \rho_M^{\frac{t_M}{2}}$ $\qquad t_i = \dfrac{\lambda}{-\log(\sqrt{\rho_i})}$

STIR: Shift-To-Improve-Rate

- Main idea to reduce:

  $$\mathsf{RS}[\mathbb{F}, L, d] \quad \Rightarrow \quad \mathsf{RS}[\mathbb{F}, L^2, d/k]$$

  where $|L^2| = |L|/2$.

- New rate: $\rho' = \dfrac{2\rho}{k}$. If $k > 2 \Rightarrow$ rate improves $\Rightarrow$ new code is easier to test

- Distance: preserve distance expect w.p. $\rho^{t/2}$ where t is number of queries

- Round-by-round errors: $\rho_0^{\frac{t_1}{2}}, \ldots, \rho_M^{\frac{t_M}{2}}$ $\qquad t_i = \dfrac{\lambda}{-\log(\sqrt{\rho_i})}$

Less queries each round!

STIR: Shift-To-Improve-Rate

- Main idea to reduce:

$$\mathrm{RS}[\mathbb{F}, L, d] \quad \Rightarrow \quad \mathrm{RS}[\mathbb{F}, L^2, d/k]$$

  where $|L^2| = |L|/2$.

- New rate: $\rho' = \dfrac{2\rho}{k}$. If $k > 2 \Rightarrow$ rate improves $\Rightarrow$ new code is easier to test
- Distance: preserve distance expect w.p. $\rho^{t/2}$ where t is number of queries
- Round-by-round errors: $\rho_0^{\frac{t_1}{2}}, \ldots, \rho_M^{\frac{t_M}{2}}$ $\quad t_i = \dfrac{\lambda}{-\log(\sqrt{\rho_i})}$ ← Less queries each round!
- Example $k = 16$, $\rho_0 = 1/2$, 100 bits of security:

STIR: Shift-To-Improve-Rate

- Main idea to reduce:

  $$\mathrm{RS}[\mathbb{F}, L, d] \quad \Rightarrow \quad \mathrm{RS}[\mathbb{F}, L^2, d/k]$$

  where $|L^2| = |L|/2$.
- New rate: $\rho' = \dfrac{2\rho}{k}$. If $k > 2 \Rightarrow$ rate improves $\Rightarrow$ new code is easier to test
- Distance: preserve distance expect w.p. $\rho^{t/2}$ where t is number of queries
- Round-by-round errors: $\rho_0^{\frac{t_1}{2}}, \ldots, \rho_M^{\frac{t_M}{2}}$ $\quad t_i = \dfrac{\lambda}{-\log(\sqrt{\rho_i})}$ ← Less queries each round!
- Example $k = 16$, $\rho_0 = 1/2$, 100 bits of security:

  $$\rho_1 = 1/16, \rho_2 = 1/128, \ \ldots$$

STIR: Shift-To-Improve-Rate

- Main idea to reduce:

$$\mathsf{RS}[\mathbb{F}, L, d] \quad \Rightarrow \quad \mathsf{RS}[\mathbb{F}, L^2, d/k]$$

  where $|L^2| = |L|/2$.

- New rate: $\rho' = \dfrac{2\rho}{k}$. If $k > 2 \Rightarrow$ rate improves $\Rightarrow$ new code is easier to test

- Distance: preserve distance expect w.p. $\rho^{t/2}$ where t is number of queries

- Round-by-round errors: $\rho_0^{\frac{t_1}{2}}, \ldots, \rho_M^{\frac{t_M}{2}}$ $\qquad t_i = \dfrac{\lambda}{-\log(\sqrt{\rho_i})}$

Less queries each round!

- Example $k = 16$, $\rho_0 = 1/2$, 100 bits of security:

$$\rho_1 = 1/16,\ \rho_2 = 1/128,\ \ldots$$

$$t_0 = 200,\ t_1 = 50,\ t_2 = 29,\ \ldots$$

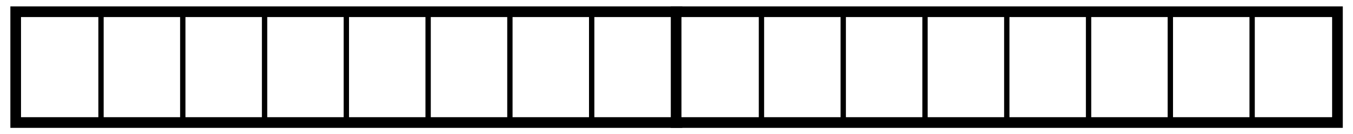
Domain shifting

Given test for $\mathrm{RS}[\mathbb{F}, L*, d]$, test $\mathrm{RS}[\mathbb{F}, L, d]$

Domain shifting

Given test for $\mathsf{RS}[\mathbb{F}, L*, d]$, test $\mathsf{RS}[\mathbb{F}, L, d]$

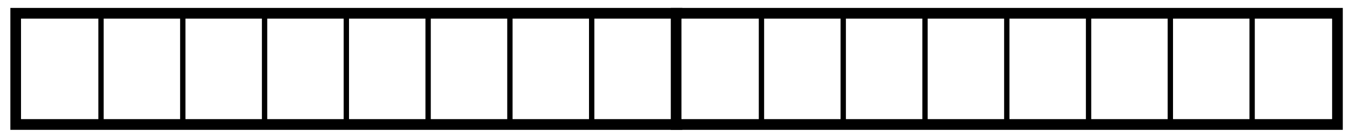
$$f : L \to \mathbb{F}$$

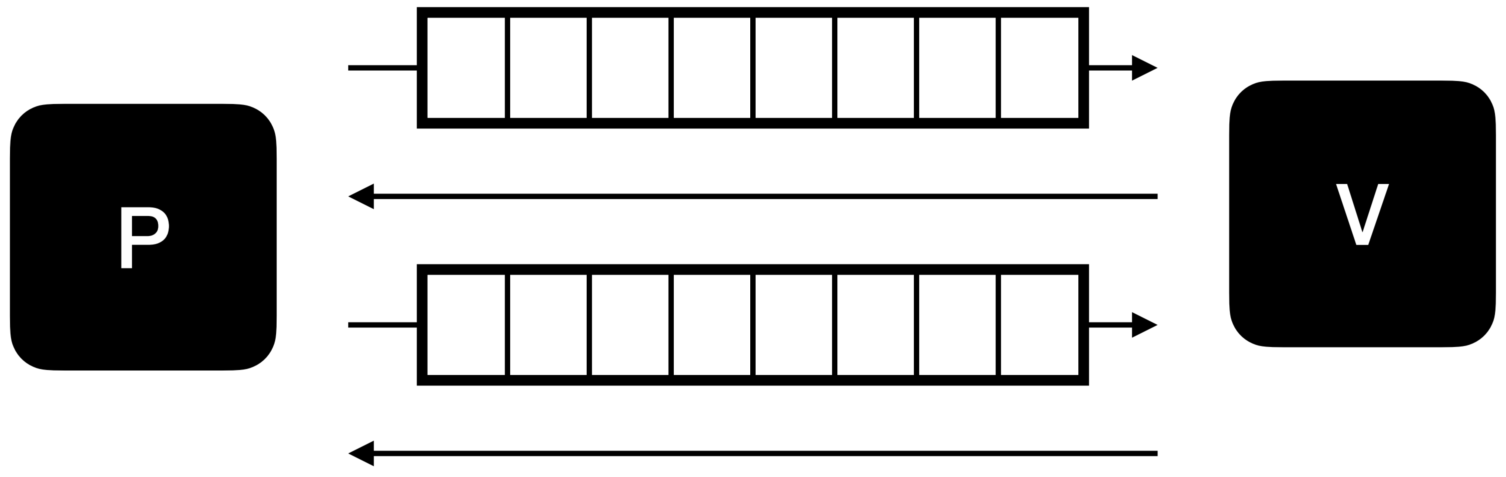


Domain shifting

Given test for $\mathsf{RS}[\mathbb{F}, L^*, d]$, test $\mathsf{RS}[\mathbb{F}, L, d]$

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

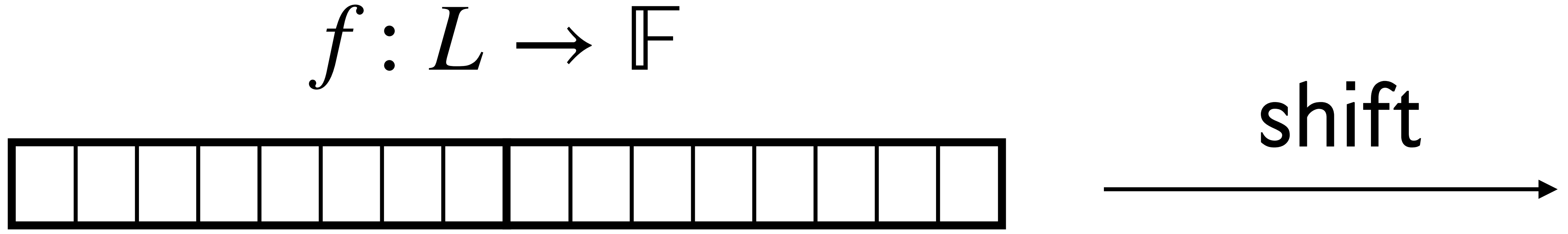


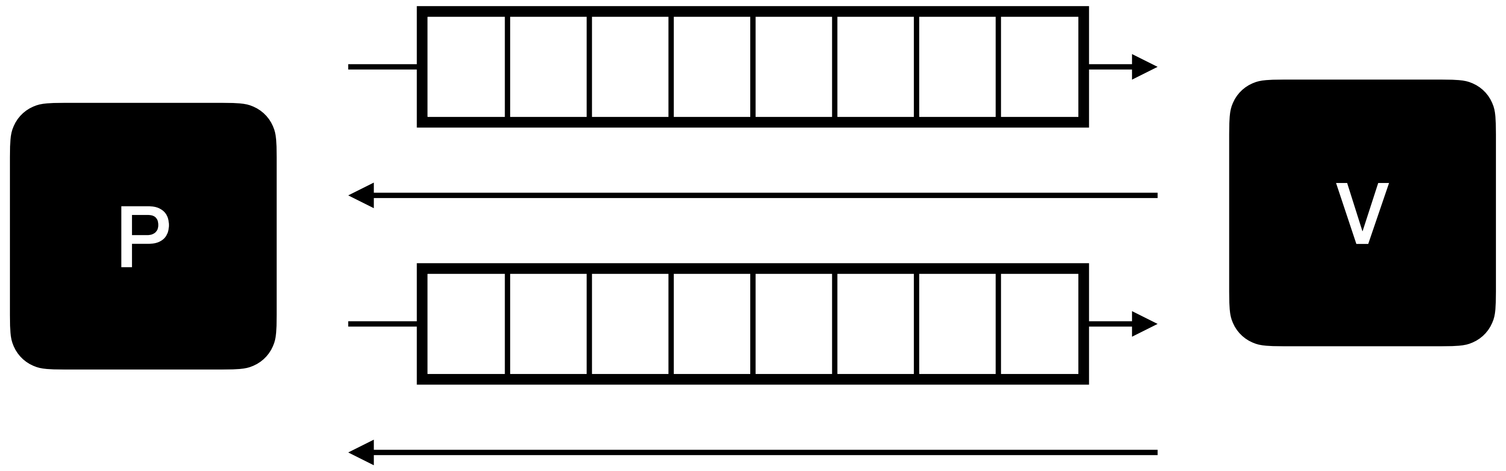


Test for $\mathsf{RS}[\mathbb{F}, L^*, d]$

Domain shifting

Given test for $\mathsf{RS}[\mathbb{F}, L^*, d]$, test $\mathsf{RS}[\mathbb{F}, L, d]$

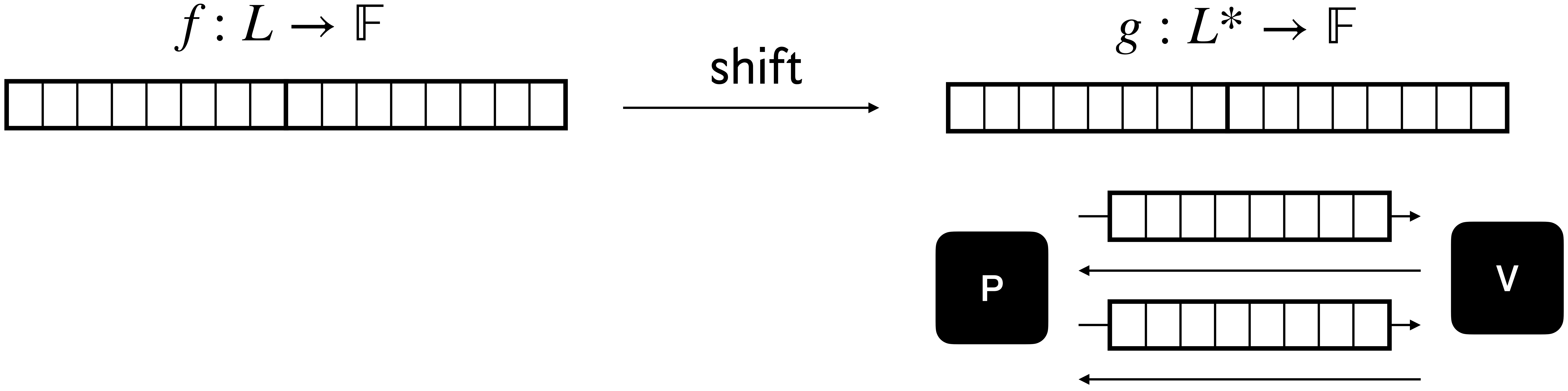




Test for $\mathsf{RS}[\mathbb{F}, L^*, d]$

Domain shifting

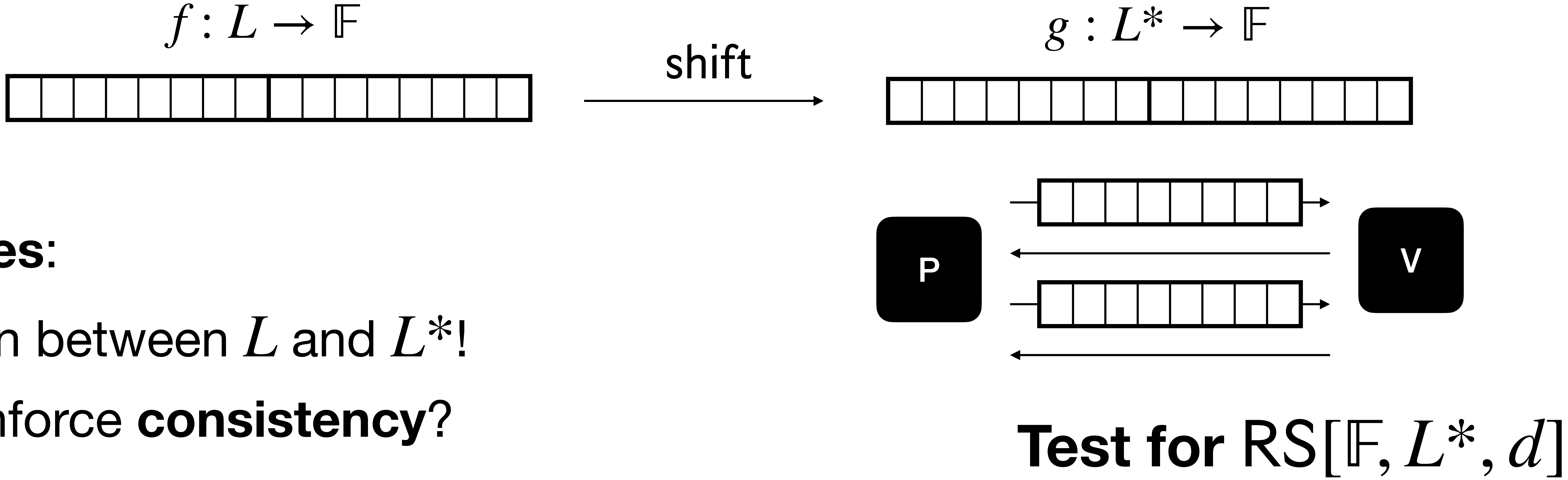
Given test for $\mathsf{RS}[\mathbb{F}, L^*, d]$, test $\mathsf{RS}[\mathbb{F}, L, d]$



Test for $\mathsf{RS}[\mathbb{F}, L^*, d]$

Domain shifting

Given test for $\mathsf{RS}[\mathbb{F}, L^*, d]$, test $\mathsf{RS}[\mathbb{F}, L, d]$



Test for $\mathsf{RS}[\mathbb{F}, L^*, d]$

Challenges:

No relation between L and L^*!

How to enforce **consistency**?

Quotienting

Enforce constraints on f or amplify distance

Quotienting

Enforce constraints on f or amplify distance

Let:

$f : L \to \mathbb{F}$ be a function

$\mathsf{Ans} : S \to \mathbb{F}$ be a list of (claimed) evaluations of (the extension of) f on S

Quotienting

Enforce constraints on f or amplify distance

Let:

$f : L \to \mathbb{F}$ be a function

$\mathsf{Ans} : S \to \mathbb{F}$ be a list of (claimed) evaluations of (the extension of) f on S

$$\mathsf{Quotient}(f, \mathsf{Ans})(x) := \frac{f(x) - \hat{\mathsf{Ans}}(x)}{V_S(x)}$$

Quotienting

Enforce constraints on f or amplify distance

Let:

$f : L \to \mathbb{F}$ be a function

$\text{Ans} : S \to \mathbb{F}$ be a list of (claimed) evaluations of (the extension of) f on S

$$\text{Quotient}(f, \text{Ans})(x) := \frac{f(x) - \hat{\text{Ans}}(x)}{V_S(x)}$$

Local

Quotienting

Enforce constraints on f or amplify distance

Let:

$f : L \to \mathbb{F}$ be a function

$\mathsf{Ans} : S \to \mathbb{F}$ be a list of (claimed) evaluations of (the extension of) f on S

$$\mathsf{Quotient}(f, \mathsf{Ans})(x) := \frac{f(x) - \hat{\mathsf{Ans}}(x)}{V_S(x)}$$

Local

$\mathbf{V}$ can compute $\mathsf{Quotient}(f, \mathsf{Ans})$ at $x \in L \backslash S$ by querying f at x

Quotienting

Enforce constraints on f or amplify distance

Let:

$f : L \to \mathbb{F}$ be a function

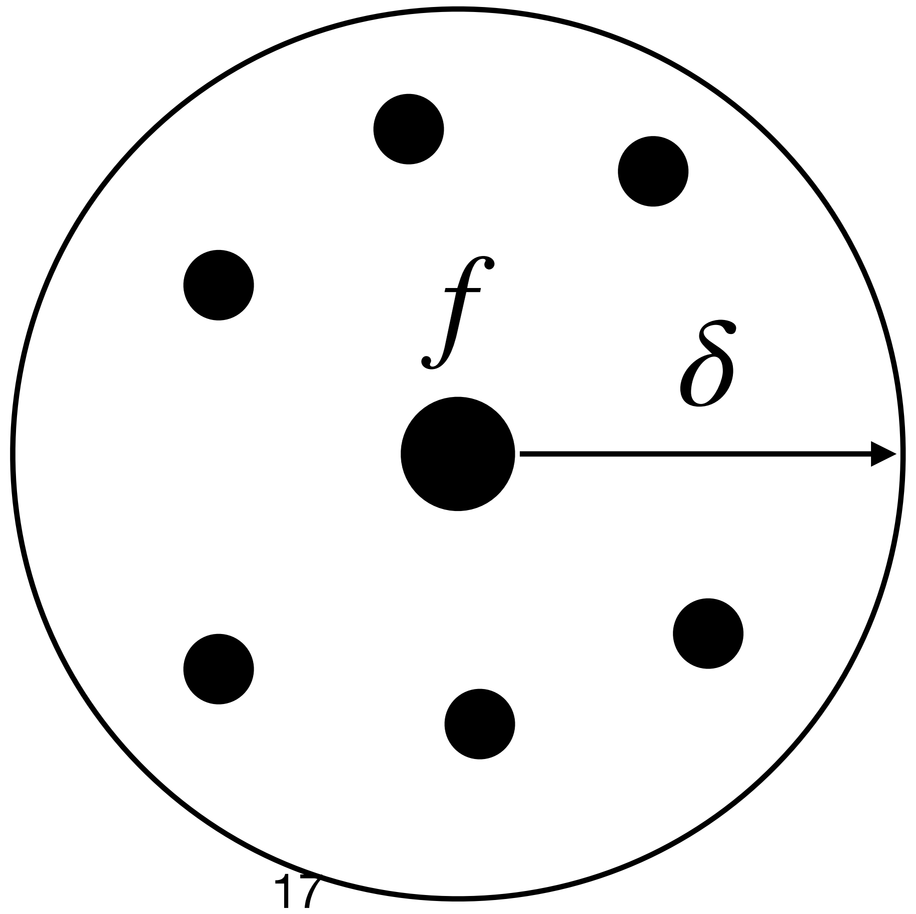
Ans $: S \to \mathbb{F}$ be a list of (claimed) evaluations of (the extension of) f on S

$$\mathrm{Quotient}(f, \mathrm{Ans})(x) := \frac{f(x) - \hat{\mathrm{Ans}}(x)}{V_S(x)}$$

Local

$\mathbf{V}$ can compute Quotient(f, Ans) at $x \in L \backslash S$ by querying f at x

Consistency



Quotienting

Enforce constraints on f or amplify distance

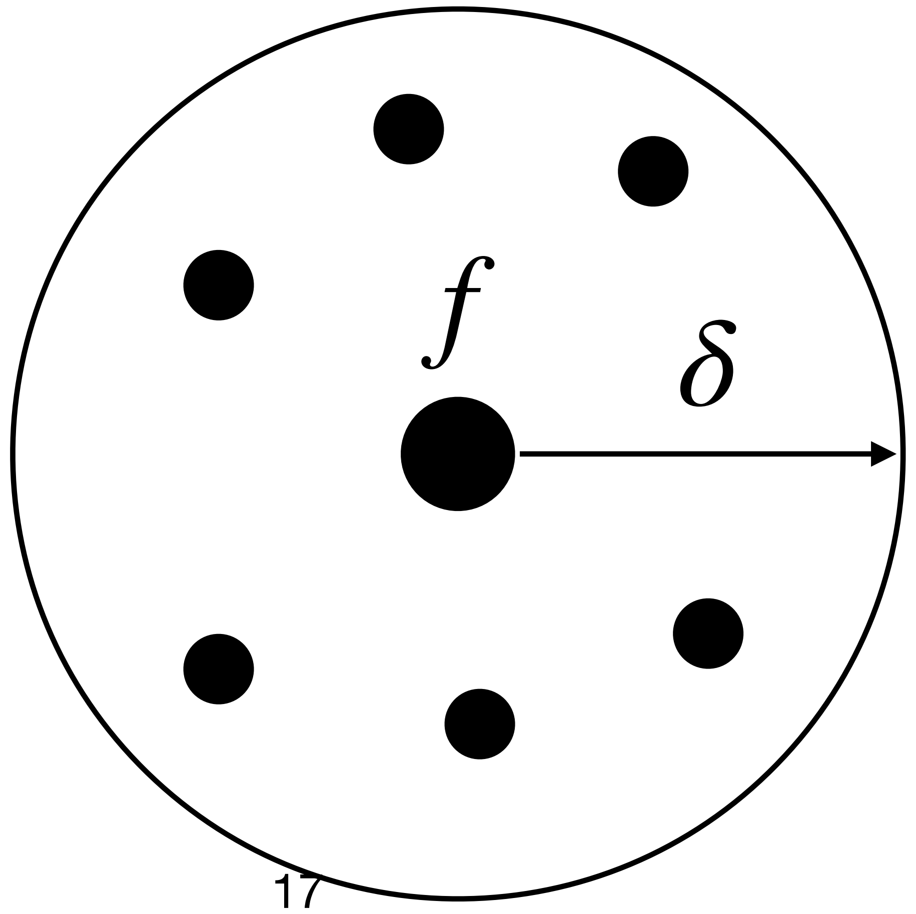
Let:
$f : L \to \mathbb{F}$ be a function
$\mathsf{Ans} : S \to \mathbb{F}$ be a list of (claimed) evaluations of (the extension of) f on S

$$\mathsf{Quotient}(f, \mathsf{Ans})(x) := \frac{f(x) - \hat{\mathsf{Ans}}(x)}{V_S(x)}$$

Local

$\mathbf{V}$ can compute $\mathsf{Quotient}(f, \mathsf{Ans})$ at $x \in L \backslash S$ by querying f at x

Consistency



If every $\hat{v} \in \mathsf{List}(f, d, \delta)$ has $\hat{v}\,|_S \not\equiv \mathsf{Ans}$ then $\mathsf{Quotient}(f, \mathsf{Ans})$ is δ-far from RS

Domain shifting in unique decoding

Let $\delta^* := \dfrac{1-\rho^*}{2}, L \cap L^* = \varnothing$.

Domain shifting in unique decoding

Let $\delta^* := \dfrac{1-\rho^*}{2}, L \cap L^* = \varnothing$.

At this distance, **at most one** codeword is close.

Domain shifting in unique decoding

Let $\delta^* := \dfrac{1-\rho^*}{2}, L \cap L^* = \varnothing$.

Domain shifting in unique decoding

Let $\delta^* := \dfrac{1-\rho^*}{2}, L \cap L^* = \emptyset$.

Want to test f on L

Domain shifting in unique decoding

Let $\delta^* := \frac{1-\rho^*}{2}, L \cap L^* = \emptyset$.

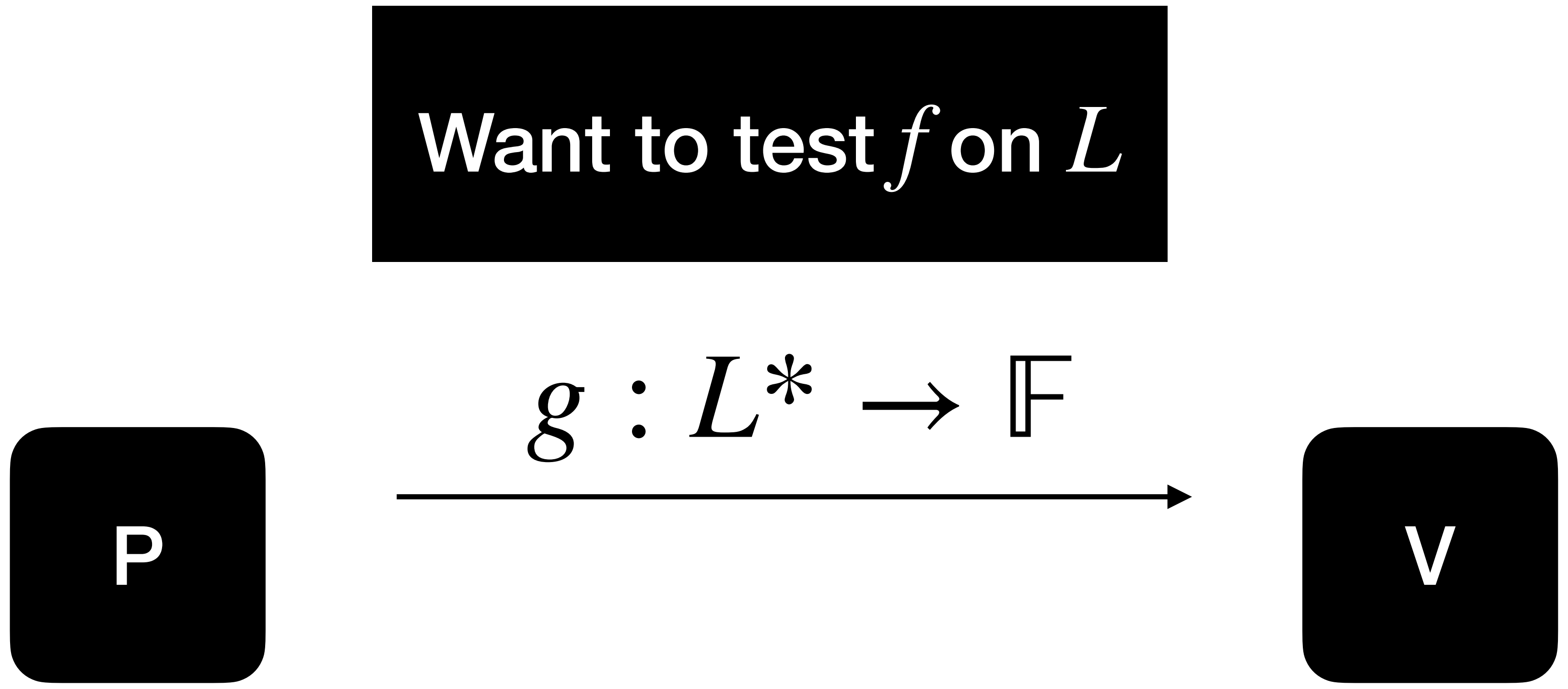
Want to test f on L

P

V

Domain shifting in unique decoding

Let $\delta^* := \dfrac{1-\rho^*}{2}, L \cap L^* = \emptyset$.



Want to test f on L

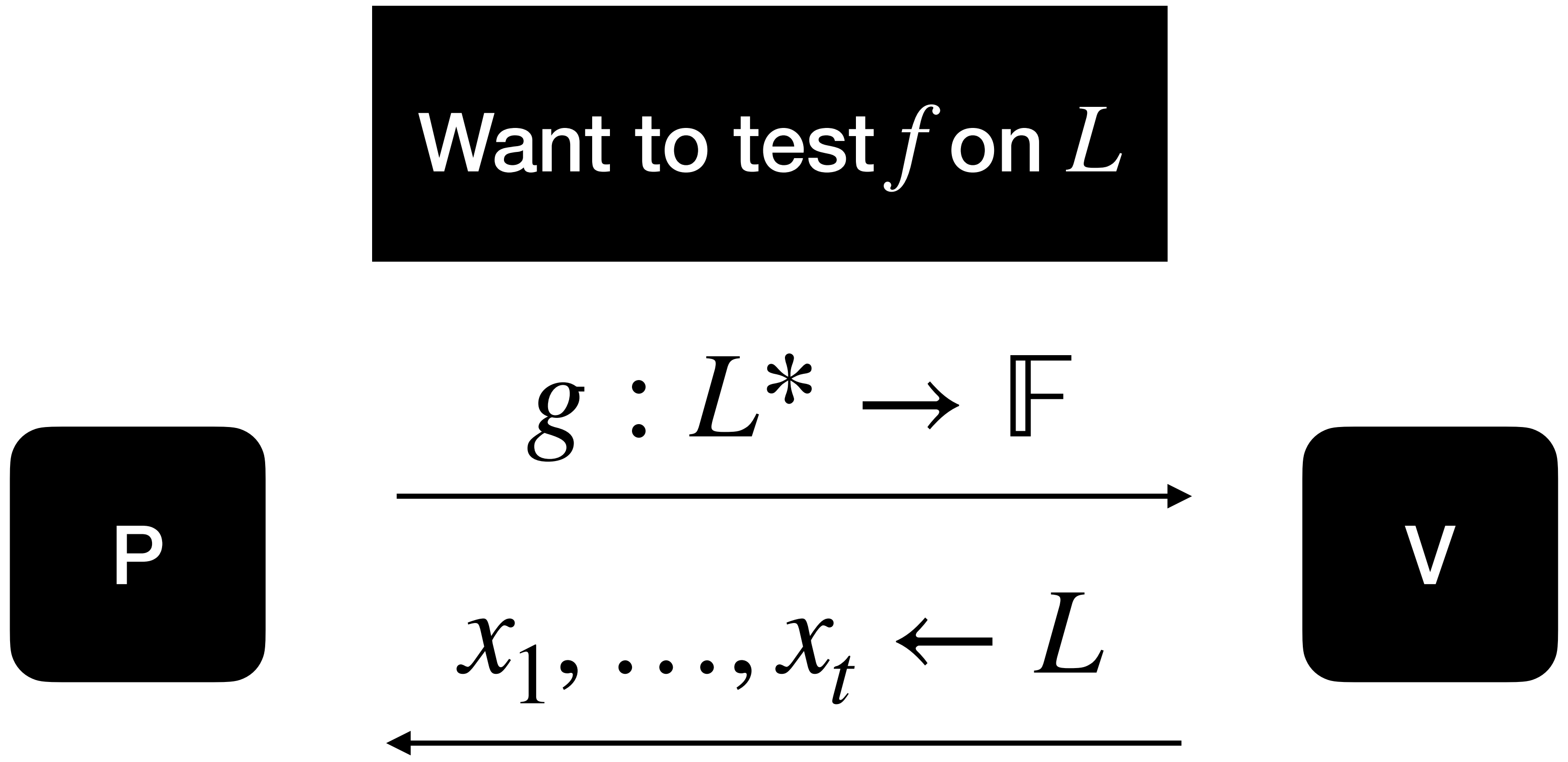
P

$g : L^* \to \mathbb{F}$

V

Domain shifting in unique decoding

Let $\delta^* := \frac{1 - \rho^*}{2}, L \cap L^* = \emptyset$.



Want to test f on L

P

$g : L^* \to \mathbb{F}$

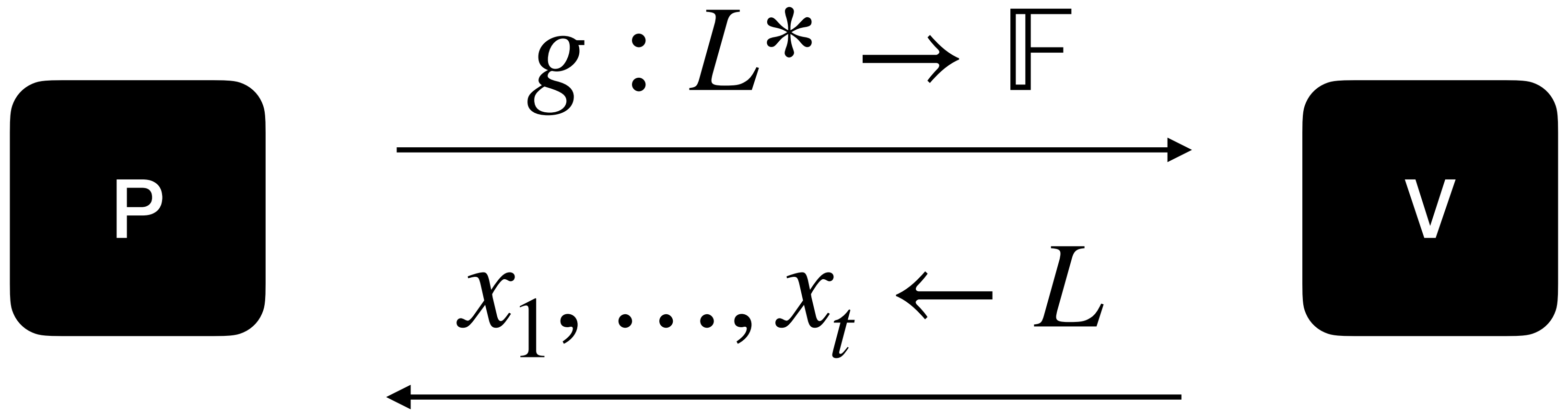
$x_1, \ldots, x_t \leftarrow L$

V

Domain shifting in unique decoding

Let $\delta^* := \dfrac{1 - \rho^*}{2}, L \cap L^* = \emptyset$.

Want to test f on L



P

$g : L^* \to \mathbb{F}$

V

$x_1, \ldots, x_t \leftarrow L$

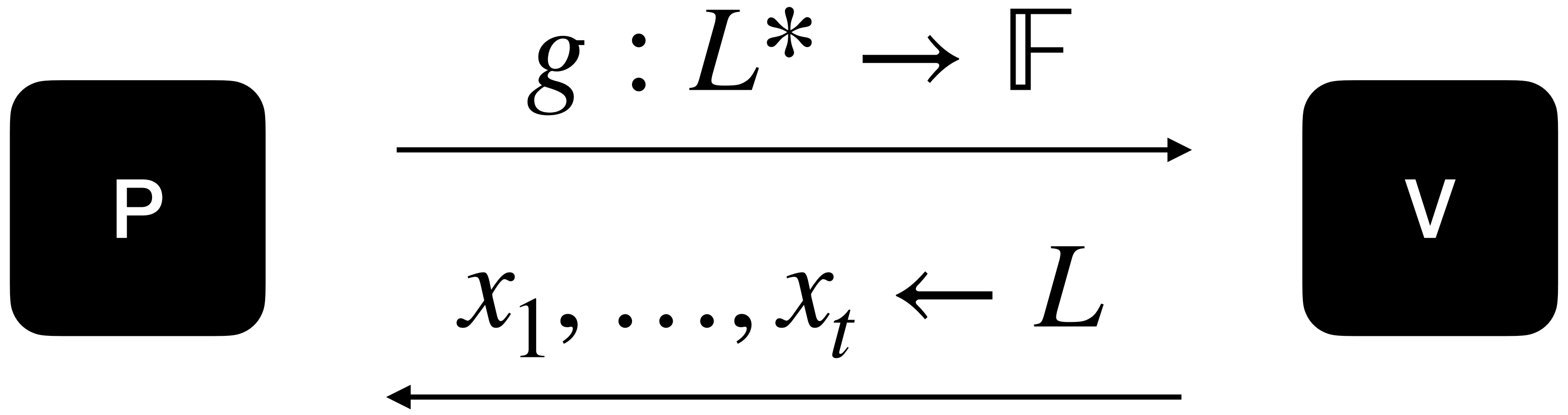
Query $y_i = f(x_i)$

$\forall\ i : \mathsf{Ans}(x_i) = y_i$

Domain shifting in unique decoding

Let $\delta^* := \dfrac{1-\rho^*}{2}$, $L \cap L^* = \emptyset$.

Want to test f on L



Query $y_i = f(x_i)$

$\forall\ i : \mathsf{Ans}(x_i) = y_i$

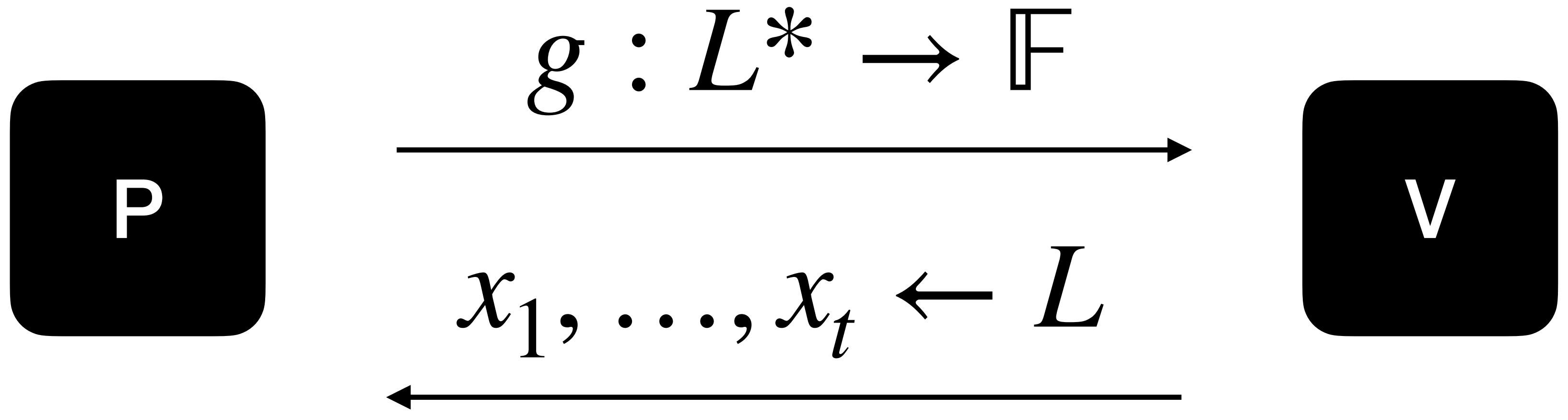
$g^* := \mathsf{Quotient}(g, \mathsf{Ans})$

Test g^* on L^*

Domain shifting in unique decoding

Let $\delta^* := \dfrac{1 - \rho^*}{2}, L \cap L^* = \varnothing$.

Want to test f on L



P

$g : L^* \to \mathbb{F}$

$x_1, \ldots, x_t \leftarrow L$

V

Query $y_i = f(x_i)$

$\forall\ i :$ Ans$(x_i) = y_i$

$g^* :=$ Quotient$(g,$ Ans$)$

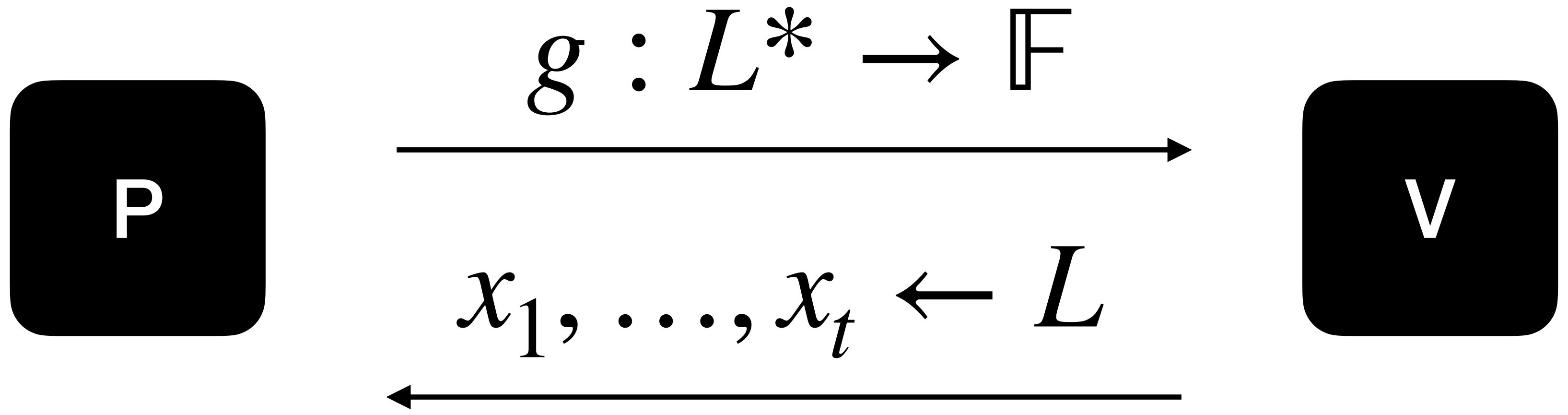
Test g^* on L^*

Since in unique decoding, there is **unique** $\hat{v}$ close to g, and:

Domain shifting in unique decoding

Let $\delta^* := \frac{1 - \rho^*}{2}, L \cap L^* = \emptyset$.

Want to test f on L



Query $y_i = f(x_i)$

$\forall\ i : \text{Ans}(x_i) = y_i$

$g^* := \text{Quotient}(g, \text{Ans})$

Test g^* on L^*

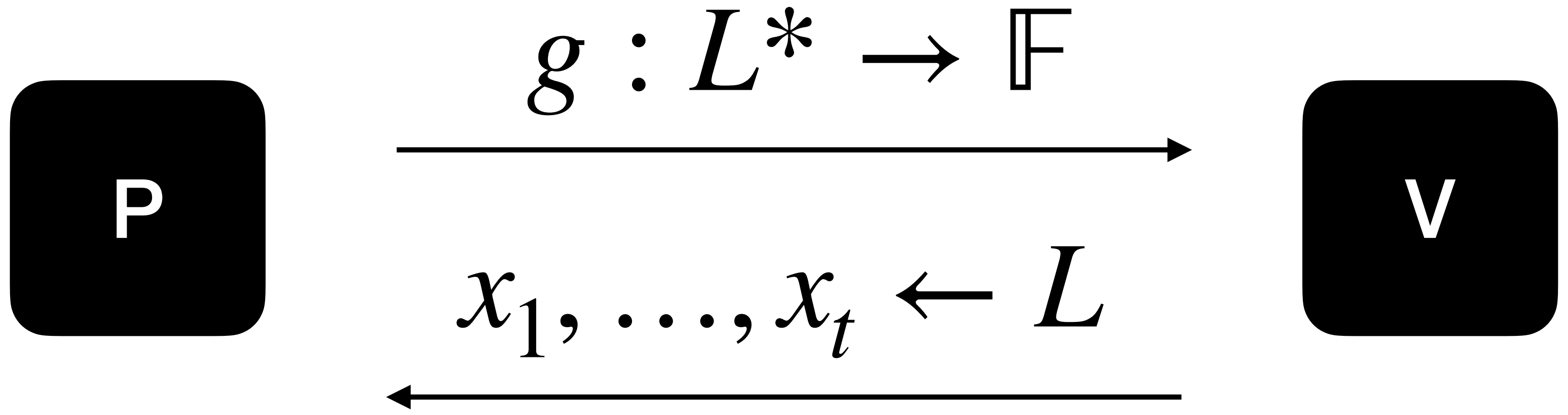
Since in unique decoding, there is **unique** $\hat{v}$ close to g, and:

If at any point $\hat{v}(x_i) \neq y_i$ then, by **quotients**, g^* is δ^*-far from C^*

Domain shifting in unique decoding

Let $\delta^* := \dfrac{1-\rho^*}{2}, L \cap L^* = \emptyset.$

Want to test f on L



P

$g : L^* \to \mathbb{F}$

V

$x_1, \ldots, x_t \leftarrow L$

Query $y_i = f(x_i)$

$\forall\ i :$ Ans$(x_i) = y_i$

$g^* :=$ Quotient$(g,$ Ans$)$

Test g^* on L^*

Since in unique decoding, there is **unique** $\hat{v}$ close to g, and:

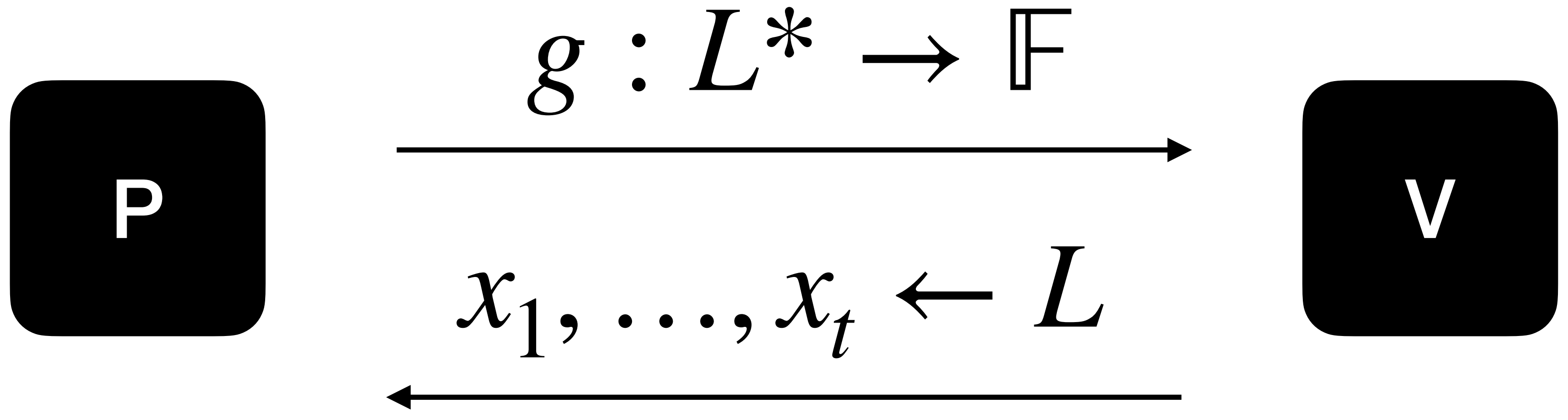
If at any point $\hat{v}(x_i) \neq y_i$ then, by **quotients**, g^* is δ^*-far from C^*

Since $\Delta(f, \hat{v}\,|_L) > \delta$, then:

Domain shifting in unique decoding

Let $\delta^* := \dfrac{1-\rho^*}{2}, L \cap L^* = \emptyset.$

Want to test f on L



P

$g : L^* \to \mathbb{F}$

V

$x_1, \ldots, x_t \leftarrow L$

Query $y_i = f(x_i)$

$\forall\ i : \text{Ans}(x_i) = y_i$

$g^* := \text{Quotient}(g, \text{Ans})$

Test g^* on $L*$

Since in unique decoding, there is **unique** $\hat{v}$ close to g, and:

If at any point $\hat{v}(x_i) \neq y_i$ then, by **quotients**, g^* is δ^*-far from C^*

Since $\Delta(f, \hat{v}\,|_L) > \delta$, then:

$$\begin{aligned}
&\Pr\left[g^* \text{ is } \delta^* \text{ close }\right] \\
&\leq \Pr\left[\forall i,\ \hat{v}(x_i) = y_i\right] \\
&= \Pr\left[\forall i,\ \hat{v}(x_i) = f(x_i)\right] \\
&\leq (1-\delta)^t
\end{aligned}$$

Out Of Domain sampling

Move to unique decoding range

Out Of Domain sampling

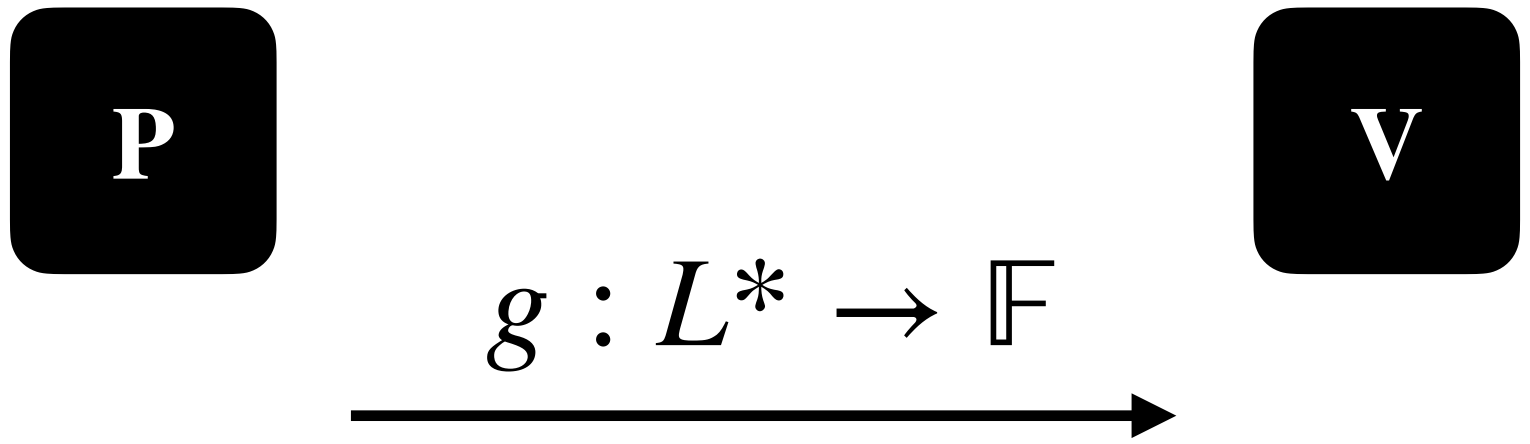
Move to unique decoding range

P

V

Out Of Domain sampling

Move to unique decoding range



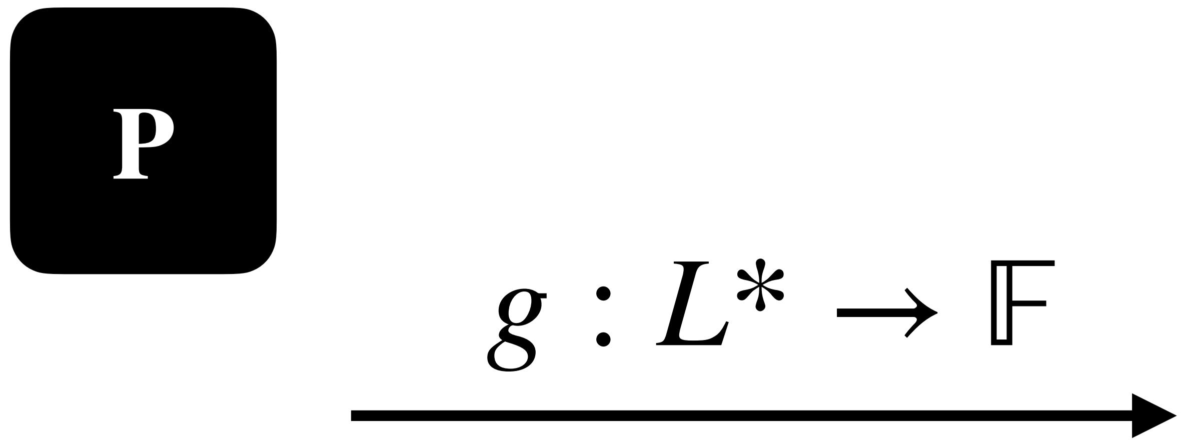
P

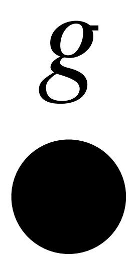
V

$g : L^* \to \mathbb{F}$

Out Of Domain sampling

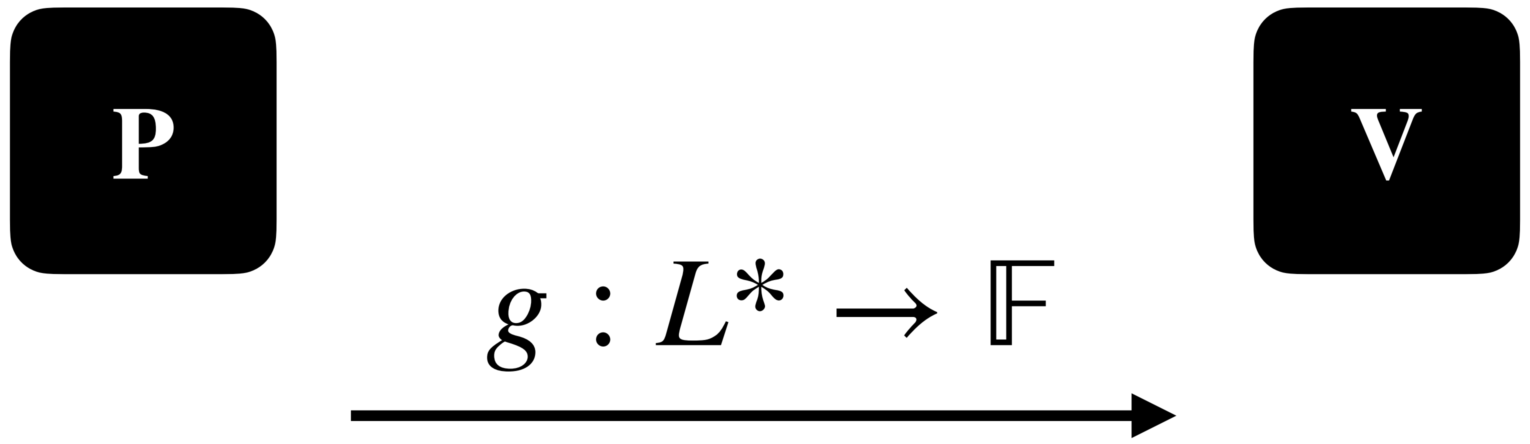
Move to unique decoding range

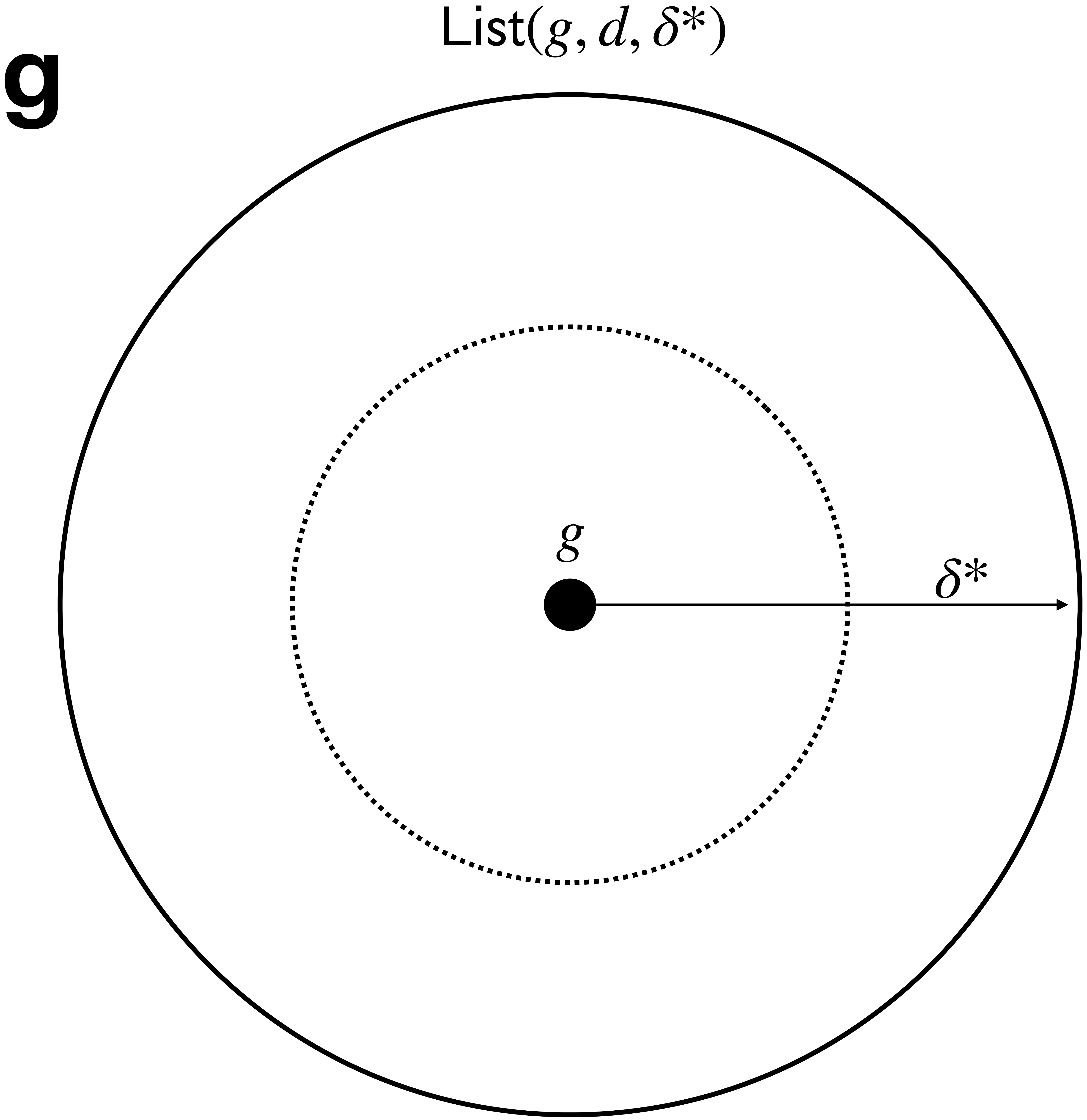




Out Of Domain sampling

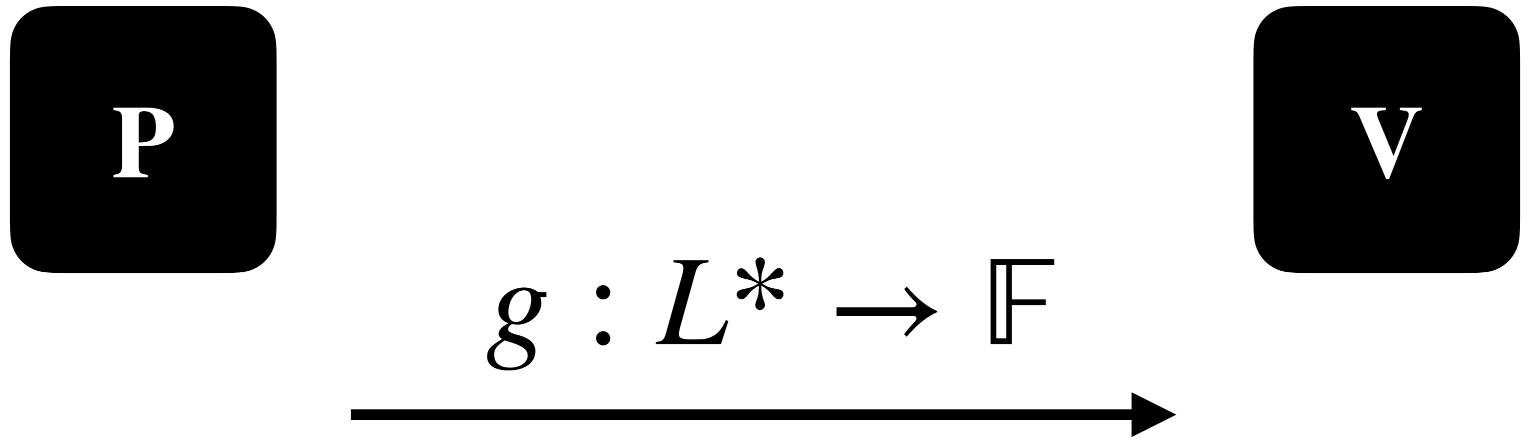
Move to unique decoding range

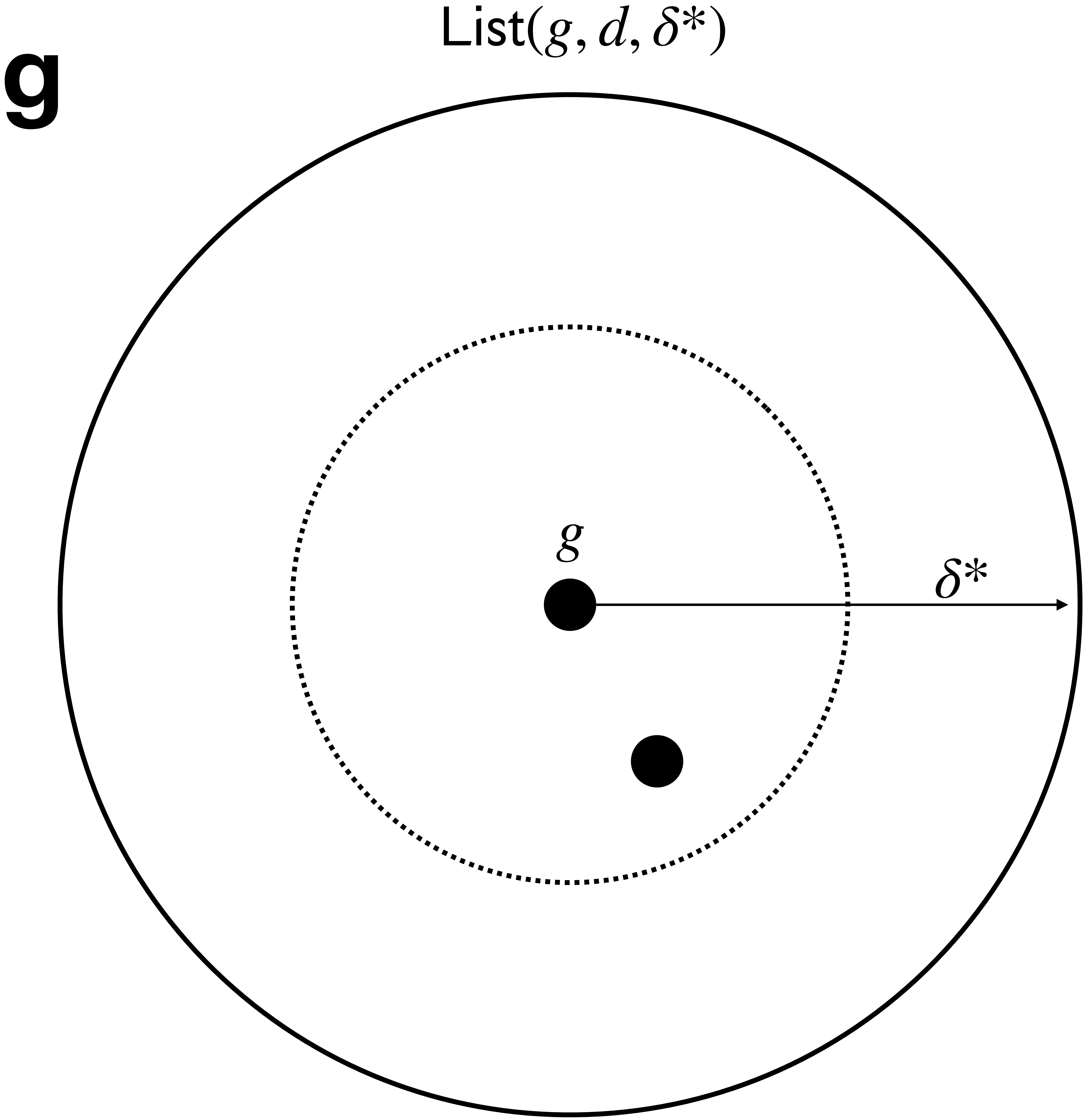




Out Of Domain sampling

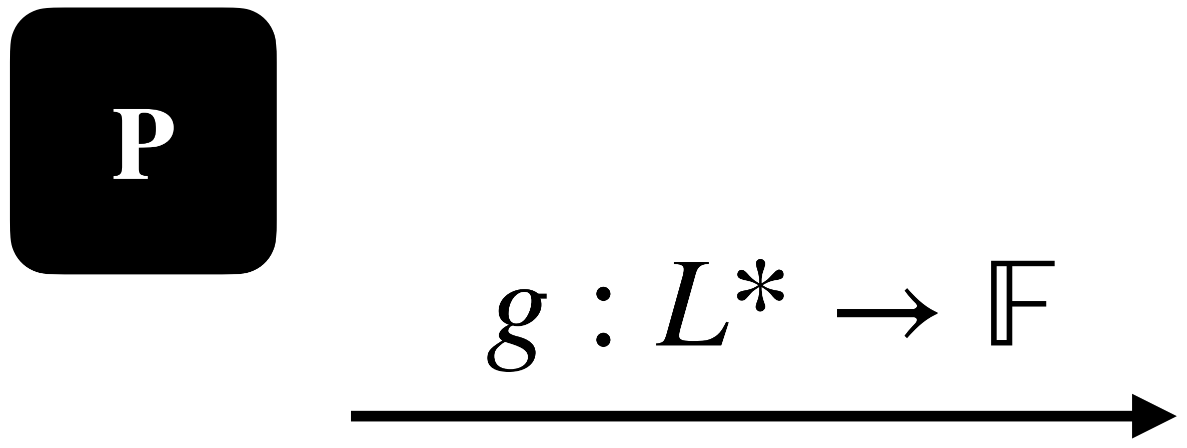
Move to unique decoding range

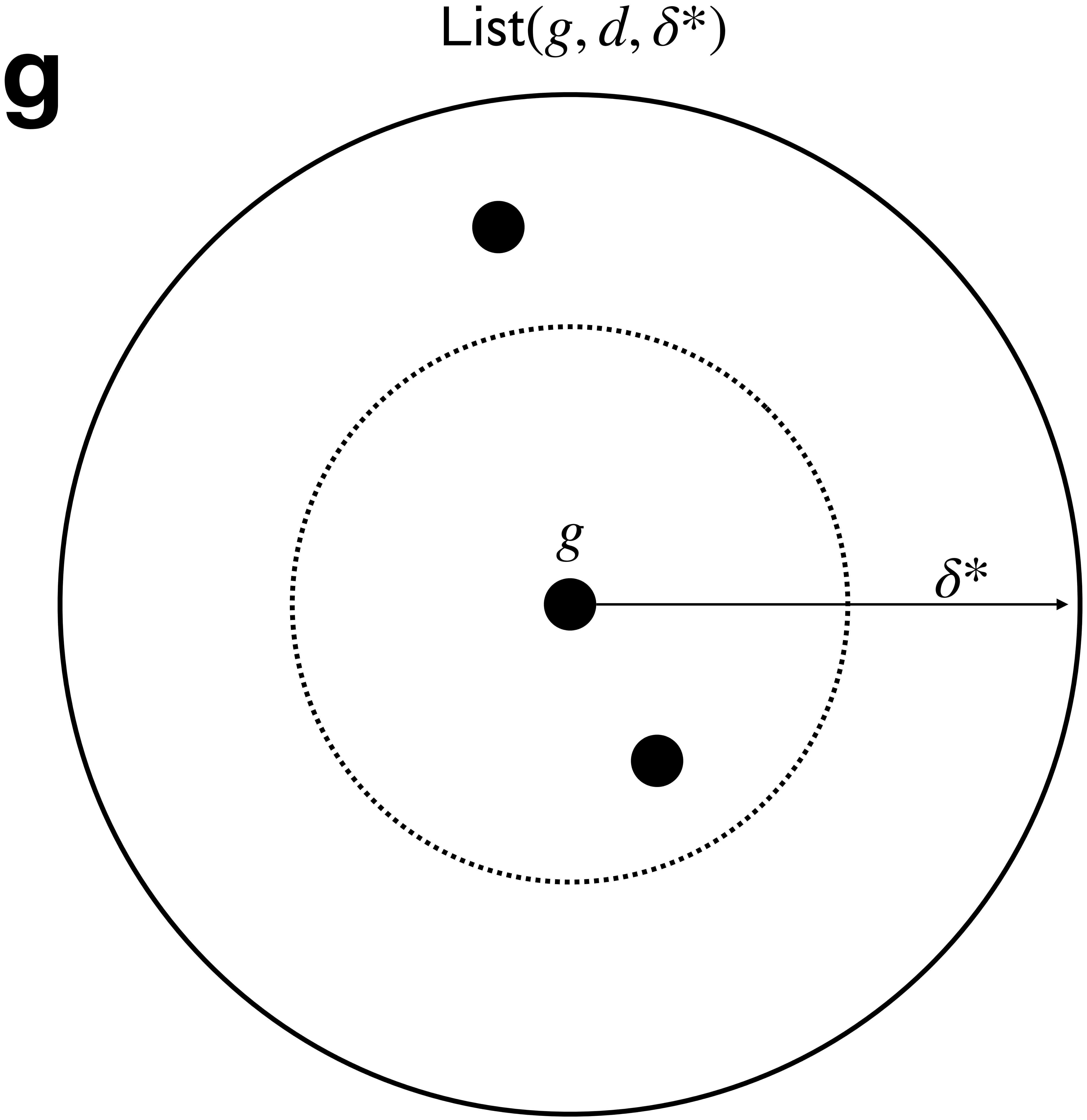




Out Of Domain sampling

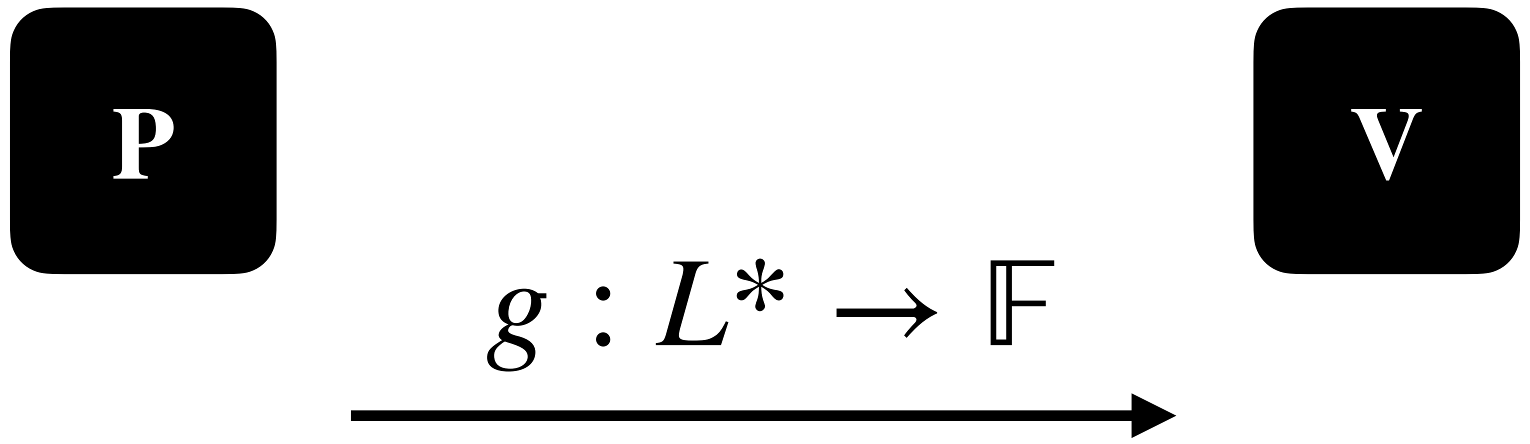
Move to unique decoding range



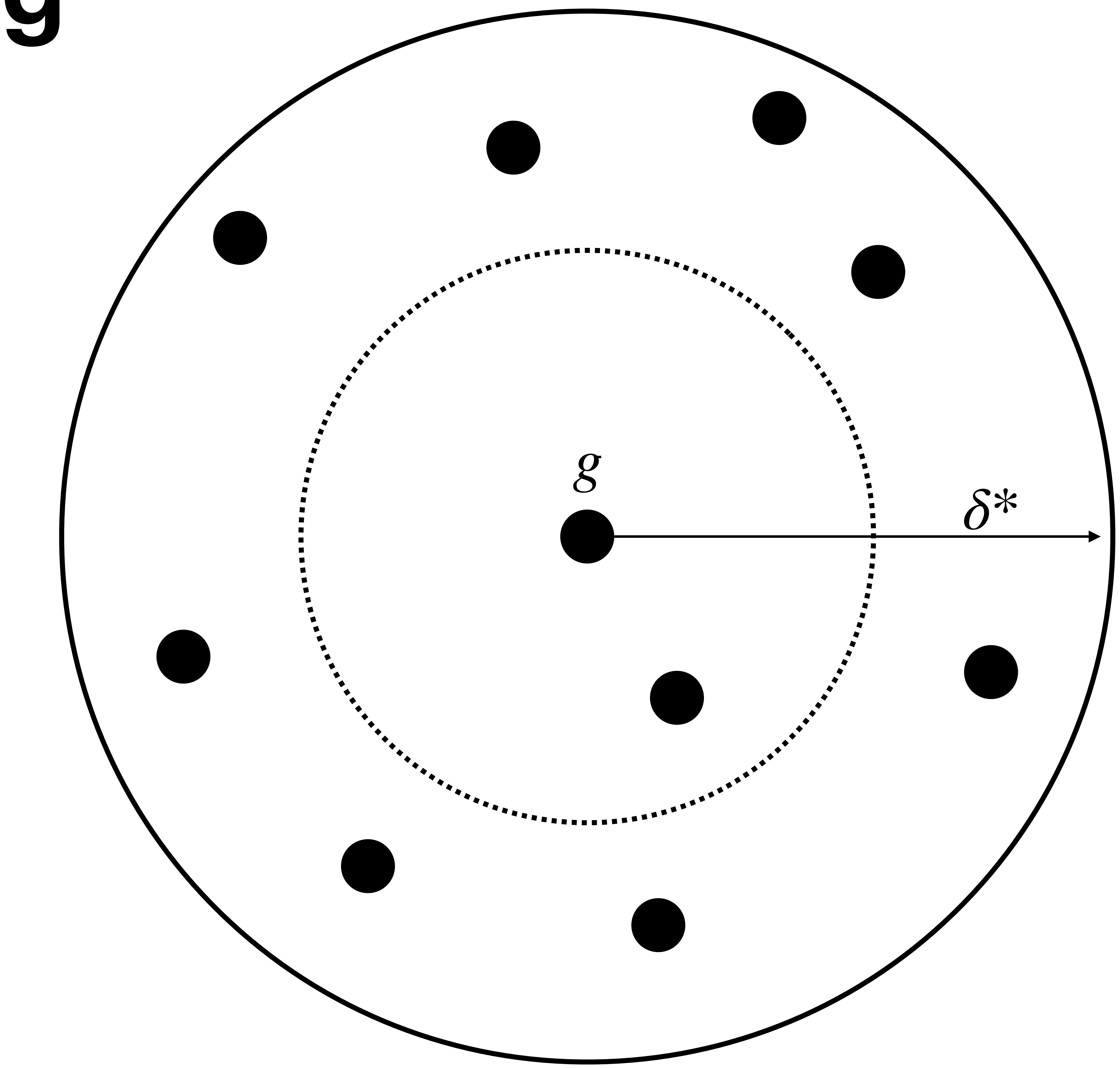


Out Of Domain sampling

Move to unique decoding range

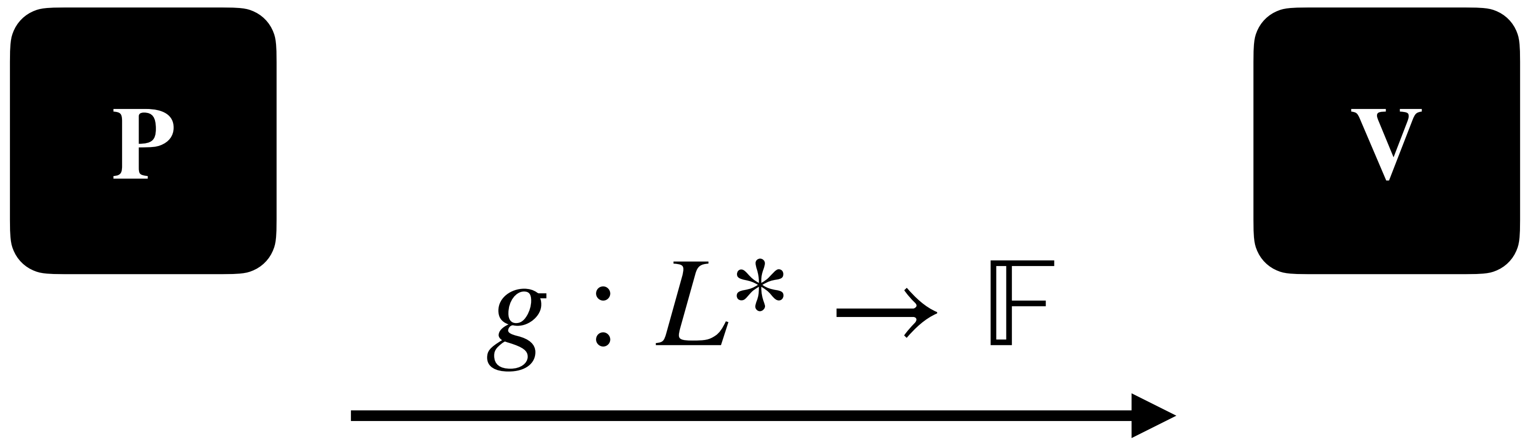


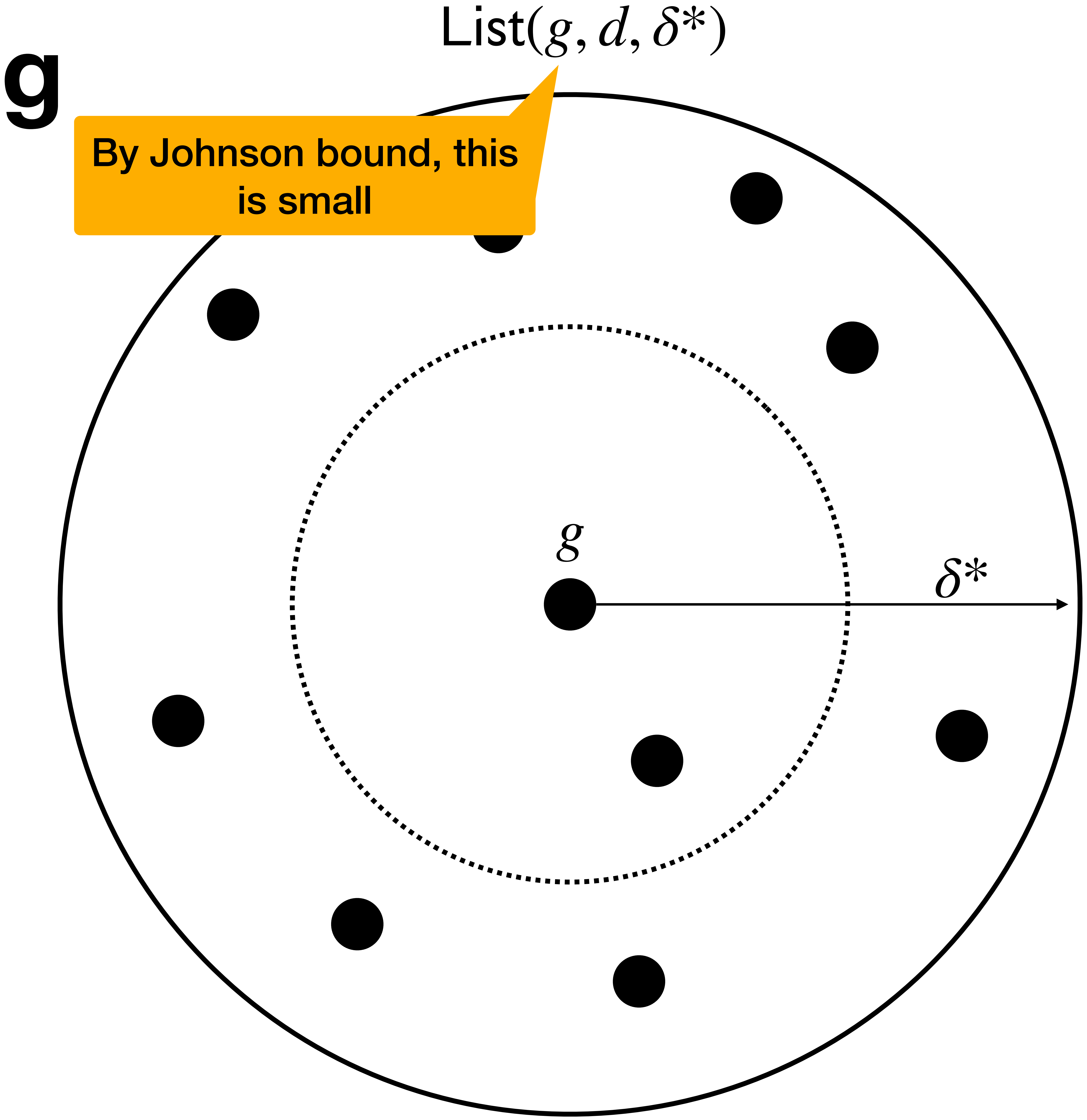
$\mathsf{List}(g, d, \delta^*)$



Out Of Domain sampling

Move to unique decoding range

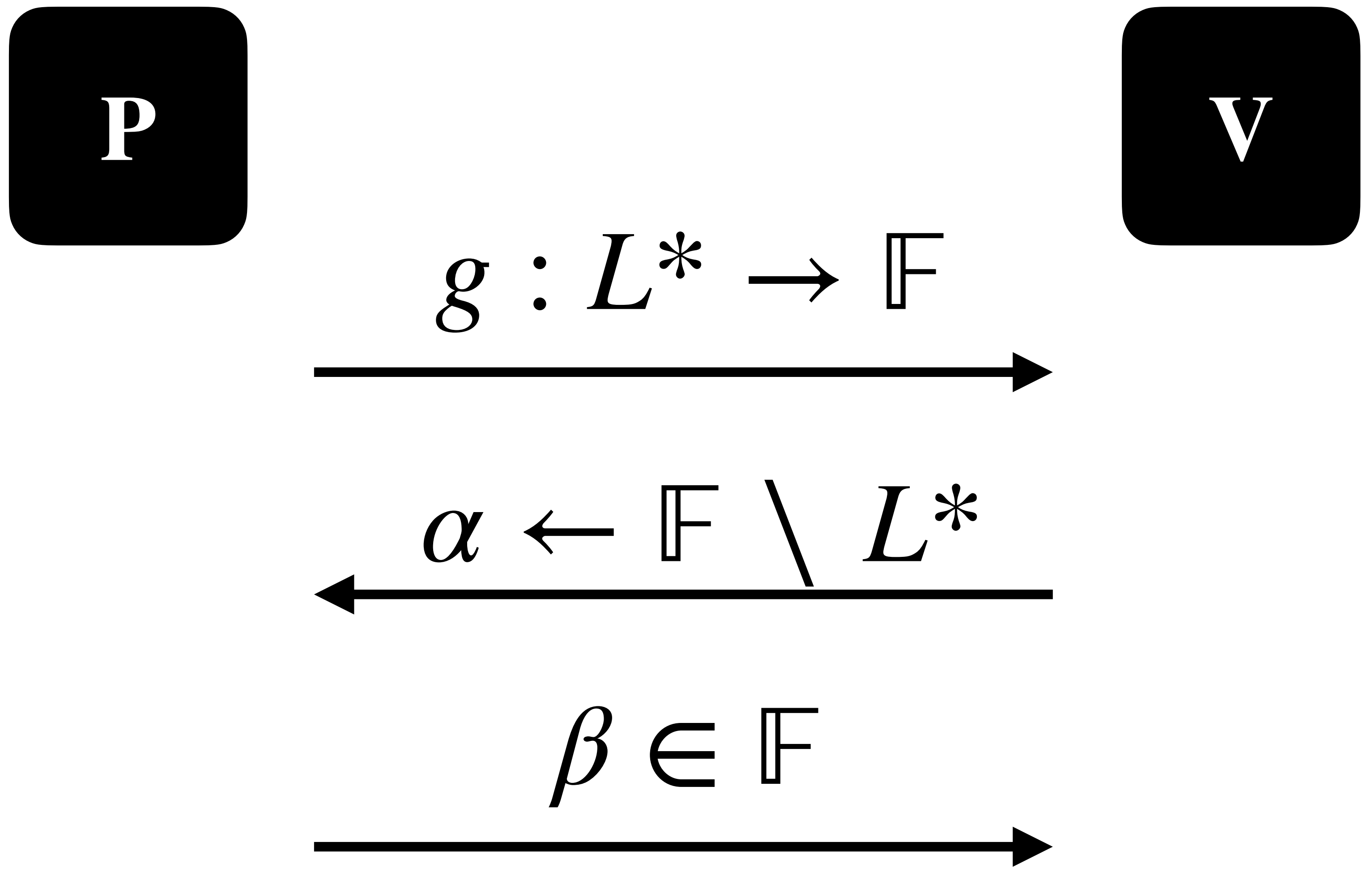




By Johnson bound, this is small

Out Of Domain sampling

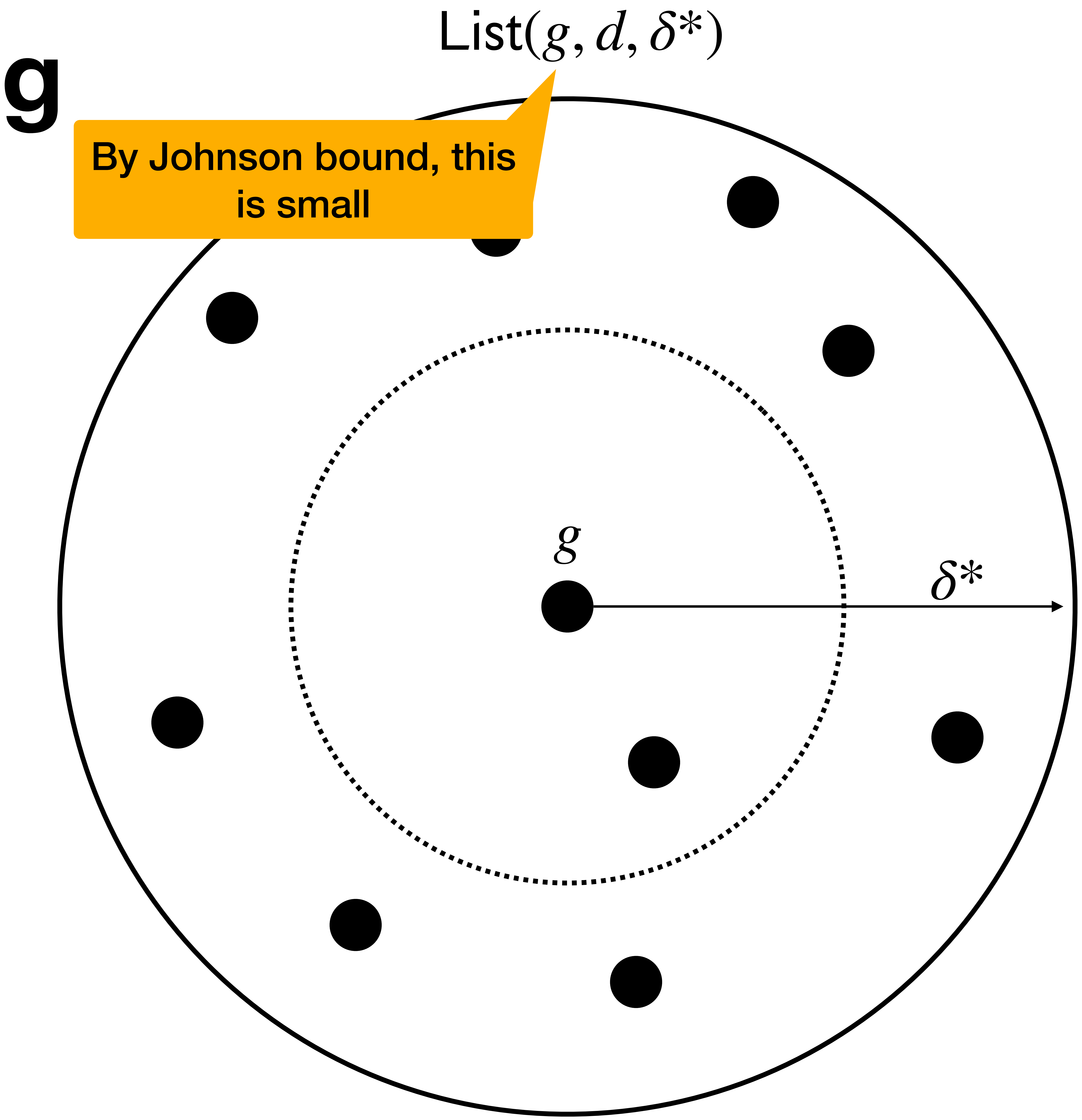
Move to unique decoding range



$g : L^* \to \mathbb{F}$

$\alpha \leftarrow \mathbb{F} \setminus L^*$

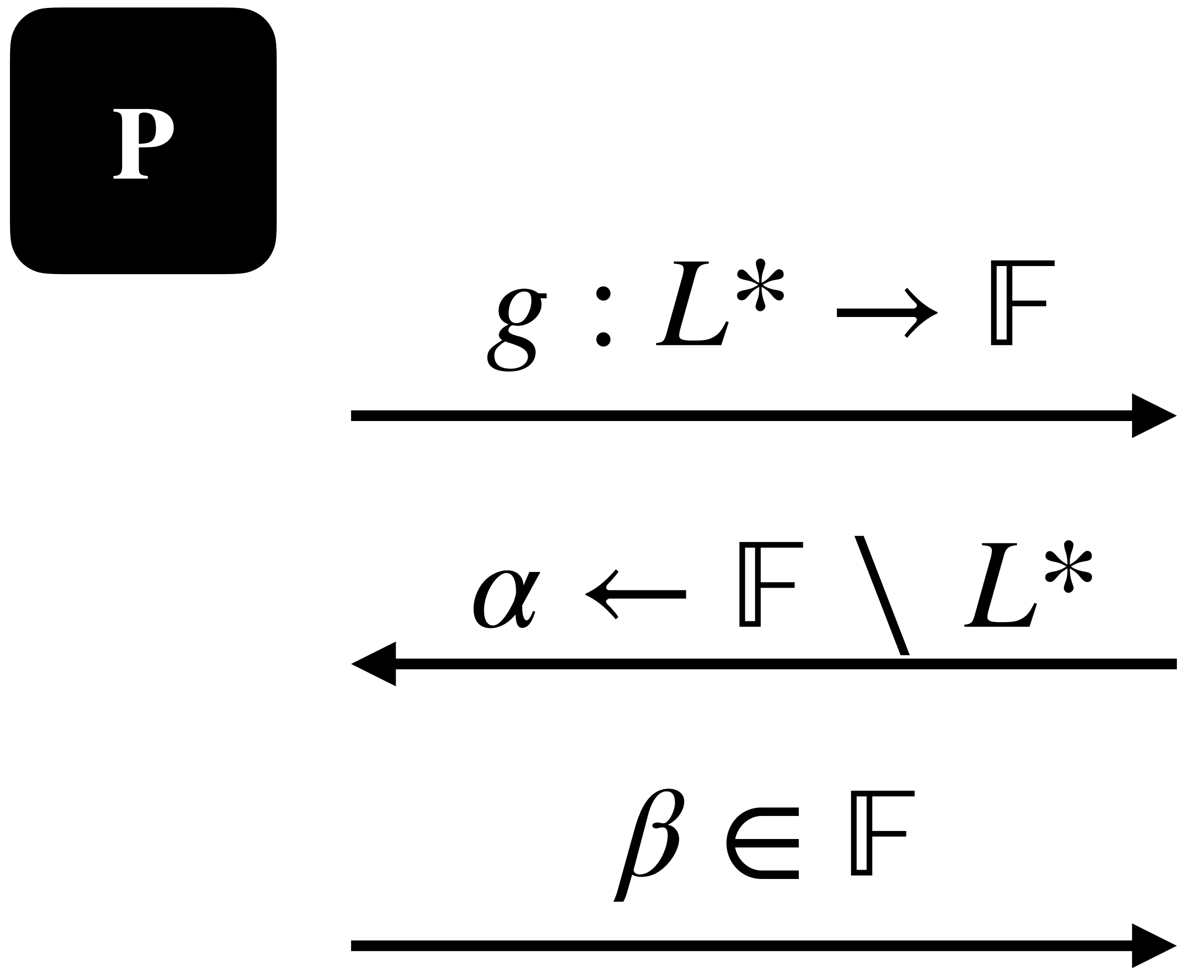
$\beta \in \mathbb{F}$



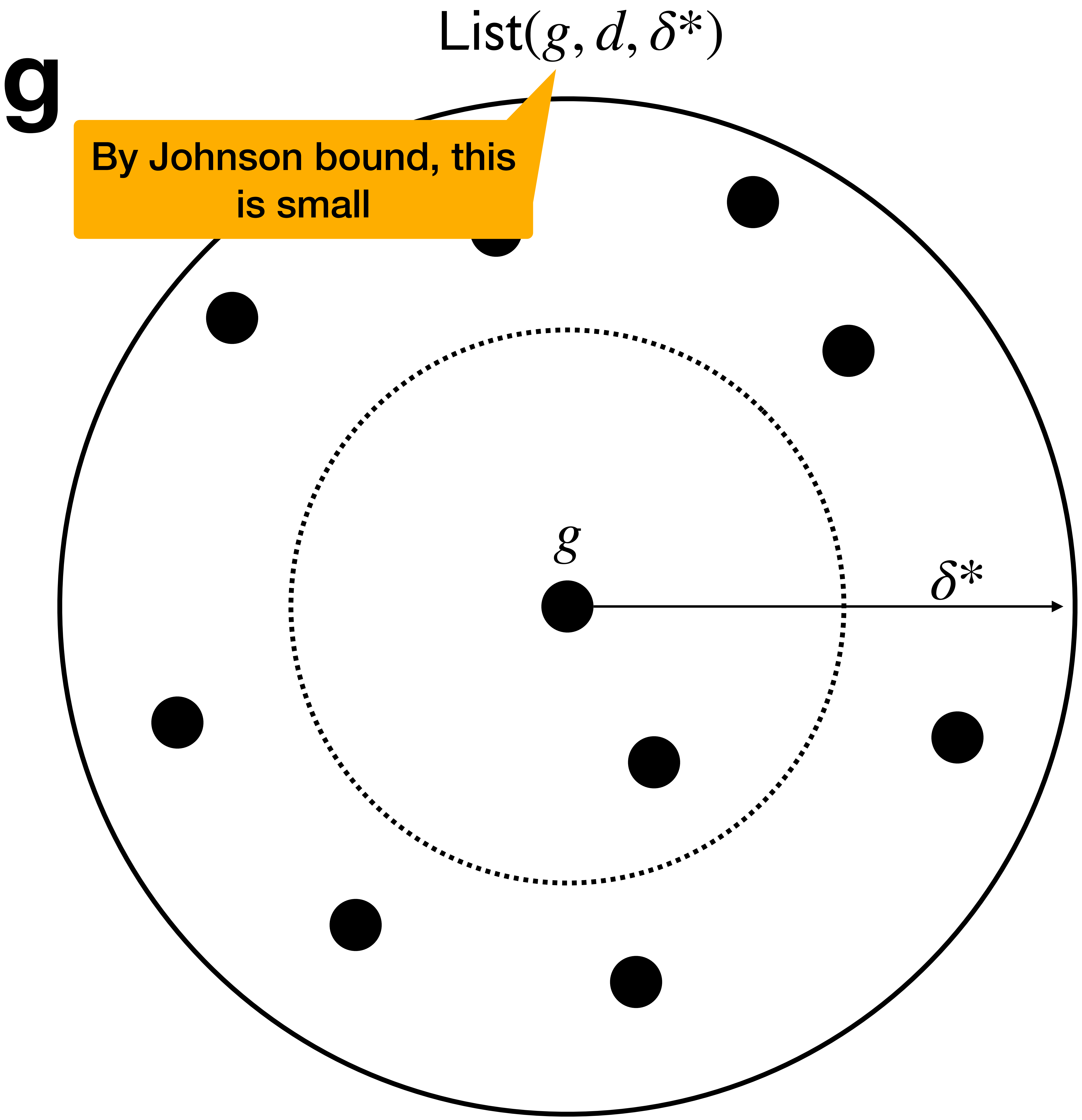
By Johnson bound, this is small

Out Of Domain sampling

Move to unique decoding range



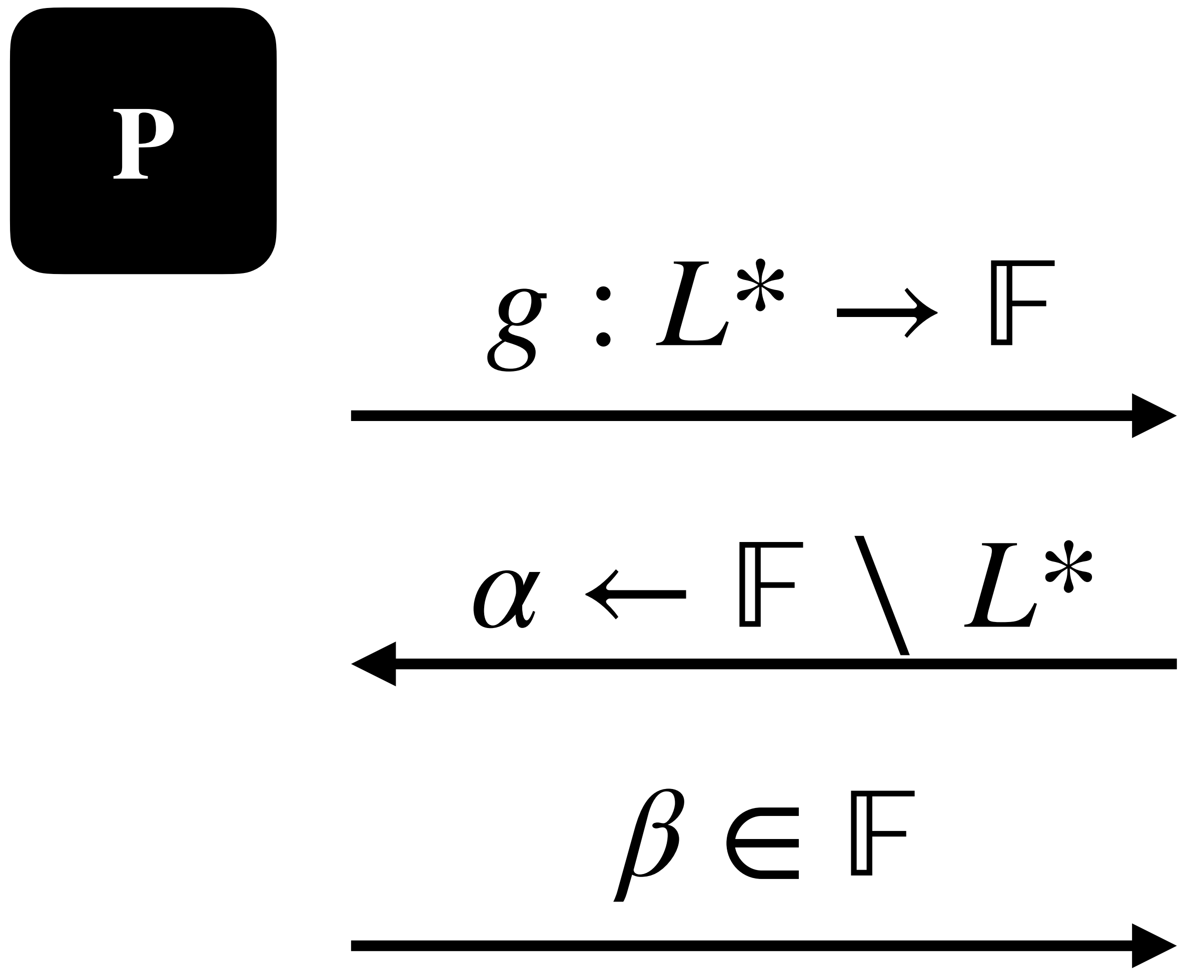
- By fundamental theorem of algebra of w.h.p. no pair $\hat{u}, \hat{v}$ with $\hat{u}(\alpha) = \hat{v}(\alpha)$
- Prover "chooses" which codeword $\hat{u}$ it "commits" to



By Johnson bound, this is small

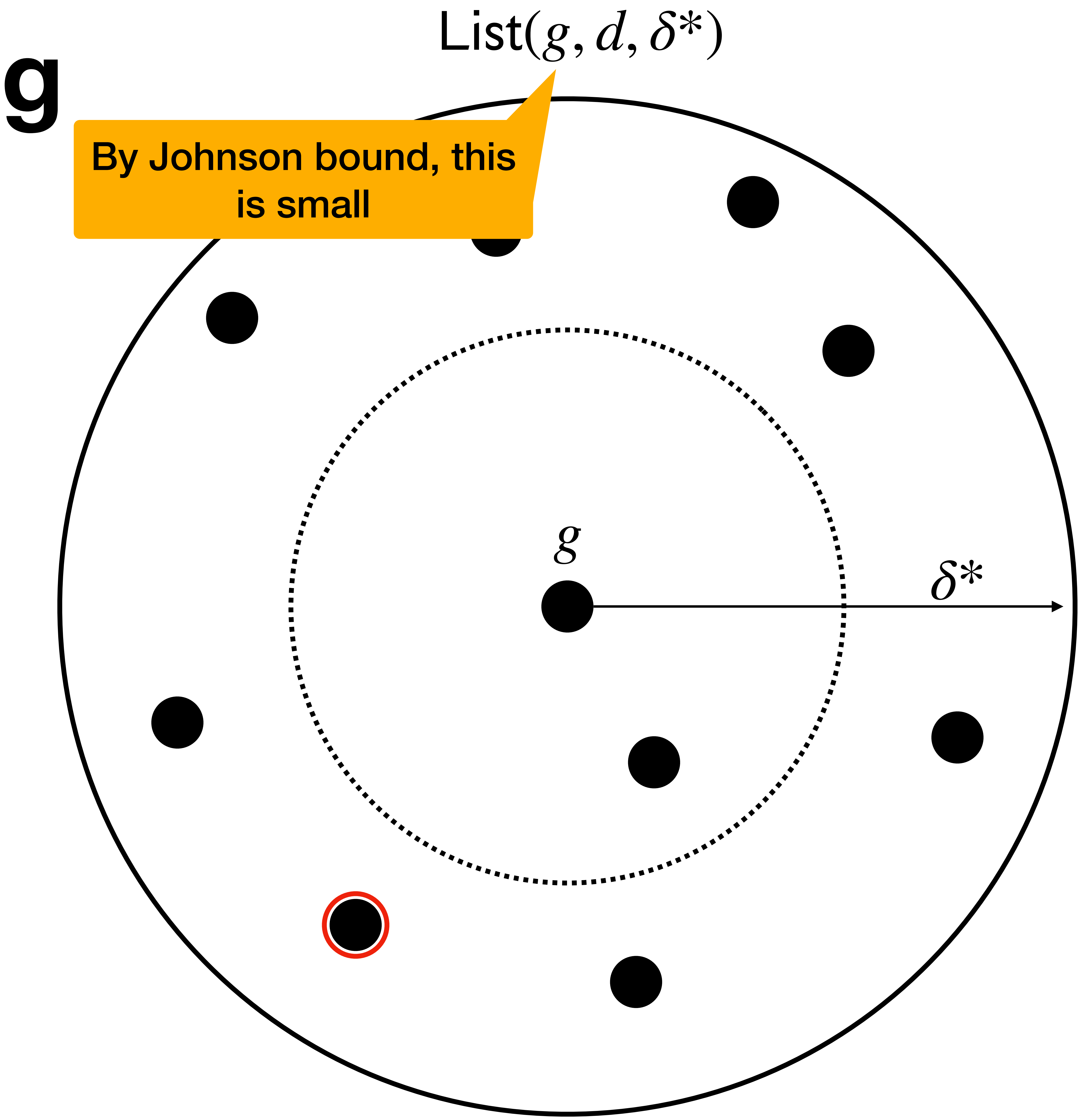
Out Of Domain sampling

Move to unique decoding range



- By fundamental theorem of algebra of w.h.p. no pair $\hat{u}, \hat{v}$ with $\hat{u}(\alpha) = \hat{v}(\alpha)$

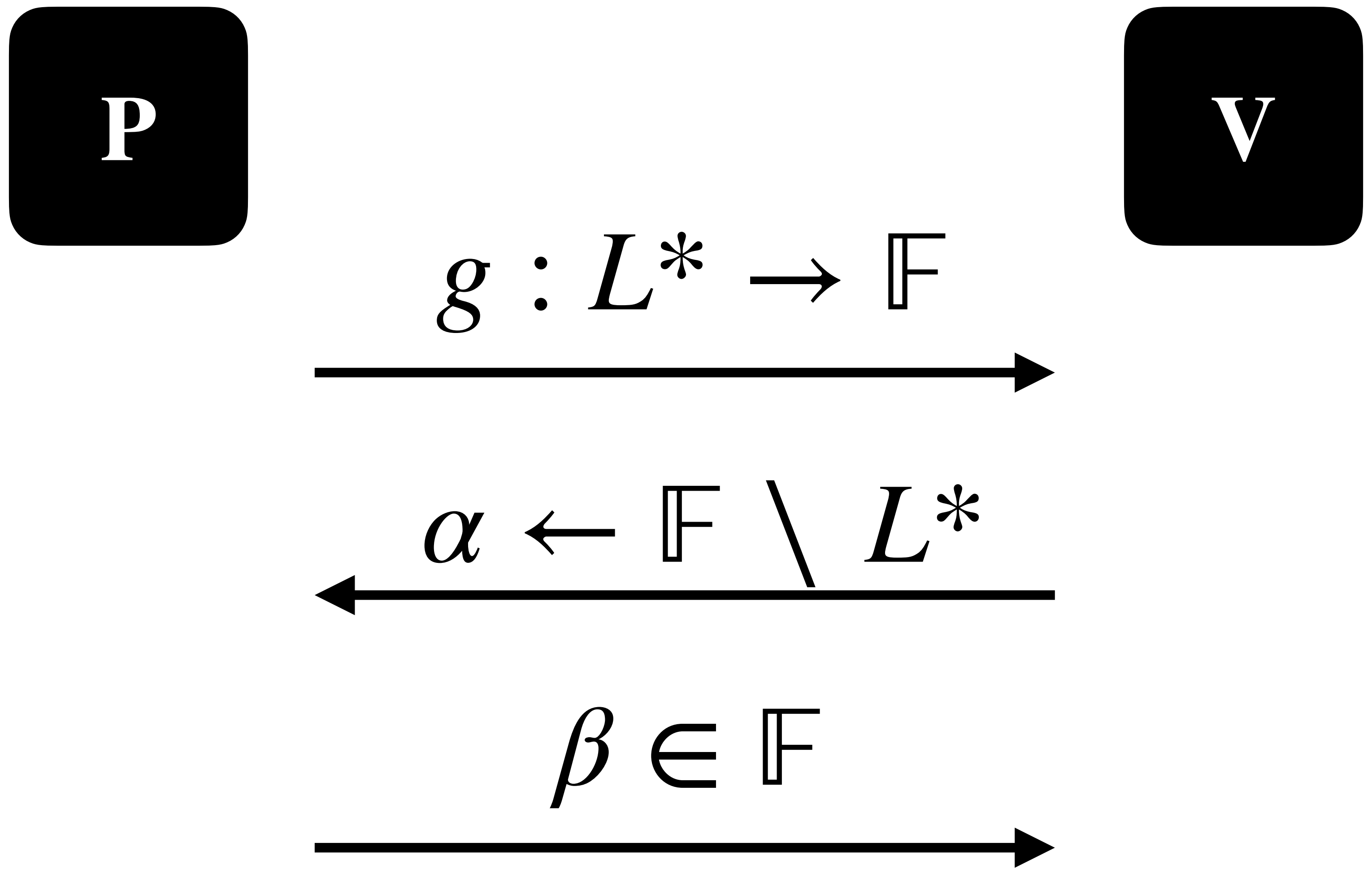
- Prover "chooses" which codeword $\hat{u}$ it "commits" to

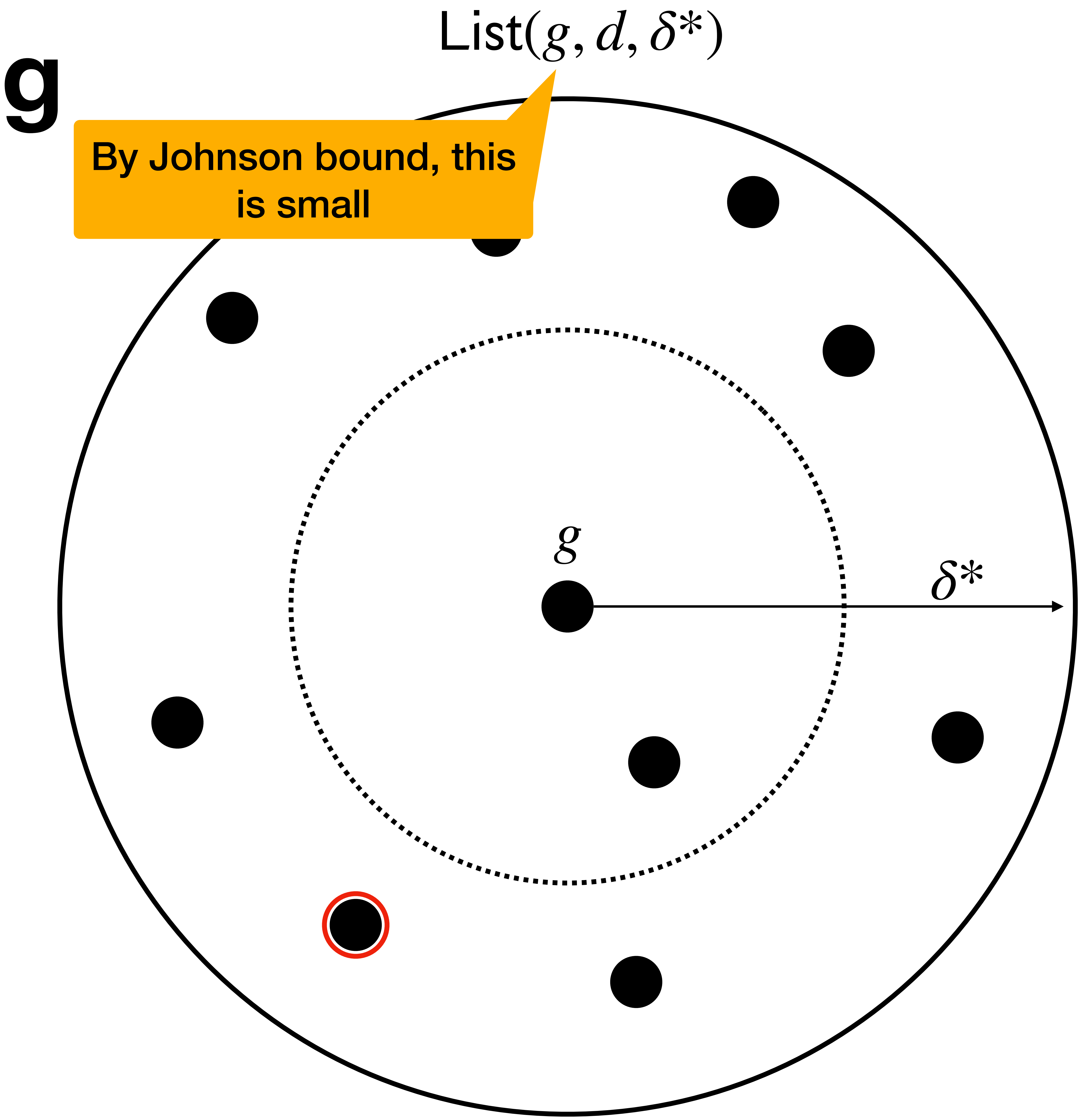


By Johnson bound, this is small

Out Of Domain sampling

Move to unique decoding range





By Johnson bound, this is small

- By fundamental theorem of algebra of w.h.p. no pair $\hat{u}, \hat{v}$ with $\hat{u}(\alpha) = \hat{v}(\alpha)$

- Prover "chooses" which codeword $\hat{u}$ it "commits" to

Use Quotient$(g, \alpha \mapsto \beta)$ to enforce the constraint

Domain shifting in list decoding

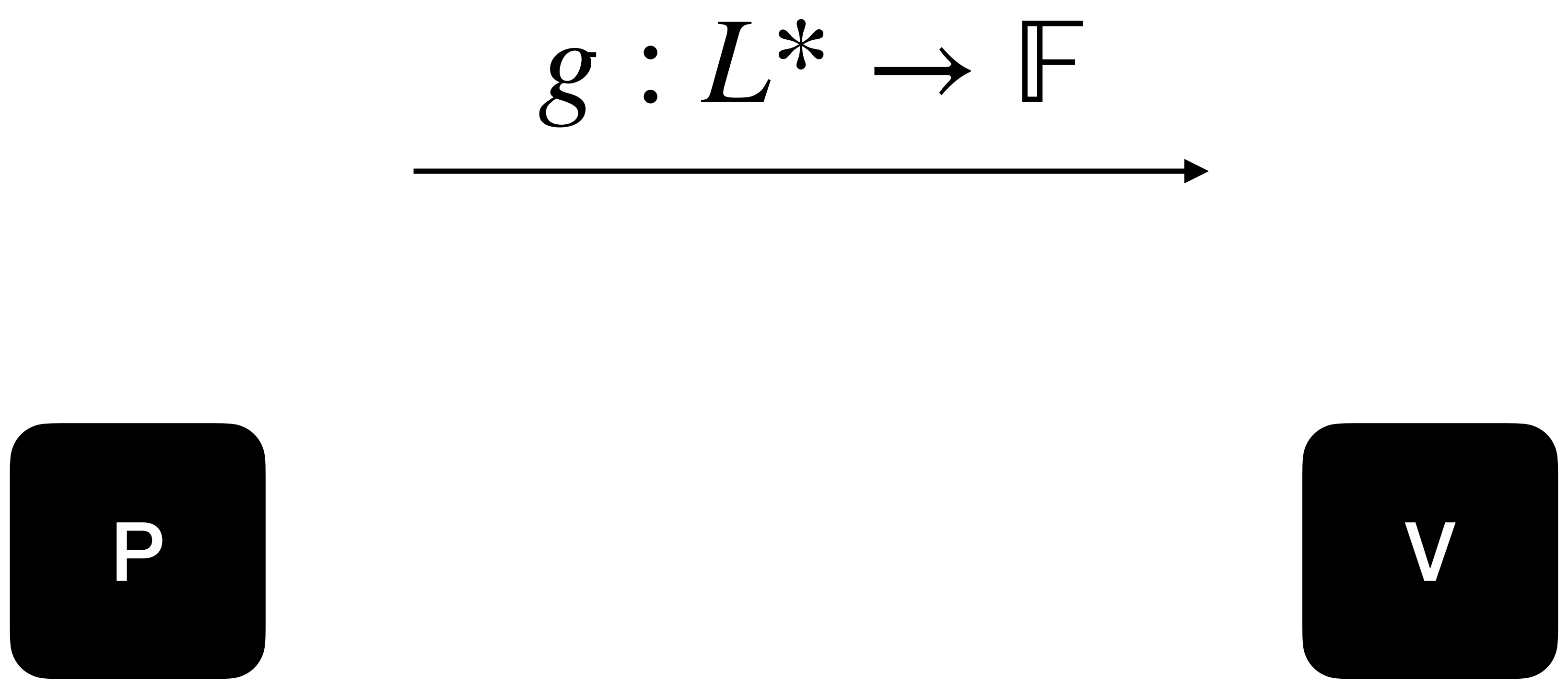
Let $\delta^* := 1 - \sqrt{\rho^*}, L \cap L^* = \varnothing$.

Domain shifting in list decoding

Let $\delta^* := 1 - \sqrt{\rho^*}$, $L \cap L^* = \varnothing$.

Domain shifting in list decoding

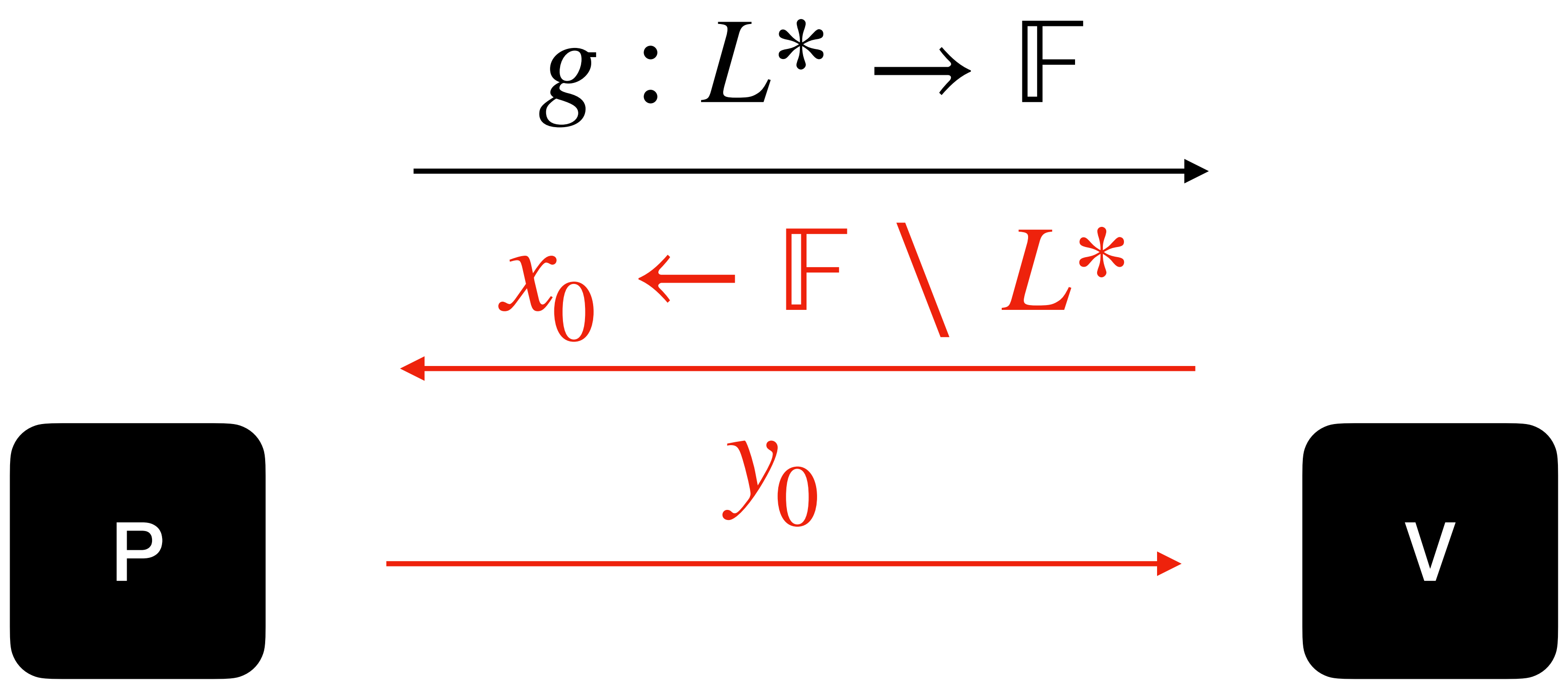
Let $\delta^* := 1 - \sqrt{\rho^*}, L \cap L^* = \varnothing$.



$g : L^* \to \mathbb{F}$

Domain shifting in list decoding

Let $\delta^* := 1 - \sqrt{\rho^*}$, $L \cap L^* = \varnothing$.



P

$g : L^* \to \mathbb{F}$

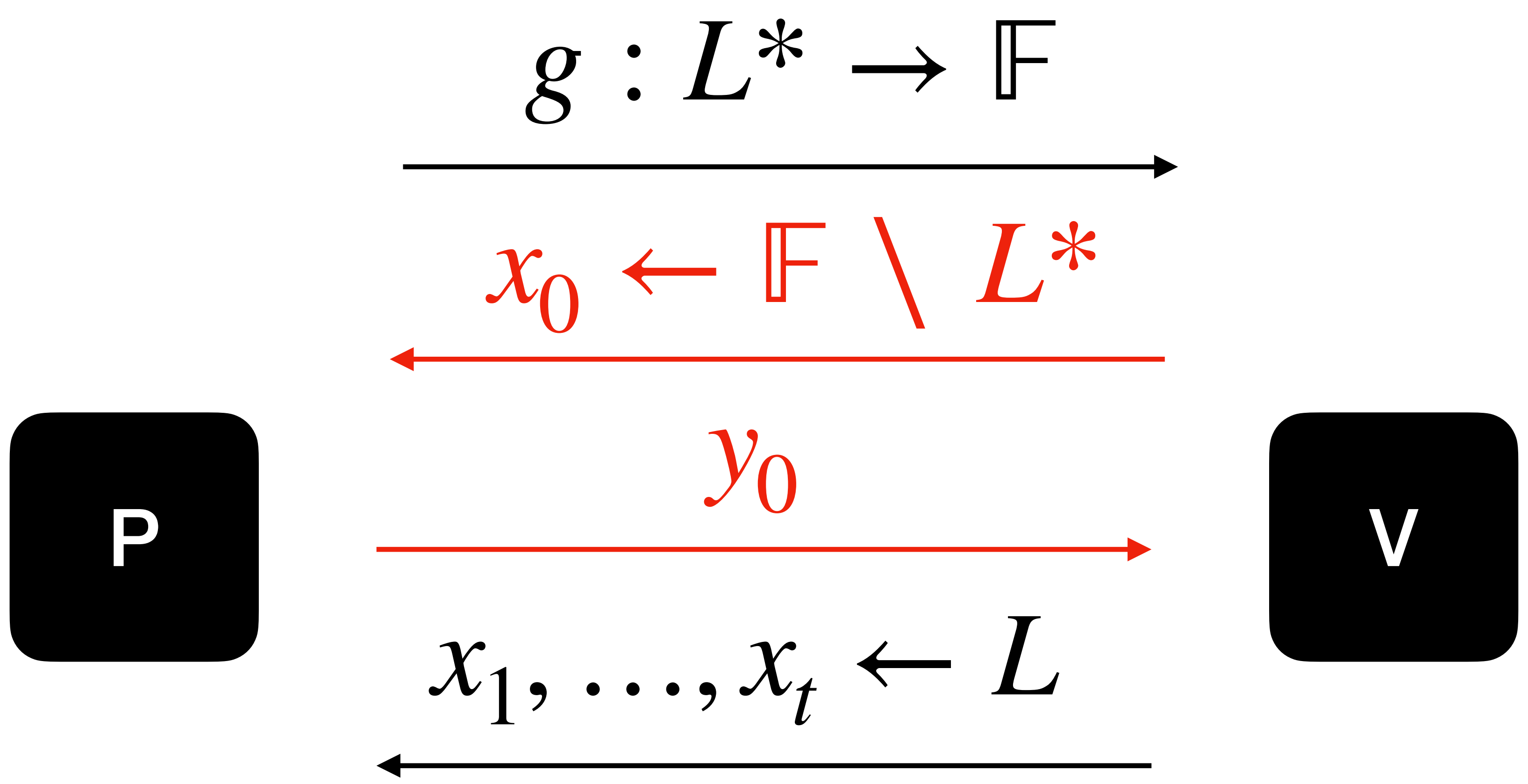
$x_0 \leftarrow \mathbb{F} \setminus L^*$

y_0

V

Domain shifting in list decoding

Let $\delta^* := 1 - \sqrt{\rho^*}$, $L \cap L^* = \varnothing$.



P

V

$g : L^* \to \mathbb{F}$

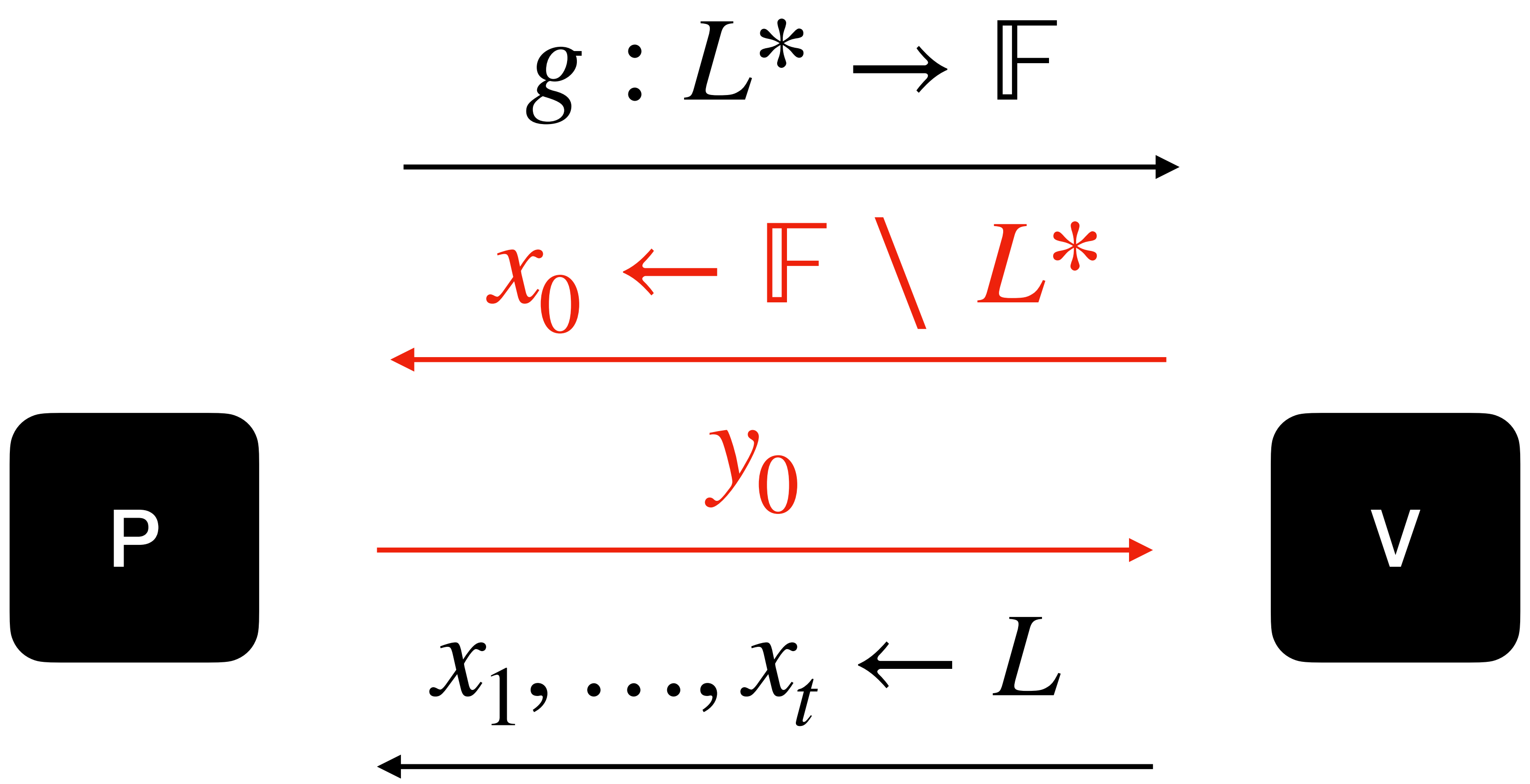
$x_0 \leftarrow \mathbb{F} \setminus L^*$

y_0

$x_1, \ldots, x_t \leftarrow L$

Domain shifting in list decoding

Let $\delta^* := 1 - \sqrt{\rho^*}$, $L \cap L^* = \varnothing$.



P

$g : L^* \to \mathbb{F}$

$x_0 \leftarrow \mathbb{F} \setminus L^*$

y_0

$x_1, \ldots, x_t \leftarrow L$

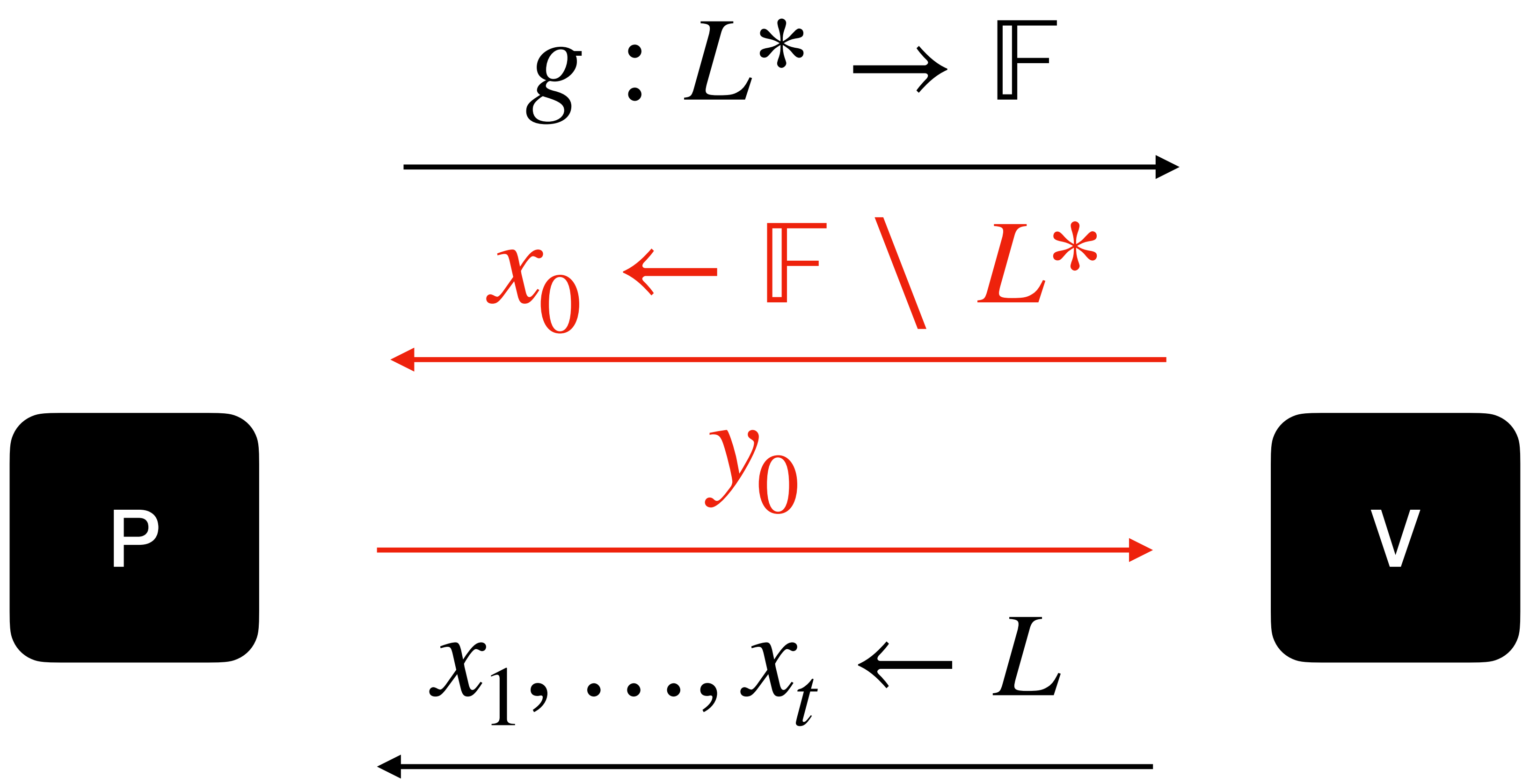
V

Query $y_i = f(x_i)$

$\forall\ i$: $\mathsf{Ans}(x_i) = y_i$

Domain shifting in list decoding

Let $\delta^* := 1 - \sqrt{\rho^*}$, $L \cap L^* = \emptyset$.



P

$g : L^* \to \mathbb{F}$

$x_0 \leftarrow \mathbb{F} \setminus L^*$

y_0

$x_1, \ldots, x_t \leftarrow L$

V

Query $y_i = f(x_i)$

$\forall\ i$: $\mathsf{Ans}(x_i) = y_i$

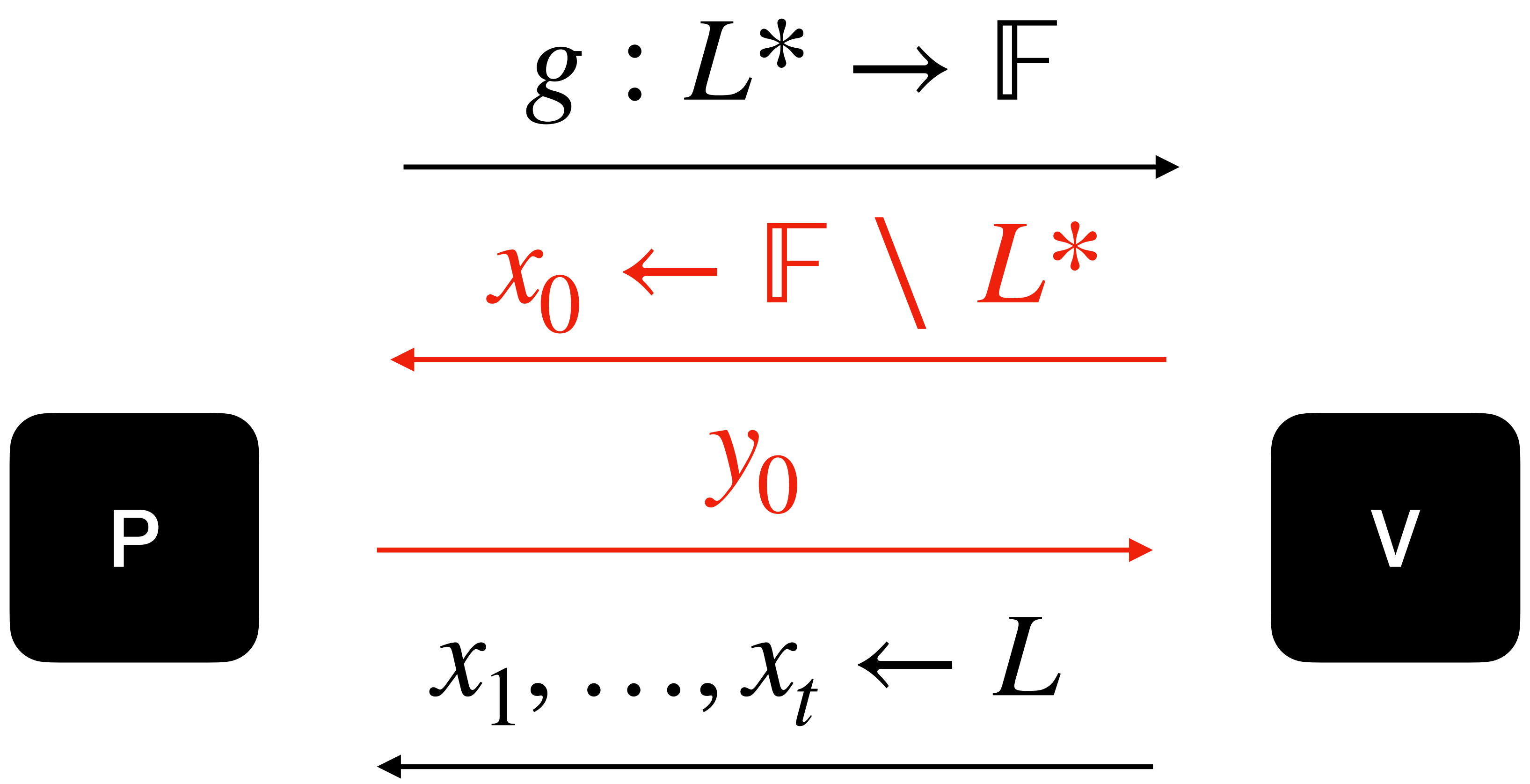
$g^* :=$ Quotient(g, Ans)

Test g^* on L^*

Domain shifting in list decoding

Let $\delta^* := 1 - \sqrt{\rho^*}$, $L \cap L^* = \varnothing$.

By OOD, there is **unique** $\hat{v}$ close to g, with $\hat{v}(x_0) = y_0$



P

V

$g : L^* \to \mathbb{F}$

$x_0 \leftarrow \mathbb{F} \setminus L^*$

y_0

$x_1, \ldots, x_t \leftarrow L$

Query $y_i = f(x_i)$

$\forall\ i$: Ans$(x_i) = y_i$

$g^* :=$ Quotient$(g,$ Ans$)$

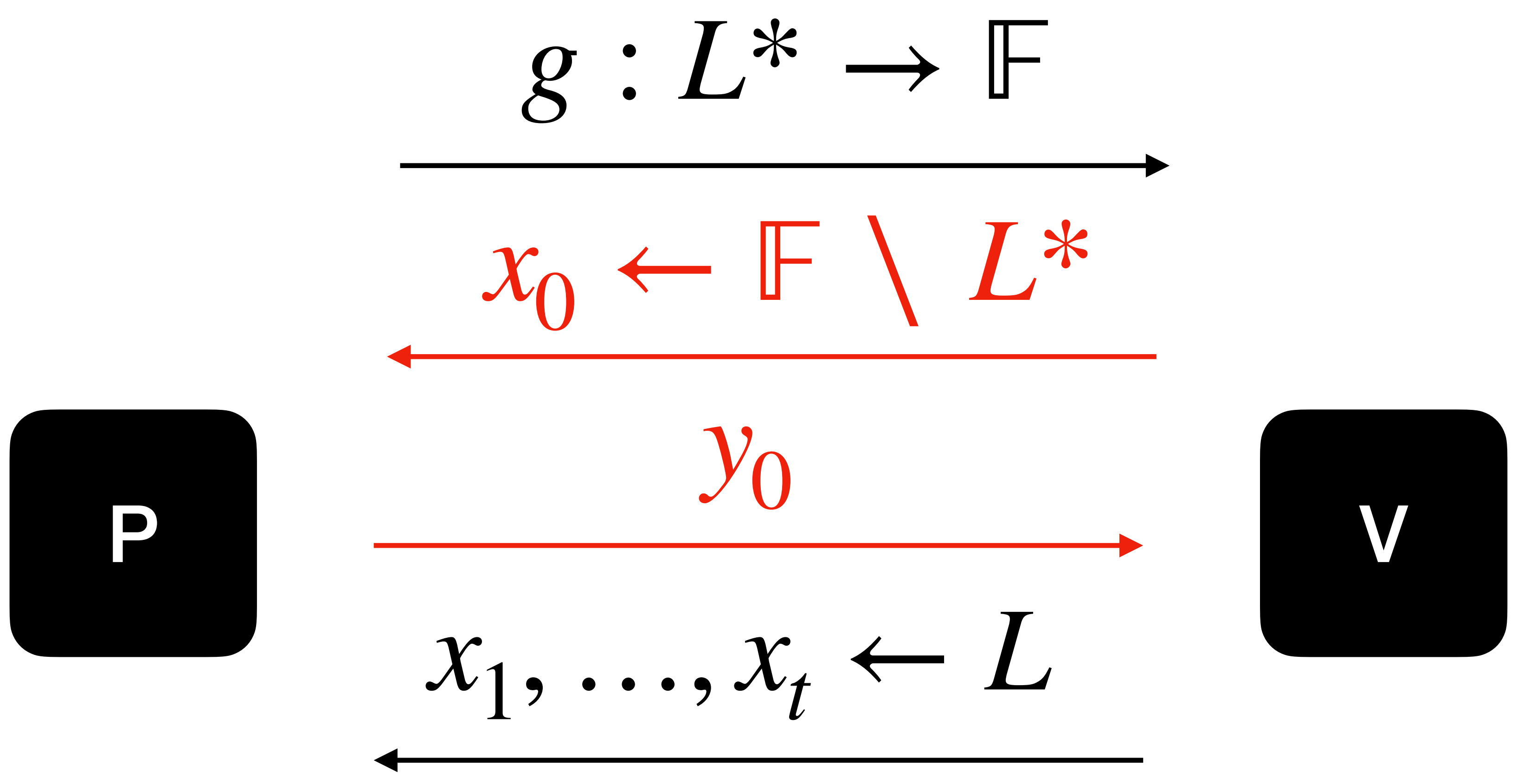
Test g^* on $L*$

Domain shifting in list decoding

Let $\delta^* := 1 - \sqrt{\rho^*}$, $L \cap L^* = \emptyset$.

By OOD, there is **unique** $\hat{v}$ close to g, with $\hat{v}(x_0) = y_0$

If at any point $\hat{v}(x_i) \neq y_i$ then, by **quotients**, g^* is δ^*-far from C^*



P

V

$g : L^* \to \mathbb{F}$

$x_0 \leftarrow \mathbb{F} \setminus L^*$

y_0

$x_1, \ldots, x_t \leftarrow L$

Query $y_i = f(x_i)$

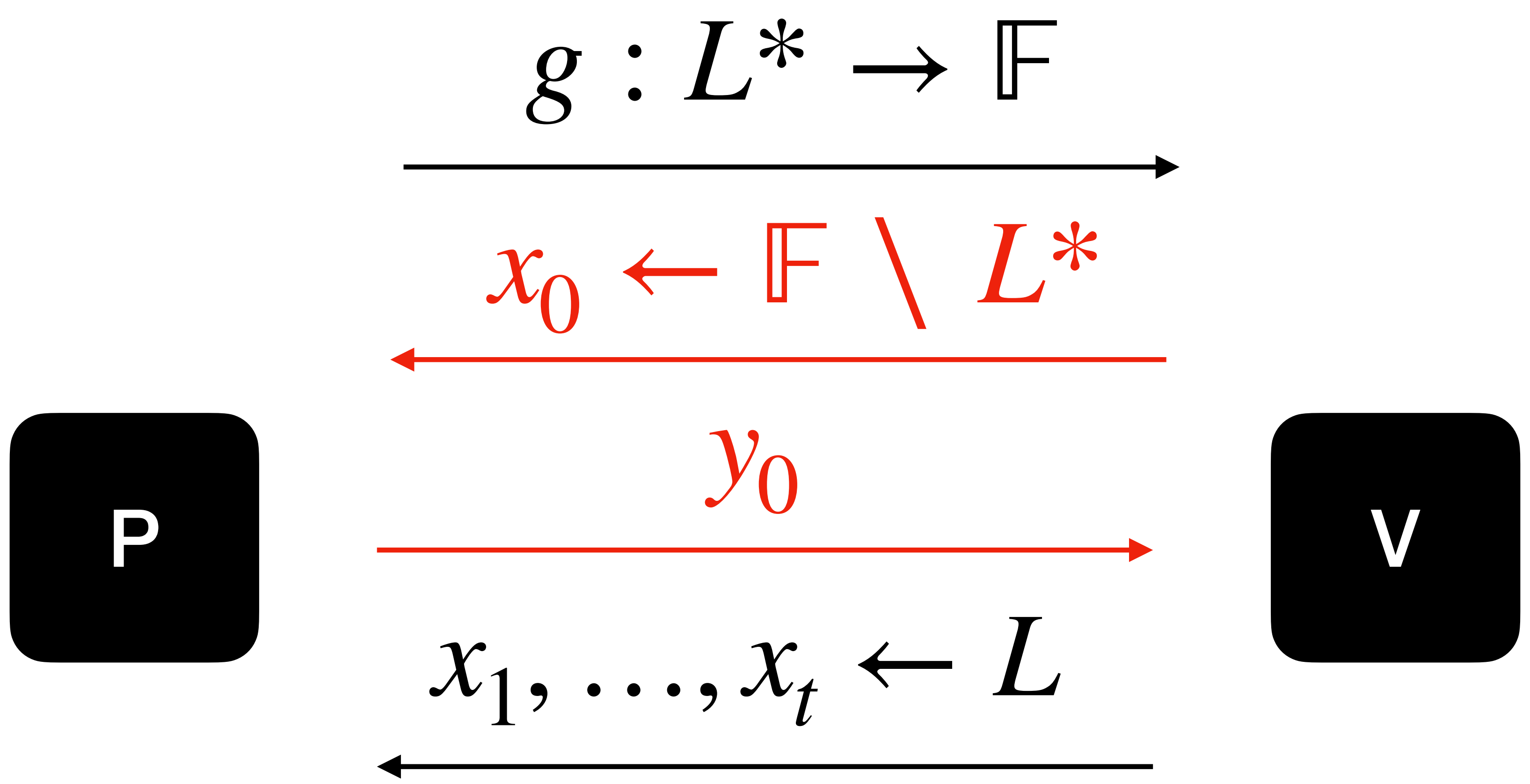
$\forall\ i : \mathrm{Ans}(x_i) = y_i$

$g^* := \mathrm{Quotient}(g, \mathrm{Ans})$

Test g^* on L^*

Domain shifting in list decoding

Let $\delta^* := 1 - \sqrt{\rho^*}$, $L \cap L^* = \varnothing$.



P

V

$g : L^* \to \mathbb{F}$

$x_0 \leftarrow \mathbb{F} \setminus L^*$

y_0

$x_1, \ldots, x_t \leftarrow L$

Query $y_i = f(x_i)$

$\forall\ i$: $\text{Ans}(x_i) = y_i$

$g^* :=$ Quotient(g, Ans)

Test g^* on L^*

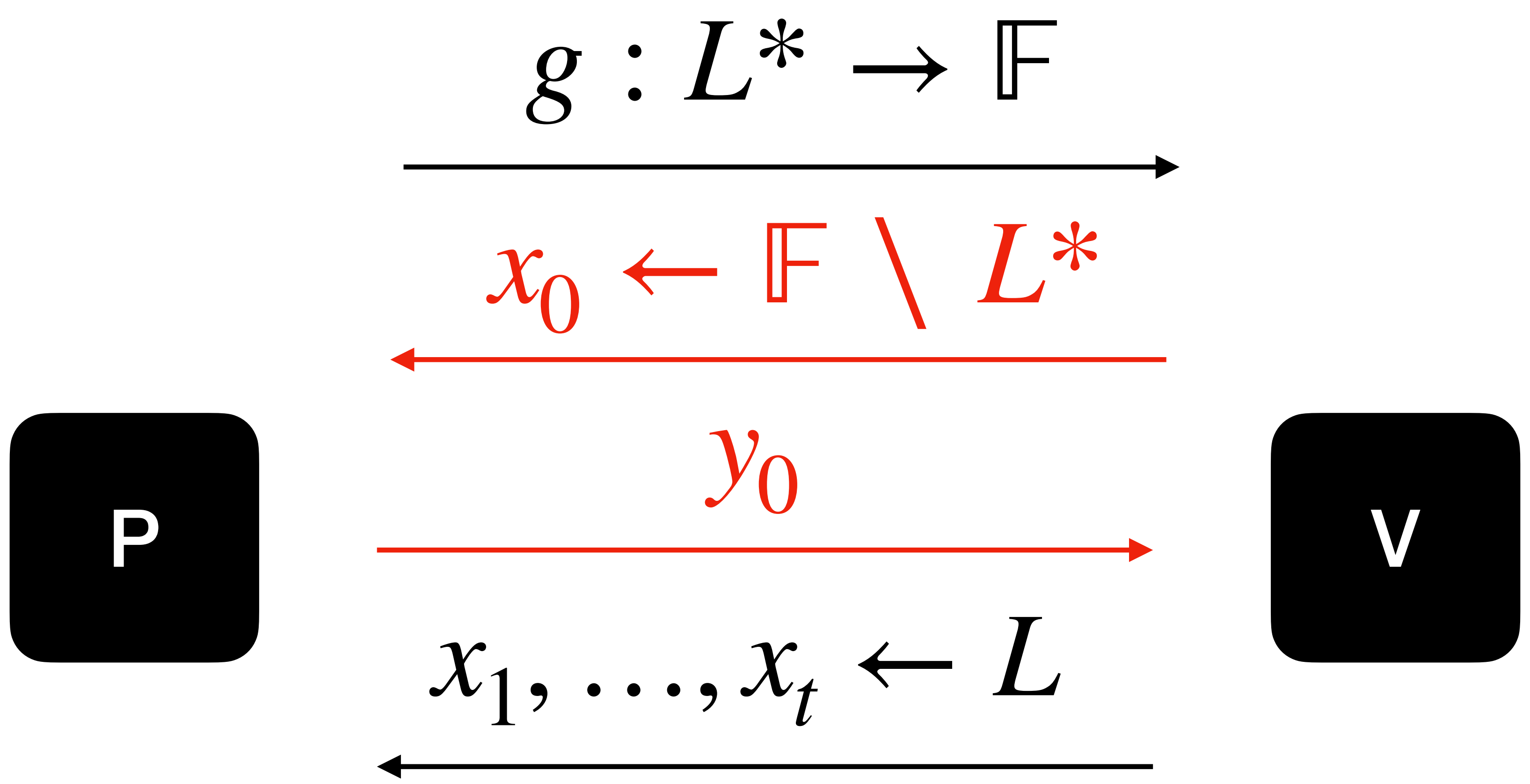
By OOD, there is **unique** $\hat{v}$ close to g, with $\hat{v}(x_0) = y_0$

If at any point $\hat{v}(x_i) \neq y_i$ then, by **quotients**, g^* is δ^*-far from C^*

Since $\Delta(f, \hat{v}|_L) > \delta$, then:

Domain shifting in list decoding

Let $\delta^* := 1 - \sqrt{\rho^*}$, $L \cap L^* = \emptyset$.



P

V

$g : L^* \to \mathbb{F}$

$x_0 \leftarrow \mathbb{F} \setminus L^*$

y_0

$x_1, \ldots, x_t \leftarrow L$

Query $y_i = f(x_i)$

$\forall\ i : \text{Ans}(x_i) = y_i$

$g^* := \text{Quotient}(g, \text{Ans})$

Test g^* on L^*

By OOD, there is **unique** $\hat{v}$ close to g, with $\hat{v}(x_0) = y_0$

If at any point $\hat{v}(x_i) \neq y_i$ then, by **quotients**, g^* is δ^*-far from C^*

Since $\Delta(f, \hat{v}|_L) > \delta$, then:

$$\Pr\left[g^* \text{ is } \delta^* \text{ close }\right]$$

$$\leq \Pr\left[\forall i,\ \hat{v}(x_i) = y_i\right]$$

$$= \Pr\left[\forall i,\ \hat{v}(x_i) = f(x_i)\right]$$

$$\leq (1 - \delta)^t$$

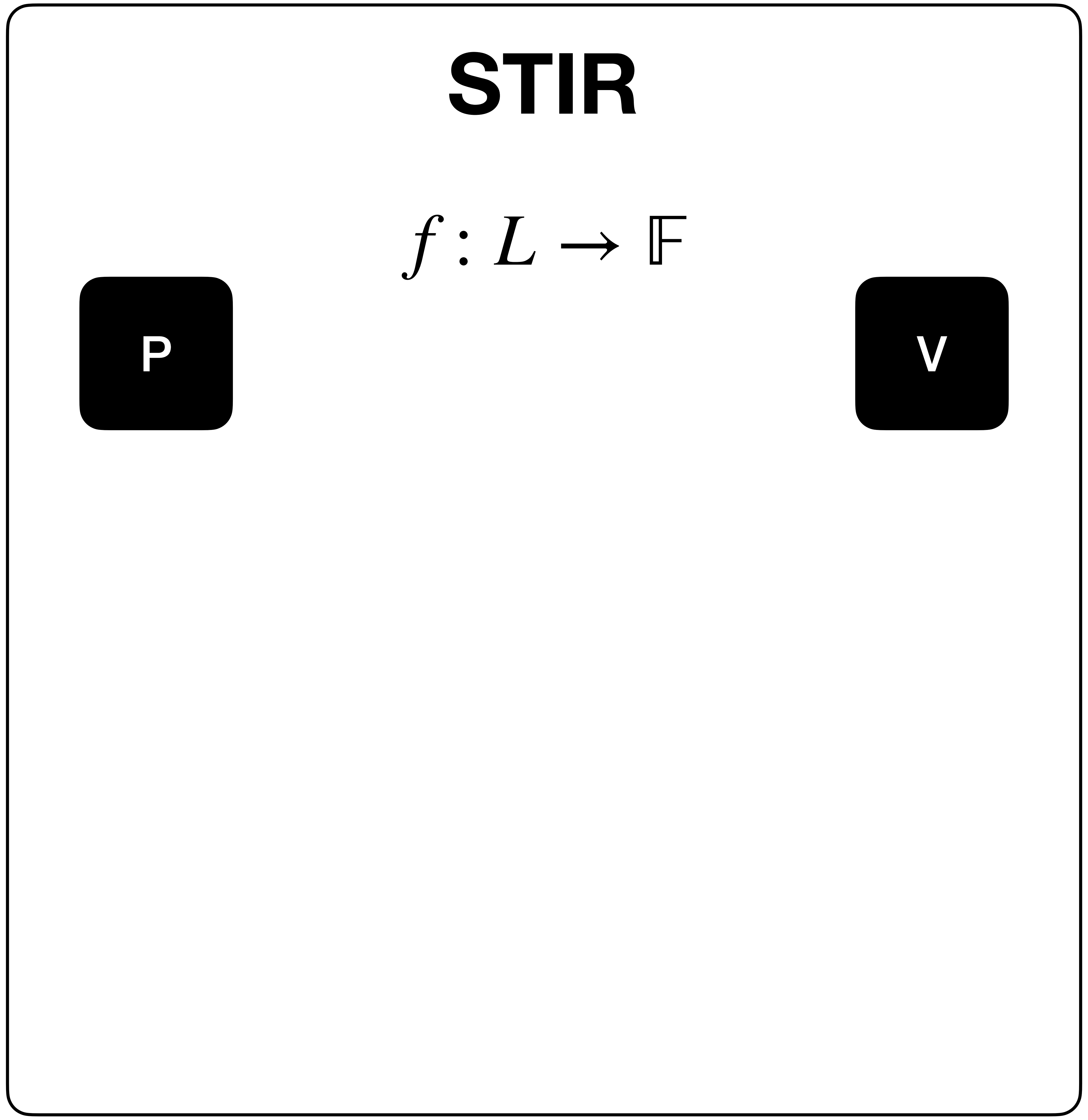
Final Protocol

STIR: domain shifting of fold

STIR

Final Protocol

STIR: domain shifting of fold

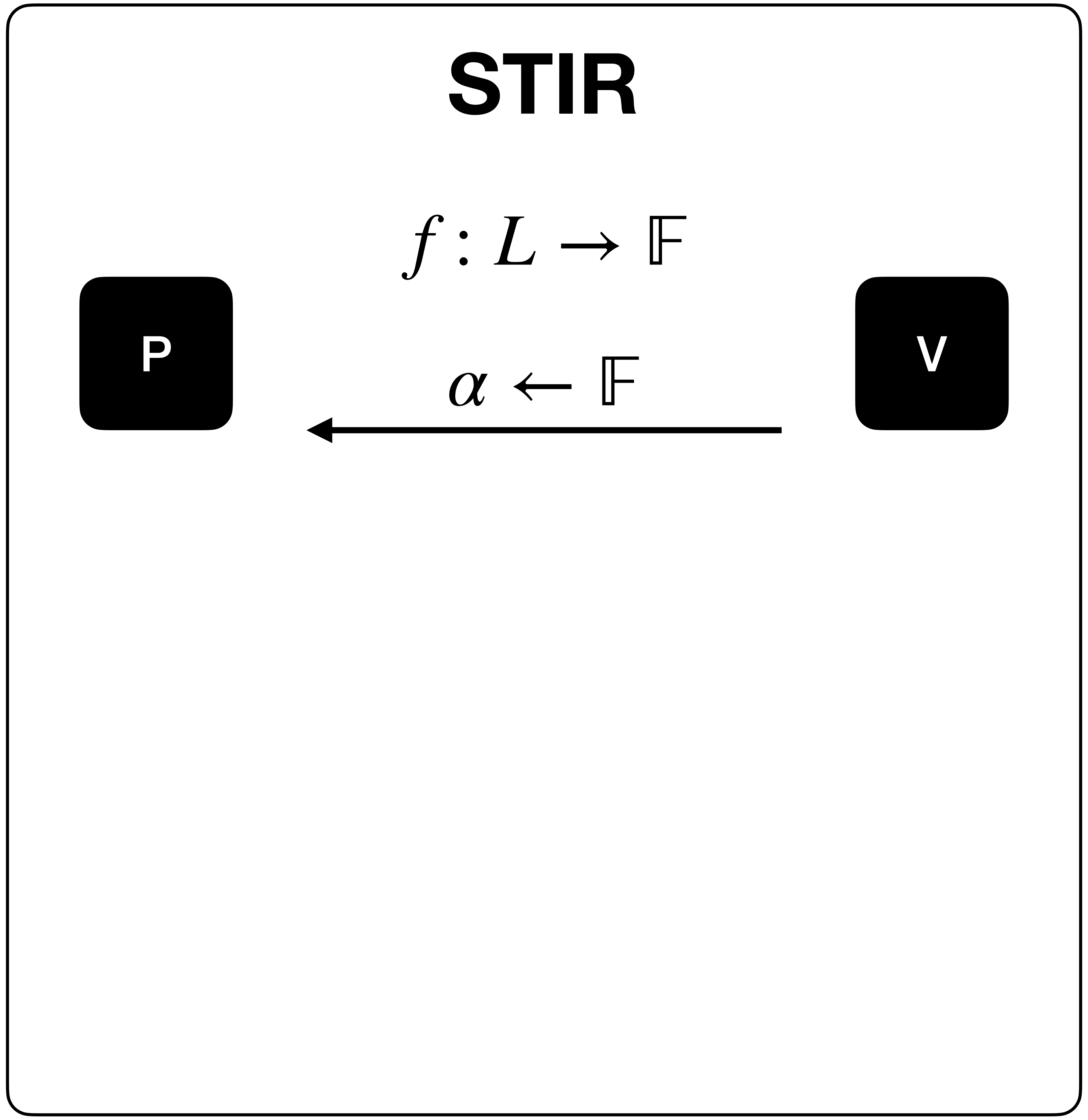


STIR

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

Final Protocol

STIR: domain shifting of fold



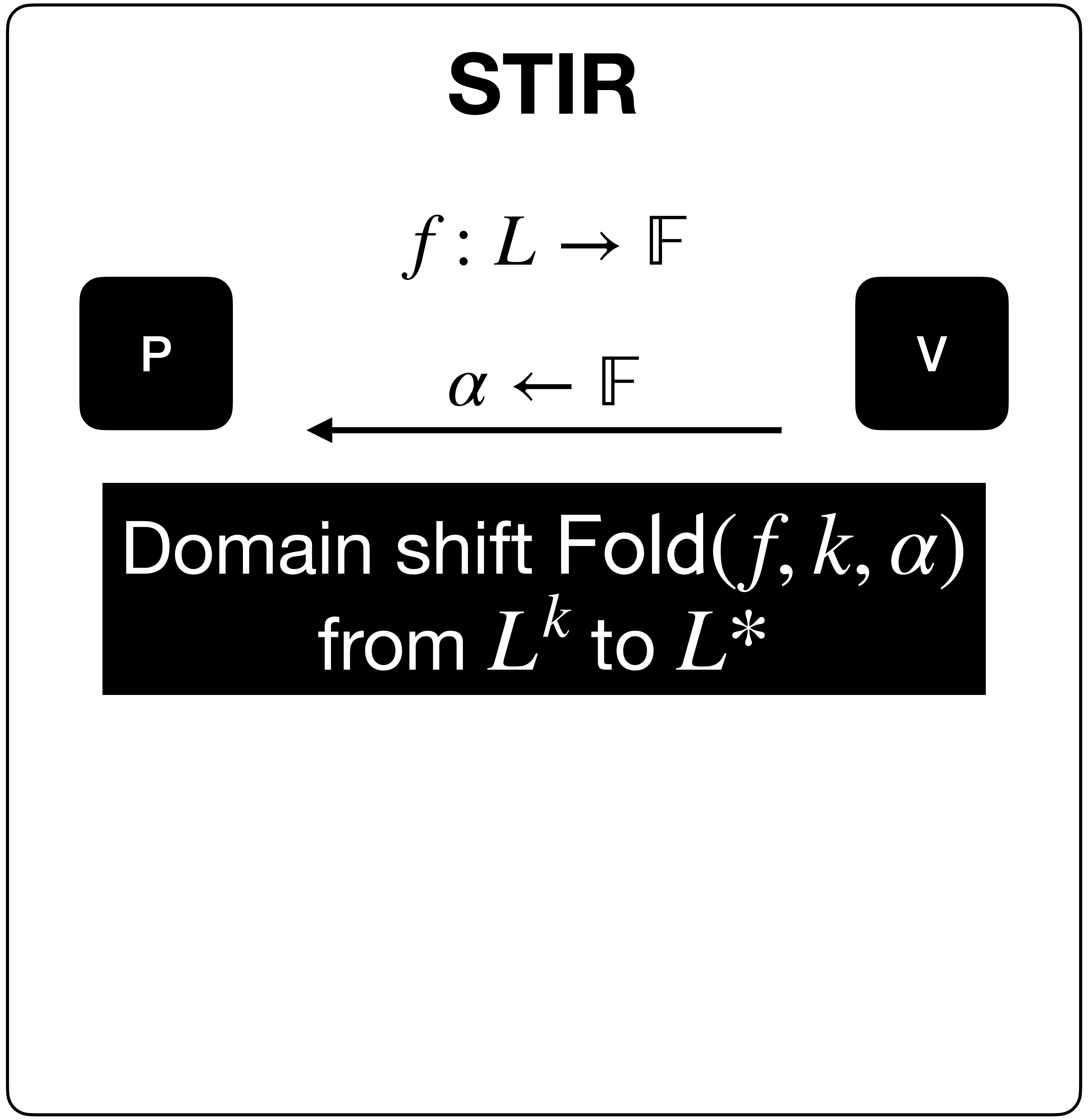
STIR

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

$\alpha \leftarrow \mathbb{F}$

Final Protocol

STIR: domain shifting of fold



STIR

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

P

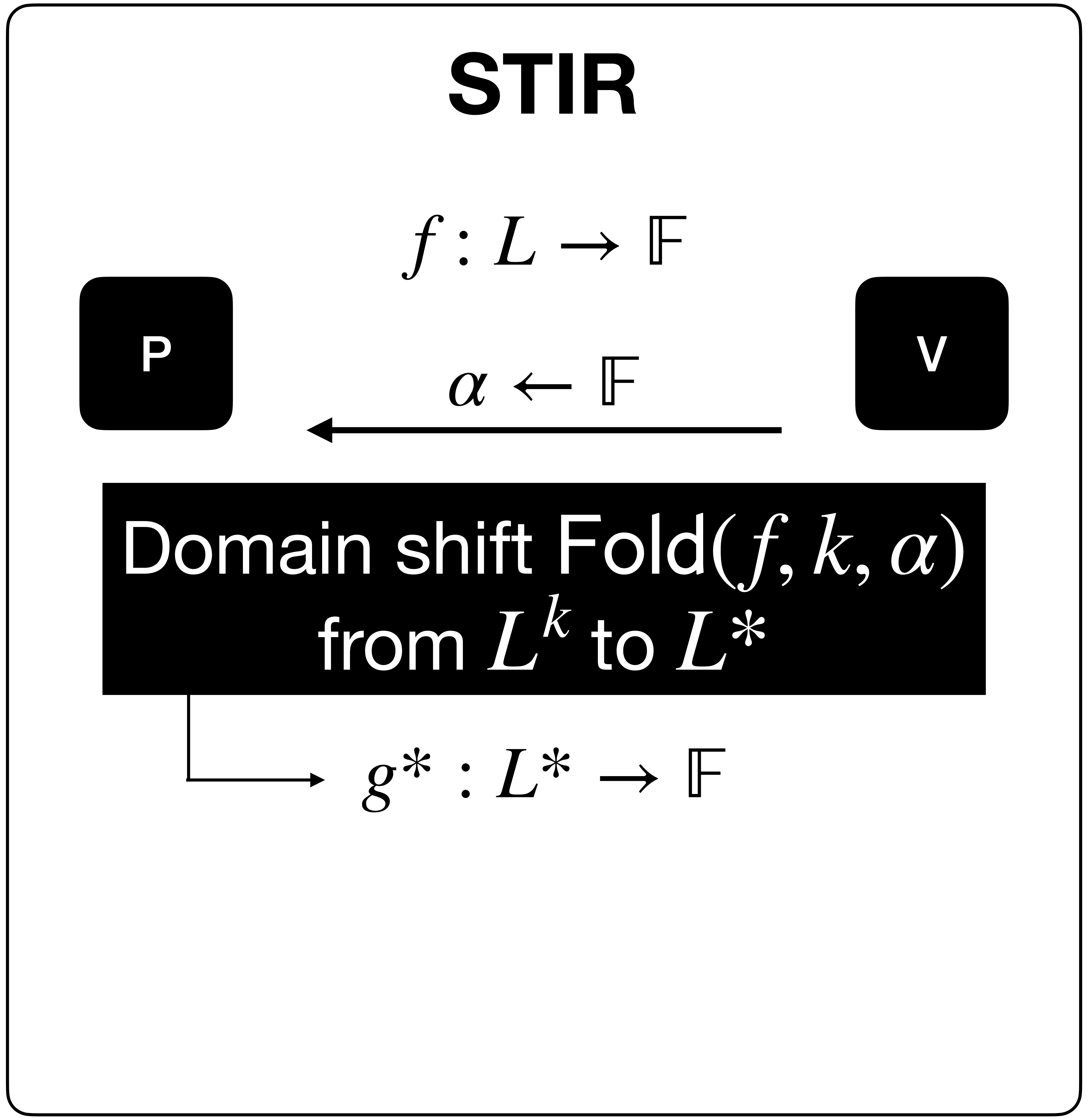
V

$\alpha \leftarrow \mathbb{F}$

Domain shift Fold(f, k, α) from L^k to L^*

Final Protocol

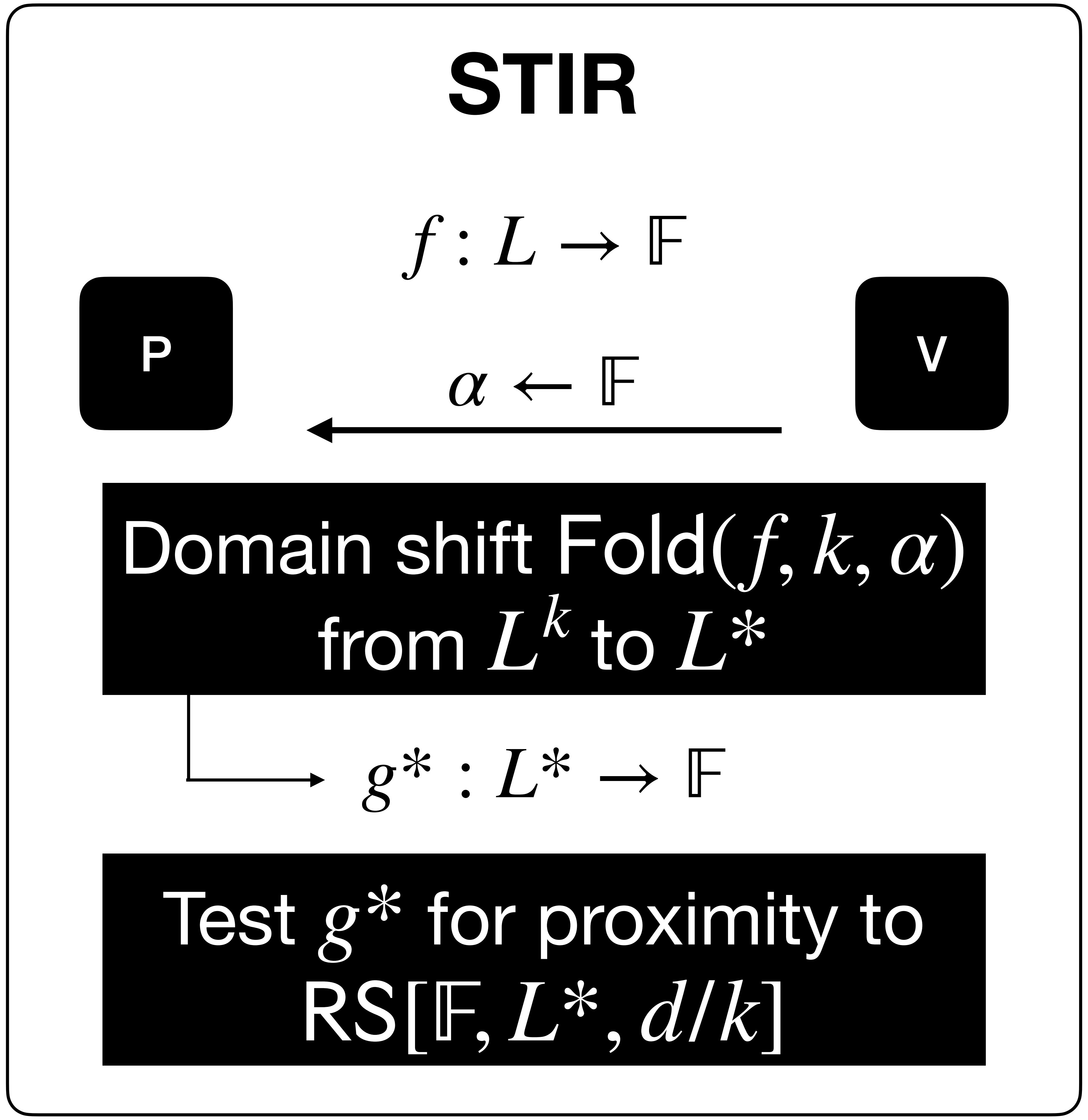
STIR: domain shifting of fold



STIR
$f : L \to \mathbb{F}$
P
V
$\alpha \leftarrow \mathbb{F}$
Domain shift Fold(f, k, α) from L^k to L^*
$g^* : L^* \to \mathbb{F}$

Final Protocol

STIR: domain shifting of fold



STIR

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

P

V

$\alpha \leftarrow \mathbb{F}$

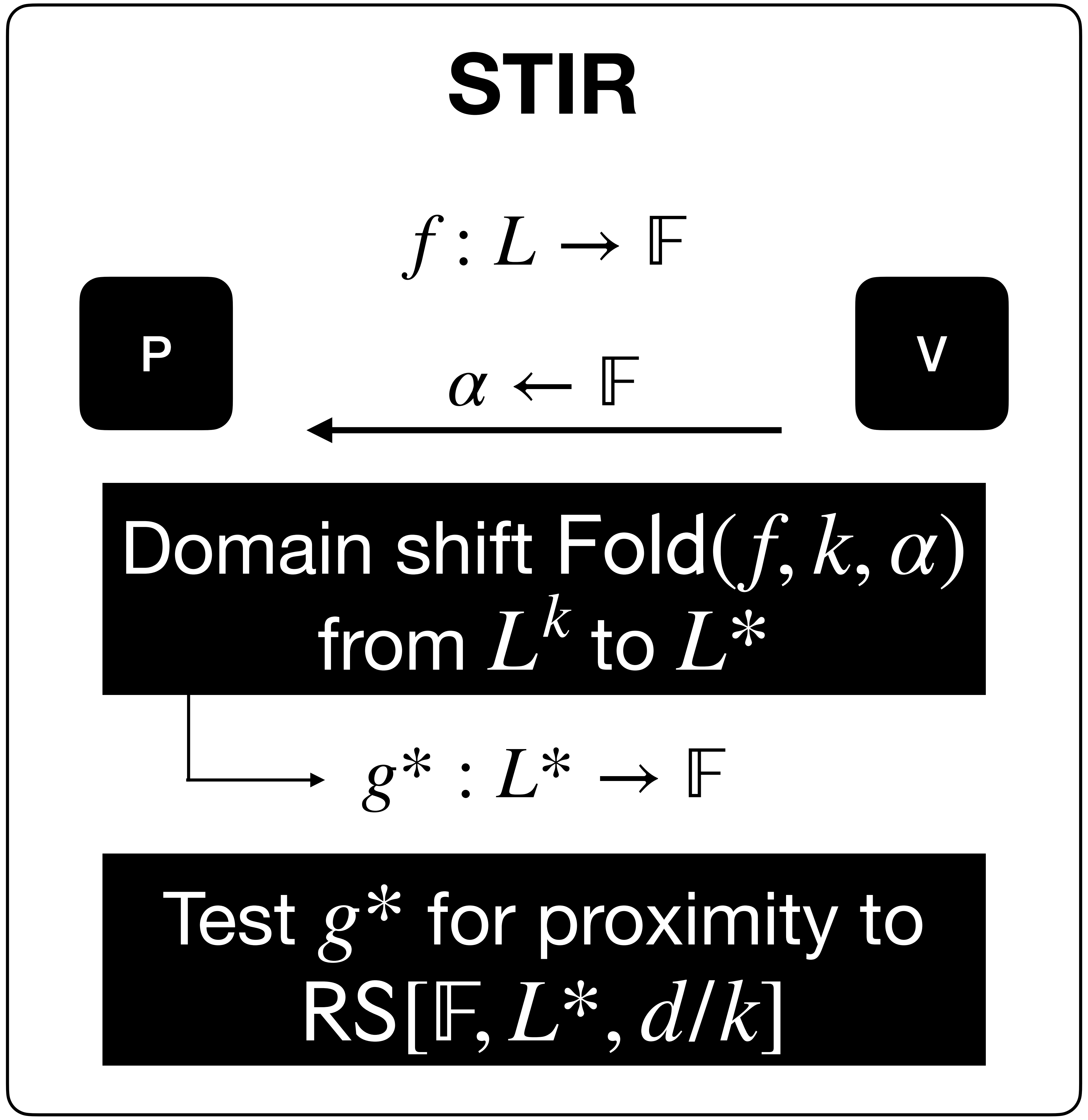
Domain shift $\text{Fold}(f, k, \alpha)$ from L^k to L^*

$g^* : L^* \to \mathbb{F}$

Test g^* for proximity to $\text{RS}[\mathbb{F}, L^*, d/k]$

Final Protocol

STIR: domain shifting of fold



STIR

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

P

V

$\alpha \leftarrow \mathbb{F}$

Domain shift $\mathsf{Fold}(f, k, \alpha)$ from L^k to $L*$

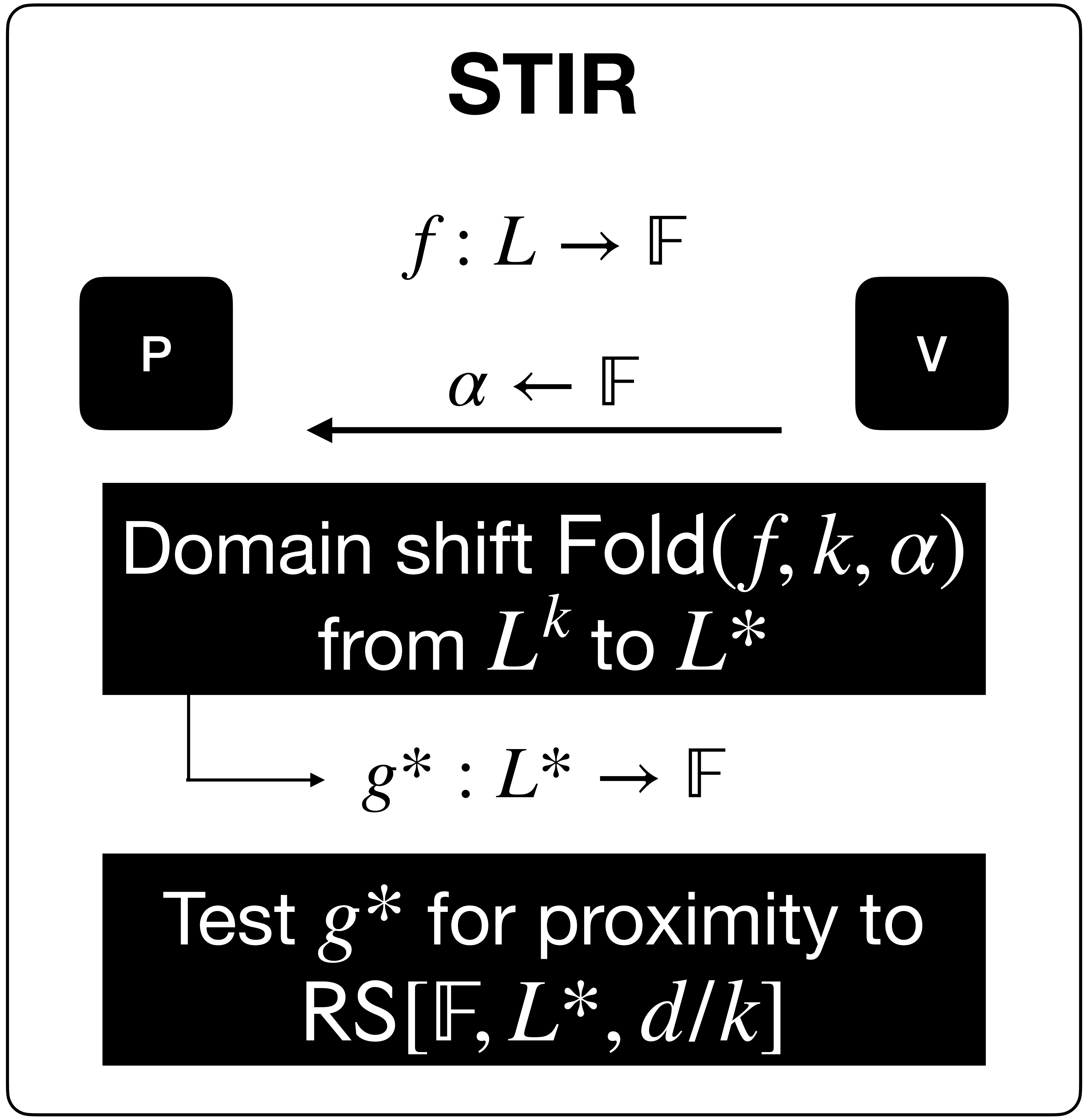
$g* : L* \to \mathbb{F}$

Test $g*$ for proximity to $\mathsf{RS}[\mathbb{F}, L*, d/k]$

Soundness

Final Protocol

STIR: domain shifting of fold



STIR

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

$\alpha \leftarrow \mathbb{F}$

Domain shift Fold(f, k, α) from L^k to L^*

$g^* : L^* \to \mathbb{F}$

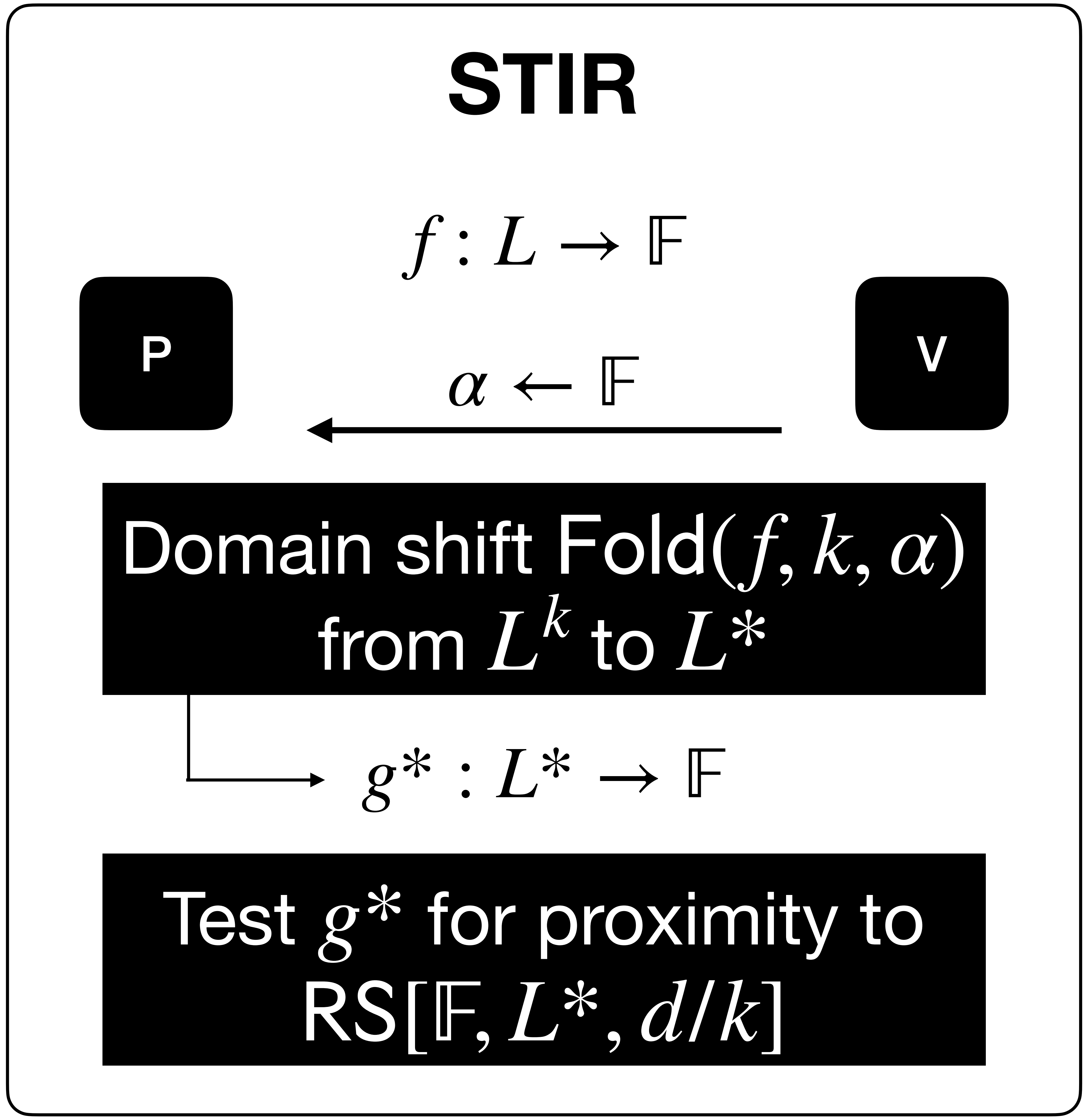
Test g^* for proximity to RS$[\mathbb{F}, L^*, d/k]$

Soundness

By distance-preservation of **folding**, $\Delta($Fold$(f, k, \alpha),$ RS$[\mathbb{F}, L^k, d/k]) > \delta$ w.h.p

Final Protocol

STIR: domain shifting of fold



STIR

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

P

V

$\alpha \leftarrow \mathbb{F}$

Domain shift Fold(f, k, α) from L^k to L^*

$g^* : L^* \to \mathbb{F}$

Test g^* for proximity to RS$[\mathbb{F}, L^*, d/k]$

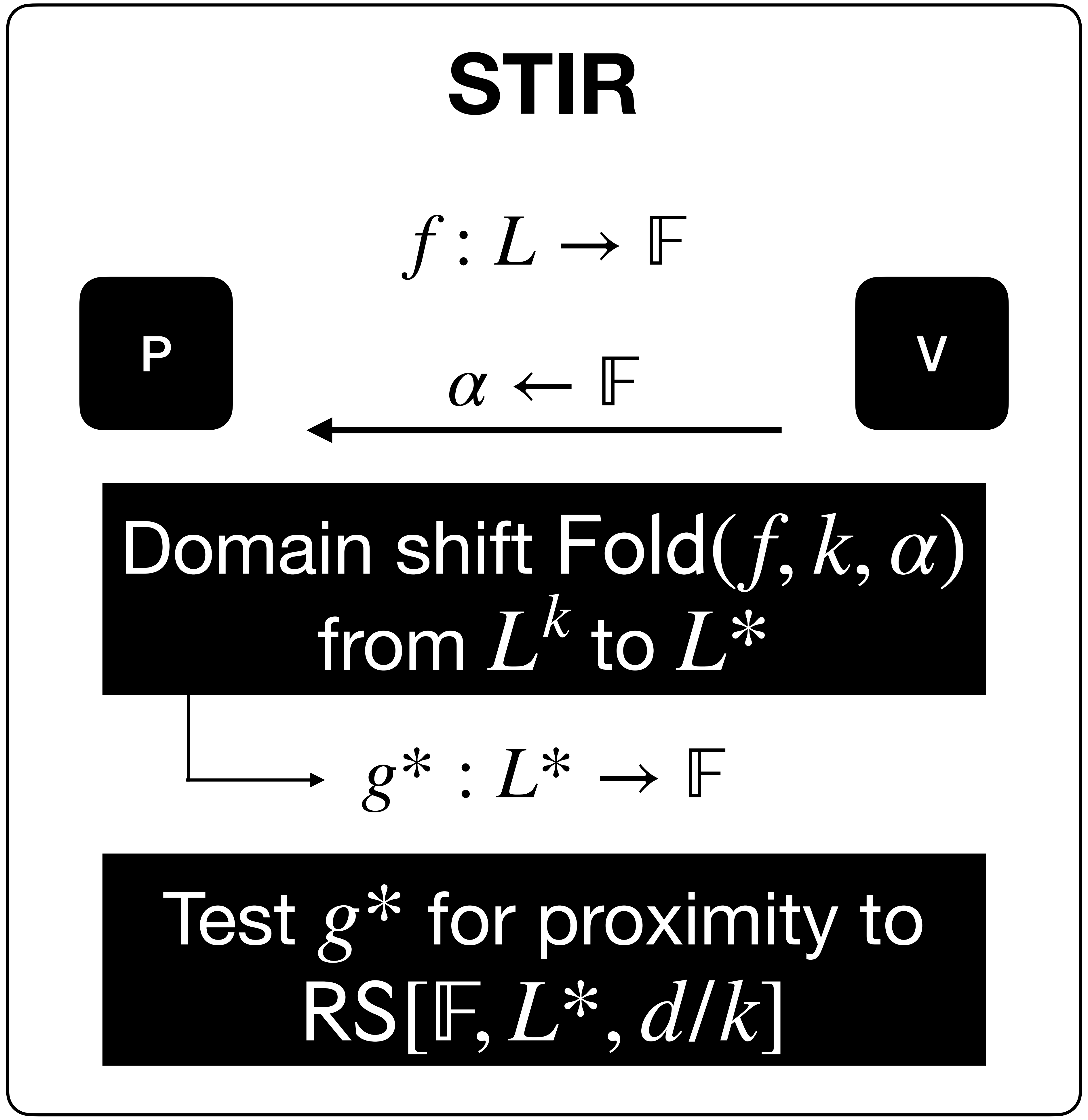
Soundness

By distance-preservation of **folding**,
$\Delta(\text{Fold}(f, k, \alpha), \text{RS}[\mathbb{F}, L^k, d/k]) > \delta$
w.h.p

By **domain-shifting**,
$\Delta(g^*, \text{RS}[\mathbb{F}, L^*, d/k]) > 1 - \sqrt{\rho^*}$
unless w.p. $(1 - \delta)^t$

Final Protocol

STIR: domain shifting of fold



STIR

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

P

V

$\alpha \leftarrow \mathbb{F}$

Domain shift $\mathsf{Fold}(f, k, \alpha)$ from L^k to L^*

$g^* : L^* \to \mathbb{F}$

Test g^* for proximity to $\mathsf{RS}[\mathbb{F}, L^*, d/k]$

Soundness

By distance-preservation of **folding**, $\Delta(\mathsf{Fold}(f, k, \alpha), \mathsf{RS}[\mathbb{F}, L^k, d/k]) > \delta$ w.h.p

By **domain-shifting**, $\Delta(g^*, \mathsf{RS}[\mathbb{F}, L^*, d/k]) > 1 - \sqrt{\rho^*}$ unless w.p. $(1 - \delta)^t$

Recursing, yields **STIR** 🥣

Conclusion

What we saw

What we did have time to talk about

What we saw

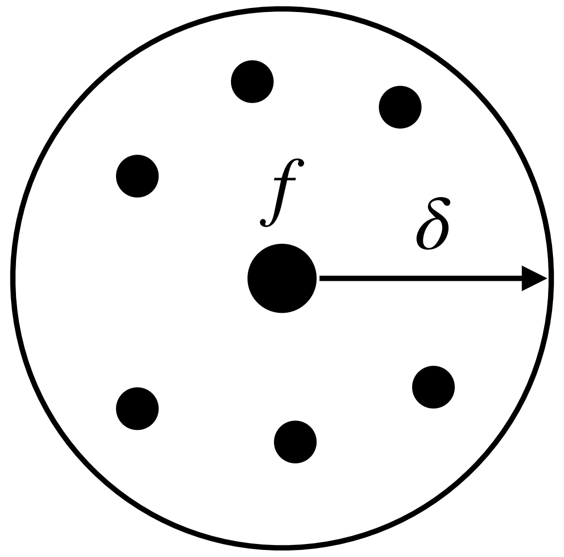
What we did have time to talk about

- Domain shifting:

What we saw

What we did have time to talk about

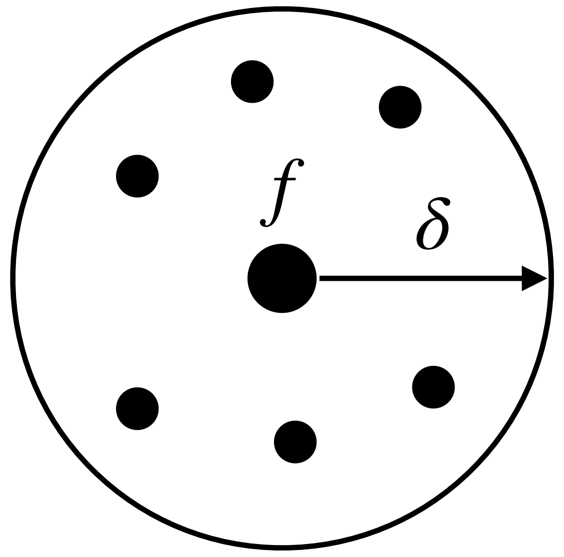
- Domain shifting:
  - Quotienting and its properties

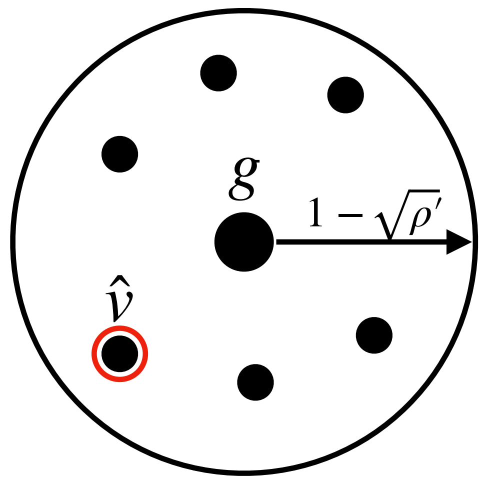


What we saw

What we did have time to talk about

- Domain shifting:
    - Quotienting and its properties
    - Out-Of-Domain sampling

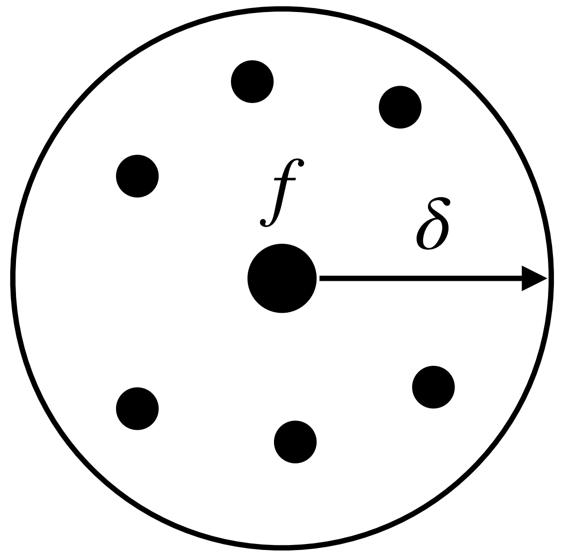


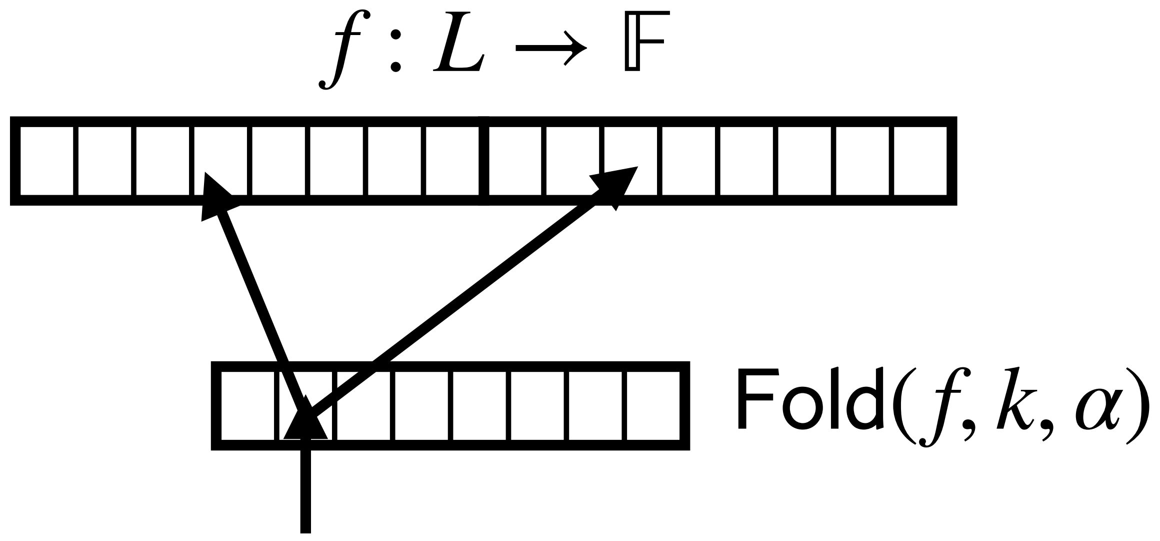


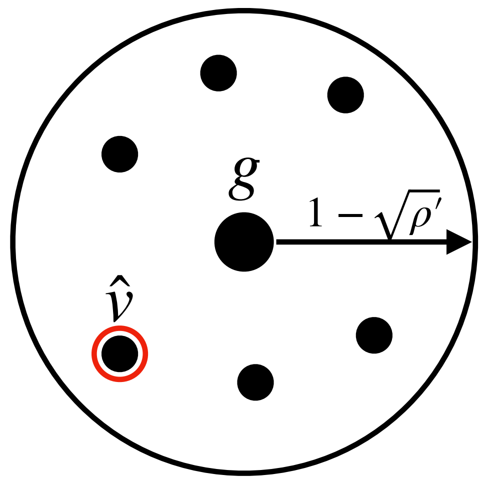
What we saw

What we did have time to talk about

- Domain shifting:
    - Quotienting and its properties
    - Out-Of-Domain sampling
- Folding and its properties





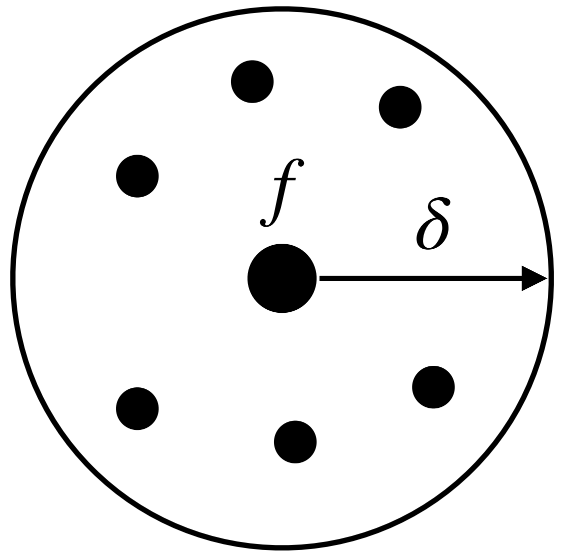


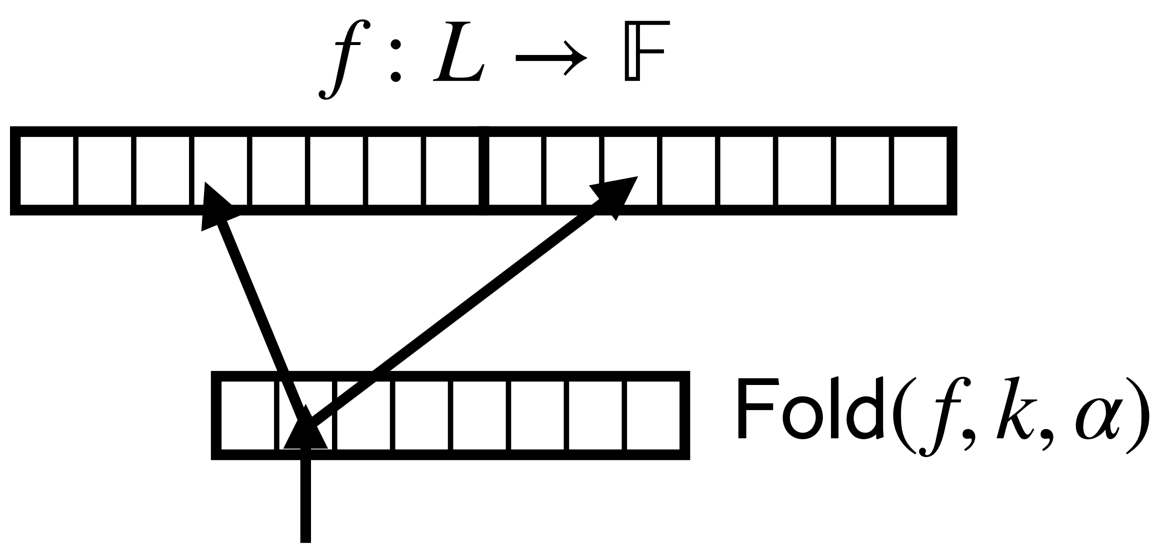
What we saw

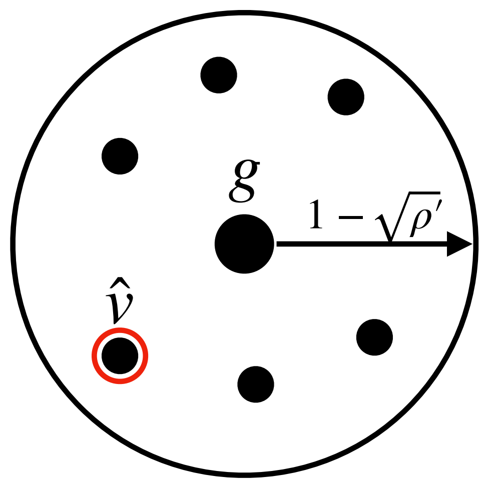
What we did have time to talk about

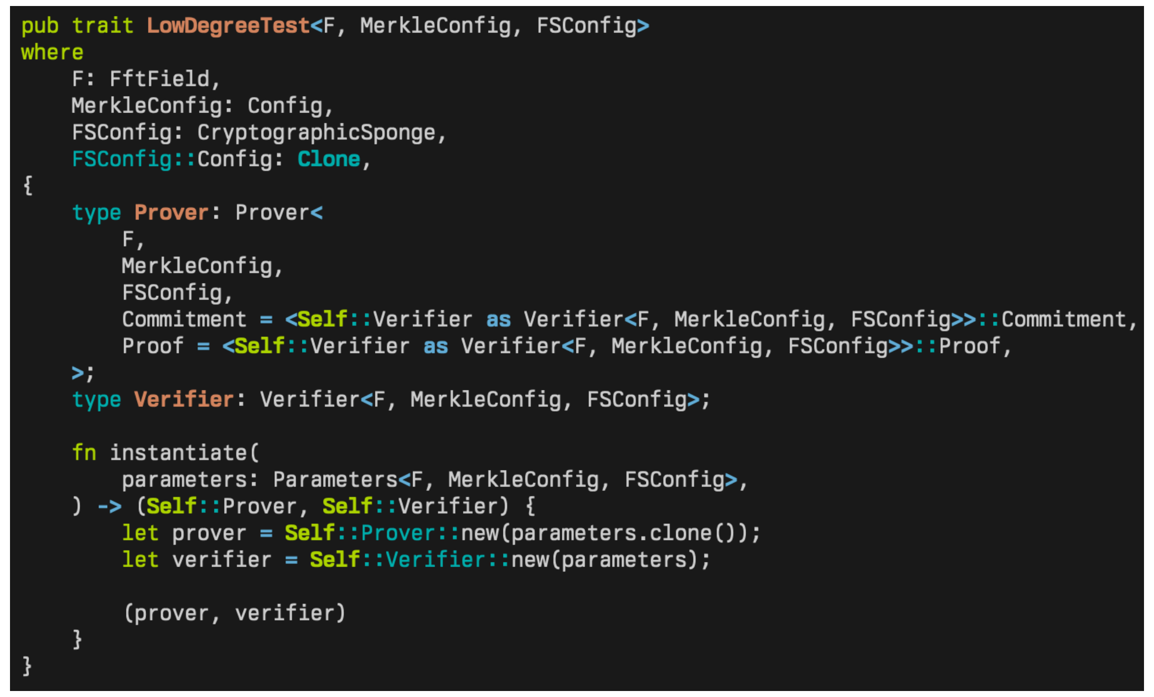
- Domain shifting:
  - Quotienting and its properties
  - Out-Of-Domain sampling
- Folding and its properties
- STIR 🥣

STIR







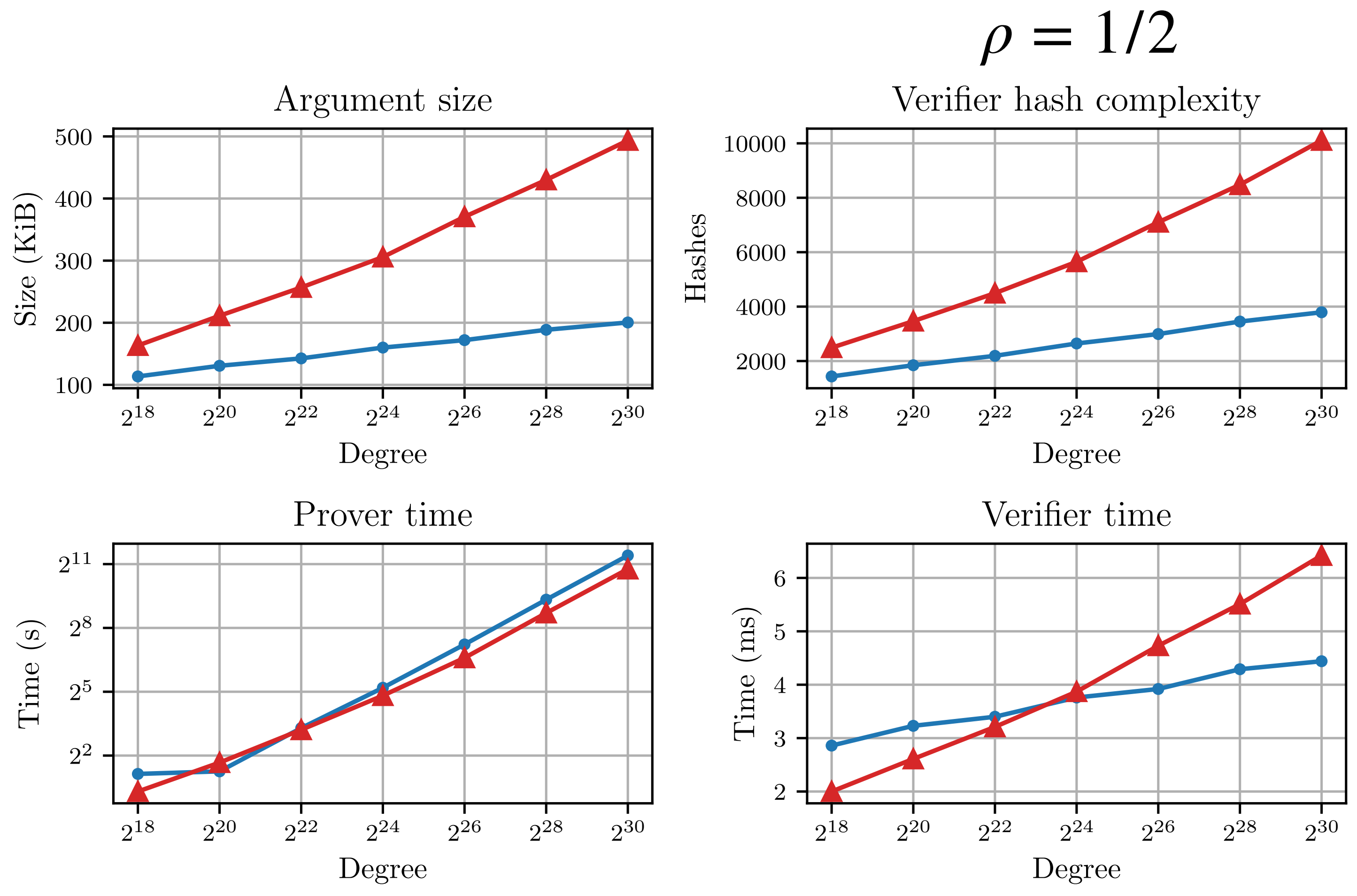


```rust
pub trait LowDegreeTest<F, MerkleConfig, FSConfig>
where
    F: FftField,
    MerkleConfig: Config,
    FSConfig: CryptographicSponge,
    FSConfig::Config: Clone,
{
    type Prover: Prover<
        F,
        MerkleConfig,
        FSConfig,
        Commitment = <Self::Verifier as Verifier<F, MerkleConfig, FSConfig>>::Commitment,
        Proof = <Self::Verifier as Verifier<F, MerkleConfig, FSConfig>>::Proof,
    >;
    type Verifier: Verifier<F, MerkleConfig, FSConfig>;

    fn instantiate(
        parameters: Parameters<F, MerkleConfig, FSConfig>,
    ) -> (Self::Prover, Self::Verifier) {
        let prover = Self::Prover::new(parameters.clone());
        let verifier = Self::Verifier::new(parameters);

        (prover, verifier)
    }
}
```

$\rho = 1/2$



There is more!

What we did not have time to talk about

There is more!

What we did not have time to talk about

Degree corrections

There is more!

What we did not have time to talk about

Degree corrections

- $\text{Quotient}(f, \text{Ans})$ has degree $d - |S|$, how to bump up to d?

There is more!

What we did not have time to talk about

Degree corrections

- Quotient(f, Ans) has degree $d - |S|$, how to bump up to d?

High-soundness compiler for Poly-IOPs

There is more!

What we did not have time to talk about

Degree corrections

- $\text{Quotient}(f, \text{Ans})$ has degree $d - |S|$, how to bump up to d?

High-soundness compiler for Poly-IOPs

- Builds on compiler in [ACY23] to achieve concrete efficiency

There is more!

What we did not have time to talk about

Degree corrections

- $\text{Quotient}(f, \text{Ans})$ has degree $d - |S|$, how to bump up to d?

High-soundness compiler for Poly-IOPs

- Builds on compiler in [ACY23] to achieve concrete efficiency

Round-by-round soundness of STIR $\implies$ secure in non-interactive setting

What’s next?

What we hope to have time to talk about next talk!

What’s next?

What we hope to have time to talk about next talk!

- Small fields:

What’s next?

What we hope to have time to talk about next talk!

- **Small fields**:
  - STIR variant of CIRCLE-STARKs?

What’s next?

What we hope to have time to talk about next talk!

- **Small fields**:
    - STIR variant of CIRCLE-STARKs?
    - Basefold STIR? Binary-STIR?

What's next?

What we hope to have time to talk about next talk!

- **Small fields**:
    - STIR variant of CIRCLE-STARKs?
    - Basefold STIR? Binary-STIR?
- **Breaking the $O(\log d)$-query barrier**

What’s next?

What we hope to have time to talk about next talk!

- **Small fields**:
    - STIR variant of CIRCLE-STARKs?
    - Basefold STIR? Binary-STIR?
- **Breaking the $O(\log d)$-query barrier**
    - Work in [ACY23] achieves $O(\log \log d)$ queries proximity tests

What’s next?

What we hope to have time to talk about next talk!

- **Small fields**:
  - STIR variant of CIRCLE-STARKs?
  - Basefold STIR? Binary-STIR?
- **Breaking the $O(\log d)$-query barrier**
  - Work in [ACY23] achieves $O(\log \log d)$ queries proximity tests
  - Not concretely efficient, lacks efficient soundness amplification

What's next?

What we hope to have time to talk about next talk!

- **Small fields**:
  - STIR variant of CIRCLE-STARKs?
  - Basefold STIR? Binary-STIR?
- **Breaking the $O(\log d)$-query barrier**
  - Work in [ACY23] achieves $O(\log \log d)$ queries proximity tests
  - Not concretely efficient, lacks efficient soundness amplification
  - Exciting!!!

Wh xt?

Wh time to talk about next talk!

-
    - S
    - Basefold S
- **Breaking the $O(\log d)$-qu**
    - Work in [ACY23] achieves $O(\log \log$
    - Not concretely efficient, lacks efficient soundn
    - Exciting!!!

USE STIR🥣!!!

Thank you!

See paper:
ia.cr/2024/390

And blog post:
gfenzi.io/papers/stir

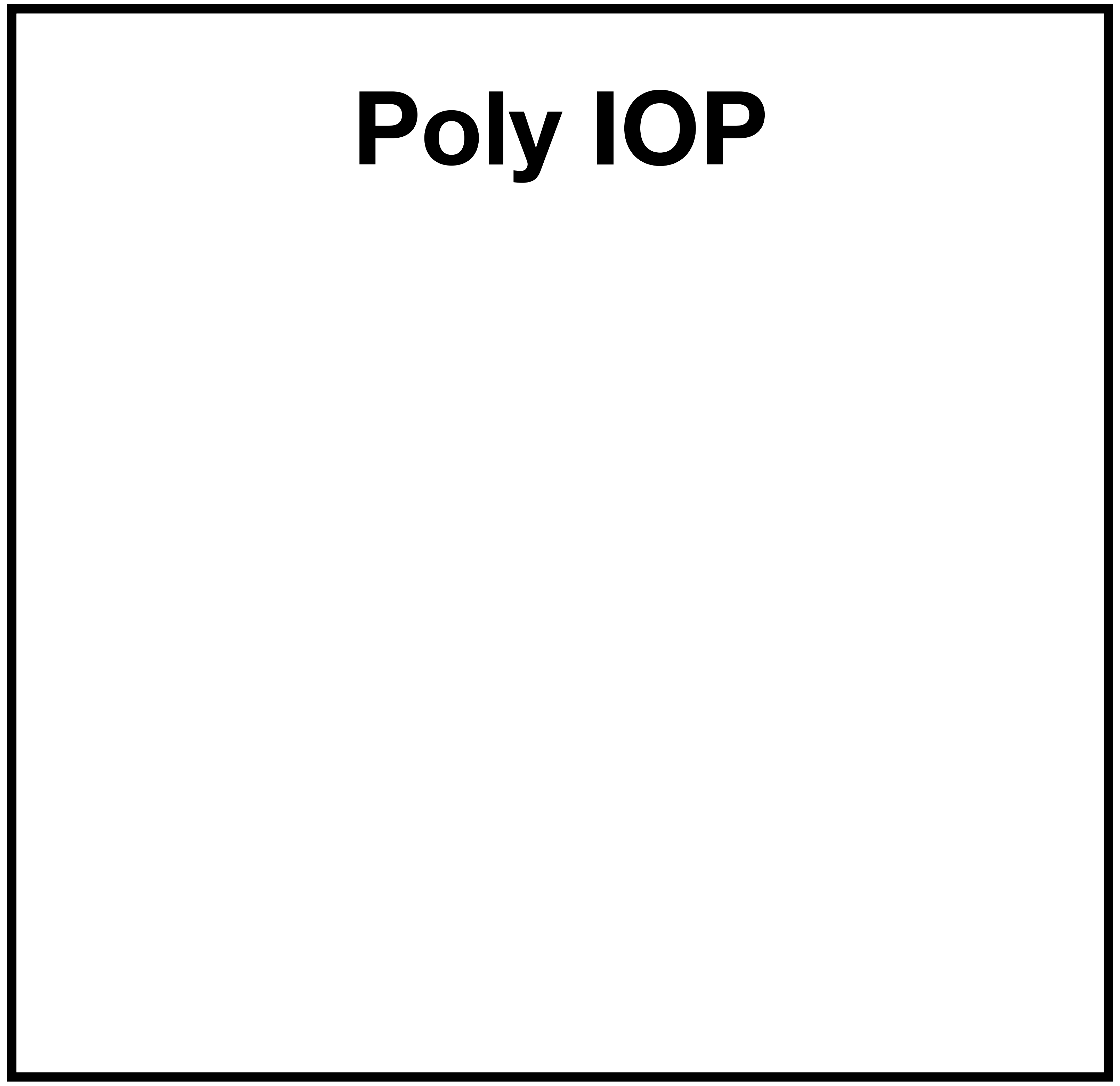
Extra slides

Anatomy of an IOP-based SNARK

Reducing to low-degree testing

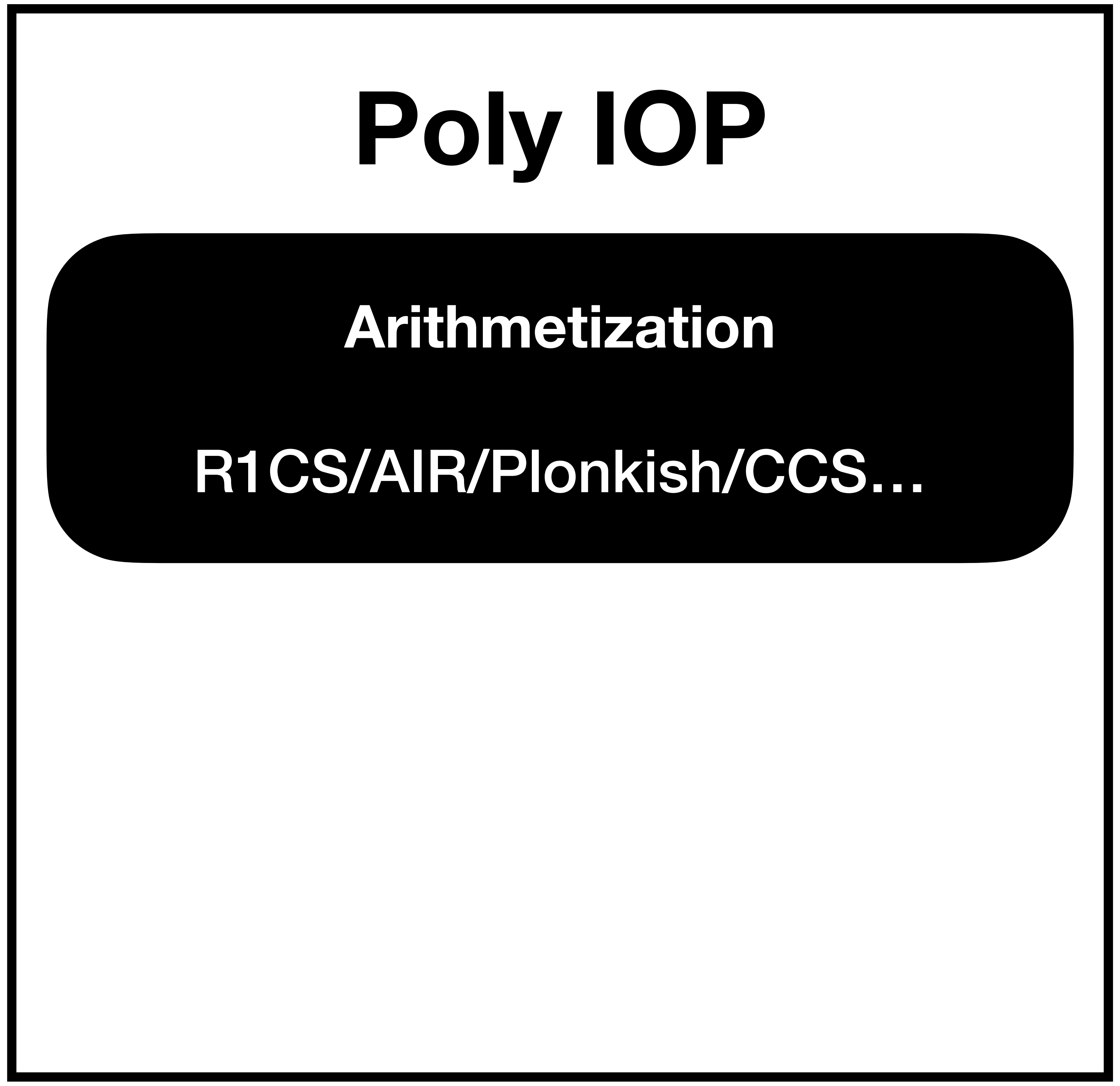
Anatomy of an IOP-based SNARK

Reducing to low-degree testing



Anatomy of an IOP-based SNARK

Reducing to low-degree testing



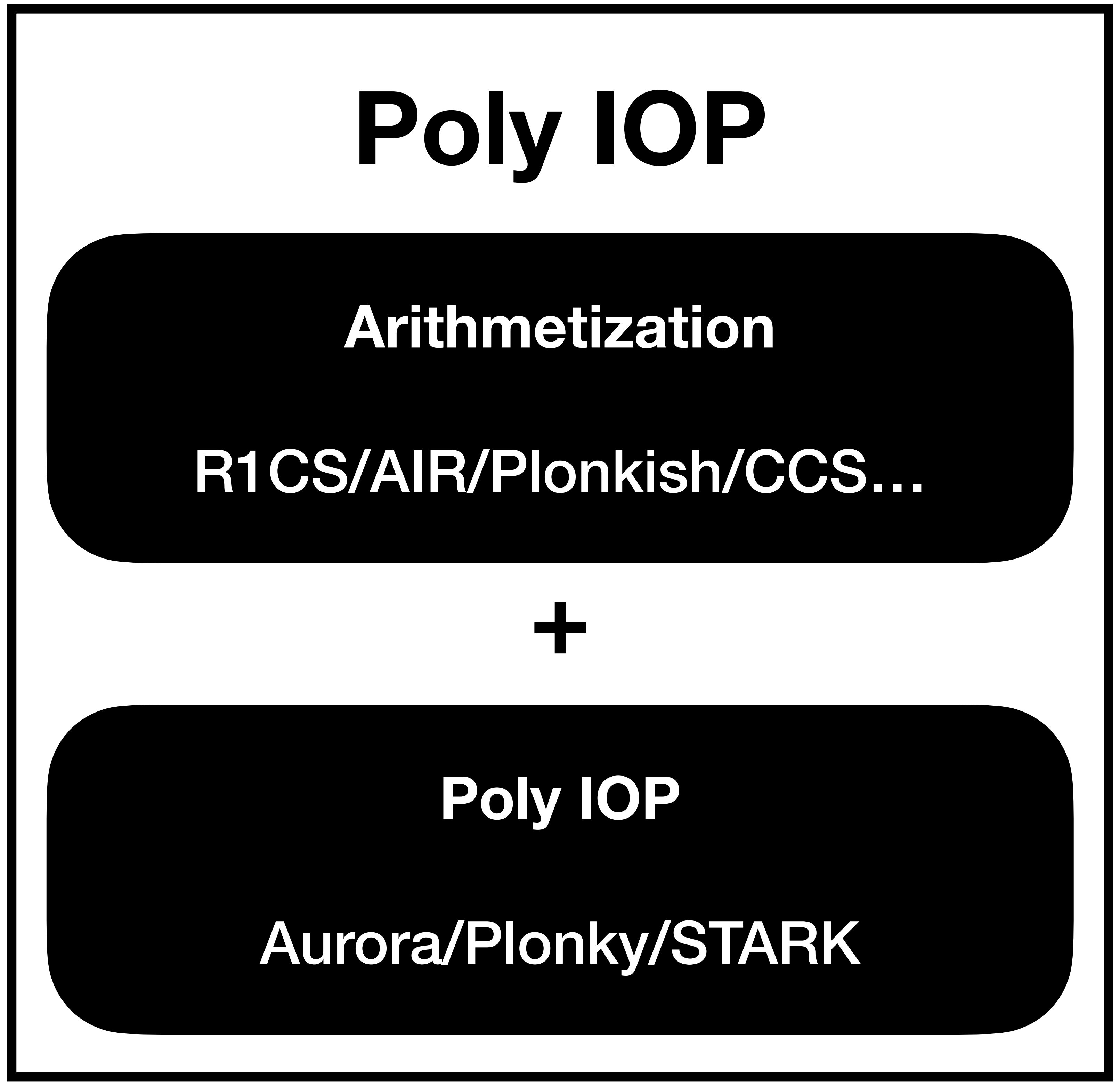
Poly IOP

Arithmetization

R1CS/AIR/Plonkish/CCS…

Anatomy of an IOP-based SNARK

Reducing to low-degree testing



Poly IOP

Arithmetization

R1CS/AIR/Plonkish/CCS…

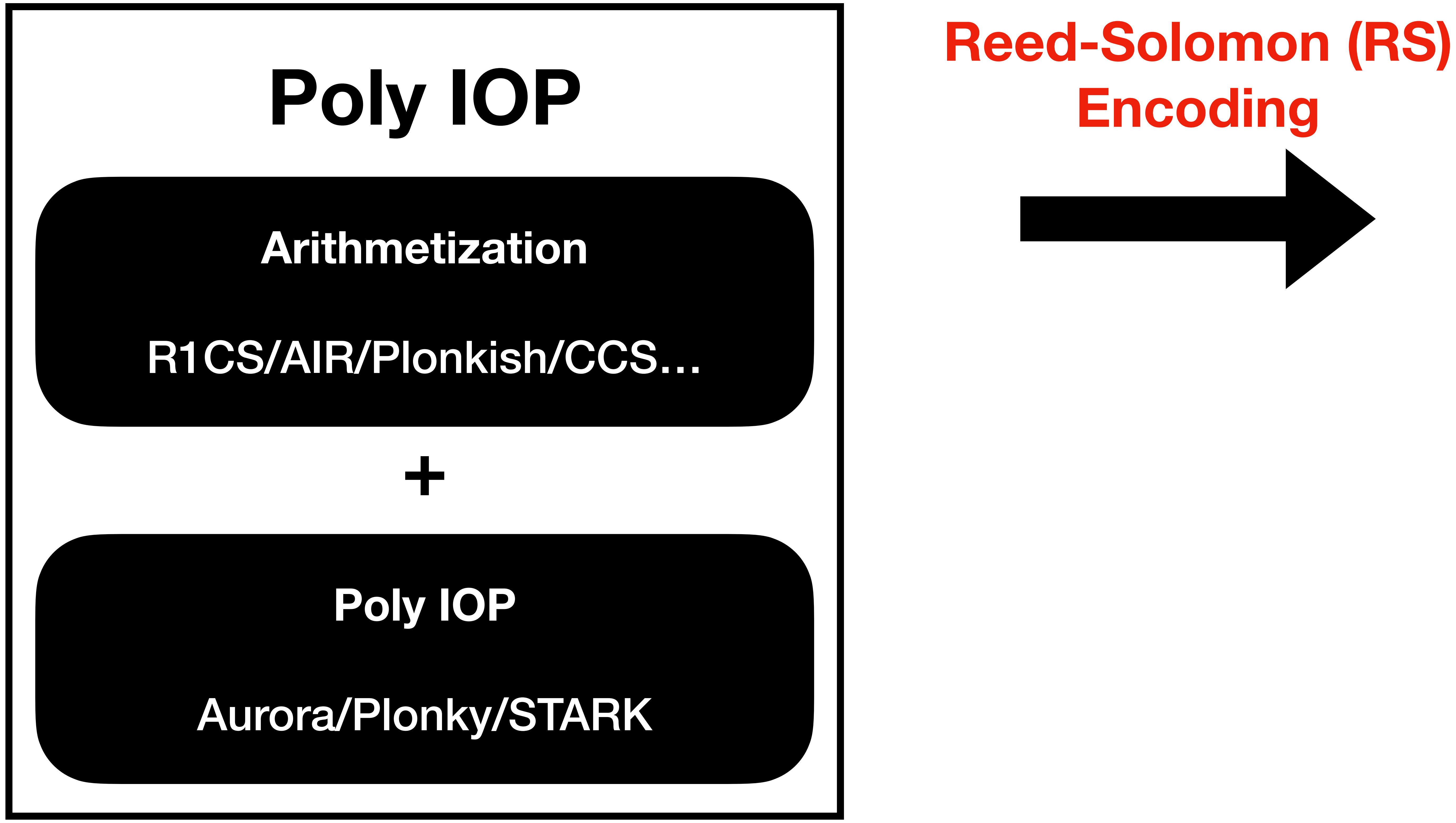
+

Poly IOP

Aurora/Plonky/STARK

Anatomy of an IOP-based SNARK

Reducing to low-degree testing



Poly IOP

Arithmetization

R1CS/AIR/Plonkish/CCS…

+

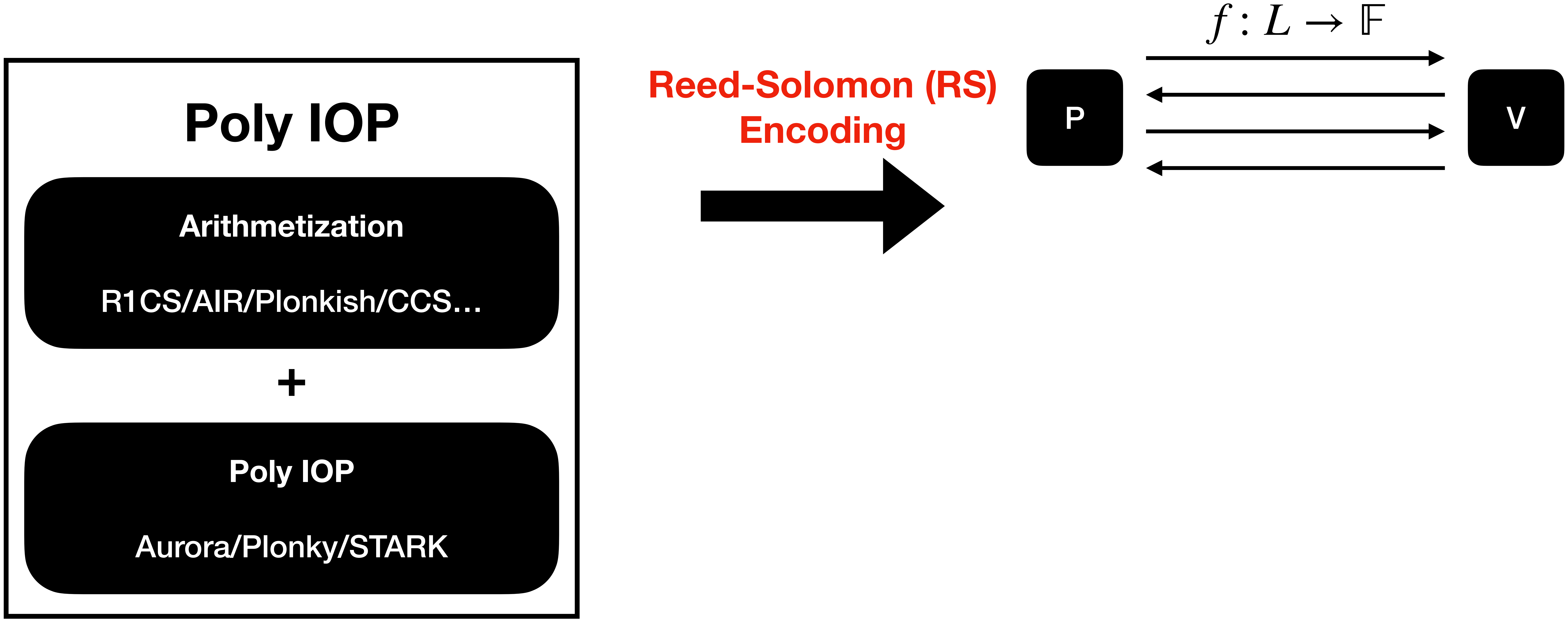
Poly IOP

Aurora/Plonky/STARK

Reed-Solomon (RS) Encoding

Anatomy of an IOP-based SNARK

Reducing to low-degree testing



Poly IOP

Arithmetization

R1CS/AIR/Plonkish/CCS…

+

Poly IOP

Aurora/Plonky/STARK

Reed-Solomon (RS) Encoding

Anatomy of an IOP-based SNARK

Reducing to low-degree testing

Poly IOP

Arithmetization

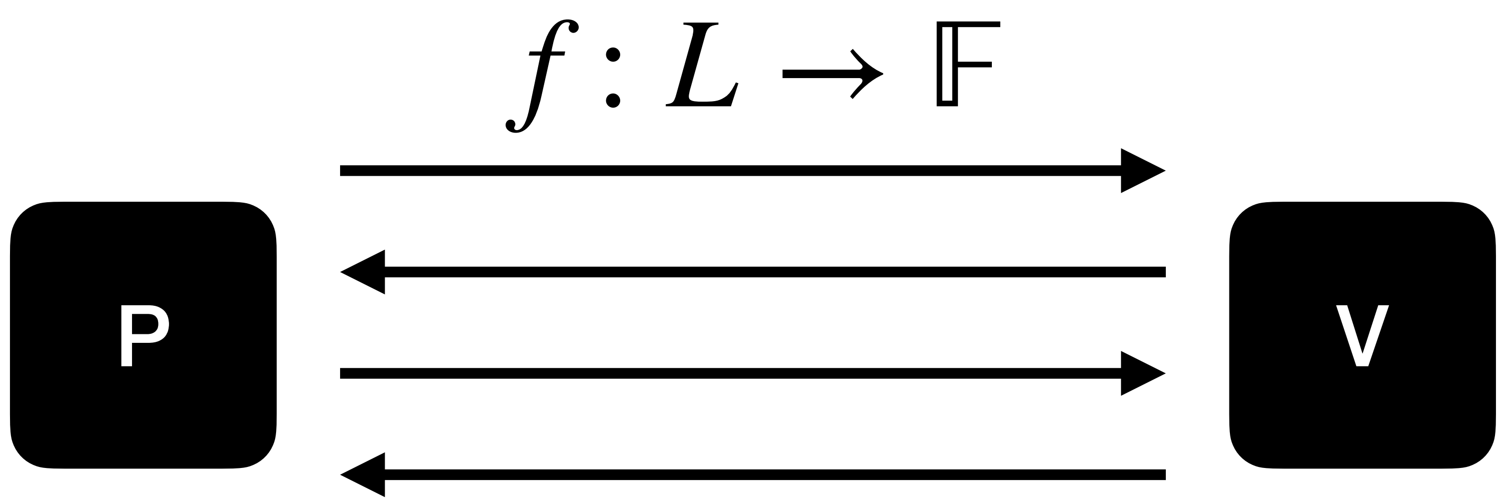
R1CS/AIR/Plonkish/CCS…

+

Poly IOP

Aurora/Plonky/STARK

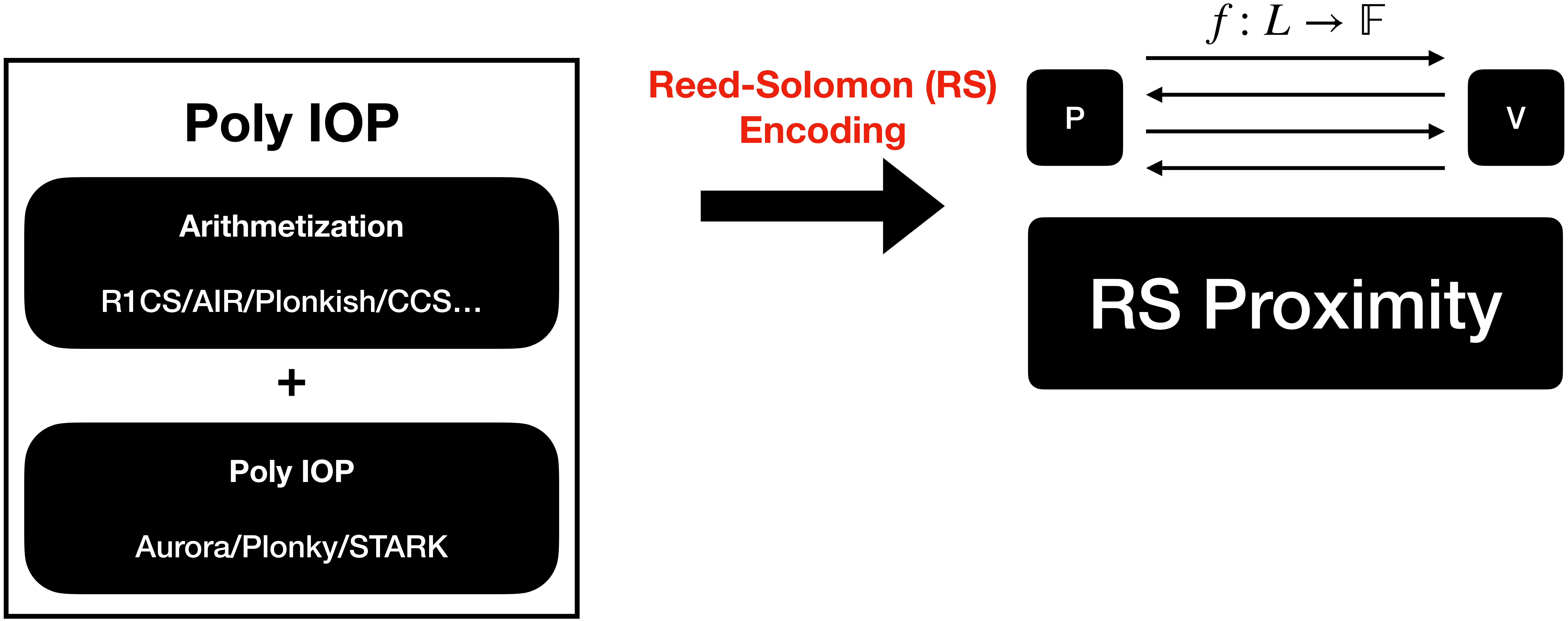
Reed-Solomon (RS) Encoding



$\mathbf{V}$ needs to test f is close to a low-degree function

Anatomy of an IOP-based SNARK

Reducing to low-degree testing



Poly IOP

Arithmetization

R1CS/AIR/Plonkish/CCS…

+

Poly IOP

Aurora/Plonky/STARK

Reed-Solomon (RS) Encoding

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

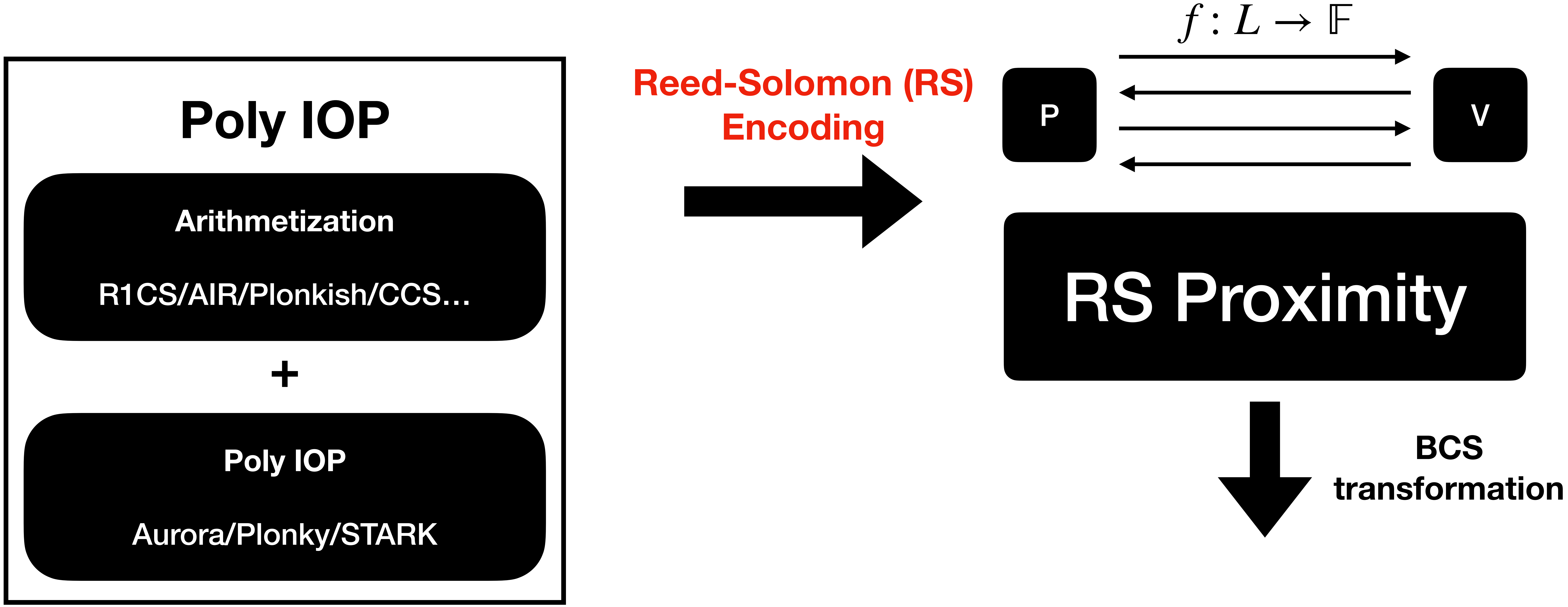
P

V

RS Proximity

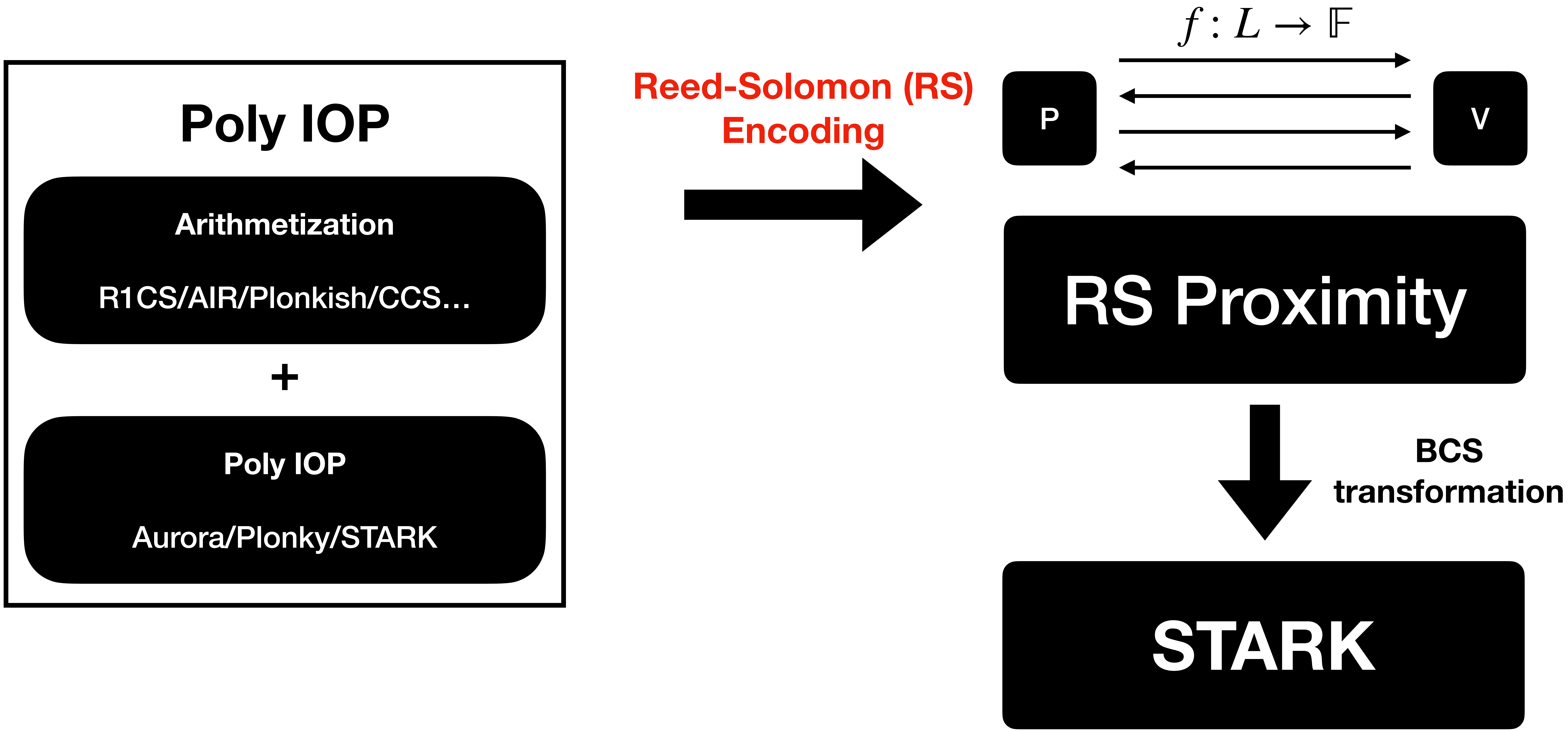
Anatomy of an IOP-based SNARK

Reducing to low-degree testing



Anatomy of an IOP-based SNARK

Reducing to low-degree testing



Poly IOP

Arithmetization

R1CS/AIR/Plonkish/CCS…

+

Poly IOP

Aurora/Plonky/STARK

Reed-Solomon (RS) Encoding

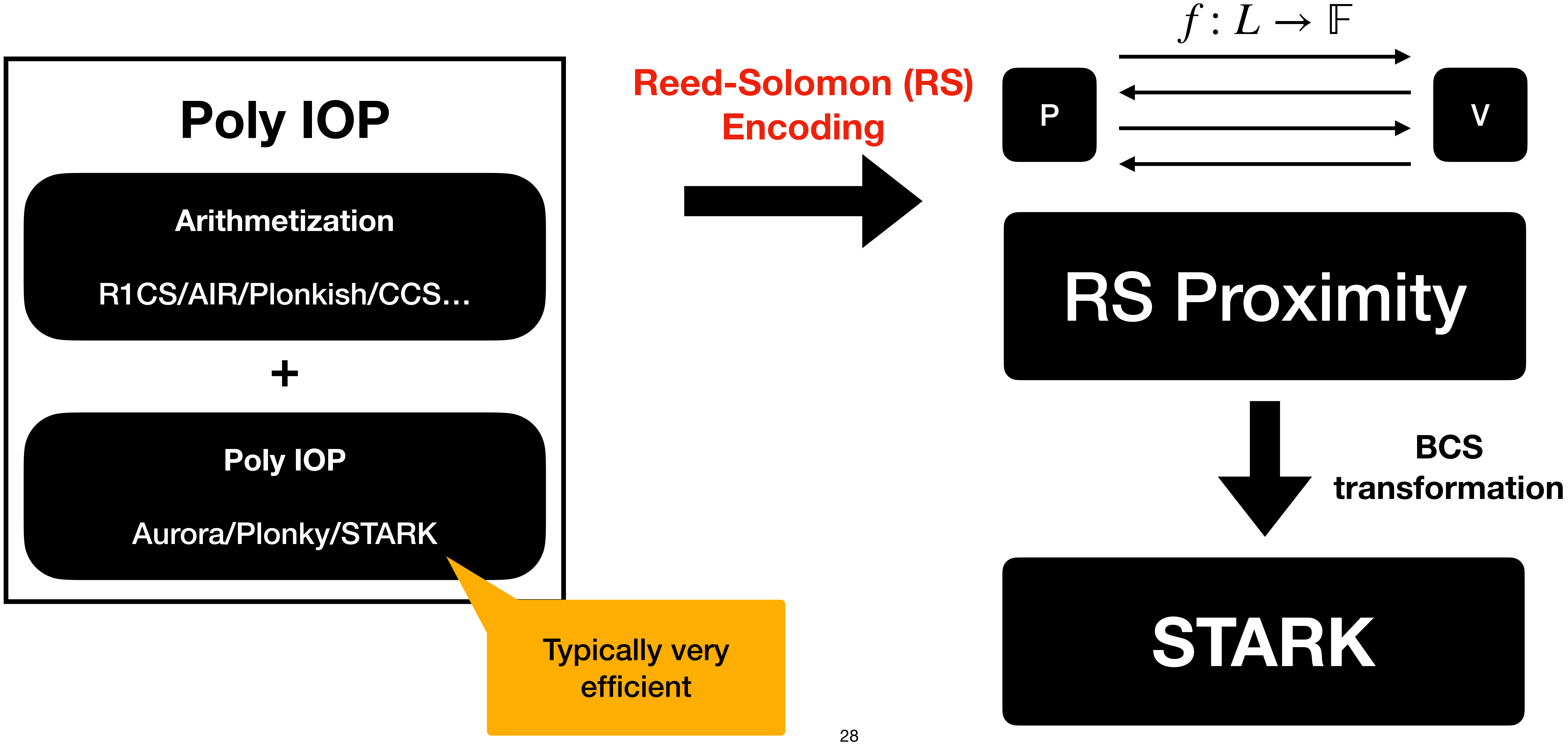
RS Proximity

BCS transformation

STARK

Anatomy of an IOP-based SNARK

Reducing to low-degree testing



Poly IOP

Arithmetization

R1CS/AIR/Plonkish/CCS…

+

Poly IOP

Aurora/Plonky/STARK

Typically very efficient

Reed-Solomon (RS) Encoding

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

P

V

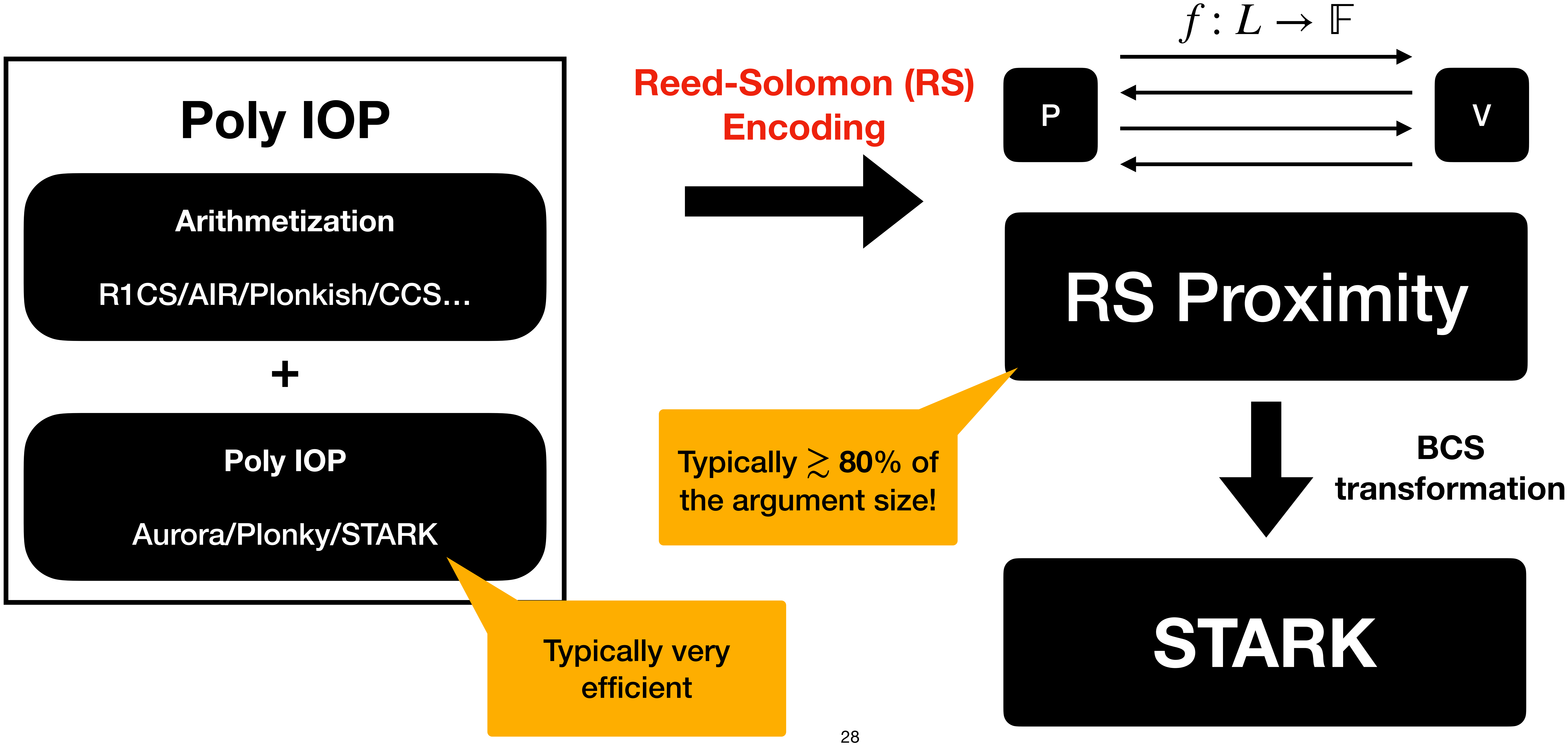
RS Proximity

BCS transformation

STARK

Anatomy of an IOP-based SNARK

Reducing to low-degree testing



Poly IOP

Arithmetization

R1CS/AIR/Plonkish/CCS…

+

Poly IOP

Aurora/Plonky/STARK

Typically very efficient

Reed-Solomon (RS) Encoding

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

P

V

RS Proximity

Typically $\gtrsim$ 80% of the argument size!

BCS transformation

STARK

STARKs & Friends

STARKs & Friends

- **IOP**-based SNARKs instantiated using the BCS transformation

STARKs & Friends

- **IOP**-based SNARKs instantiated using the BCS transformation
- Secure in the ROM: can be instantiated with only hash-functions

STARKs & Friends

- **IOP**-based SNARKs instantiated using the BCS transformation
- Secure in the ROM: can be instantiated with only hash-functions
- Transparent (public-coin) setup

STARKs & Friends

- **IOP**-based SNARKs instantiated using the BCS transformation
- Secure in the ROM: can be instantiated with only hash-functions
- Transparent (public-coin) setup
- Fast proving, not tied to ECC fields, post quantum secure in QROM [CMS].

STARKs & Friends

- **IOP**-based SNARKs instantiated using the BCS transformation
- Secure in the ROM: can be instantiated with only hash-functions
- Transparent (public-coin) setup
- Fast proving, not tied to ECC fields, post quantum secure in QROM [CMS].

Rollups: STARKWARE polygon zkSync

STARKs & Friends

- **IOP**-based SNARKs instantiated using the BCS transformation
- Secure in the ROM: can be instantiated with only hash-functions
- Transparent (public-coin) setup
- Fast proving, not tied to ECC fields, post quantum secure in QROM [CMS].

Rollups: STARKWARE polygon zkSync

zkVMs: Succinct RISC ZERO

STARKs & Friends

- **IOP**-based SNARKs instantiated using the BCS transformation
- Secure in the ROM: can be instantiated with only hash-functions
- Transparent (public-coin) setup
- Fast proving, not tied to ECC fields, post quantum secure in QROM [CMS].

Rollups: STARKWARE polygon zkSync

zkVMs: Succinct RISC ZERO **And more…**

What about the conjecture?

FRI and STIR benefit in roughly the same way

- Conjecture on list-decoding up to distance $1 - \rho$ (instead of $1 - \sqrt{\rho}$)
- FRI queries:

$$O\left(\lambda \cdot \frac{\log d}{-\log \rho}\right)$$

In both, for $\delta = 1 - \rho$,

reduces queries by ~2x

- STIR queries:

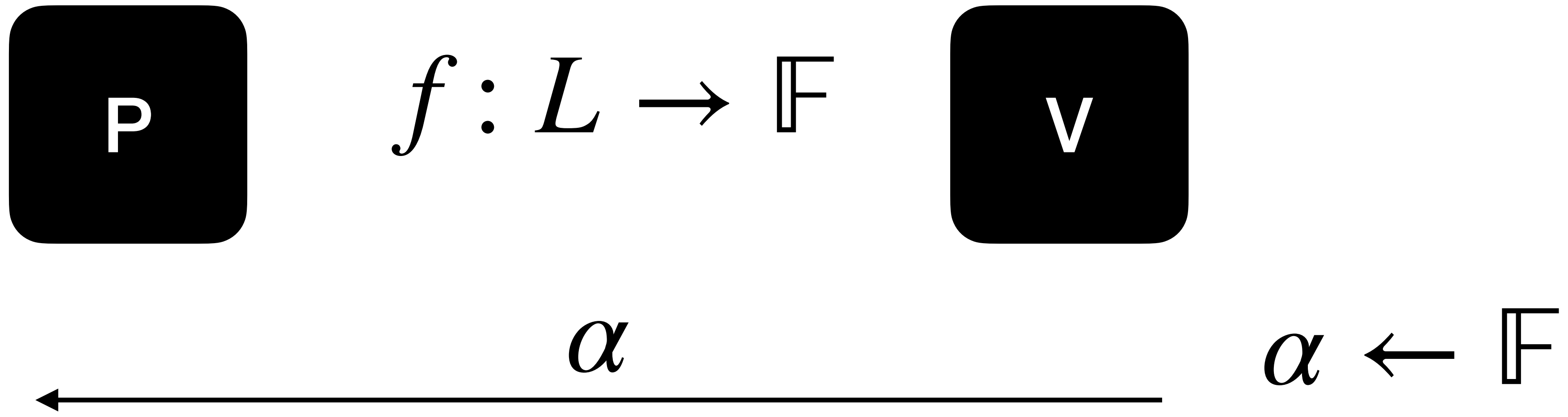
$$O\left(\lambda \cdot \log\left(\frac{\log d}{-\log \rho}\right) + \log d\right)$$

STIR iteration

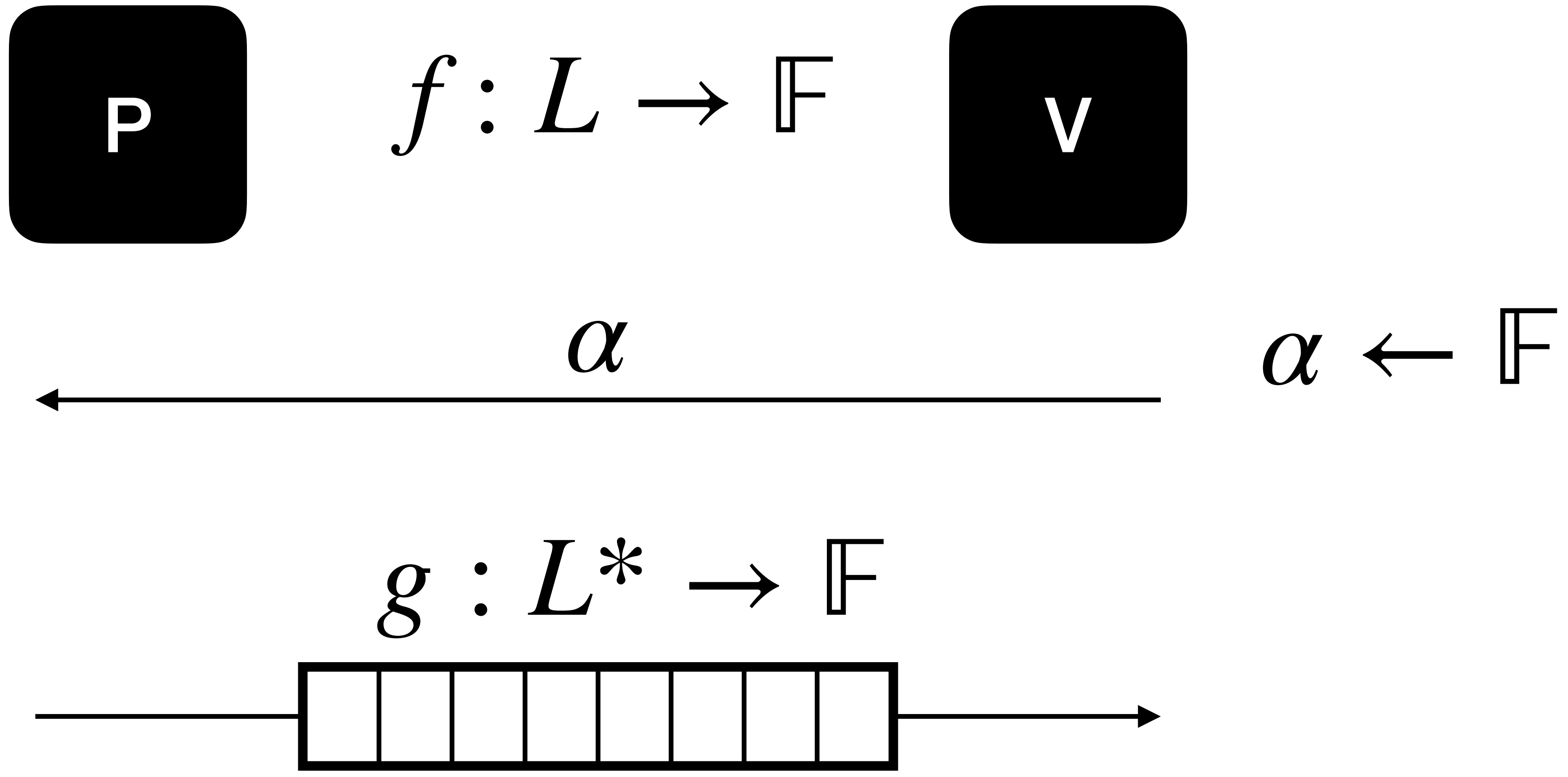
STIR iteration

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

STIR iteration



STIR iteration



P

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

V

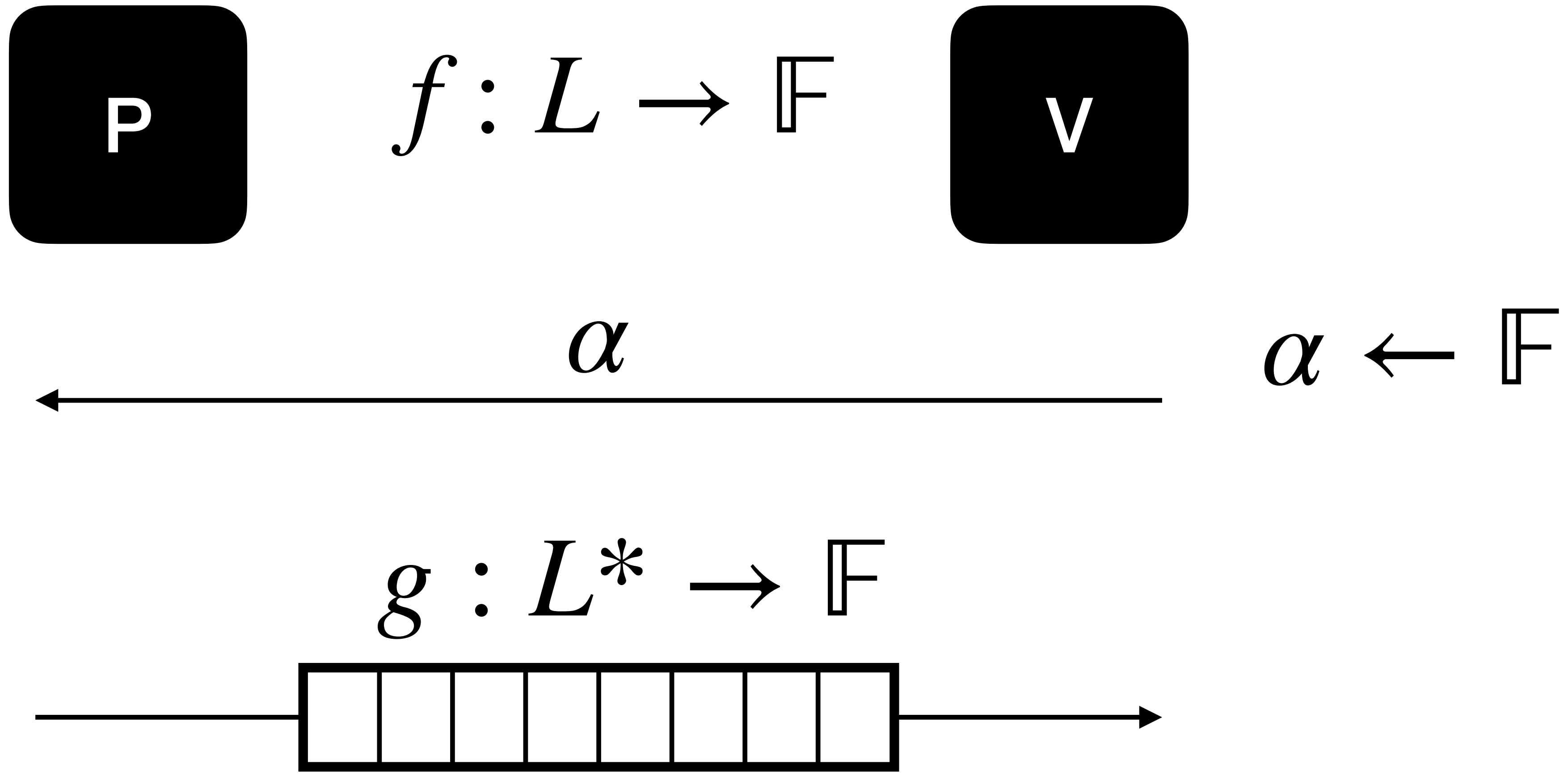
α

$\alpha \leftarrow \mathbb{F}$

$g : L^* \to \mathbb{F}$

STIR iteration

g is *claimed* to be equal to (the extension of) Fold(f, k, α) on L^*



P

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

V

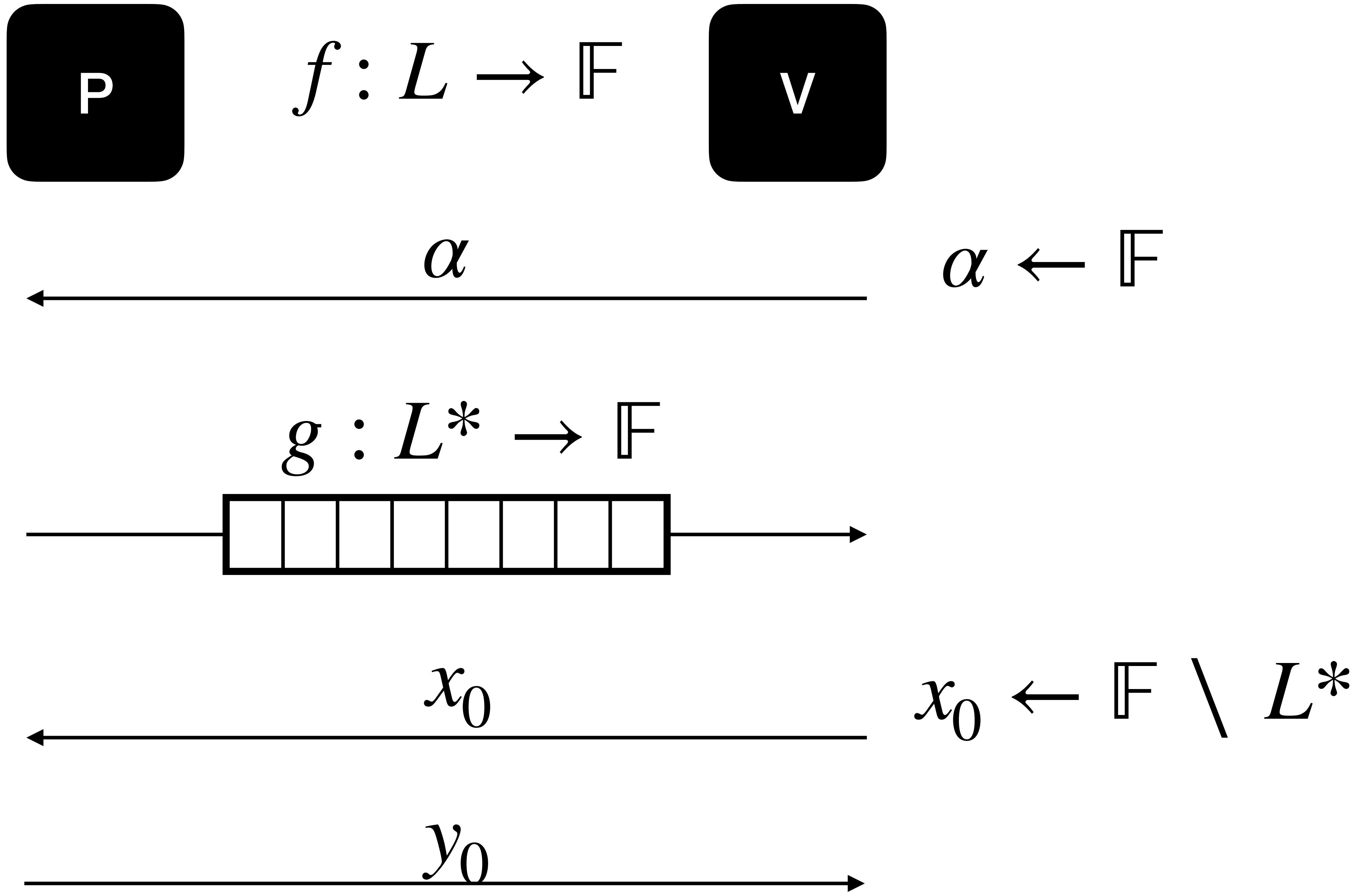
α

$\alpha \leftarrow \mathbb{F}$

$g : L^* \to \mathbb{F}$

STIR iteration

g is *claimed* to be equal to (the extension of) Fold(f, k, α) on L^*



P

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

V

α

$\alpha \leftarrow \mathbb{F}$

$g : L^* \to \mathbb{F}$

x_0

$x_0 \leftarrow \mathbb{F} \setminus L^*$

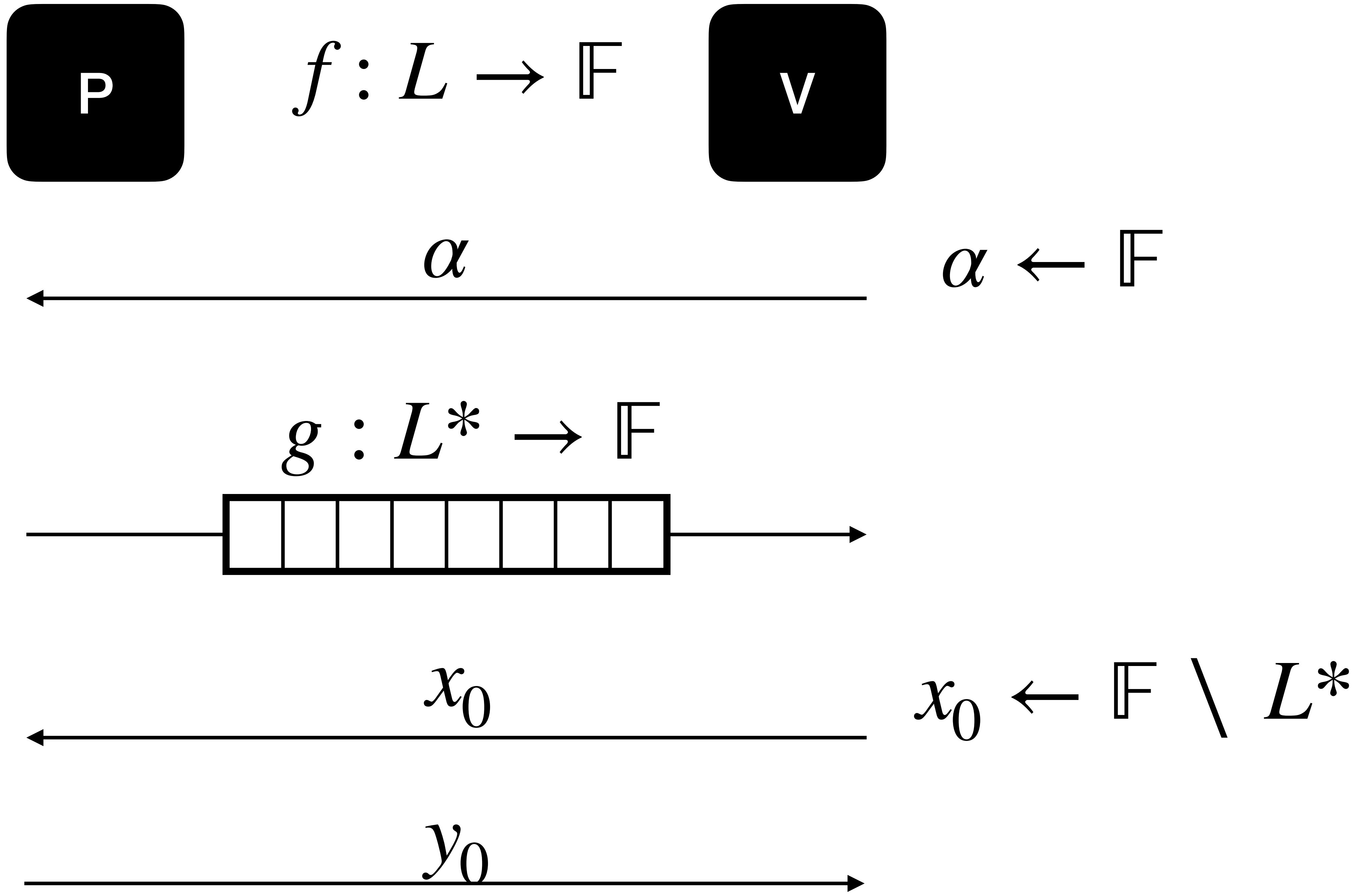
y_0

STIR iteration

g is *claimed* to be equal to (the extension of) Fold(f, k, α) on L^*

Problem: We can only query Fold(f, k, α) on $L^k \neq L^*$.

Enforce consistency via Quotient!


P
$f : L \to \mathbb{F}$
V
α
$\alpha \leftarrow \mathbb{F}$
$g : L^* \to \mathbb{F}$
x_0
$x_0 \leftarrow \mathbb{F} \setminus L^*$
y_0

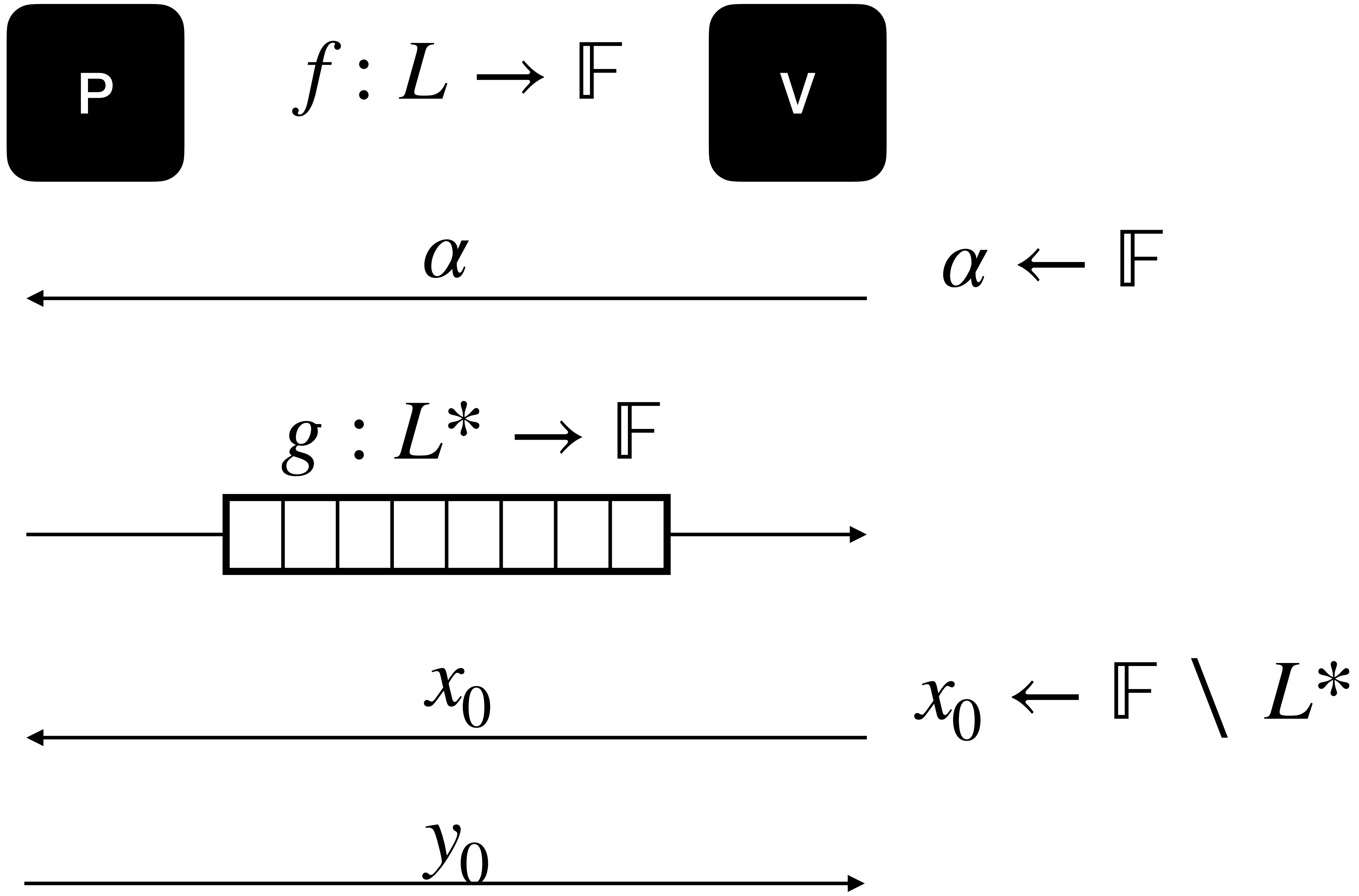
STIR iteration

g is *claimed* to be equal to (the extension of) Fold(f, k, α) on L^*

Problem: We can only query Fold(f, k, α) on $L^k \neq L^*$.

Enforce consistency via Quotient!

Query Fold(f, k, α) at $x_1, \ldots, x_t \in L^k$ to get $y_1, \ldots, y_t$


P
$f : L \to \mathbb{F}$
V
α
$\alpha \leftarrow \mathbb{F}$
$g : L^* \to \mathbb{F}$
x_0
$x_0 \leftarrow \mathbb{F} \setminus L^*$
y_0

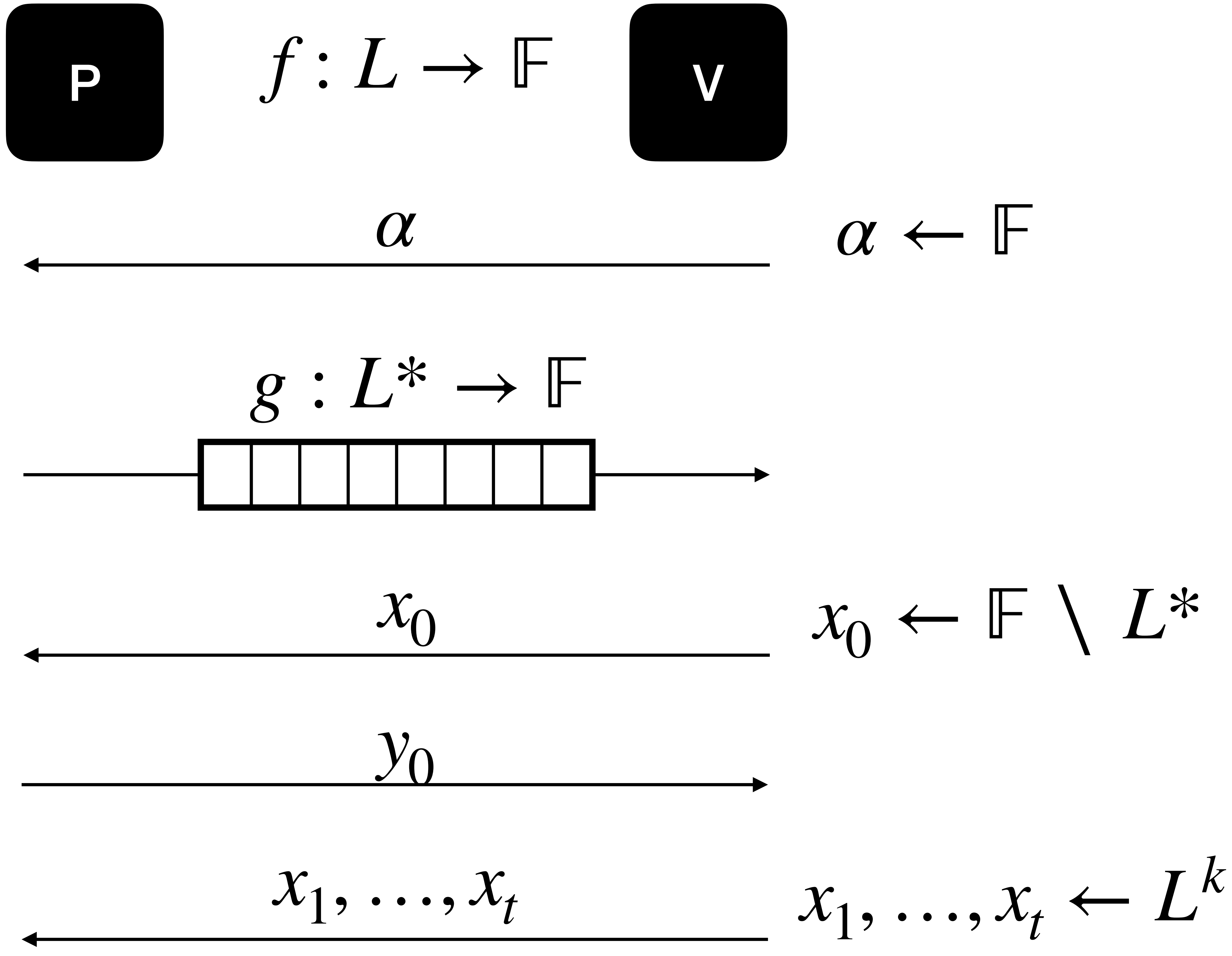
STIR iteration

g is *claimed* to be equal to (the extension of) Fold(f, k, α) on L^*

Problem: We can only query Fold(f, k, α) on $L^k \neq L^*$.

Enforce consistency via Quotient!

Query Fold(f, k, α) at $x_1, \ldots, x_t \in L^k$ to get $y_1, \ldots, y_t$


P
$f : L \to \mathbb{F}$
V
α
$\alpha \leftarrow \mathbb{F}$
$g : L^* \to \mathbb{F}$
x_0
$x_0 \leftarrow \mathbb{F} \setminus L^*$
y_0
$x_1, \ldots, x_t$
$x_1, \ldots, x_t \leftarrow L^k$

$\forall\ i$: Ans(x_i) $= y_i$

f' := Quotient(g, Ans)

Query f to get

y_i = Fold(f, k, α)(x_i)

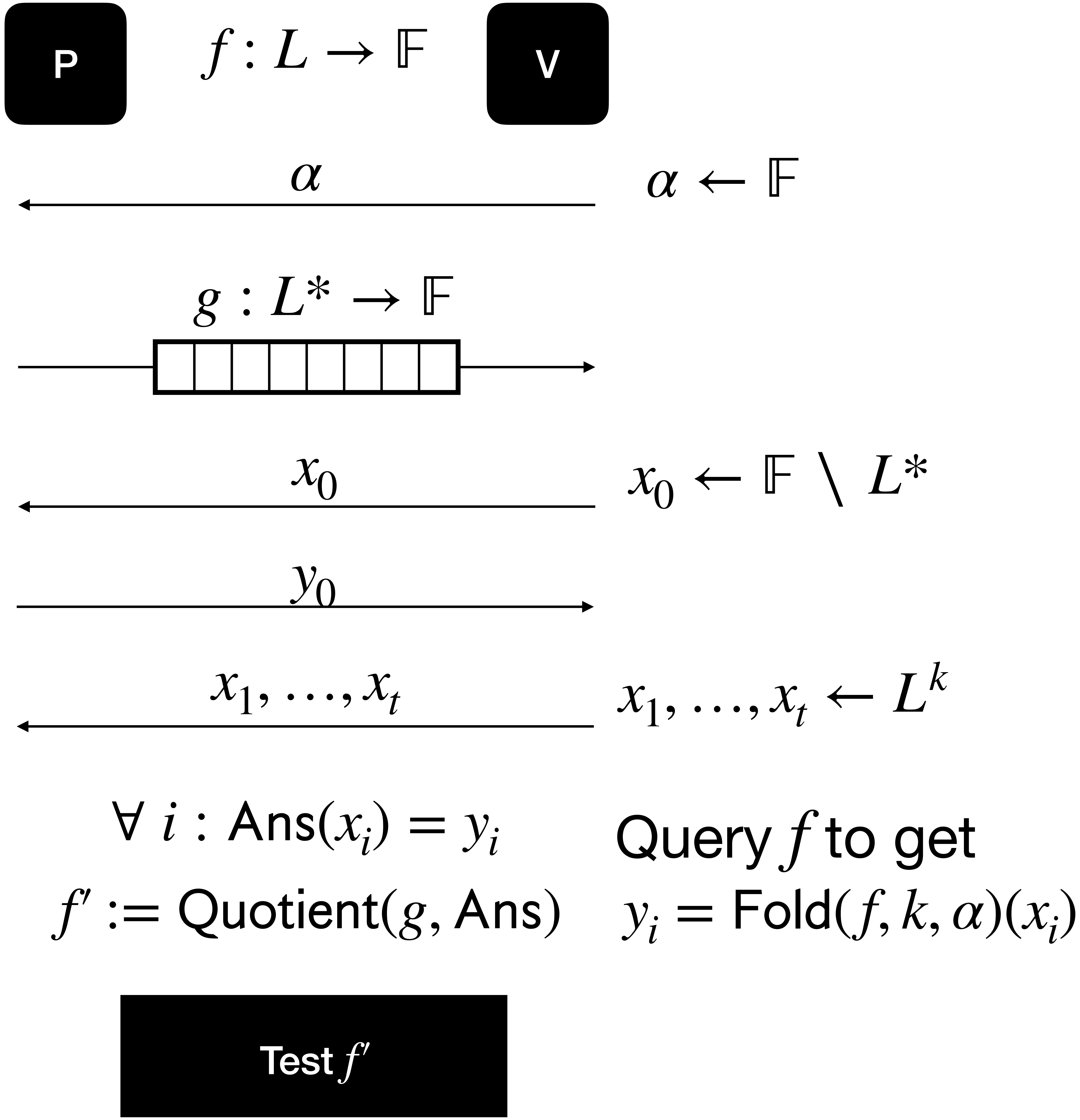
STIR iteration

g is *claimed* to be equal to (the extension of) Fold(f, k, α) on L^*

Problem: We can only query Fold(f, k, α) on $L^k \neq L^*$.

Enforce consistency via Quotient!

Query Fold(f, k, α) at $x_1, \ldots, x_t \in L^k$ to get $y_1, \ldots, y_t$



P

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

V

α

$\alpha \leftarrow \mathbb{F}$

$g : L^* \to \mathbb{F}$

x_0

$x_0 \leftarrow \mathbb{F} \setminus L^*$

y_0

$x_1, \ldots, x_t$

$x_1, \ldots, x_t \leftarrow L^k$

$\forall\ i\ :\ \text{Ans}(x_i) = y_i$

$f' :=$ Quotient(g, Ans)

Query f to get

$y_i =$ Fold$(f, k, \alpha)(x_i)$

Test f'

STIR iteration

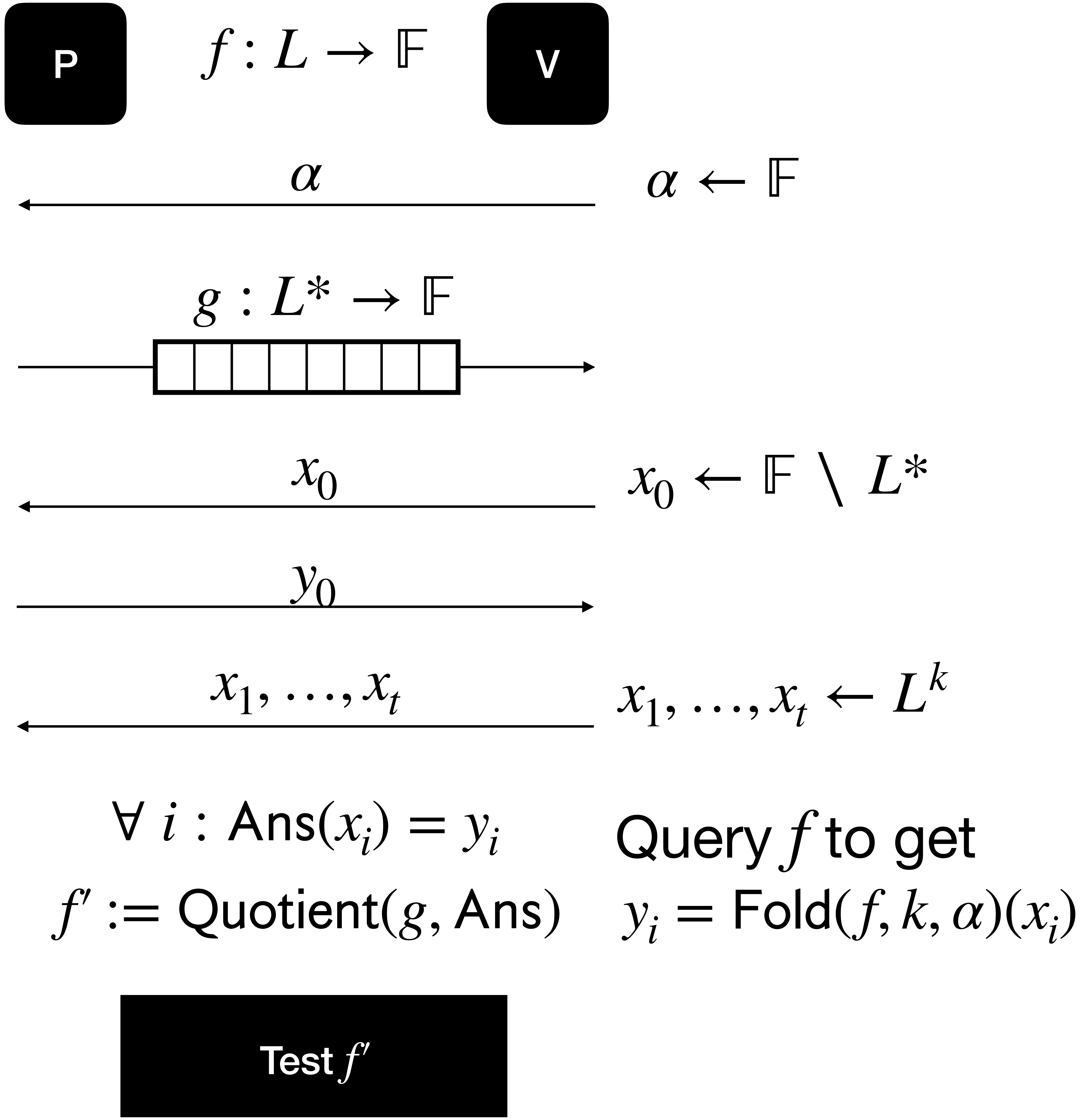
g is *claimed* to be equal to (the extension of) Fold(f, k, α) on L^*

Problem: We can only query Fold(f, k, α) on $L^k \neq L^*$.

Enforce consistency via Quotient!

Query Fold(f, k, α) at $x_1, \ldots, x_t \in L^k$ to get $y_1, \ldots, y_t$

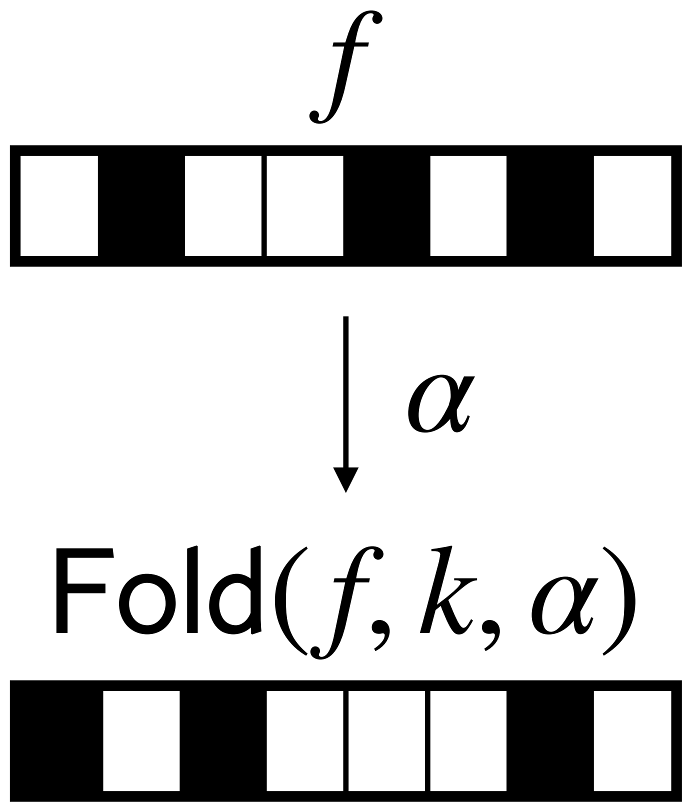
New function is quotient of g w.r.t. to these points + OOD sample


P
$f : L \to \mathbb{F}$
V
α
$\alpha \leftarrow \mathbb{F}$
$g : L^* \to \mathbb{F}$
x_0
$x_0 \leftarrow \mathbb{F} \setminus L^*$
y_0
$x_1, \ldots, x_t$
$x_1, \ldots, x_t \leftarrow L^k$
$\forall\ i :$ Ans$(x_i) = y_i$
$f' :=$ Quotient$(g,$ Ans$)$
Query f to get
$y_i =$ Fold$(f, k, \alpha)(x_i)$
Test f'

Claim: if f is δ-far from C, unless with probability $\approx (1 - \delta)^t, f'$ is $(1 - \sqrt{\rho'})$ far from C'

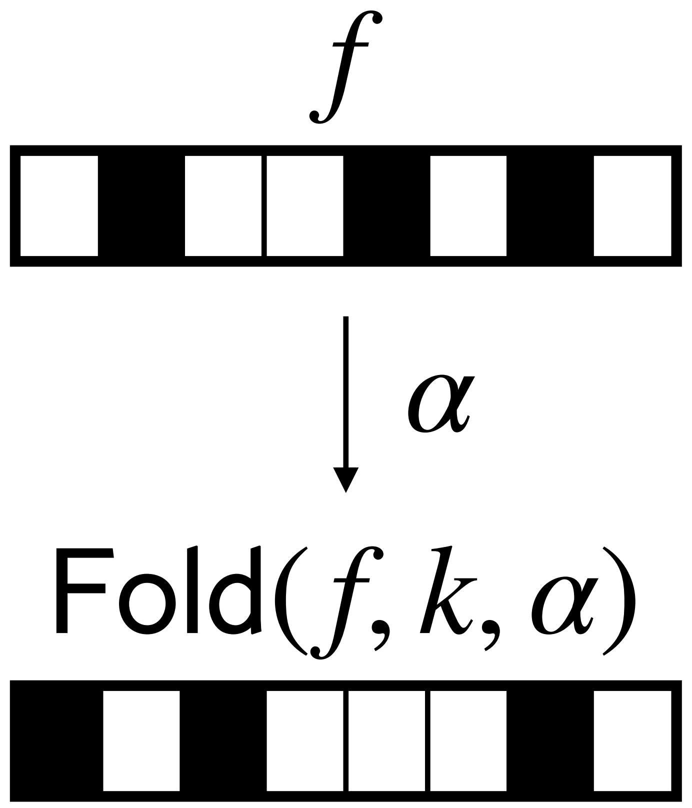
Claim: if f is δ-far from C, unless with probability $\approx (1-\delta)^t, f'$ is $(1-\sqrt{\rho'})$ far from C'

Folding

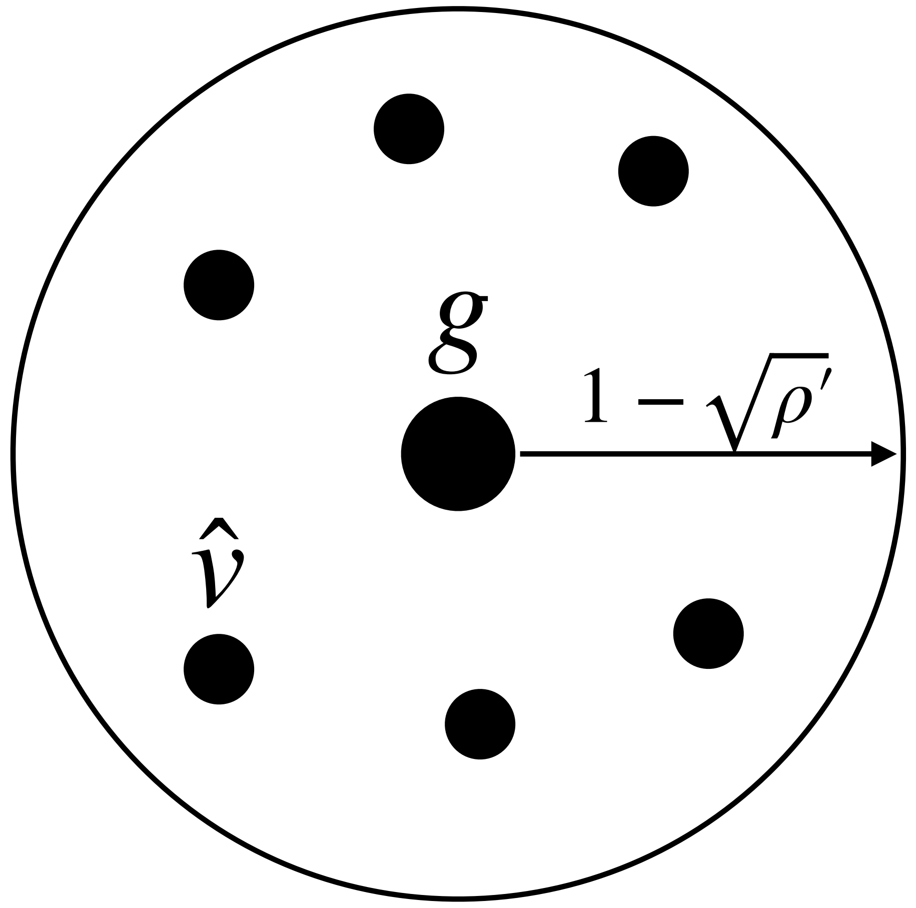


Claim: if f is δ-far from C, unless with probability $\approx (1-\delta)^t, f'$ is $(1-\sqrt{\rho'})$ far from C'

Folding

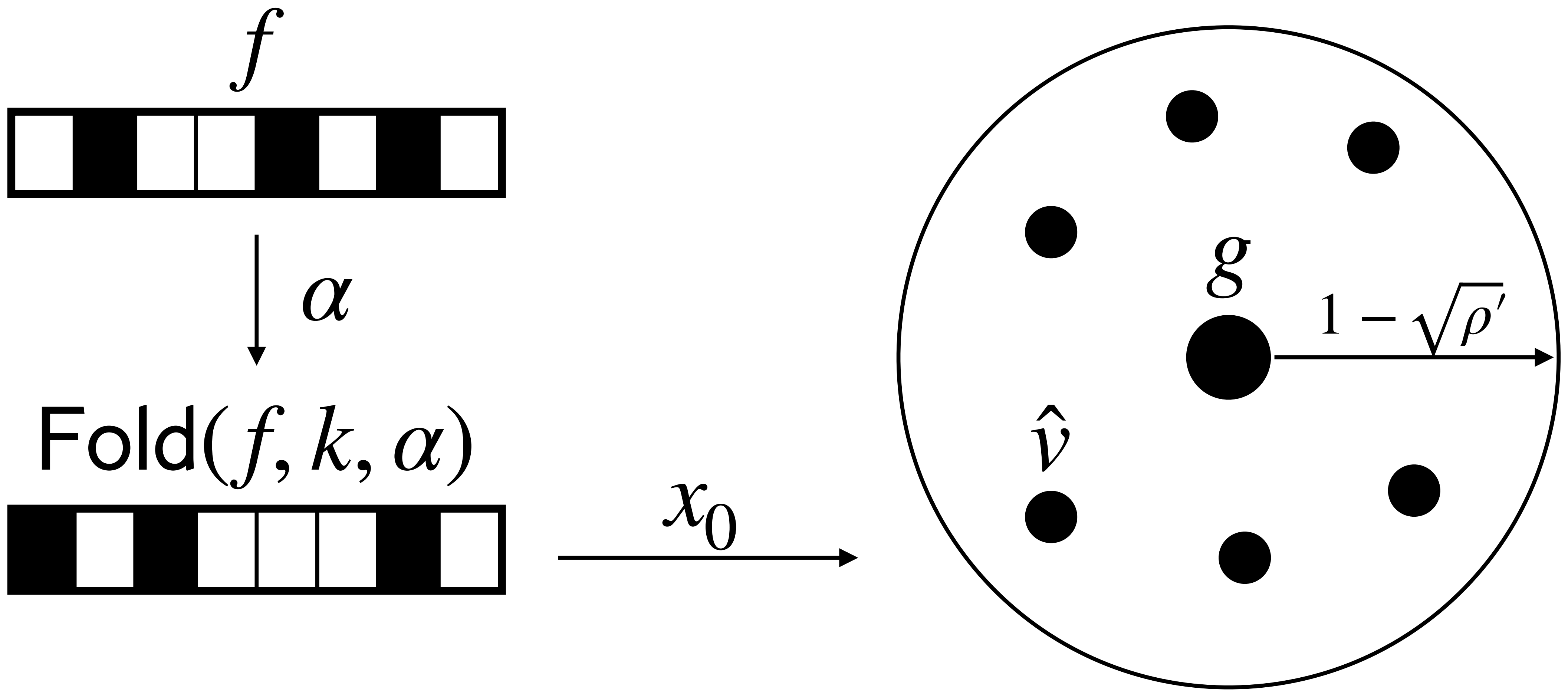


OOD



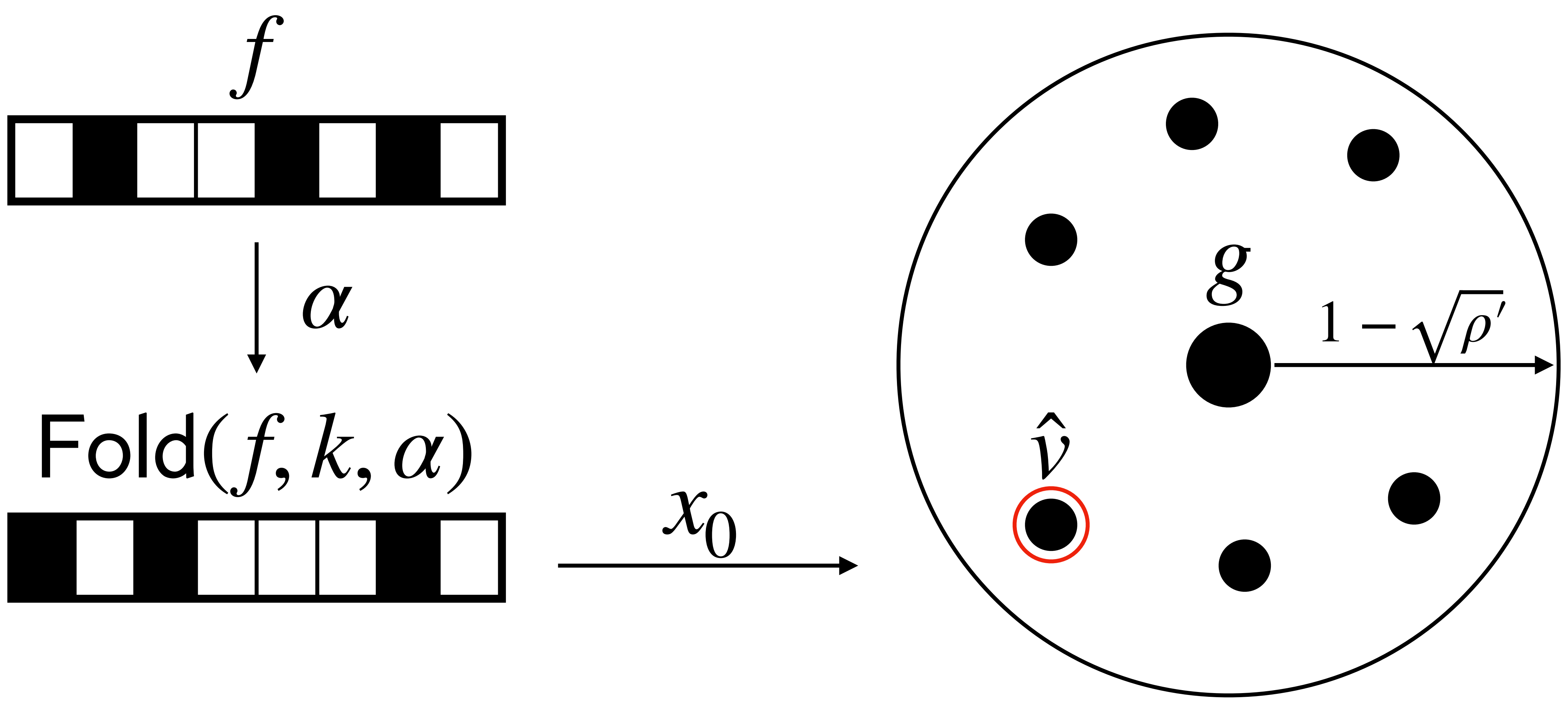
Claim: if f is δ-far from C, unless with probability $\approx (1-\delta)^t$, f' is $(1-\sqrt{\rho'})$ far from C'

Folding **OOD**



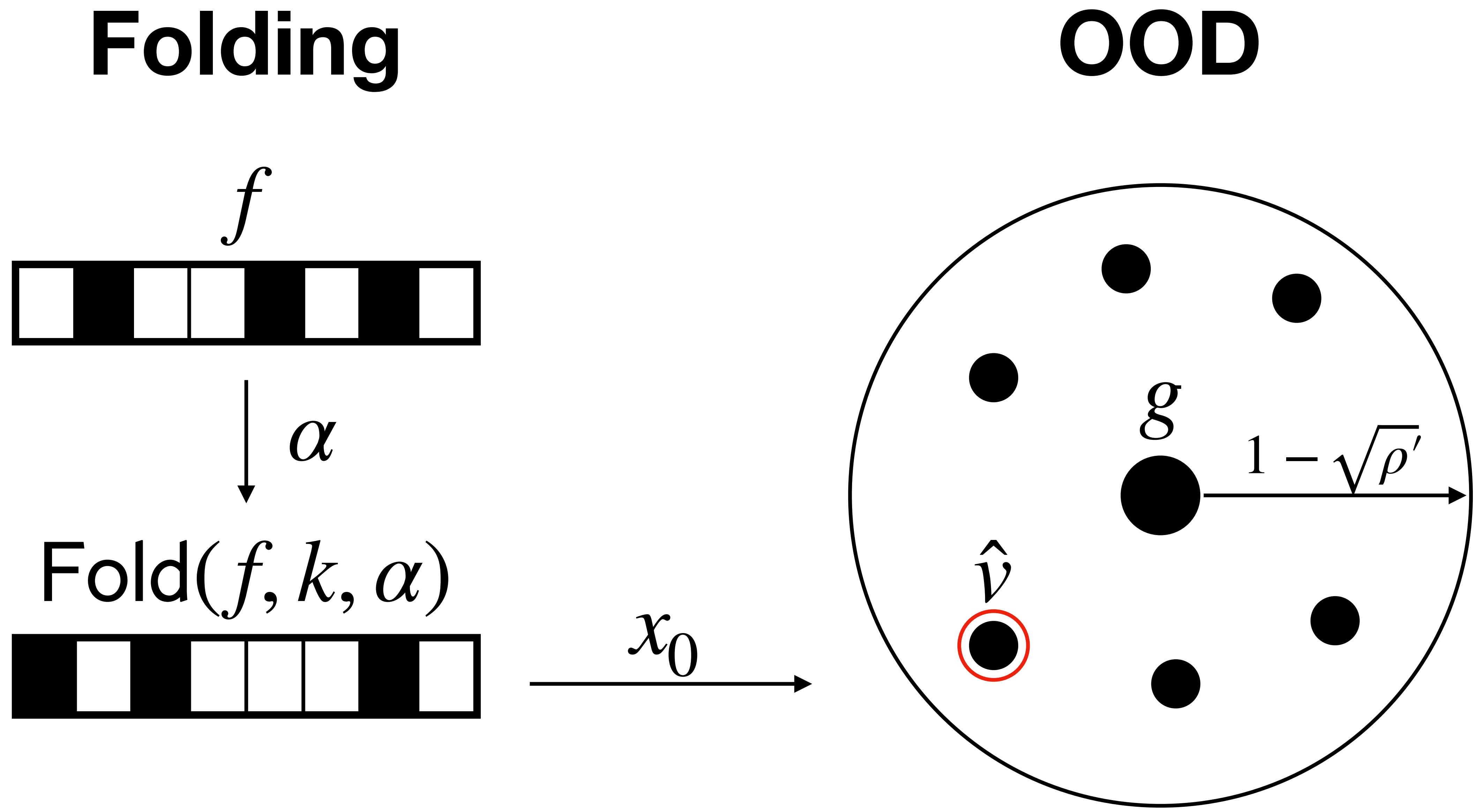
Claim: if f is δ-far from C, unless with probability $\approx (1-\delta)^t, f'$ is $(1-\sqrt{\rho'})$ far from C'

Folding **OOD**



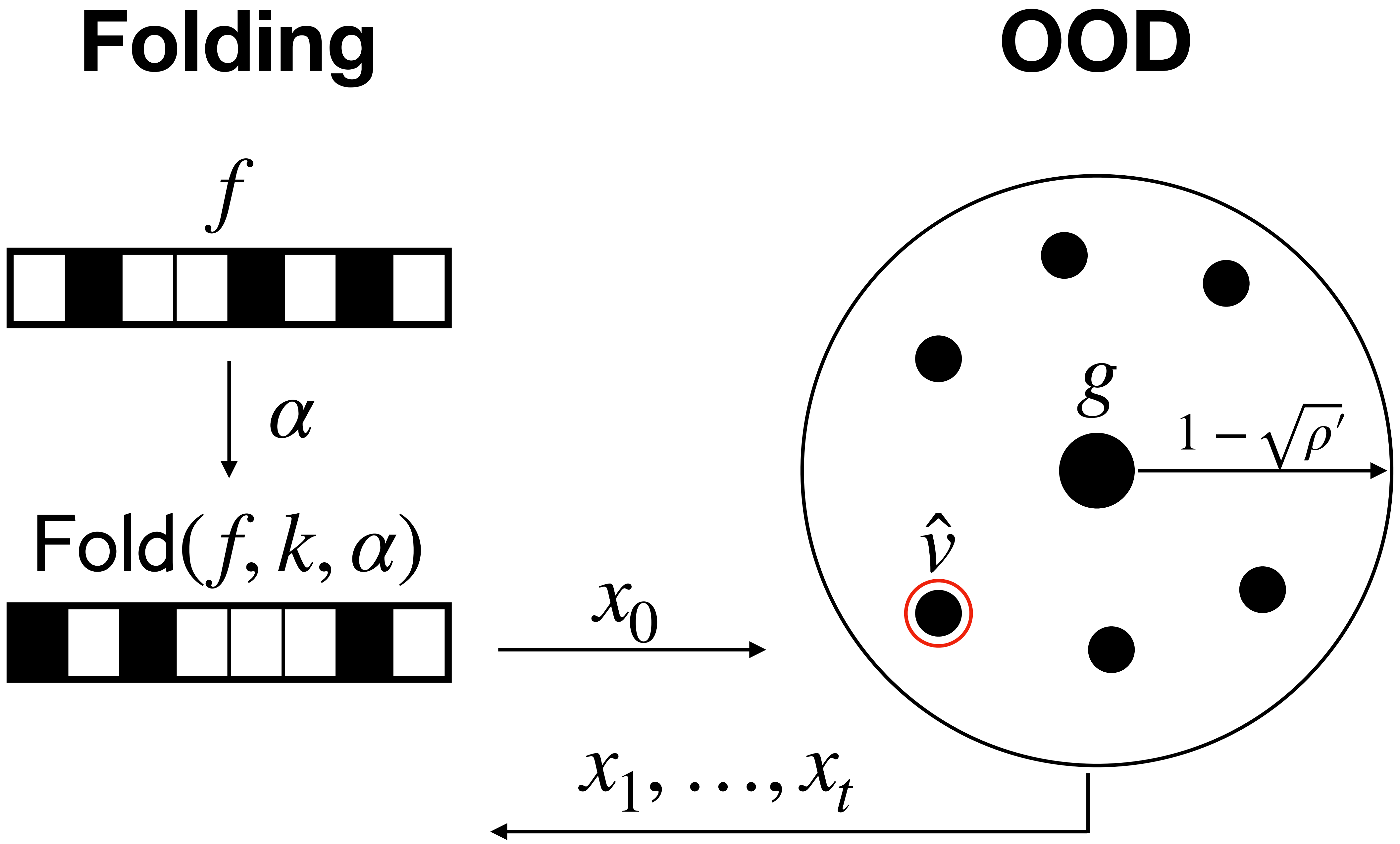
$\hat{v}$ is **unique** close codeword to g with $\hat{v}(x_0) = y_0$

Claim: if f is δ-far from C, unless with probability $\approx (1-\delta)^t, f'$ is $(1-\sqrt{\rho'})$ far from C'



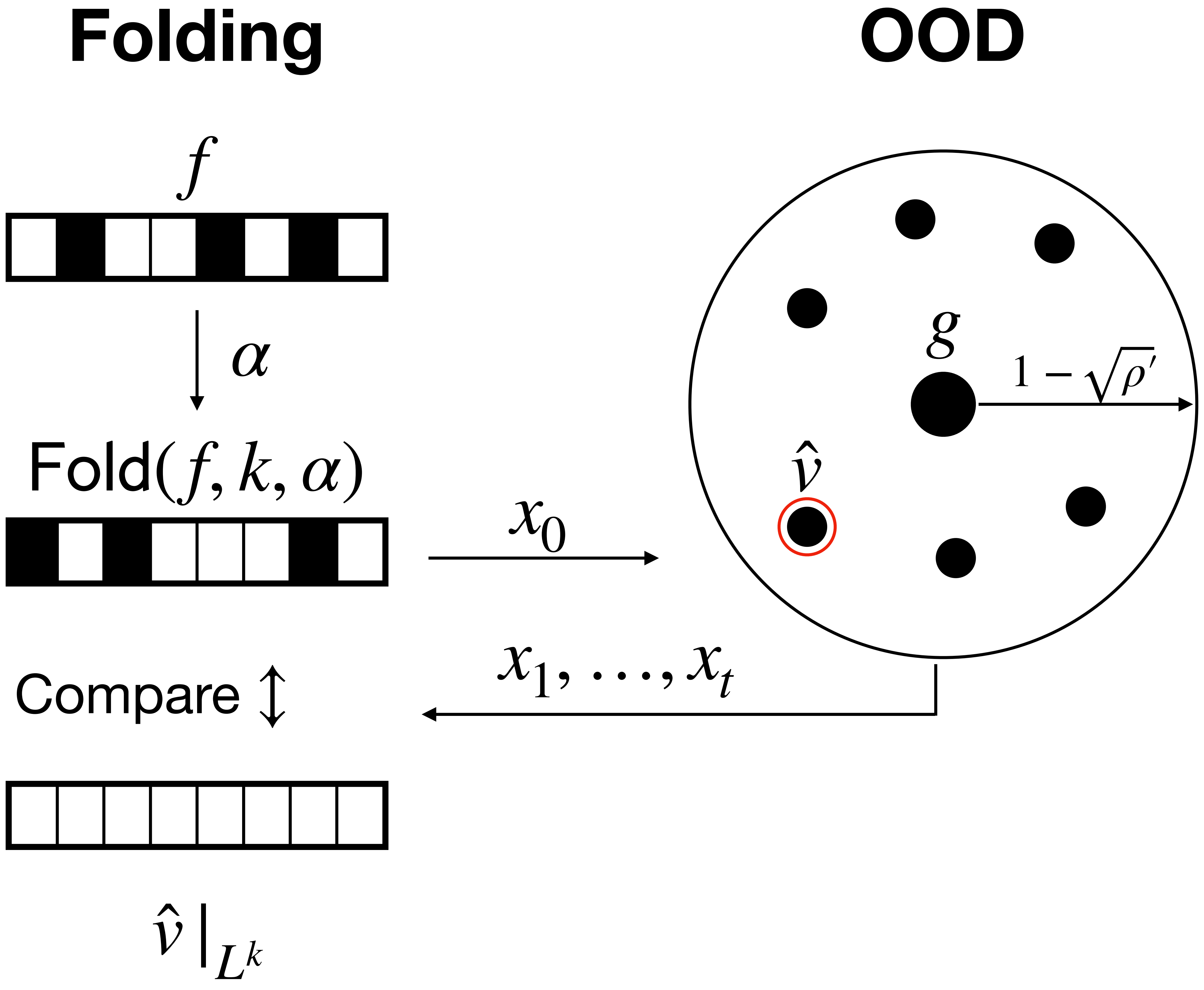
$\hat{v}$ is **unique** close codeword to g with $\hat{v}(x_0) = y_0$

Claim: if f is δ-far from C, unless with probability $\approx (1-\delta)^t$, f' is $(1-\sqrt{\rho'})$ far from C'

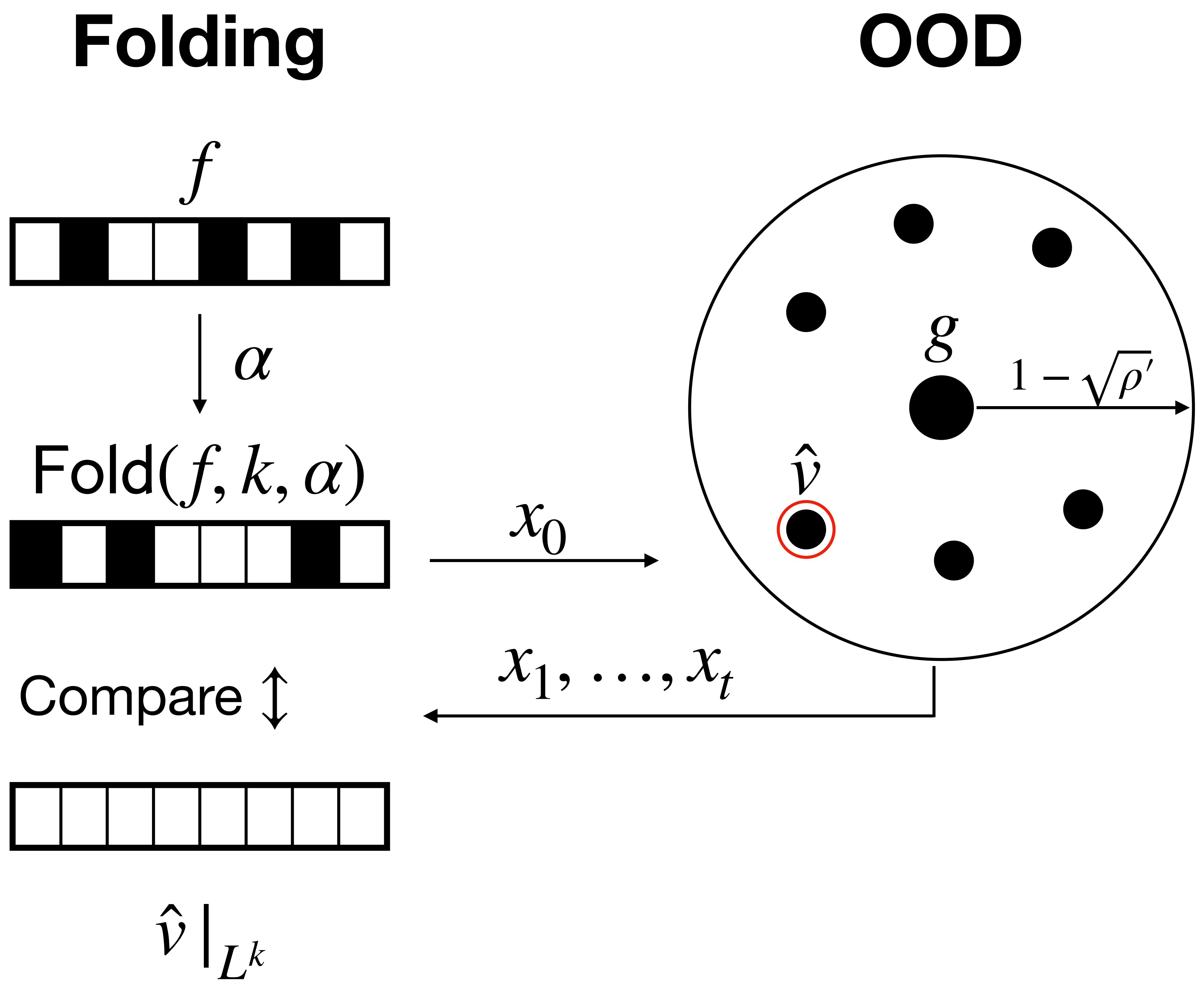


Claim: if f is δ-far from C, unless with probability $\approx (1-\delta)^t$, f' is $(1-\sqrt{\rho'})$ far from C'

$\hat{v}$ is **unique** close codeword to g with $\hat{v}(x_0) = y_0$



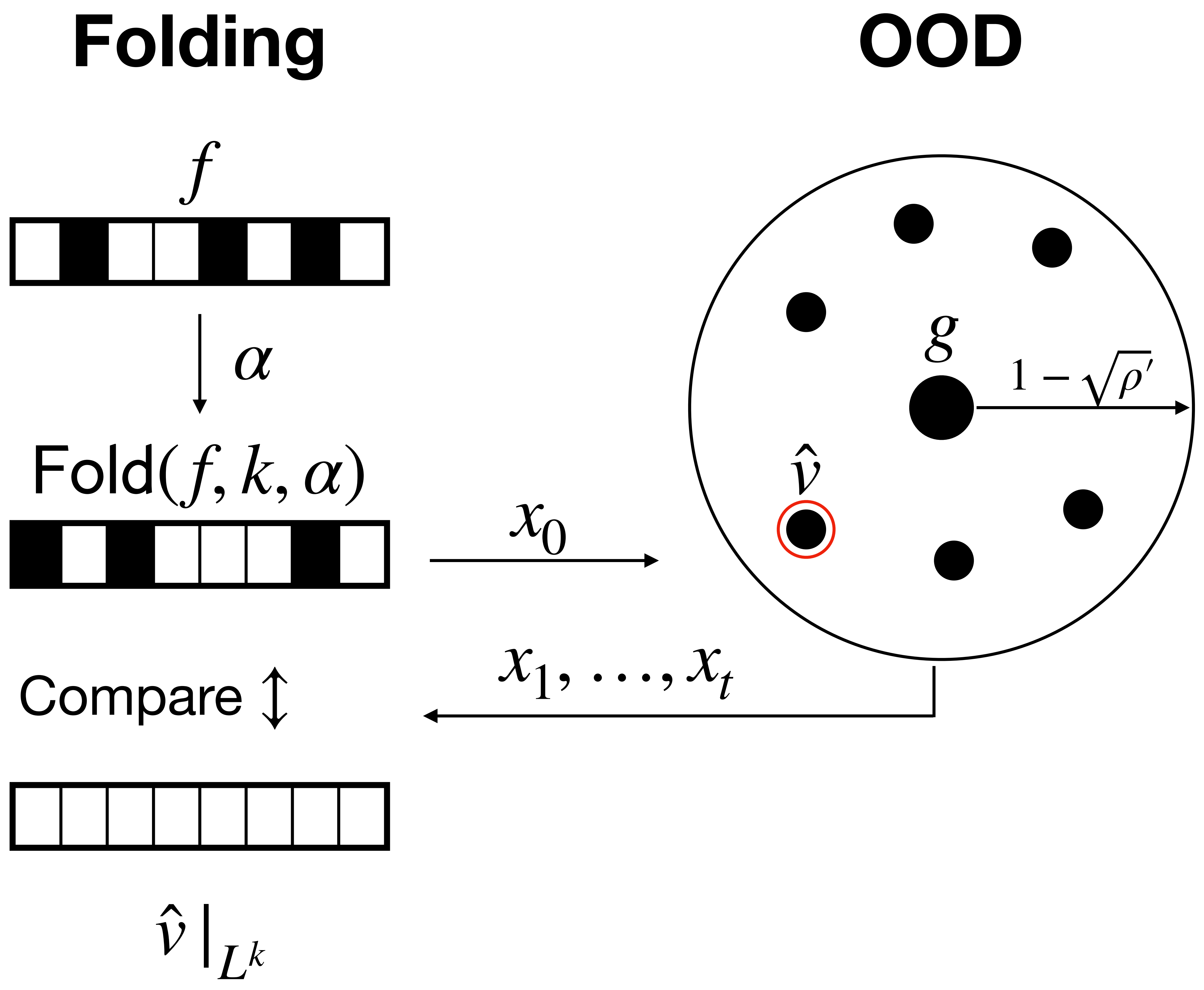
Claim: if f is δ-far from C, unless with probability $\approx (1-\delta)^t$, f' is $(1-\sqrt{\rho'})$ far from C'



$\hat{v}$ is **unique** close codeword to g with $\hat{v}(x_0) = y_0$

If at any point $\hat{v}(x_i) \neq y_i$ then, by **quotients**, f' is $(1-\sqrt{\rho'})$-far from C'

Claim: if f is δ-far from C, unless with probability $\approx (1-\delta)^t$, f' is $(1-\sqrt{\rho'})$ far from C'



Folding

OOD

f

α

Fold(f, k, α)

x_0

g

$1-\sqrt{\rho'}$

$\hat{v}$

$x_1, \ldots, x_t$

Compare ↕

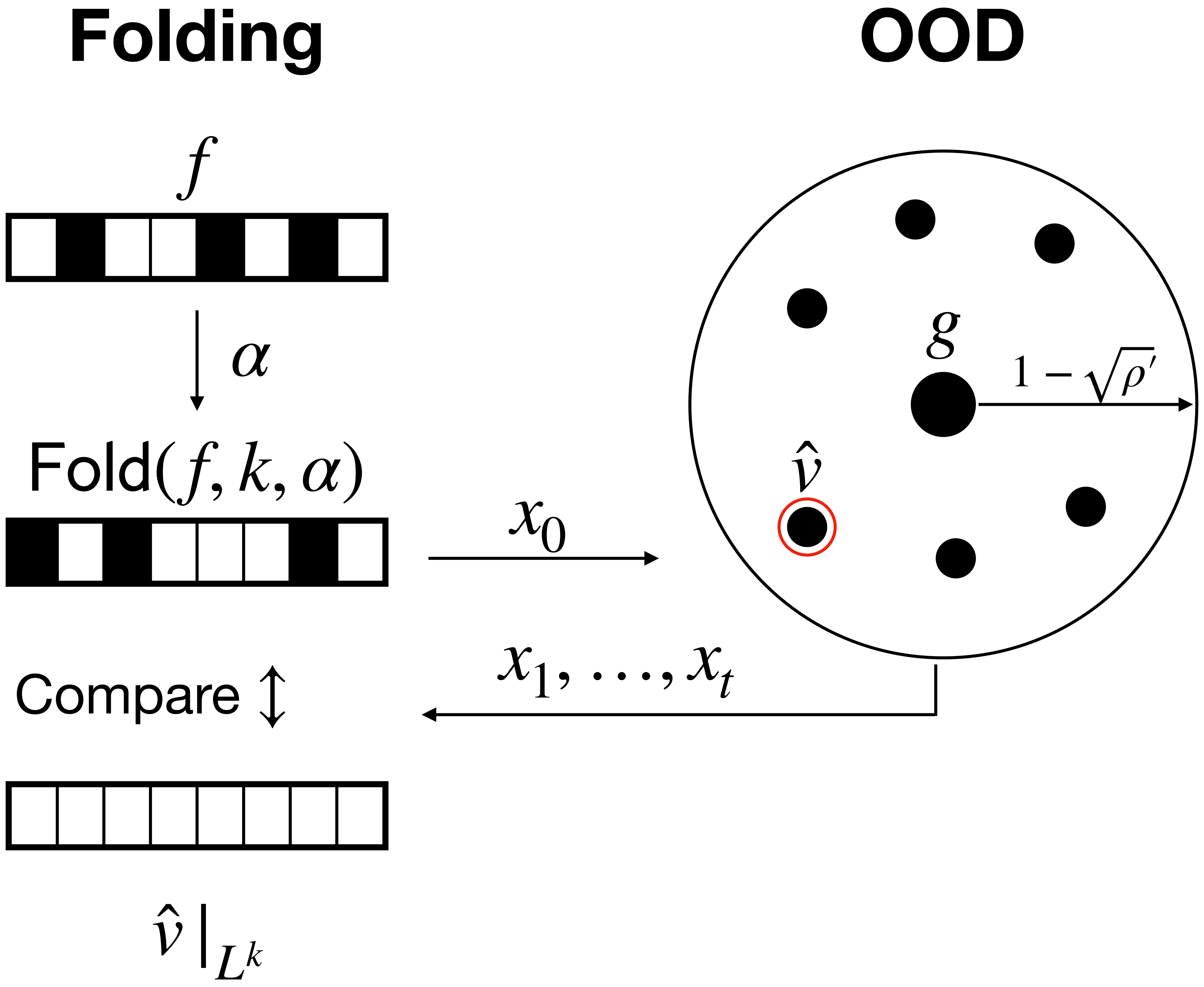
$\hat{v}\,|_{L^k}$

$\hat{v}$ is **unique** close codeword to g with $\hat{v}(x_0) = y_0$

If at any point $\hat{v}(x_i) \neq y_i$ then, by **quotients,** f' is $(1-\sqrt{\rho'})$-far from C'

Since Fold is δ-far from the code, $\Delta(\hat{v}\,|_{L^k}, \text{Fold}(f, k, \alpha)) > \delta$

Claim: if f is δ-far from C, unless with probability $\approx (1-\delta)^t$, f' is $(1-\sqrt{\rho'})$ far from C'



$\hat{v}$ is **unique** close codeword to g with $\hat{v}(x_0) = y_0$

If at any point $\hat{v}(x_i) \neq y_i$ then, by **quotients,** f' is $(1-\sqrt{\rho'})$-far from C'

Since Fold is δ-far from the code, $\Delta(\hat{v}|_{L^k}, \text{Fold}(f, k, \alpha)) > \delta$

$$\Pr\left[f' \text{ is } 1-\sqrt{\rho'} \text{ close }\right]$$
$$\leq \Pr\left[\forall i,\ \hat{v}(x_i) = y_i\right]$$
$$= \Pr\left[\forall i,\ \hat{v}(x_i) = \text{Fold}(f, k, \alpha)(x_i)\right]$$
$$\leq (1-\delta)^t$$