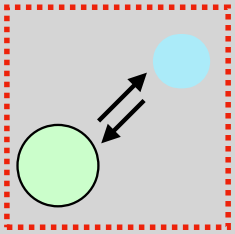
A Fiat–Shamir Transformation from Duplex Sponges

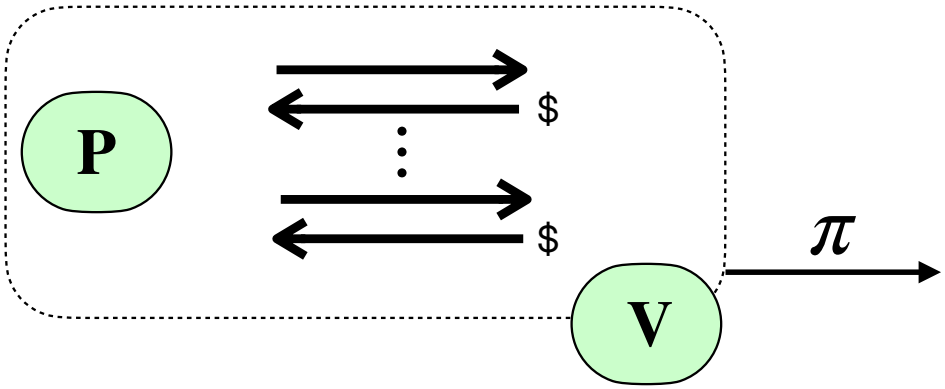
Alessandro Chiesa (EPFL), Michele Orrù (CNRS)



[FS87]

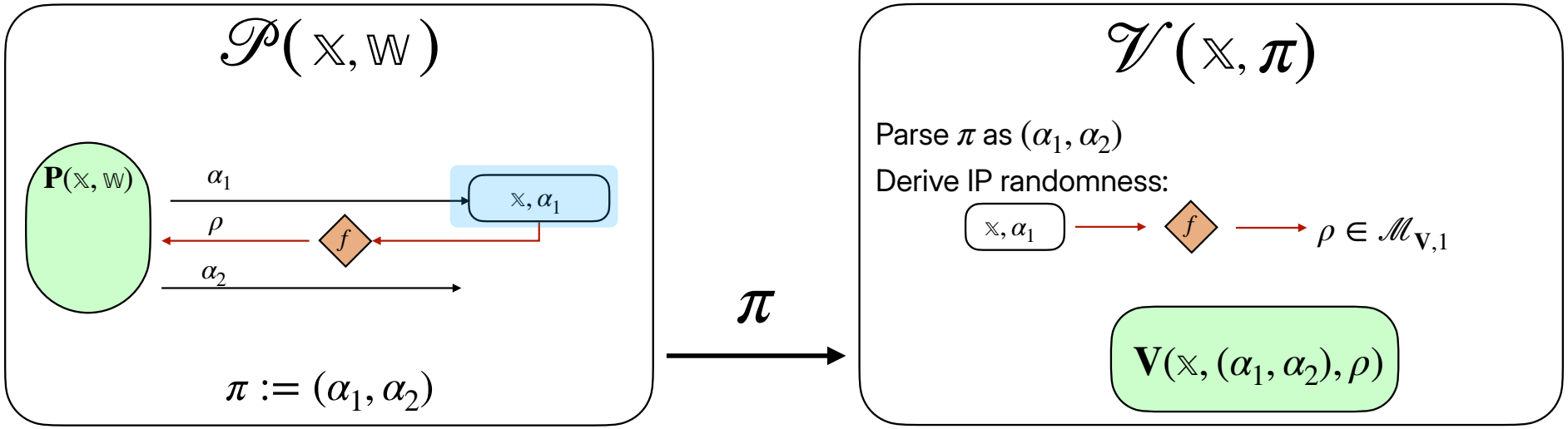
The Fiat–Shamir Transformation

A public-coin interactive proof between a prover $\mathbf{P}$ and a verifier $\mathbf{V}$.

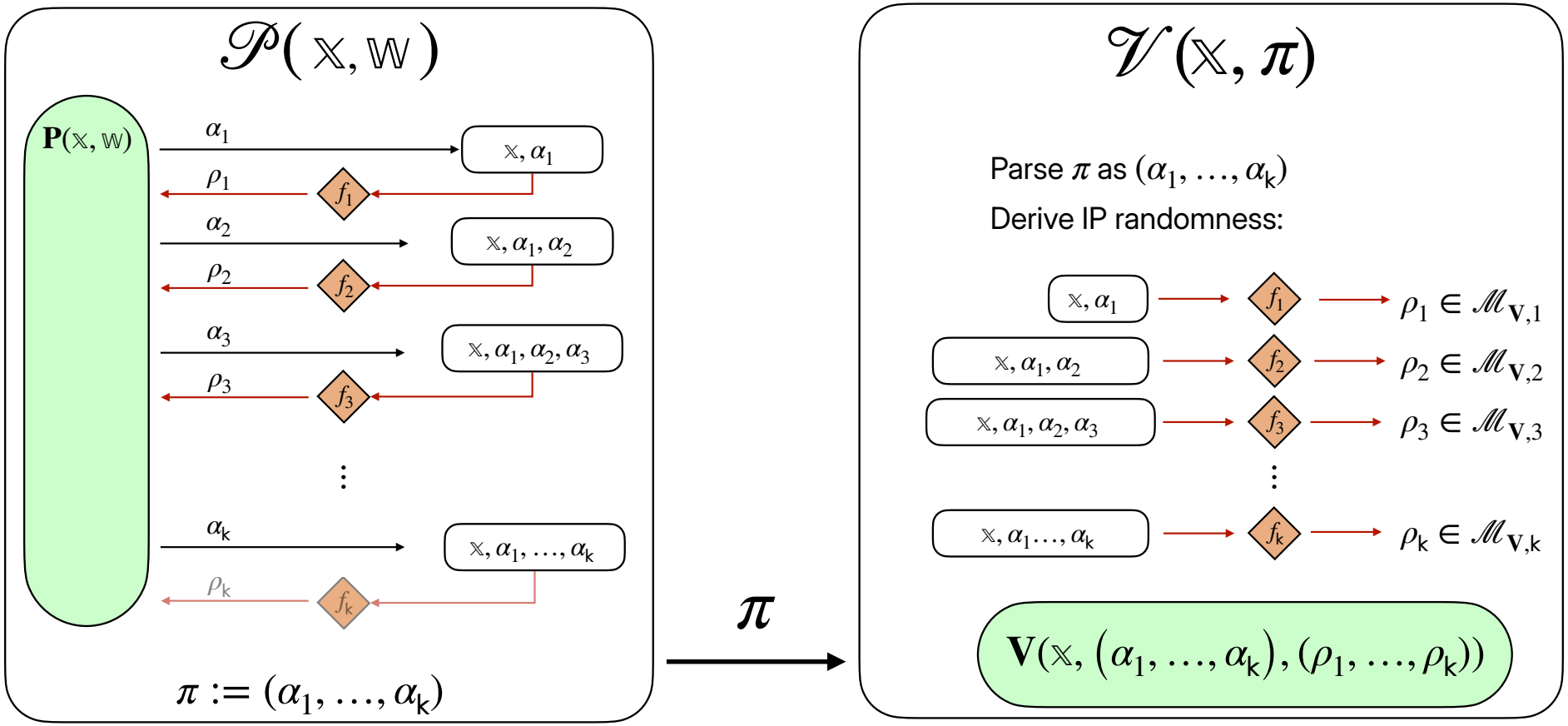


A hash function.

Fiat–Shamir is an umbrella term for multiple transformations.



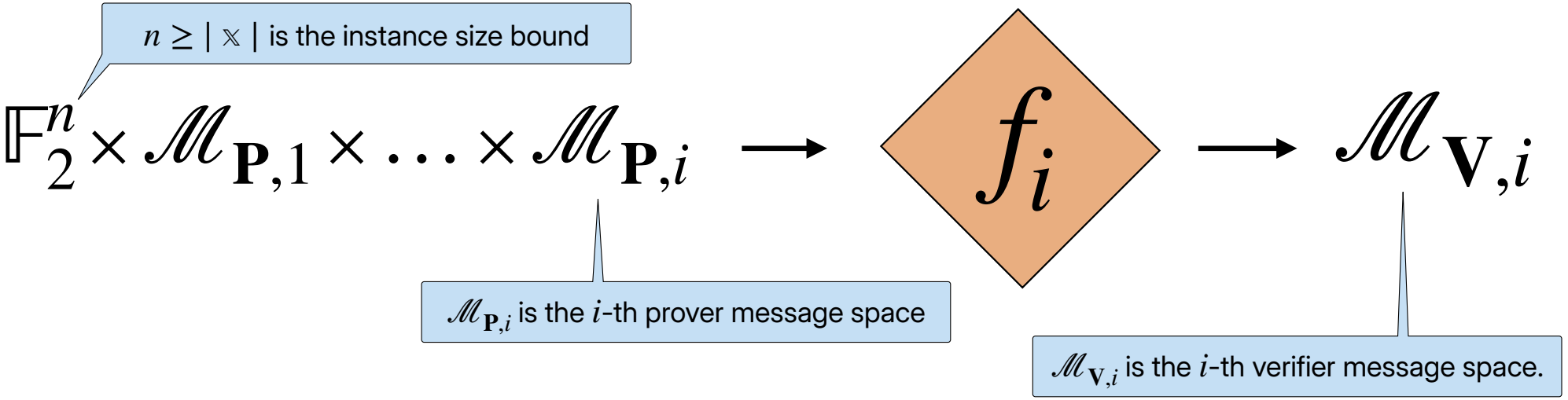
The canonical Fiat-Shamir transformation for a Σ-protocol.



The canonical Fiat–Shamir transformation for **multi-round** protocols.

What Kind of Random Oracles?

The canonical Fiat–Shamir transformation uses k **random oracles**.



Limitations

of the canonical Fiat–Shamir transformation.

Quadratic Blowup

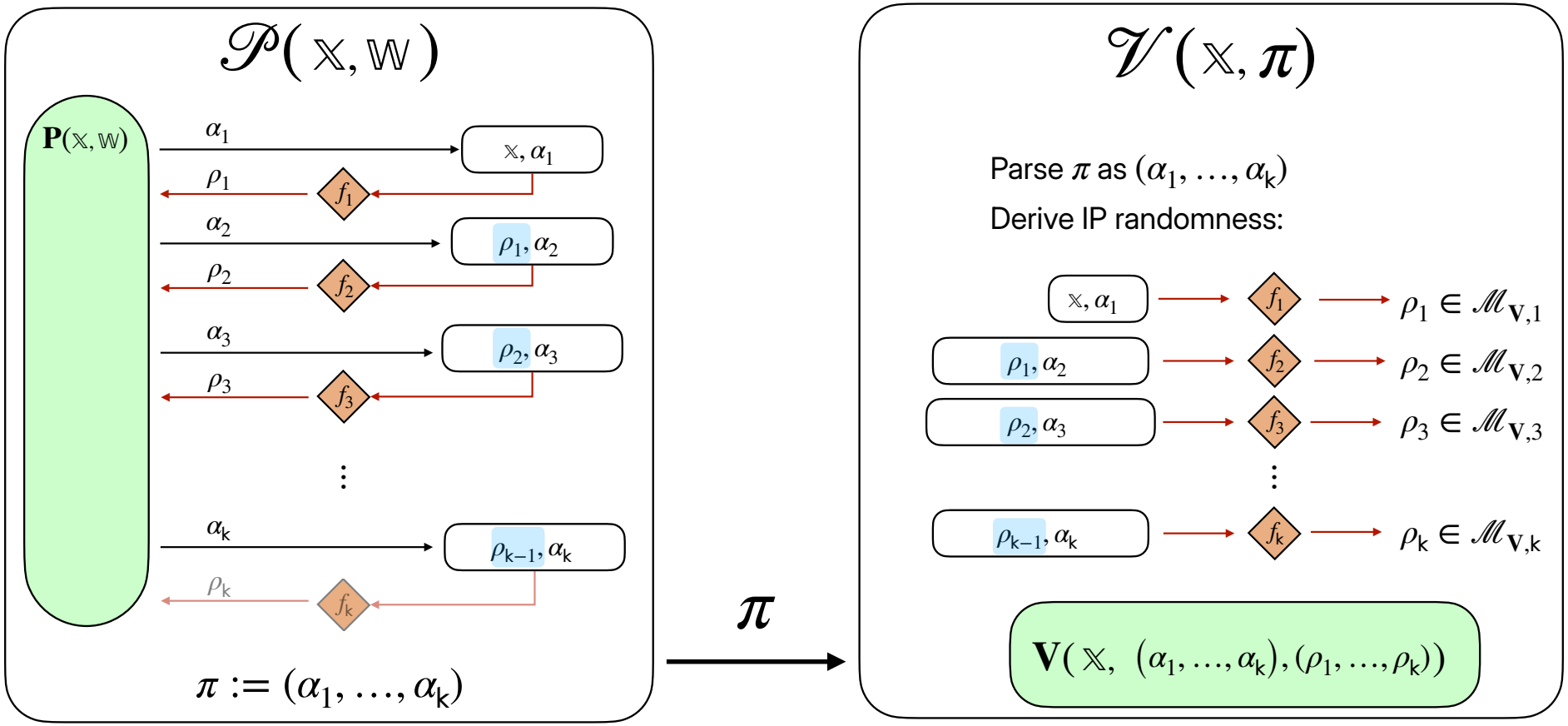
The i-th query is contained in the $(i+1)$-th query. The overall query size is

$$\mathsf{k} \cdot \mathrm{len}(\mathbb{x}) + \sum_{i \in [\mathsf{k}]} (\mathsf{k} - i + 1) \cdot \mathrm{len}(\alpha_i) = \Omega\left(\frac{\mathsf{k}(\mathsf{k}-1)}{2}\right)$$

Design Mismatch

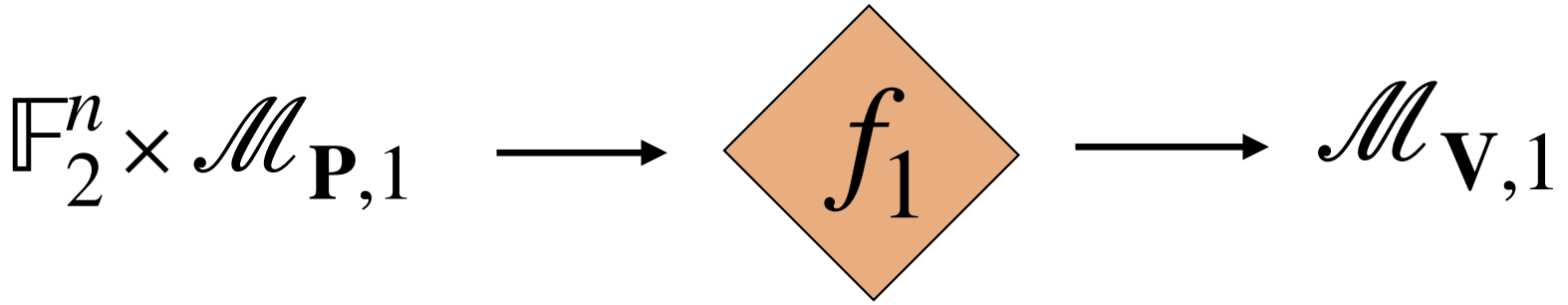
Cryptographic hash functions rely on *blocks of fixed length.*

In canonical variant: $\mathbb{F}_2^n \times \mathscr{M}_{\mathbf{P},1} \times \ldots \times \mathscr{M}_{\mathbf{P},i} \to \mathscr{M}_{\mathbf{V},i}$.
In real world: $\Sigma^m \to \Sigma^n$.

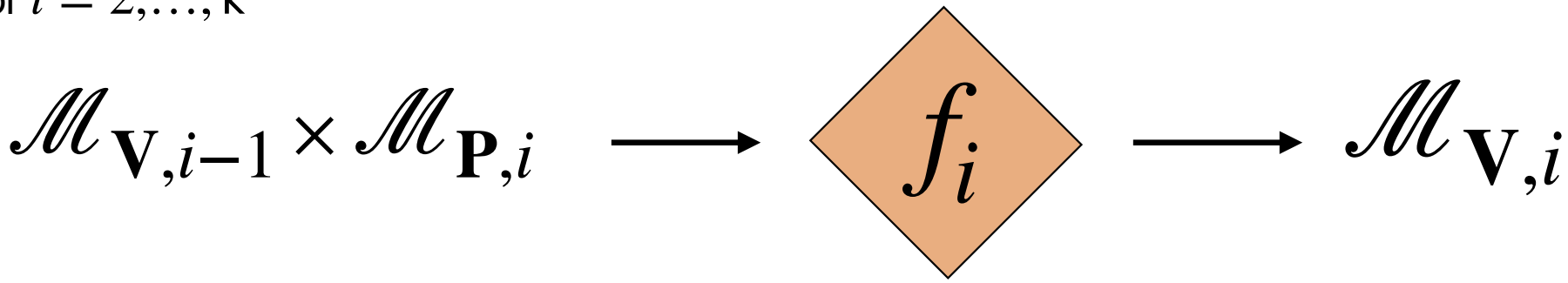


The hash-chain Fiat–Shamir transformation for multi-round protocols.

Oracles in the Hash-Chain Variant



For $i = 2,\ldots,$ k



Design Mismatch:

Oracles still depend on message spaces.

Formally-studied Fiat-Shamir tranformations are not used in practice

Fiat-Shamir

Oracle p ⇒ Real-world p

Negative results:
[Bar01, GK03, BBHMR19, KRS25]

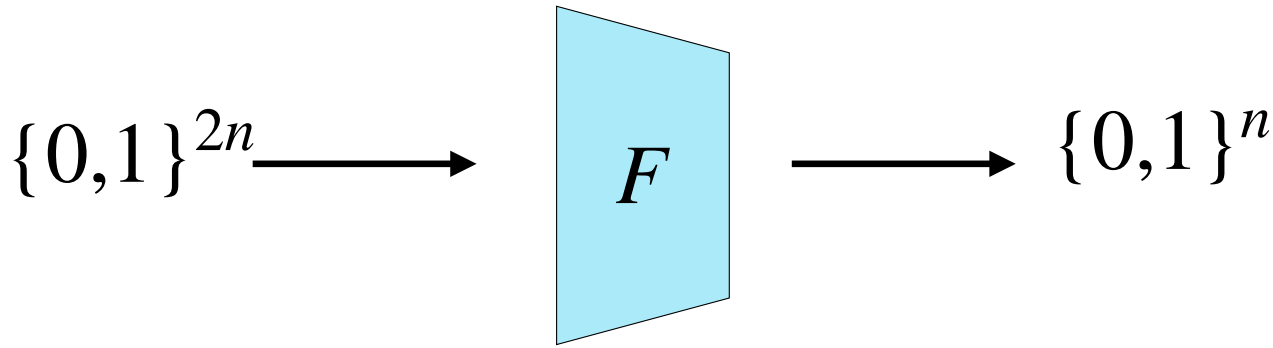
Fiat-Shamir

statistical soundness of IP
+
correlation intractability

[IKRR16, CCRR18, HL18, CCHLRRW19, PS19, LVW19, GJJM19, BFJKS19, LNPT19, CKU20, LV20, BKM20, JKKZ20, CLMQ20, LNPY20, JJ21, HLR21, CJJ21a, CJJ21b, LV22, HJKS22, GLS22, BCHKLPR22, KLV23, CGJJZ23, DJJ24,IL25, ...]

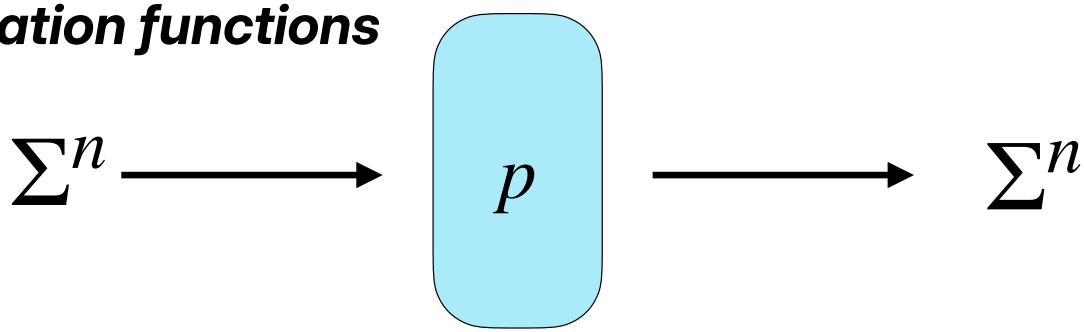
In practice:

compression functions



ex: SHA-256, BLAKE2

permutation functions

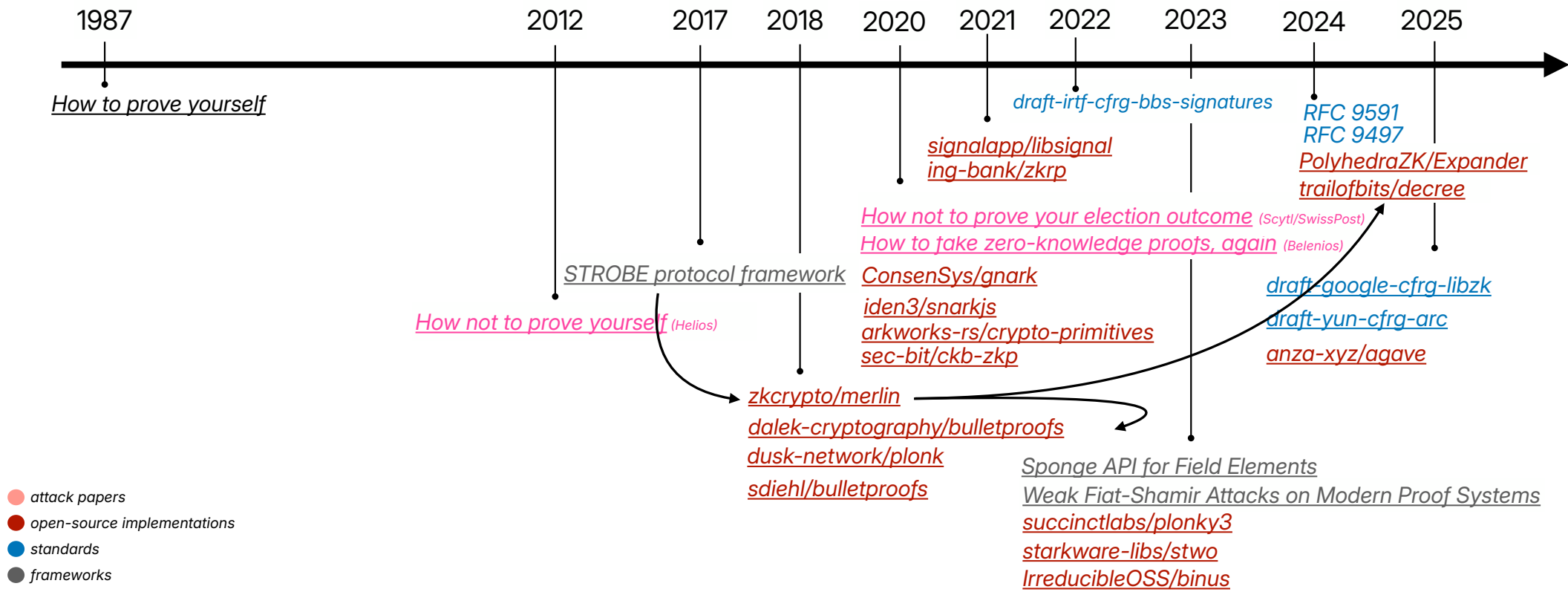


ex: Keccak-f, Poseidon

Unclear what are the security guarantees of modes of operation on top of these.

So... what do people do in practice?

Implementors have been left developing heuristics for themselves.



There's no rigorous analysis of any of these Fiat–Shamir transformations.

Our Contribution

A Fiat–Shamir transformation from ideal permutations

Formal analysis with precise security bounds.

A Fiat–Shamir Transformation From Duplex Sponges

Alessandro Chiesa
alessandro.chiesa@epfl.ch
EPFL

Michele Orrù
m@orru.net
CNRS

Abstract

The Fiat–Shamir transformation underlies numerous non-interactive arguments, with variants that differ in important ways. This paper addresses a gap between variants analyzed by theoreticians and variants implemented (and deployed) by practitioners. Specifically, theoretical analyses typically assume parties have access to random oracles with sufficiently large input and output size, while cryptographic hash functions in practice have fixed input and output sizes (pushing practitioners towards other variants).

In this paper we propose and analyze a variant of the Fiat–Shamir transformation that is based on an ideal permutation of fixed size. The transformation relies on the popular duplex sponge paradigm, and minimizes the number of calls to the permutation (given the amount of information to absorb and to squeeze). Our variant closely models deployed variants of the Fiat–Shamir transformation, and our analysis provides concrete security bounds that can be used to set security parameters in practice.

We additionally contribute `spongefish`, an open-source Rust library implementing our Fiat–Shamir transformation. The library is interoperable across multiple cryptographic frameworks, and works with any choice of permutation. The library comes equipped with Keccak and Poseidon permutations, as well as several "codecs" for re-mapping prover and verifier messages to the permutation's domain.

Keywords: Fiat–Shamir transformation; duplex sponge

</>

Open-source implementation.

spongefish Public — Edit Pins — Unwatch 2

main — 2 Branches — 1 Tag — Go to file t — Add file — Code

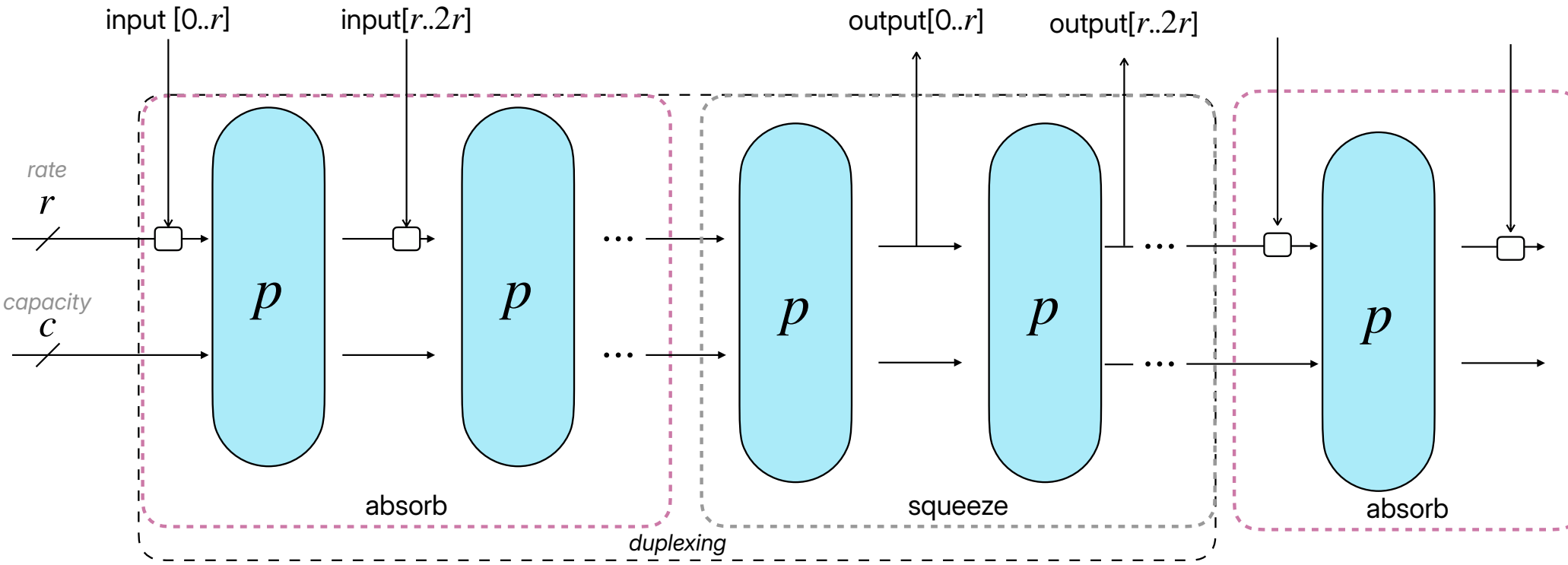
tcoratger Simplify some map patterns (#55) ✓ — 68b21ae · last week — 195 Commits

.github/workflows	Attempt update lint-fmt.yml	3 weeks ago
doc	Add latex rendering.	2 years ago
spongefish-anemoi	edits: clippy: add clippy constraints to other crates (#40)	3 weeks ago
spongefish-poseidon	edits: clippy: add clippy constraints to other crates (#40)	3 weeks ago
spongefish-pow	Add unit tests for blake3 Proof of Work (#53)	2 weeks ago
spongefish	Simplify some map patterns (#55)	last week
.gitignore	Remove Cargo.lock	11 months ago
Cargo.toml	Remove unused anyhow dep (#56)	last week
LICENSE	Create LICENSE	2 years ago
README.md	Refactor using naming conventions of the academic paper.	3 weeks ago

Duplexing the sponge

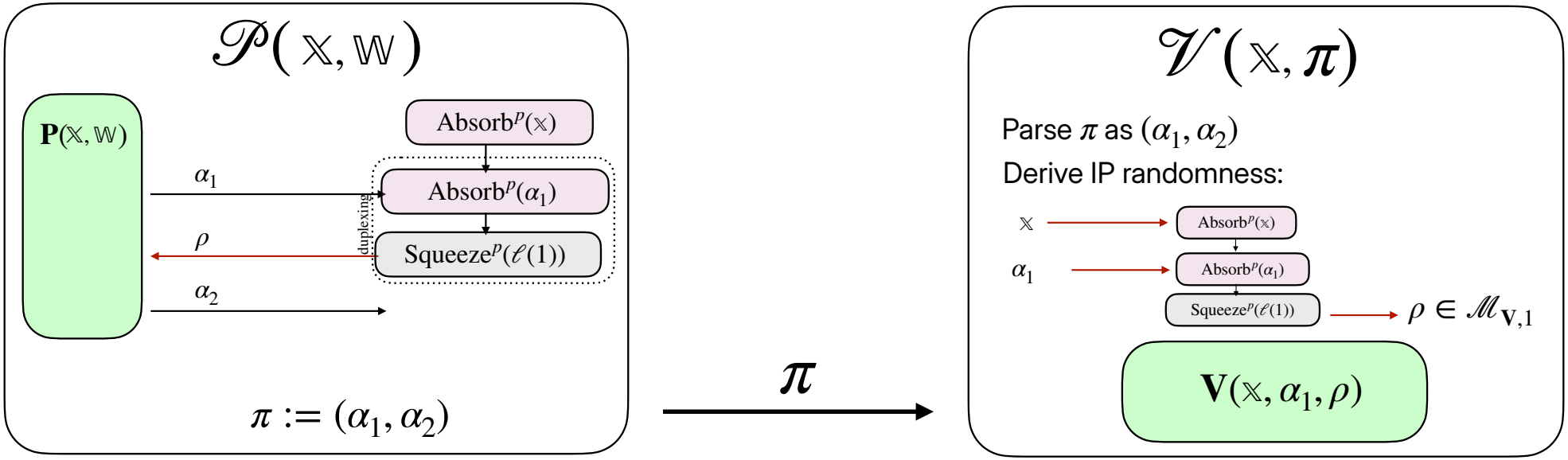
[BDPV07]
[BDPV11]

Let $p : \Sigma^{r+c} \to \Sigma^{r+c}$ be a permutation function.

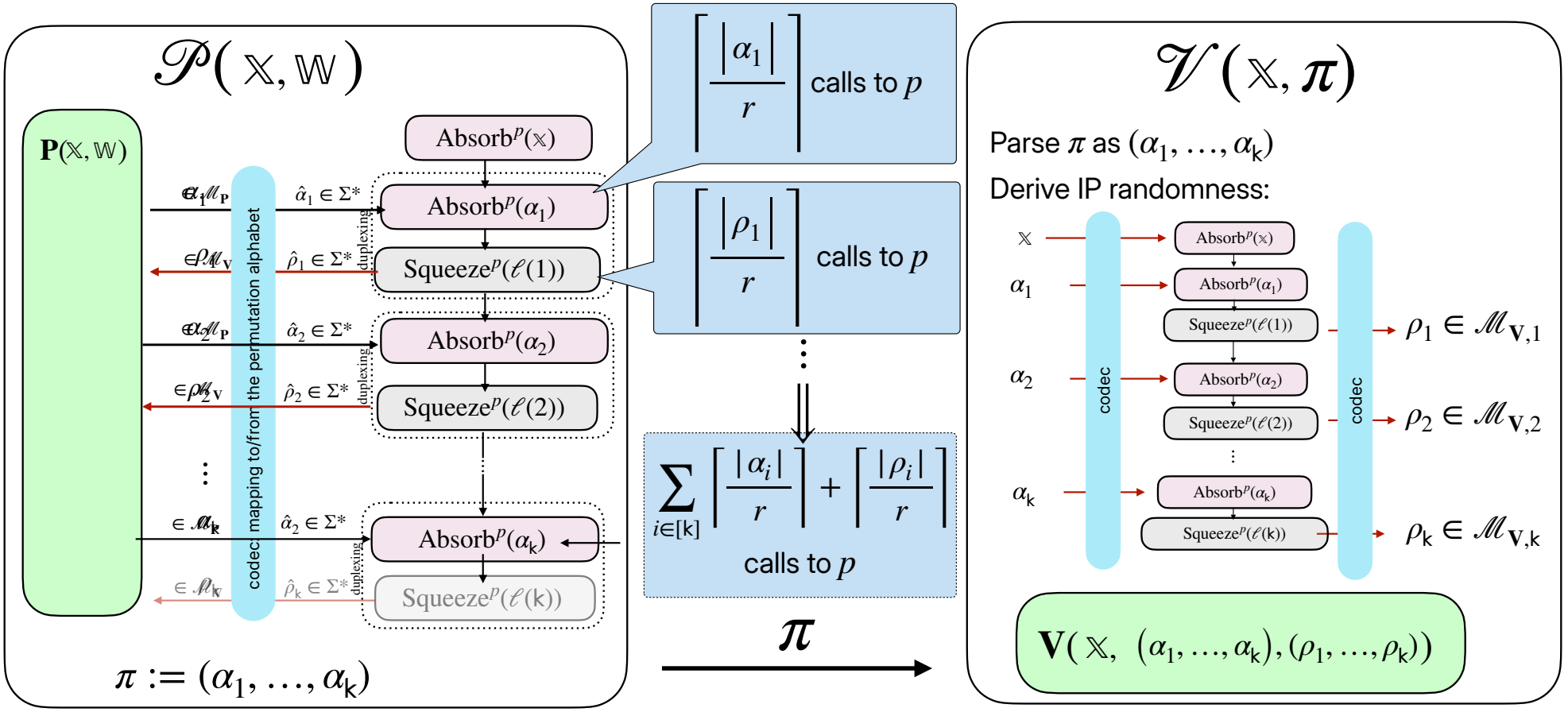


Absorb prover messages.
Squeeze verifier messages.

DSFS[IP] for Σ-protocols



DSFS[IP] for multi-round protocols



Proving security

In an oracle model, where $p : \Sigma^{r+c} \to \Sigma^{r+c}$ is an ideal permutation.

(knowledge) soundness

[malicious prover]

$$\mathbb{x}, \pi \implies \mathbb{W}$$

$$\varepsilon_{\text{srsnd}} + O\left(\frac{t^2}{|\Sigma|^c}\right)$$

zero-knowledge

[malicious verifier]

$$\mathcal{P}(\mathbb{x}, \mathbb{W}) \approx \mathcal{S}(\mathbb{x})$$

$$\varepsilon_{\text{hvzk}} + O\left(\frac{t}{|\Sigma|^{\min(c,\delta)}}\right)$$

δ is the **privacy parameter**

where t is the number of queries by adversary to p.

Soundness

$$\mathbb{x}, \pi \implies \mathbb{W}$$

(Knowledge) Soundness

$$p \xleftarrow{\text{tr}} \mathscr{D}_{\mathfrak{S}}$$
$$(\mathbb{x}, \pi) \leftarrow \tilde{\mathscr{P}}^{p,p^{-1}} \longrightarrow \tilde{\mathbf{P}}^{\mathrm{SR}}$$

- Soundness:

$$\varepsilon_{\mathrm{snd}} = \Pr\left[\pi \text{ valid and } \mathbb{x} \notin \mathscr{L}\right]$$

- Straightline Knowledge Soundness:

$$\kappa = \Pr\left[\mathbb{w} \leftarrow \mathscr{E}(\mathbb{x}, \pi, \mathsf{tr}) \text{ such that } \pi \text{ valid and } (\mathbb{x}, \mathbb{w}) \notin \mathscr{R}\right]$$

- Rewinding Knowledge Soundness:

$$\kappa_{\mathrm{rw}} = \Pr\left[\mathbb{w} \leftarrow \mathscr{E}(\mathbb{x}, \pi, \mathsf{tr}, \tilde{\mathscr{P}}) \text{ such that } \pi \text{ valid and } (\mathbb{x}, \mathbb{w}) \notin \mathscr{R}\right]$$

$$\varepsilon \leq \kappa \leq \kappa_{\mathrm{rw}}$$

- State-Restoration Soundness

$$\varepsilon^{(\mathrm{sr})} \leq \varepsilon$$

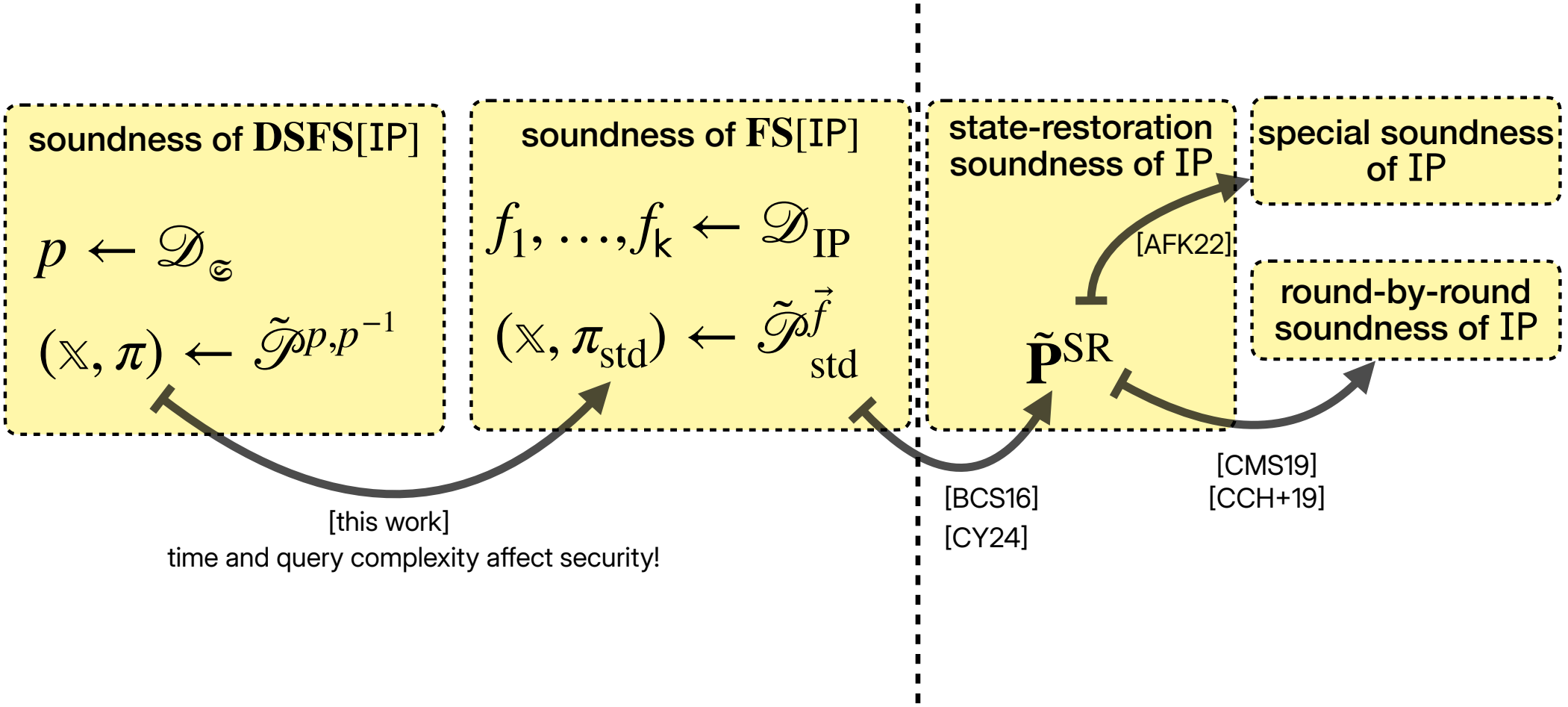
- State-Restoration Knowledge Soundness

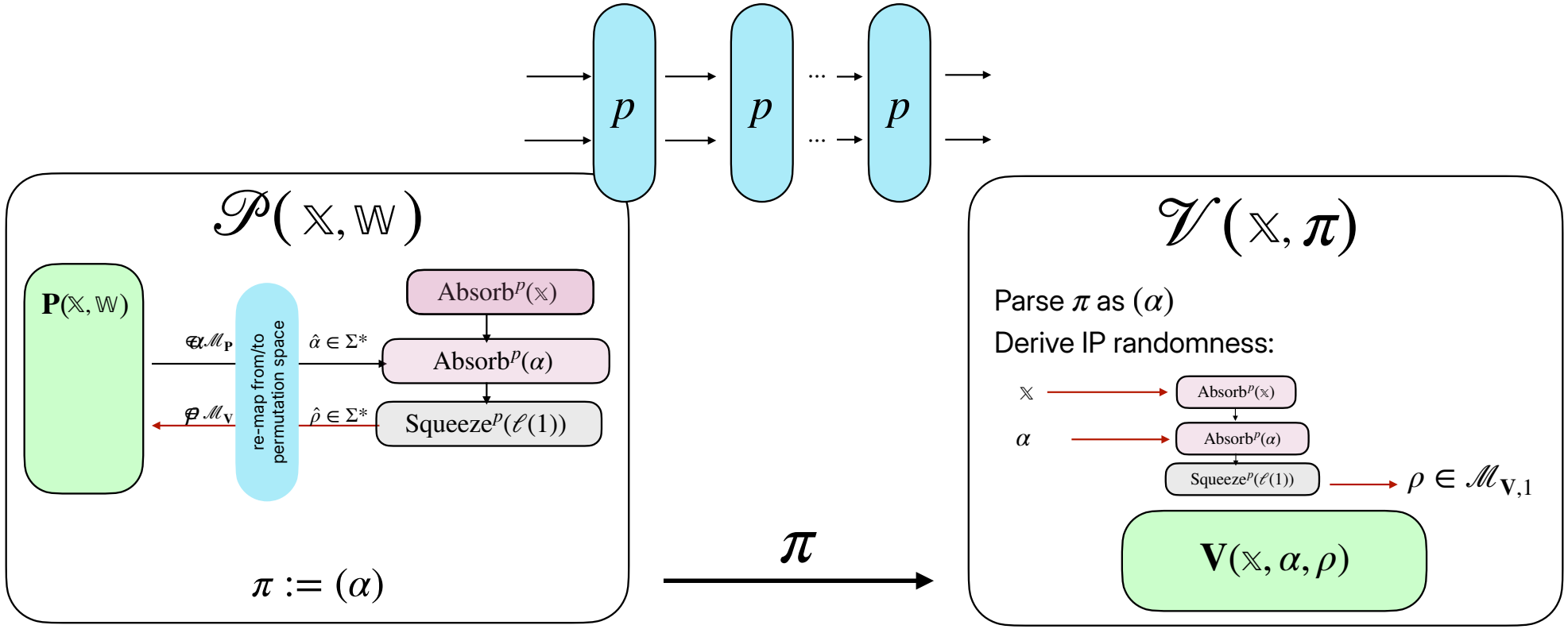
$$\kappa^{(\mathrm{sr})} \leq \kappa$$

- State-Restoration Rewinding Knowledge Soundness

$$\kappa_{\mathrm{rw}}^{(\mathrm{sr})} \leq \kappa_{\mathrm{rw}}$$

Proving soundness: strategy




p
p
p
$\mathscr{P}(\mathbb{X}, \mathbb{W})$
$\mathbf{P}(\mathbb{X}, \mathbb{W})$
$\alpha \in \mathscr{M}_{\mathbf{P}}$
$\hat{\alpha} \in \Sigma^*$
re-map from/to permutation space
$\rho \in \mathscr{M}_{\mathbf{V}}$
$\hat{\rho} \in \Sigma^*$
Absorb$^p(\mathbb{X})$
Absorb$^p(\alpha)$
Squeeze$^p(\ell(1))$
$\pi := (\alpha)$
π
$\mathscr{V}(\mathbb{X}, \pi)$
Parse π as (α)
Derive IP randomness:
$\mathbb{X}$
α
Absorb$^p(\mathbb{X})$
Absorb$^p(\alpha)$
Squeeze$^p(\ell(1))$
$\rho \in \mathscr{M}_{\mathbf{V},1}$
$\mathbf{V}(\mathbb{X}, \alpha, \rho)$

Proving soundness: main lemma

We seek a reduction $\mathscr{P}_{\text{std}}$ such that:

$p \leftarrow \mathscr{D}_{\mathfrak{S}}$ $\qquad$ $f_1, \ldots, f_k \leftarrow \mathscr{D}_{\text{IP}}$

$(\mathbb{x}, \pi) \xleftarrow{\text{tr}} \tilde{\mathscr{P}}^{p, p^{-1}}$ $\qquad$ $(\mathbb{x}, \pi_{\text{std}}) \xleftarrow{\text{tr}_{\text{std}}} \tilde{\mathscr{P}}^{\vec{f}}_{\text{std}}$

Proving soundness: main lemma

Find $\mathrm{ReduceP}$ and $\mathrm{ReduceT}$ such that:

$p \leftarrow \mathscr{D}_{\mathfrak{S}}$		$f_1, \ldots, f_k \leftarrow \mathscr{D}_{\mathrm{IP}}$
$(\mathbb{x}, \pi) \overset{\mathsf{tr}}{\leftarrow} \tilde{\mathscr{P}}^{p, p^{-1}}$	$\leftrightarrow$	$(\mathbb{x}, \pi_{\mathrm{std}}) \overset{\mathsf{tr}_{\mathrm{std}}}{\leftarrow} \mathrm{ReduceP}(\tilde{\mathscr{P}})^{\vec{f}}$
$b \leftarrow \mathscr{V}(\mathbb{x}, \pi)$	$\approx$	$b_{\mathrm{std}} \leftarrow \mathscr{V}_{\mathrm{std}}(\mathbb{x}, \pi_{\mathrm{std}})$
$\mathsf{tr}_{\mathrm{std}} \leftarrow \mathrm{ReduceT}(\mathsf{tr})$		$\mathsf{tr}_{\mathrm{std}}$

Additive error: $\eta_\star = O\left(\frac{t^2}{|\Sigma|^c}\right)$.

Isn't this already known?

After all, we already had the sponge construction, and know how to build XOFs.

Indistinguishability

Let $\mathbf{f} \leftarrow \mathscr{D}_{\mathrm{IP}}$ *and* $p \leftarrow \mathscr{D}_{\mathscr{C}}$. *A construction* $\mathscr{C}$ *is* ***indistinguishable*** *if:*

$$\left\{ \begin{array}{c} b \leftarrow \mathscr{A}^{\mathbf{f}} \\ b \end{array} \right\} \approx \left\{ \begin{array}{c} b \leftarrow \mathscr{A}^{\mathscr{C}^p} \\ b \end{array} \right\}$$

The adversary $\mathscr{A}^{p,p^{-1}}$ *can query the permutation function.*

[MRH03]

The Indifferentiability Framework

Let $\mathbf{f} \leftarrow \mathscr{D}_{\mathrm{IP}}$ *and* $p \leftarrow \mathscr{D}_{\mathscr{C}}$. *A construction* $\mathscr{C}$ *is* ***indifferentiable*** $\exists\, \mathscr{S}$:

$$\left\{ \begin{array}{c} b \leftarrow \mathscr{A}^{\mathbf{f}, \mathscr{S}^{\mathbf{f}}} \\ b \end{array} \right\} \approx \left\{ \begin{array}{c} b \leftarrow \mathscr{A}^{\mathscr{C}^{p}, p} \\ b \end{array} \right\}$$

Soundness can be proven with this property

The extractor $\mathscr{E}$ *requires the random oracle trace.*

$$\varepsilon_{\mathrm{ksnd}} = \Pr\left[\mathbb{w} \leftarrow \mathscr{E}(\mathbb{x}, \pi, \mathsf{tr}) \text{ such that } \pi \text{ valid and } (\mathbb{x}, \mathbb{w}) \notin \mathscr{R}\right]$$

Our notion: double indifferentiability

Let $\mathbf{f} \leftarrow \mathscr{D}_{\mathrm{IP}}$ *and* $p \leftarrow \mathscr{D}_{\mathscr{C}}$*. A construction* $\mathscr{C}$ *is* ***doubly indifferentiable*** *if* $\exists\, \mathscr{S}, \mathscr{T}$*:*

$$\left\{ \begin{array}{c} \mathsf{out} \xleftarrow{\mathsf{tr}} \mathscr{A}^{\mathbf{f},\mathscr{S}^{\mathbf{f}}} \\ \mathsf{tr}, \mathsf{out} \end{array} \right\} \approx \left\{ \begin{array}{c} \mathsf{out} \xleftarrow{\mathsf{tr}_\diamond} \mathscr{A}^{\mathscr{C}^p,p} \\ \mathscr{T}(\mathsf{tr}_\diamond), \mathsf{out} \end{array} \right\}$$

Our notion: double indifferentiability

Let $\mathbf{f} \leftarrow \mathscr{D}_{\mathrm{IP}}$ *and* $p \leftarrow \mathscr{D}_{\mathscr{C}}$. *A construction* $\mathscr{C}$ *is* ***doubly indifferentiable*** *if* $\exists \mathscr{S}, \mathscr{T}$:

$$\left\{ \begin{array}{c} \mathsf{out} \xleftarrow{\mathsf{tr}} \mathscr{A}^{\mathbf{f}, \mathscr{S}^{\mathbf{f}}} \\ \mathsf{tr}, \mathsf{out} \end{array} \right\} \approx \left\{ \begin{array}{c} \mathsf{out} \xleftarrow{\mathsf{tr}_\diamond} \mathscr{A}^{\mathscr{C}^p, p} \\ \mathscr{T}(\mathsf{tr}_\diamond), \mathsf{out} \end{array} \right\}$$

$$\kappa_{\mathbf{DSFS}[\mathrm{IP}]} = \Pr\left[b = 1 \ \wedge \ (\mathbb{x}, \pi) \ \middle| \ \begin{array}{c} p \leftarrow \mathscr{D}_{\mathfrak{G}} \\ (\mathbb{X}, \pi) \xleftarrow{\mathsf{tr}_\diamond} \mathscr{A}^p \\ \mathbb{W} \leftarrow \mathscr{E}_{\mathsf{DSFS}}(\mathbb{X}, \pi, \mathsf{tr}_\diamond) \\ b \leftarrow \mathscr{V}^p(\mathbb{X}, \pi) \end{array} \right] = \Pr\left[b = 1 \ \wedge \ (\mathbb{x}, \pi) \ \middle| \ \begin{array}{c} p \leftarrow \mathscr{D}_{\mathfrak{G}} \\ (\mathbb{X}, \pi) \xleftarrow{\mathsf{tr}_\diamond} \mathscr{A}^p \\ \mathsf{tr} \leftarrow \mathscr{T}(\mathsf{tr}_\diamond) \\ \mathbb{W} \leftarrow \mathscr{E}_{\mathsf{IP}}(\mathbb{X}, \pi, \mathsf{tr}_\diamond) \\ b \leftarrow \mathscr{V}^p(\mathbb{X}, \pi) \end{array} \right]$$

Our notion: double indifferentiability

Let $\mathbf{f} \leftarrow \mathscr{D}_{\mathrm{IP}}$ *and* $p \leftarrow \mathscr{D}_{\mathfrak{C}}$. *A construction* $\mathscr{C}$ *is* ***doubly indifferentiable*** *if* $\exists\, \mathscr{S}, \mathscr{T}$:

$$\left\{ \begin{array}{c} \mathsf{out} \xleftarrow{\mathsf{tr}} \mathscr{A}^{\mathbf{f}, \mathscr{S}^{\mathbf{f}}} \\ \mathsf{tr}, \mathsf{out} \end{array} \right\} \approx \left\{ \begin{array}{c} \mathsf{out} \xleftarrow{\mathsf{tr}_\diamond} \mathscr{A}^{\mathscr{C}^p, p} \\ \mathscr{T}(\mathsf{tr}_\diamond), \mathsf{out} \end{array} \right\}$$

$$\kappa_{\mathbf{DSFS}[\mathrm{IP}]} = \Pr\left[b = 1 \ \wedge\ (\mathbb{x}, \pi) \;\middle|\; \begin{array}{c} p \leftarrow \mathscr{D}_{\mathfrak{G}} \\ (\mathbb{x}, \pi) \leftarrow \mathscr{A}^p \\ \mathsf{tr} \leftarrow \mathscr{T}(\mathsf{tr}_\diamond) \\ \mathbb{W} \leftarrow \mathscr{E}_{\mathrm{IP}}(\mathbb{x}, \pi, \mathsf{tr}_\diamond) \\ b \leftarrow \mathscr{V}^p(\mathbb{x}, \pi) \end{array} \right] \leq \Pr\left[b = 1 \ \wedge\ (\mathbb{x}, \pi) \;\middle|\; \begin{array}{c} \mathbf{f} \leftarrow \mathscr{D}_{\mathrm{IP}} \\ (\mathbb{x}, \pi) \xleftarrow{\mathsf{tr}} \mathscr{A}^{\mathscr{S}^{\mathbf{f}}} \\ \mathbb{W} \leftarrow \mathscr{E}_{\mathrm{IP}}(\mathbb{x}, \pi, \mathsf{tr}) \\ b \leftarrow \mathscr{V}^{\mathscr{S}^{\mathbf{f}}}(\mathbb{x}, \pi) \end{array} \right] + \mu_\star$$

Our notion: double indifferentiability

Let $\mathbf{f} \leftarrow \mathscr{D}_{\mathrm{IP}}$ *and* $p \leftarrow \mathscr{D}_{\mathscr{C}}$. *A construction* $\mathscr{C}$ *is* ***doubly indifferentiable*** *if* $\exists\, \mathscr{S}, \mathscr{T}$ *:*

$$\left\{ \begin{array}{c} \mathsf{out} \xleftarrow{\mathsf{tr}} \mathscr{A}^{\mathbf{f}, \mathscr{S}^{\mathbf{f}}} \\ \mathsf{tr}, \mathsf{out} \end{array} \right\} \approx \left\{ \begin{array}{c} \mathsf{out} \xleftarrow{\mathsf{tr}_\diamond} \mathscr{A}^{\mathscr{C}^p, p} \\ \mathscr{T}(\mathsf{tr}_\diamond), \mathsf{out} \end{array} \right\}$$

$$\kappa_{\mathbf{DSFS}[\mathrm{IP}]} \leq \Pr\left[b = 1 \wedge (\mathbb{x}, \pi) \;\middle|\; \begin{array}{c} \mathbf{f} \leftarrow \mathscr{D}_{\mathrm{IP}} \\ (\mathbb{x}, \pi) \xleftarrow{\mathsf{tr}} \mathscr{A}^{\mathscr{S}^{\mathbf{f}}} \\ \mathbb{W} \leftarrow \mathscr{E}_{\mathrm{IP}}(\mathbb{x}, \pi, \mathsf{tr}) \\ b \leftarrow \mathscr{V}^{\mathscr{S}^{\mathbf{f}}}(\mathbb{x}, \pi) \end{array} \right] + \mu_\star = \kappa_{\mathrm{snd}} + \mu_\star$$

Zero-Knowledge

$$\mathscr{P}(\mathbb{x}, \mathbb{w}) \approx \mathscr{S}(\mathbb{x})$$

Zero-Knowledge

In the explicitly-programmable random oracle model.

$$p \leftarrow \mathscr{D}_{\mathfrak{S}}$$

$$(\mathbb{x}, \mathbb{w}) \leftarrow \mathscr{A}^{p,p^{-1}}$$

Real-world prover

$$\pi \leftarrow \mathscr{P}^{p}(\mathbb{x}, \mathbb{w})$$

$$b_0 \leftarrow \mathscr{A}^{p,p^{-1}}(\pi)$$

$$\approx$$

Ideal-world simulator

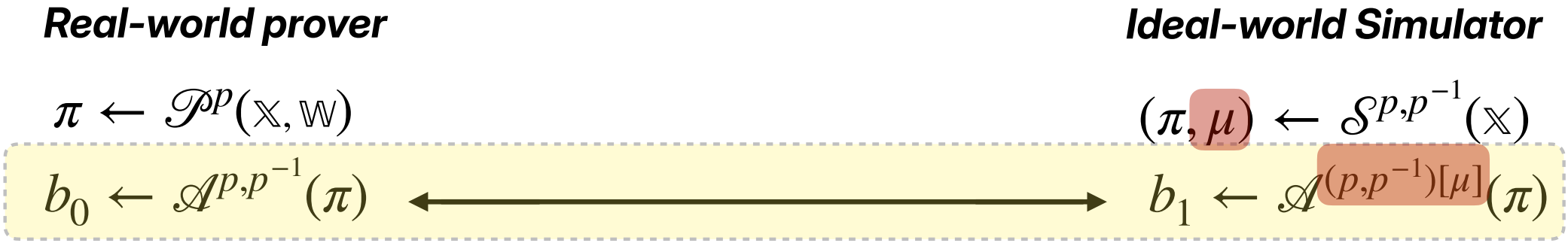
$$(\pi, \mu) \leftarrow \mathcal{S}^{p,p^{-1}}(\mathbb{x})$$

$$b_1 \leftarrow \mathscr{A}^{(p,p^{-1})[\mu]}(\pi)$$

Proving zero-knowledge: what about indifferentiability?

Indifferentiability is insufficient.

1. The simulator must reprogram the random oracle.

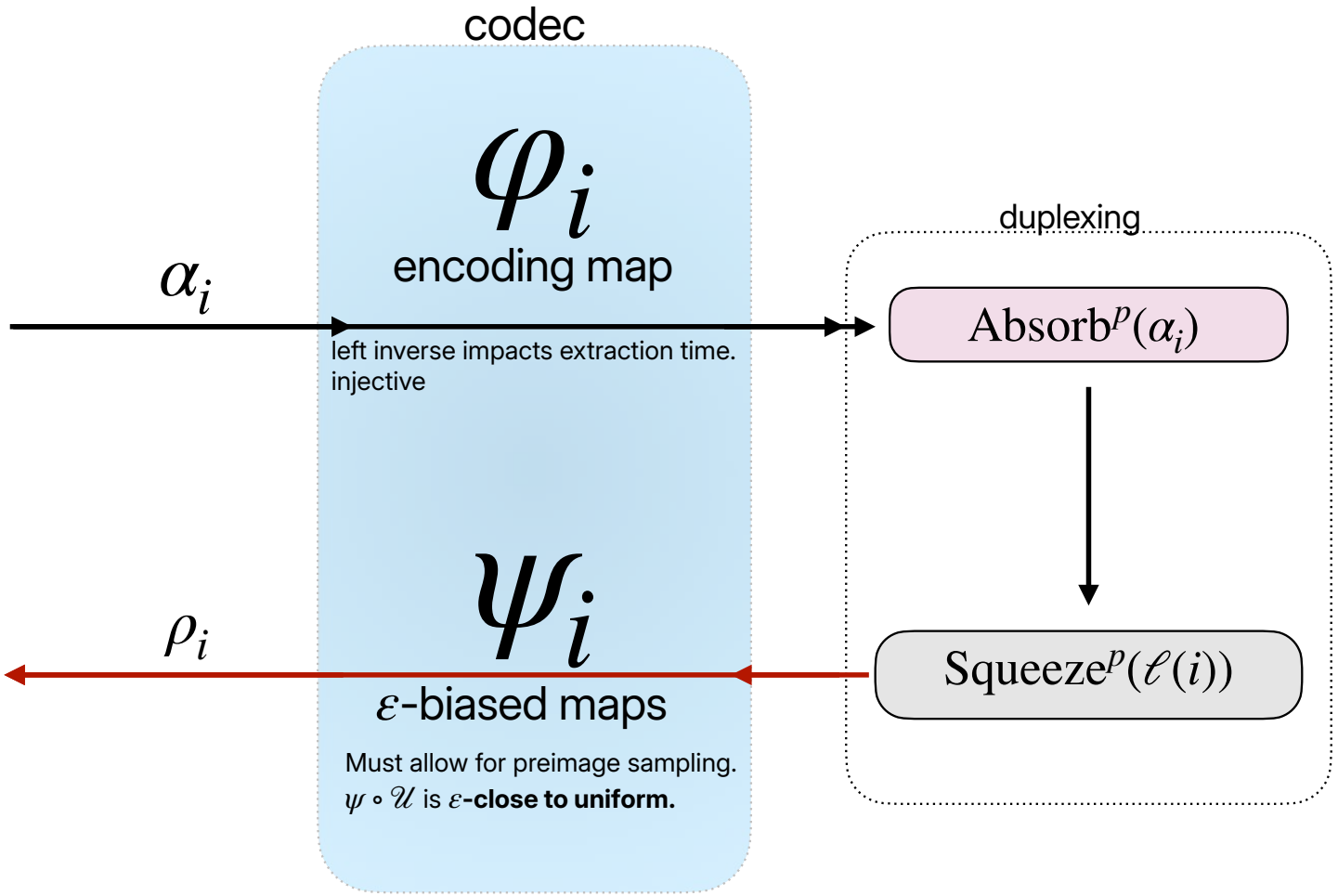


2. We need to translate the positions to be re-programmed

Converting from/to the permutation domain

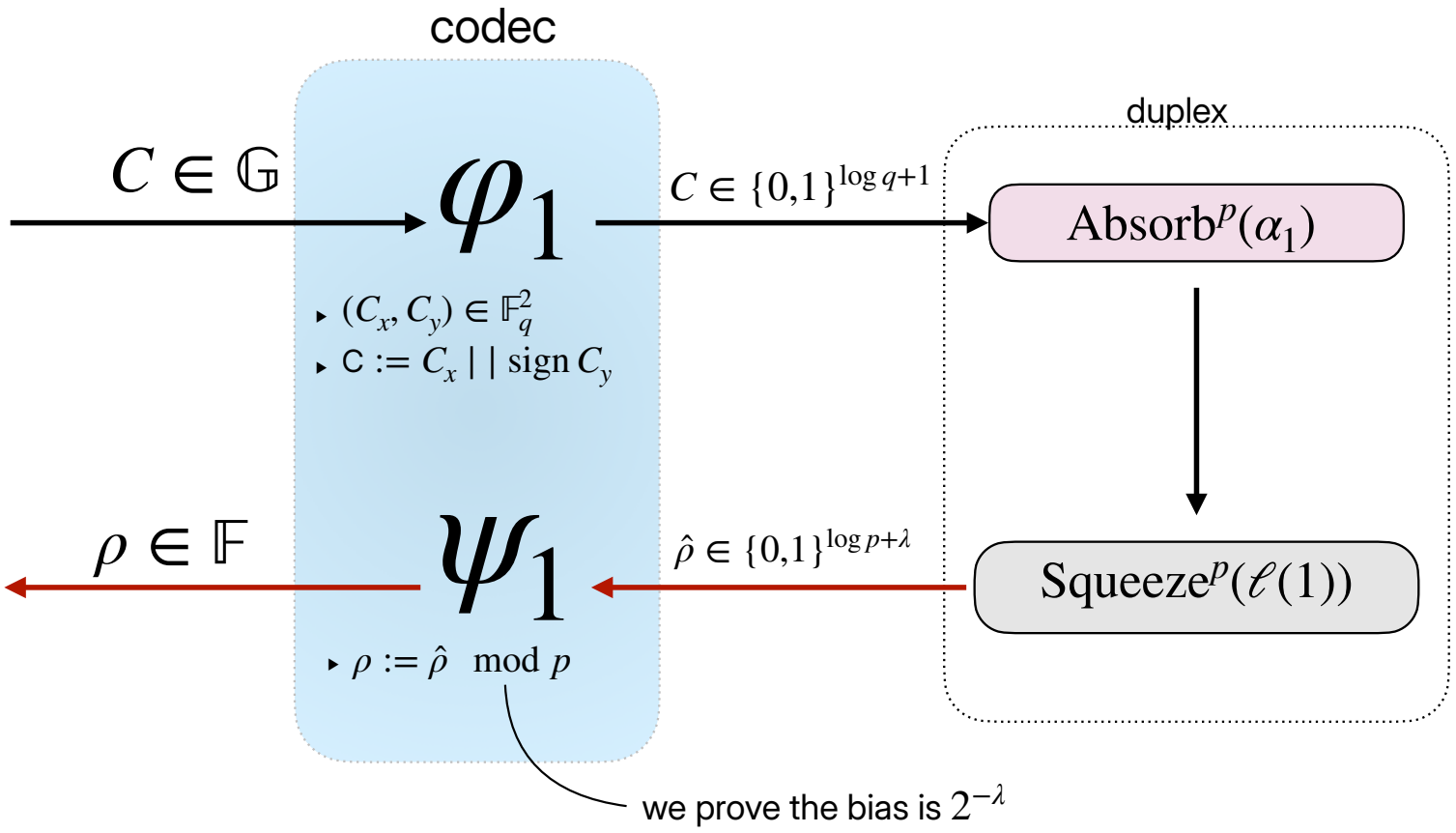
Codecs

Map from/to permutation domain



Example:

A codec for Schnorr proofs



Actual security bounds

Taking into account codecs, soundness incurs into an additive loss.

knowledge soundness:

$$\varepsilon_{\text{srsnd}} + O\left(\frac{t^2}{|\Sigma|^c}\right) + O(t) \cdot \max_{i \in [k]} \varepsilon_{\text{cdc},i}$$

(depends on the adversarial queries)

zero knowledge:

$$\varepsilon_{\text{hvzk}} + O\left(\frac{t}{|\Sigma|^{\min(c,\delta)}}\right) + \sum_{i \in [k]} \varepsilon_{\text{cdc},i}$$

(only needed for the programmed queries)

Fiat-Shamir

Oracle p ⇒ Real-world p

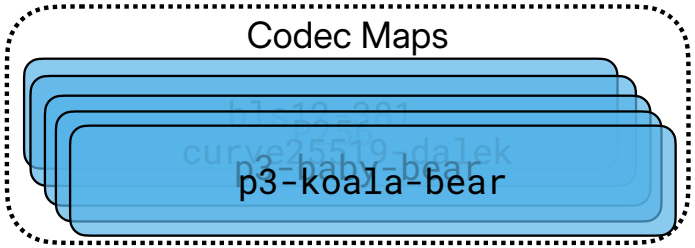
Negative results:
[Bar01, GK03, BBHMR19, KRS25]

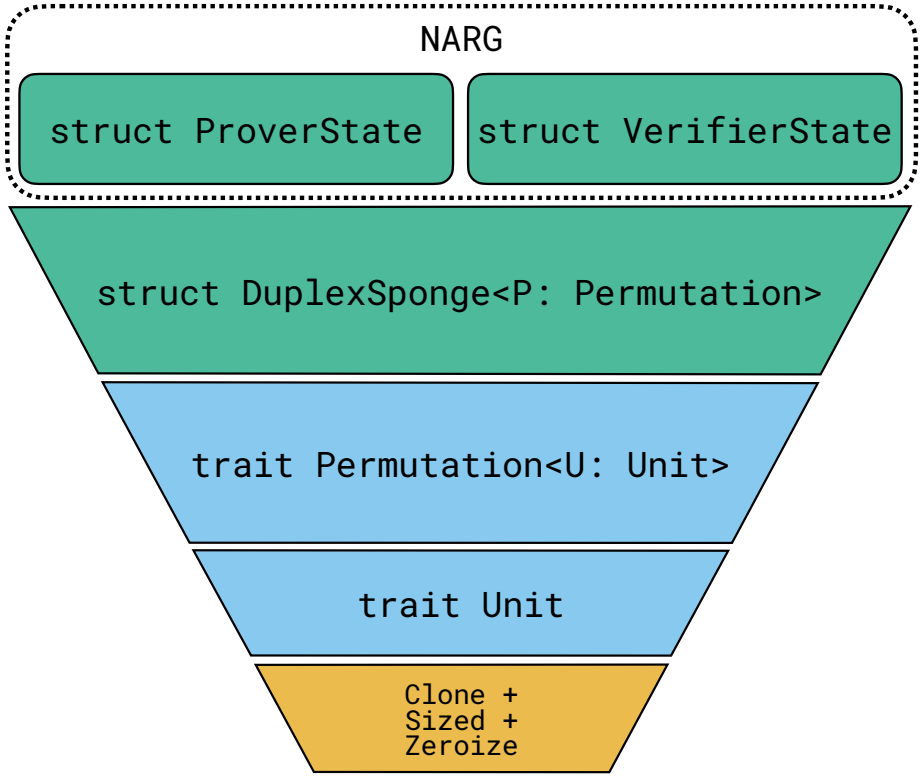
Fiat-Shamir

statistical soundness of IP
+
correlation intractability

[IKRR16, CCRR18, HL18, CCHLRRW19, PS19, LVW19, GJJM19, BFJKS19, LNPT19, CKU20, LV20, BKM20, JKKZ20, CLMQ20, LNPY20, JJ21, HLR21, CJJ21a, CJJ21b, LV22, HJKS22, GLS22, BCHKLPR22, KLV23, CGJJZ23, DJJ24,IL25, ...]

Implementation





Implementation

The software stack of github.com/arkworks-rs/spongefish

Implementation: some perks

`pub fn rng(&mut self) -> &mut (impl CryptoRng + RngCore)`

Provide the private random coins of the prover using also the witness' entropy.

`pub fn narg_string(&self) -> &[u8]`

Offer (inter-operable) proof serialization.

`!Clone, !Copy`

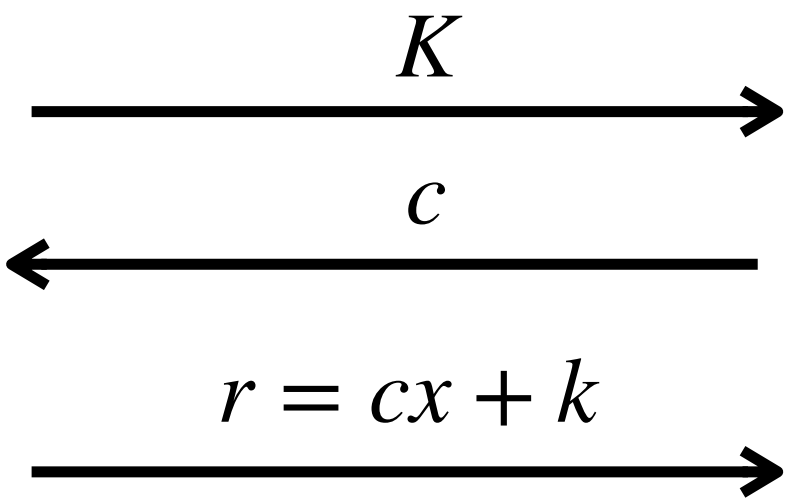
Do not implement copy to prevent accidental leaks.

Example: Schnorr proofs

$$\mathrm{DL}(G, X) = \mathrm{NIZK}\{x : xG = X\}$$

$k \leftarrow \mathbb{F}_p$

$K = kG$



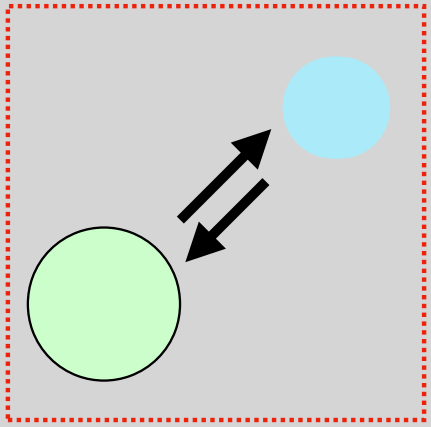

```rust
let k = G::ScalarField::rand(prover_state.rng());
let K = P * k;

prover_state.prover_message(&K)?;
```

```rust
let c = prover_state.verifier_message::<G::ScalarField>()?;
```

```rust
let r = k + c * x;
prover_state.prover_message(&r)?;

Ok(prover_state.narg_string())
```



ia.cr/2025/536

</>

arkworks.rs/spongefish

sigma.zkproof.org/fiat-shamir

A Fiat–Shamir transformation from Duplex Sponges

Open problems

Universal composability

Is it possible to prove universal composability in the global "ideal permutation" model (in analogy to GROM)?

Diagonalization attacks

How does this interact with recent attacks and fixes for Fiat-Shamir, and what about the duplex sponge transformation?