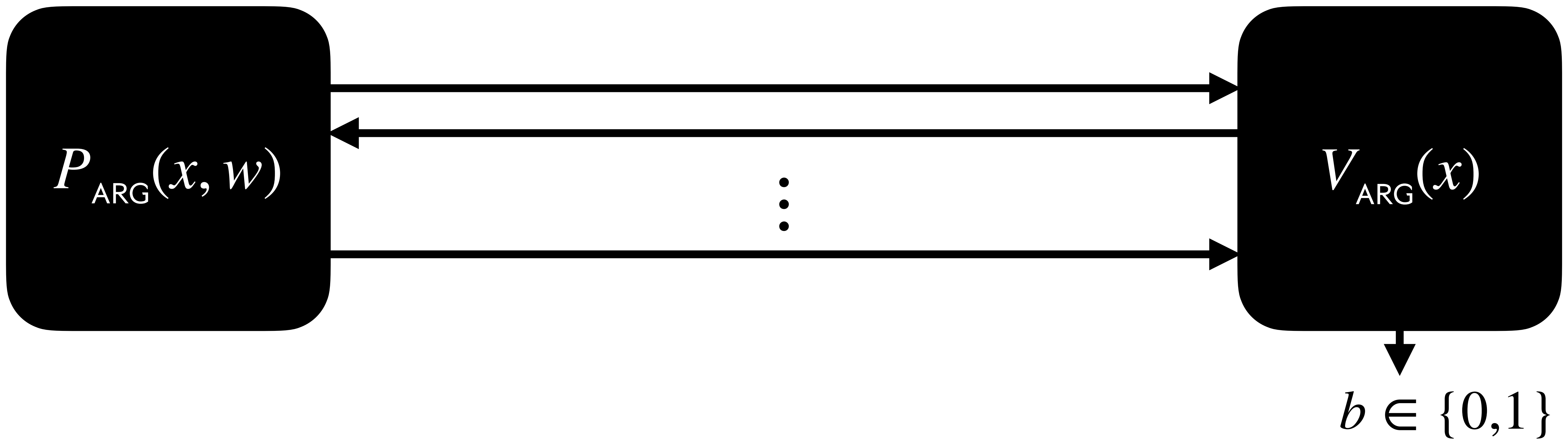
On the Fiat–Shamir Security of Succinct Arguments from Functional Commitments

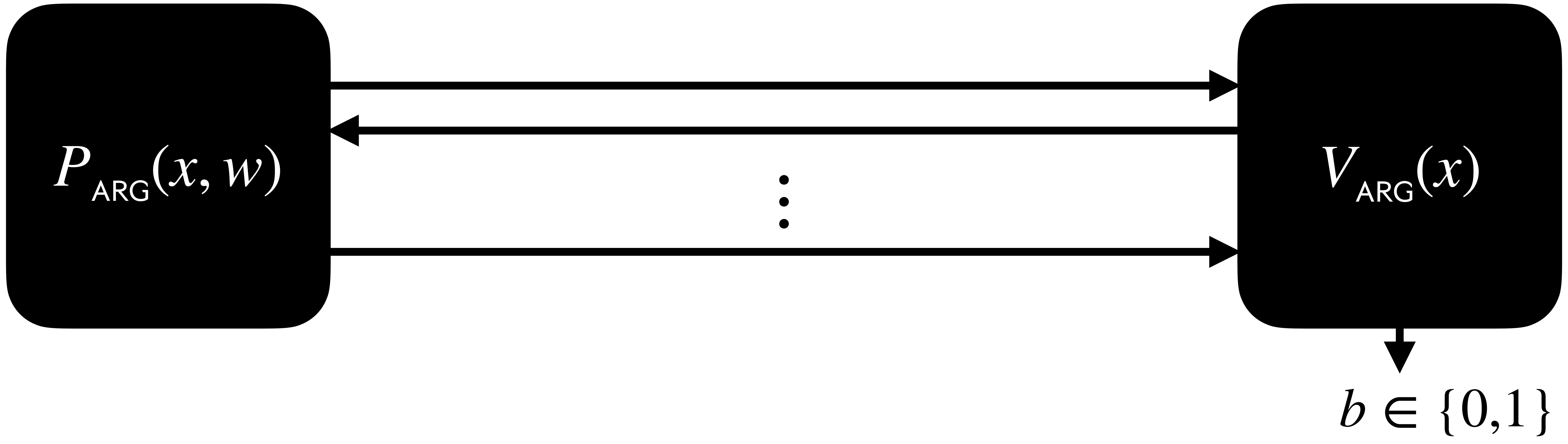
Alessandro Chiesa, Ziyi Guan, Christian Knabenhans, Zihan Yu

EPFL

Succinct interactive arguments

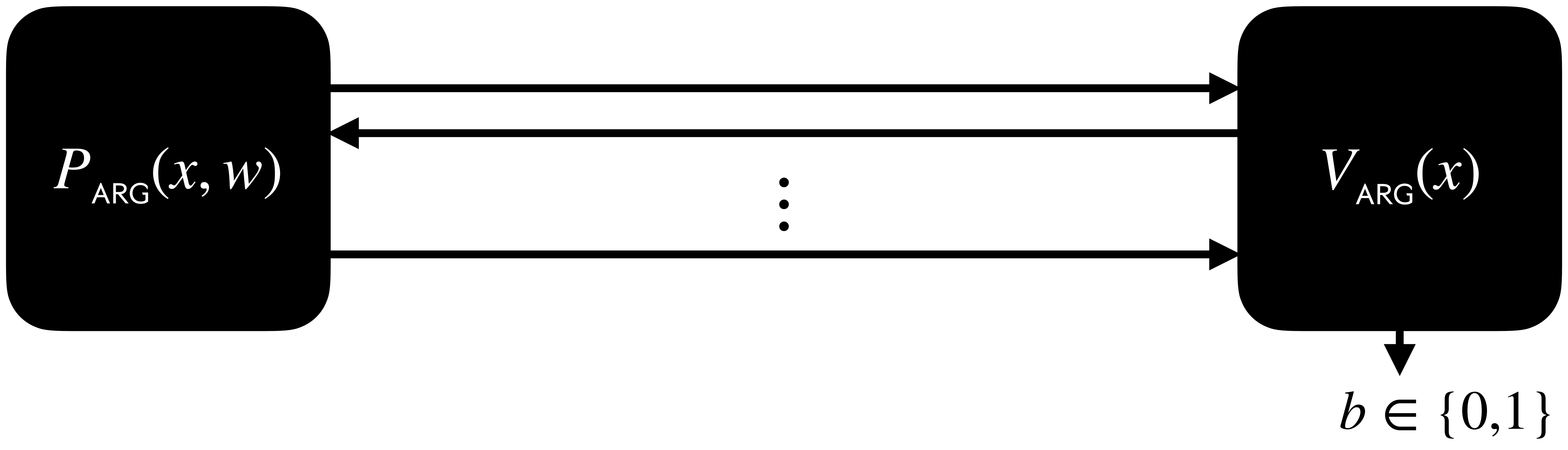


Succinct interactive arguments



Goal: succinctness $|\text{communication}| \ll |w|$

Succinct interactive arguments



Goal: succinctness $|\text{communication}| \ll |w|$

Applications: SNARGs in the ROM (via Fiat–Shamir in the ROM),
zero-knowledge with non-black-box simulation,
malicious MPC, etc.

How to build succinct interactive arguments?

Some approaches…	Proof string	Query class	Answer

How to build succinct interactive arguments?

Some approaches…	**Proof string**	**Query class**	**Answer**
PCP+VC [Kilian92] **IOP+VC** [BCS16,CDGS23]	$\Pi \in \Sigma^{\ell}$	point queries $\mathbf{Q}_{\text{point}}$	$\beta = \Pi[\alpha]$ for $\alpha \in [\ell]$

How to build succinct interactive arguments?

Some approaches…	**Proof string**	**Query class**	**Answer**
PCP+VC [Kilian92] **IOP+VC** [BCS16,CDGS23]	$\Pi \in \Sigma^{\ell}$	point queries $\mathbf{Q}_{\text{point}}$	$\beta = \Pi[\alpha]$ for $\alpha \in [\ell]$
LPCP+LC [LM19]	$\Pi \in \mathbb{F}^{\ell}$	linear queries $\mathbf{Q}_{\text{lin}}$	$\beta = \sum_{i \in [\ell]} \Pi[i] \cdot \alpha[i]$ for $\alpha \in \mathbb{F}^{\ell}$

How to build succinct interactive arguments?

Some approaches…	**Proof string**	**Query class**	**Answer**
PCP+VC [Kilian92] **IOP+VC** [BCS16,CDGS23]	$\Pi \in \Sigma^{\ell}$	point queries $\mathbf{Q}_{\text{point}}$	$\beta = \Pi[\alpha]$ for $\alpha \in [\ell]$
LPCP+LC [LM19]	$\Pi \in \mathbb{F}^{\ell}$	linear queries $\mathbf{Q}_{\text{lin}}$	$\beta = \sum_{i \in [\ell]} \Pi[i] \cdot \alpha[i]$ for $\alpha \in \mathbb{F}^{\ell}$
PIOP+PC [CHM+20,BFS20]	$\Pi \in \mathbb{F}[X]^{\leq \mathrm{D}}$	evaluation queries on polynomials $\mathbf{Q}_{\text{poly}}$	$\beta = \sum_{i \in [\ell]} \Pi[i] \cdot \alpha^{i-1}$ for $\alpha \in \mathbb{F}$

How to build succinct interactive arguments?

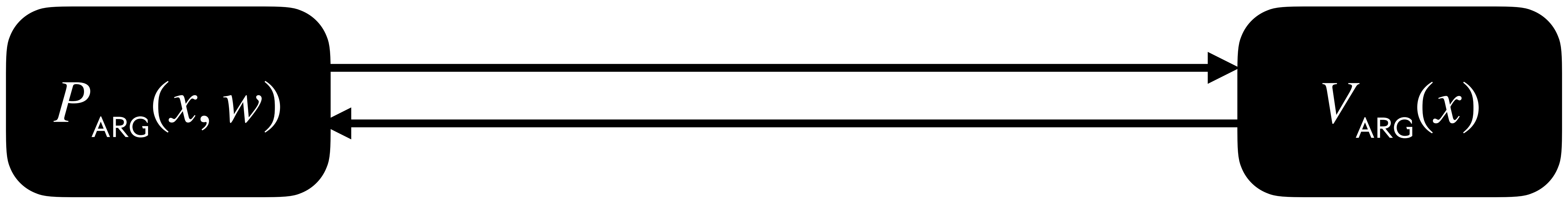
Some approaches…	**Proof string**	**Query class**	**Answer**
PCP+VC [Kilian92] **IOP+VC** [BCS16,CDGS23]	$\Pi \in \Sigma^{\ell}$	point queries $\mathbf{Q}_{\text{point}}$	$\beta = \Pi[\alpha]$ for $\alpha \in [\ell]$
LPCP+LC [LM19]	$\Pi \in \mathbb{F}^{\ell}$	linear queries $\mathbf{Q}_{\text{lin}}$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

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

How to build succinct interactive arguments?

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

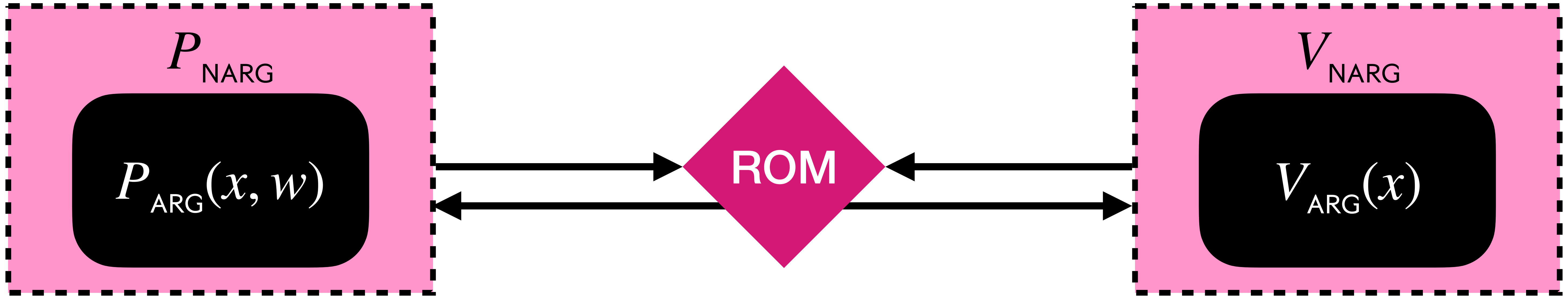
What security for interactive arguments?



soundness

$$\Pr\left[\begin{array}{c} x \notin L(R) \\ \wedge\ b = 1 \end{array} \middle|\ \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda) \\ x \leftarrow \tilde{P}_{\text{ARG}}(\mathsf{pp}) \\ b \leftarrow \left\langle \tilde{P}_{\text{ARG}}, V_{\text{ARG}}(\mathsf{pp}, x) \right\rangle \end{array}\right]$$

What security for interactive arguments?



soundness

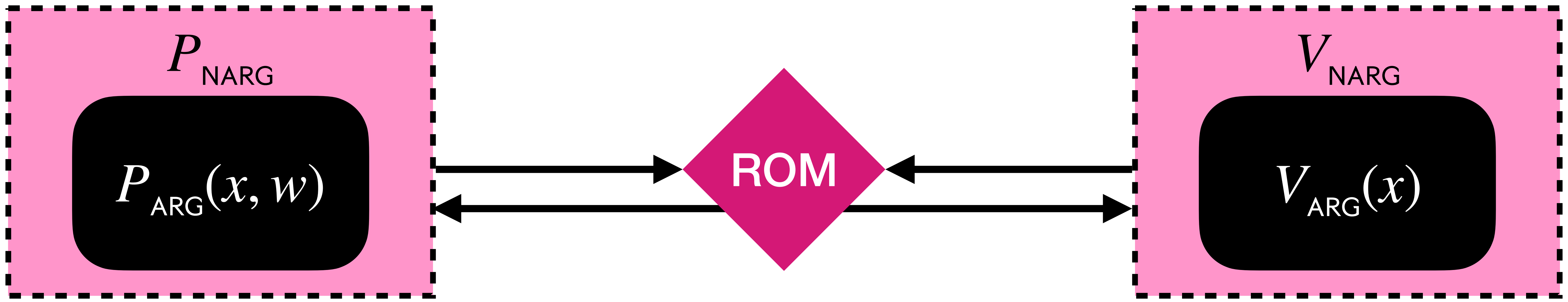
$$\Pr\left[\begin{array}{c} x \notin L(R) \\ \wedge\, b = 1 \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda) \\ x \leftarrow \tilde{P}_{\mathsf{ARG}}(\mathsf{pp}) \\ b \leftarrow \left\langle \tilde{P}_{\mathsf{ARG}}, V_{\mathsf{ARG}}(\mathsf{pp}, x) \right\rangle \end{array}\right]$$

state-restoration soundness

$$\Pr\left[\begin{array}{l} x \notin L(R) \\ \wedge\, V_{\mathsf{ARG}}(\mathsf{pp}, x, (m_i)_{i \in [k]}, (\rho_i)_{i \in [k]}) = 1 \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda) \\ (x, (m_i, \rho_i)_{i \in [\mathsf{k}]}) \leftarrow \mathsf{SRGame}(\tilde{P}_{\mathsf{ARG}}, \mathsf{pp}) \end{array}\right]$$

captures Fiat–Shamir security in the ROM

What security for interactive arguments?



soundness

$$\Pr\left[\begin{array}{c} x \notin L(R) \\ \wedge\, b = 1 \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda) \\ x \leftarrow \tilde{P}_{\text{ARG}}(\mathsf{pp}) \\ b \leftarrow \left\langle \tilde{P}_{\text{ARG}}, V_{\text{ARG}}(\mathsf{pp}, x) \right\rangle \end{array}\right]$$

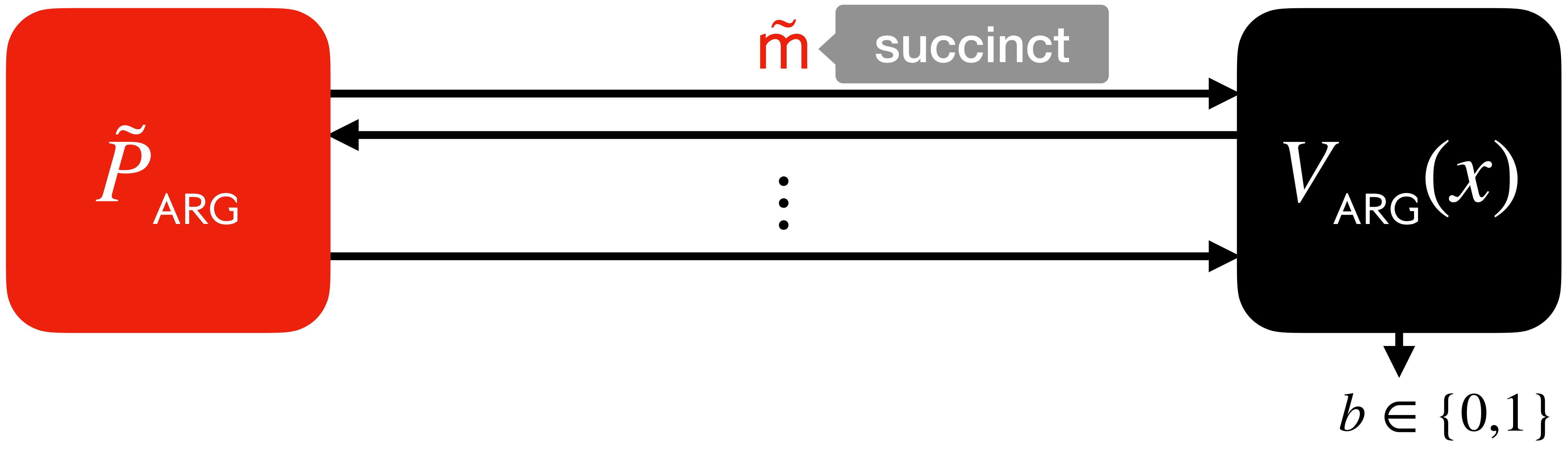
state-restoration soundness

$$\Pr\left[\begin{array}{l} x \notin L(R) \\ \wedge\, V_{\text{ARG}}(\mathsf{pp}, x, (m_i)_{i\in[k]}, (\rho_i)_{i\in[k]}) = 1 \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda) \\ (x, (m_i, \rho_i)_{i\in[\mathrm{k}]}) \leftarrow \mathsf{SRGame}(\tilde{P}_{\text{ARG}}, \mathsf{pp}) \end{array}\right]$$

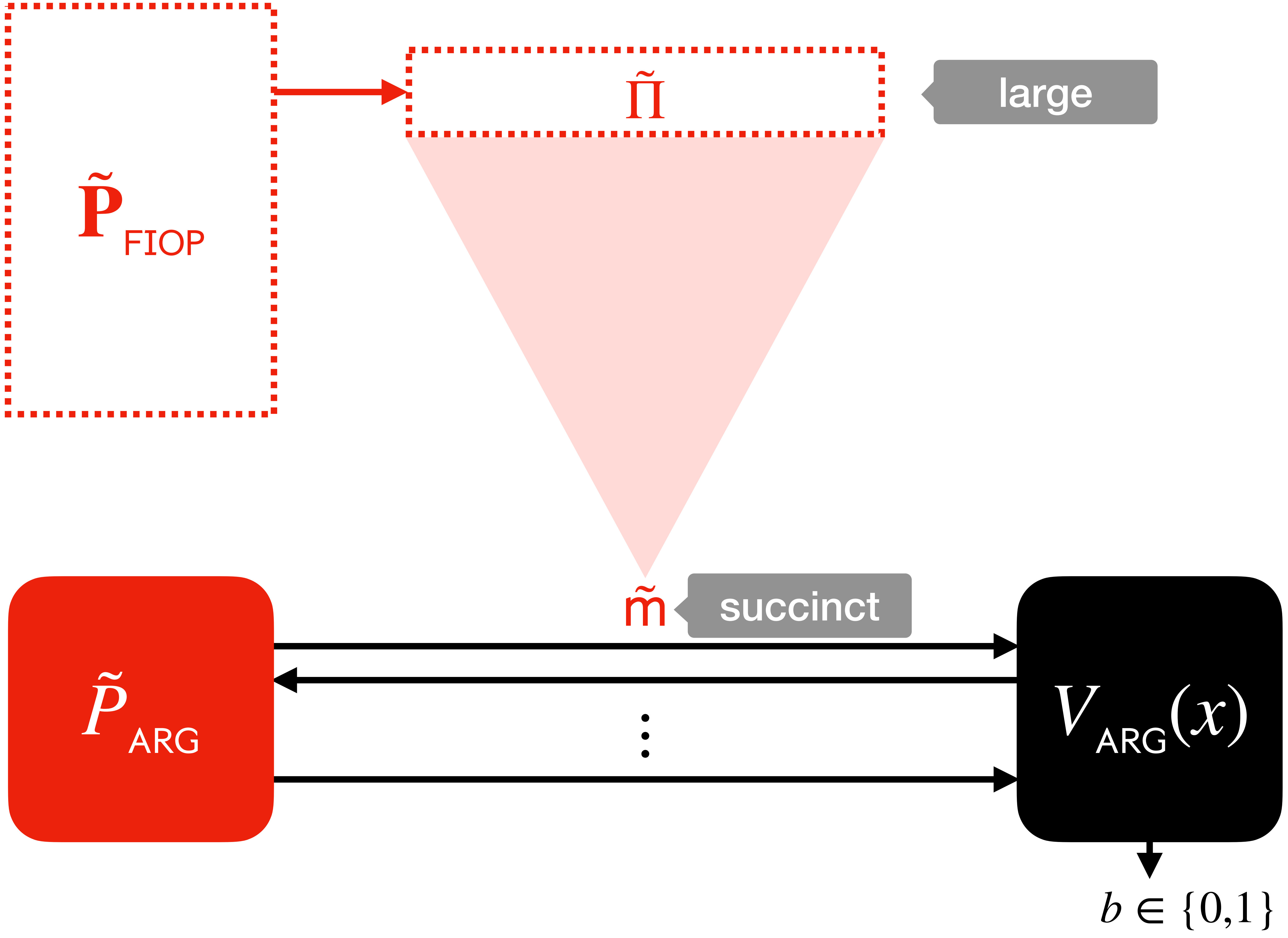
We only discuss the soundness case in this talk, but all results apply to knowledge soundness as well.

captures Fiat–Shamir security in the ROM

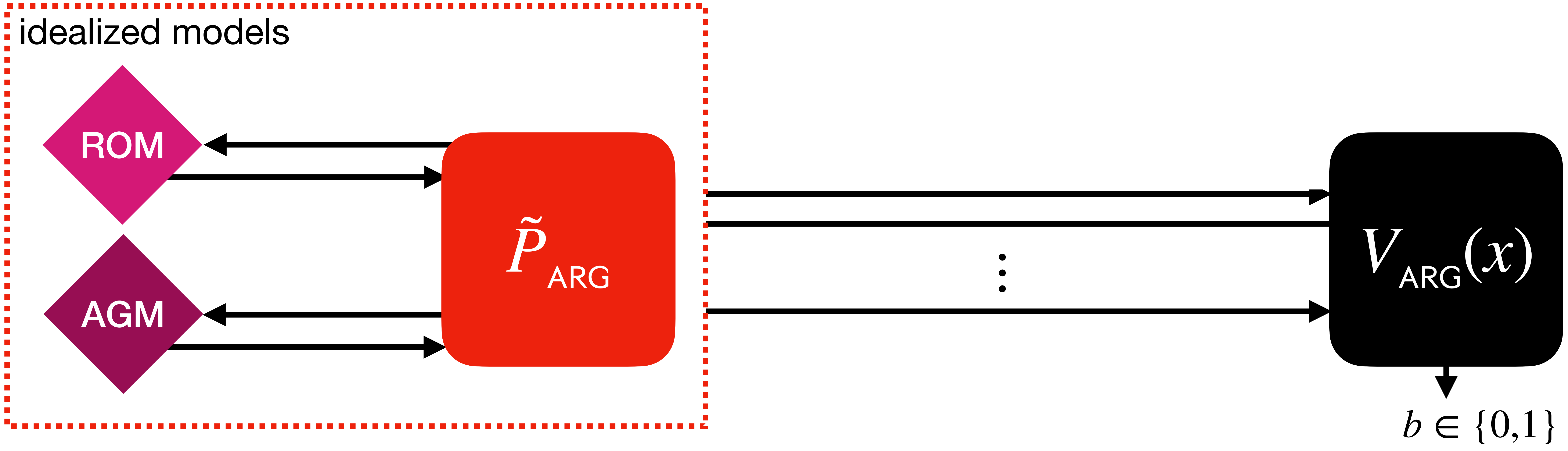
How to analyze succinct interactive arguments?



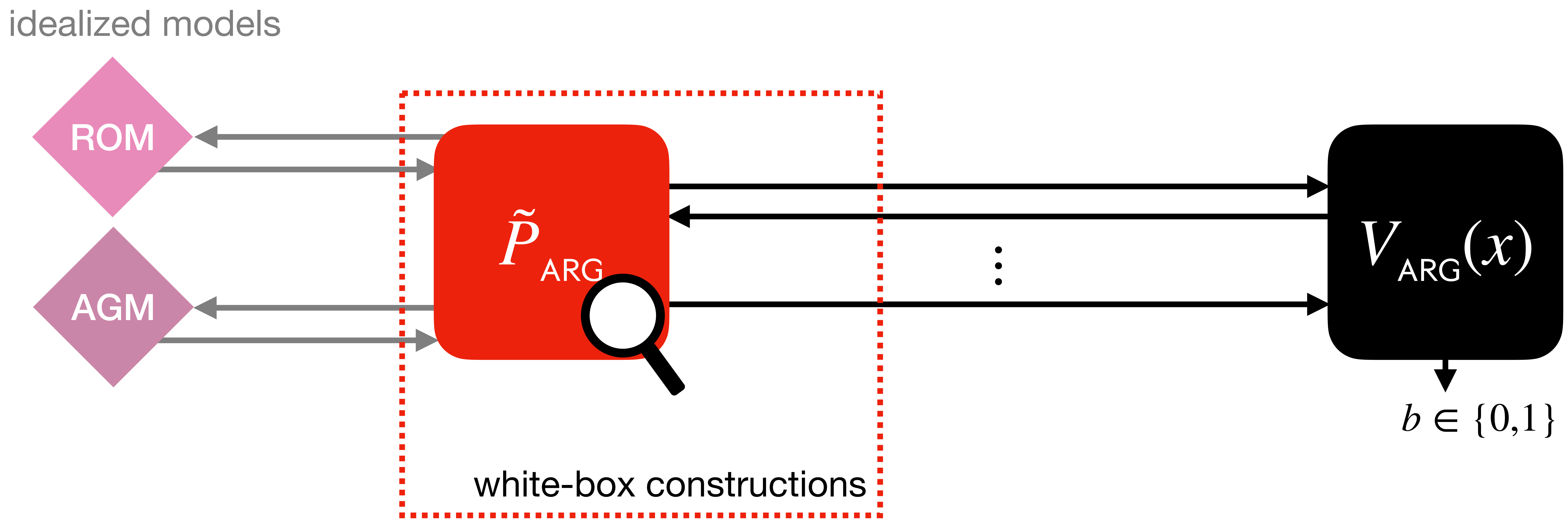
How to analyze succinct interactive arguments?



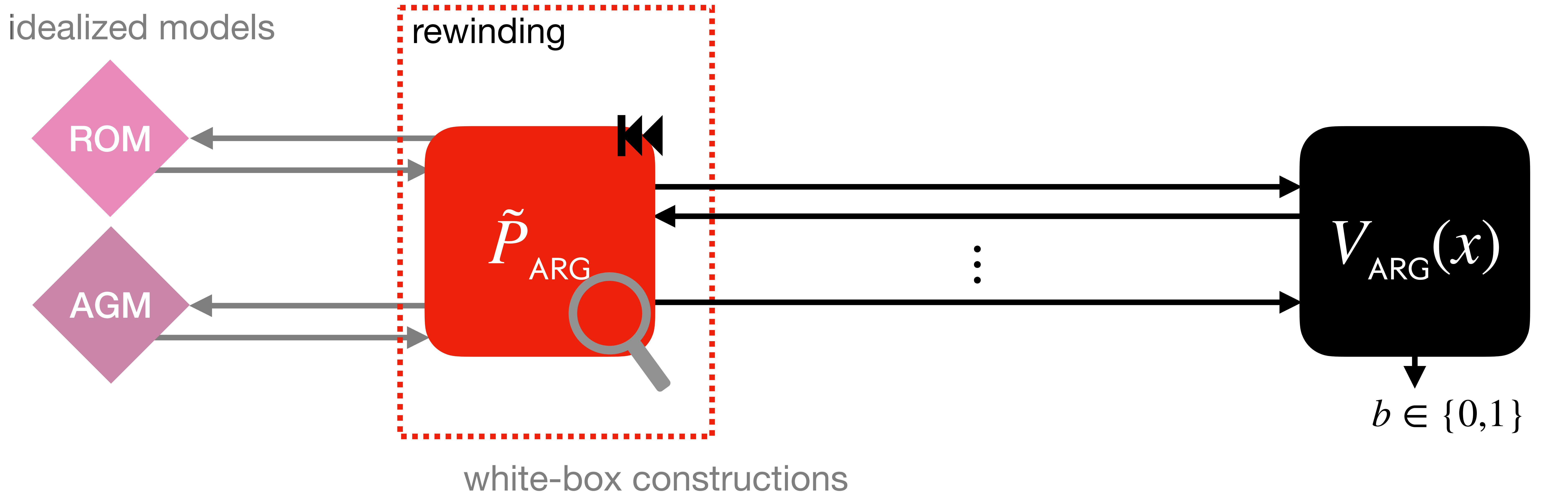
How to analyze succinct interactive arguments?



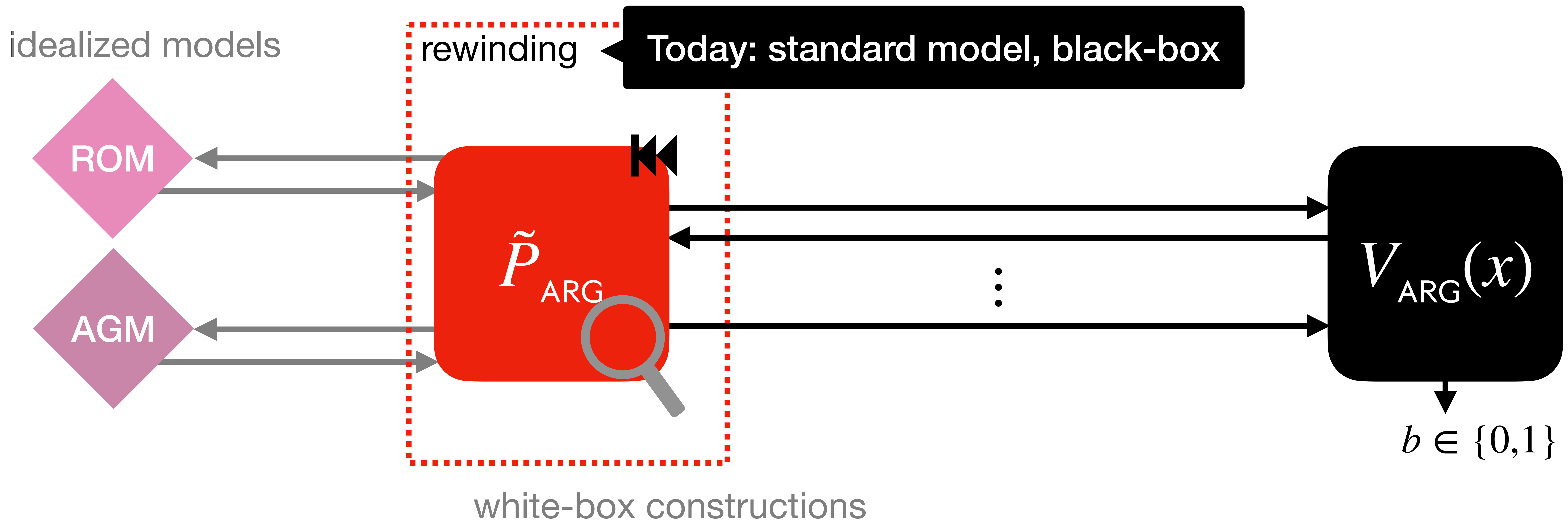
How to analyze succinct interactive arguments?



How to analyze succinct interactive arguments?



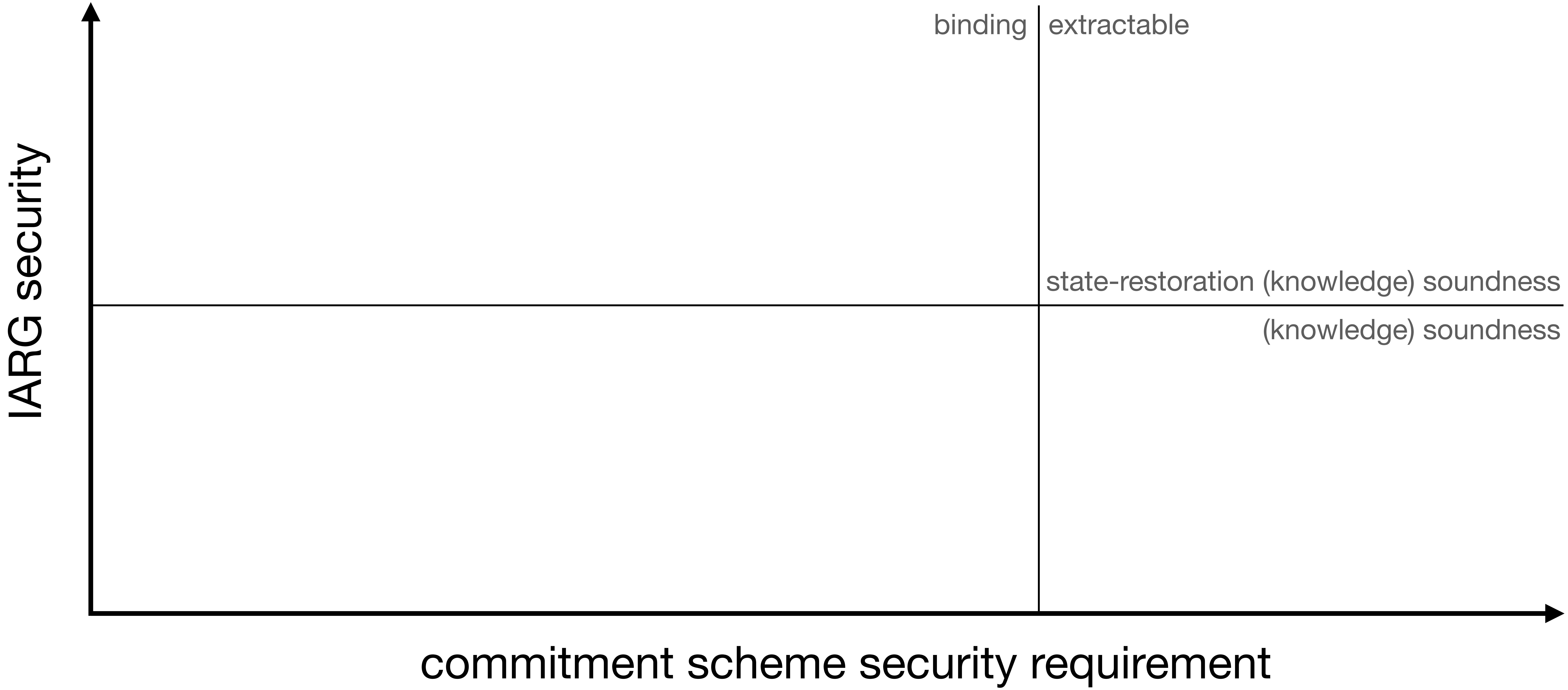
How to analyze succinct interactive arguments?



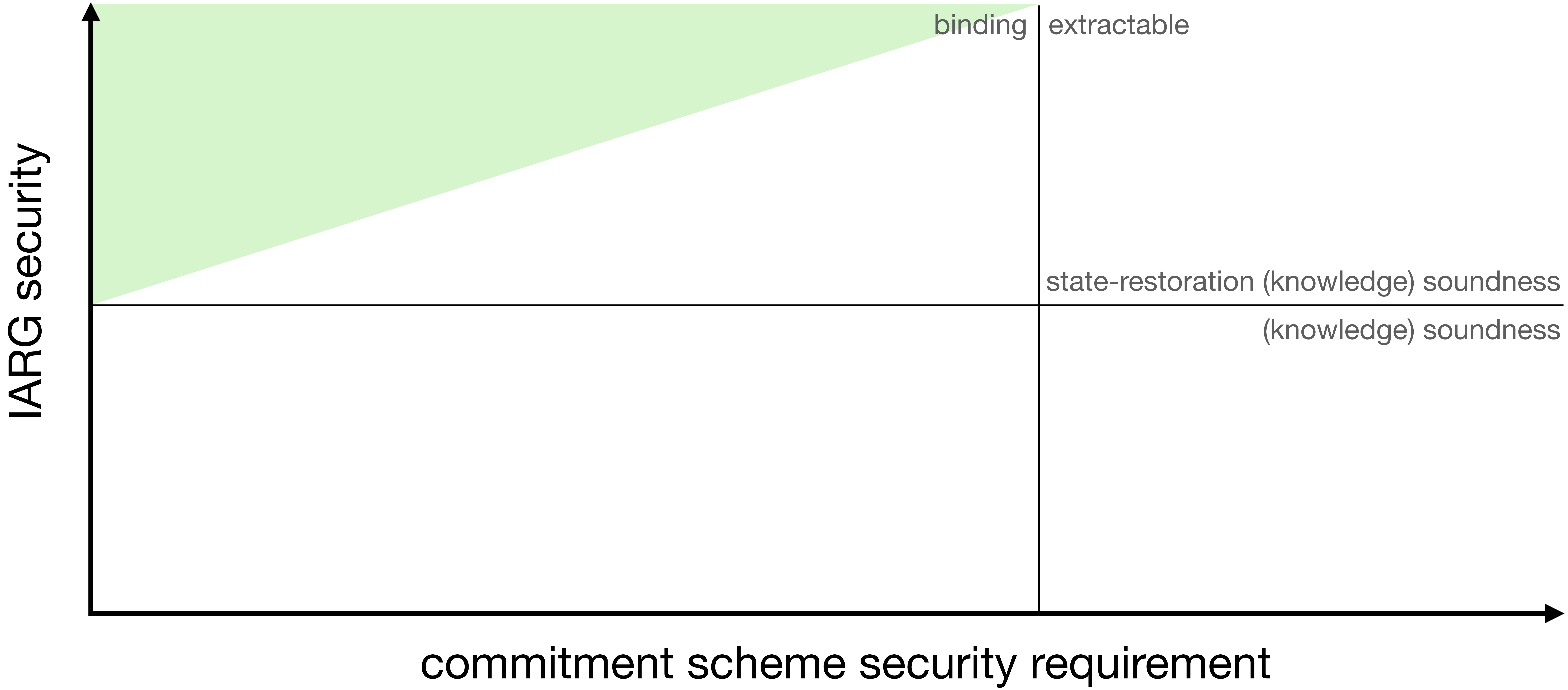
Previous standard-model analyses

Previous standard-model analyses

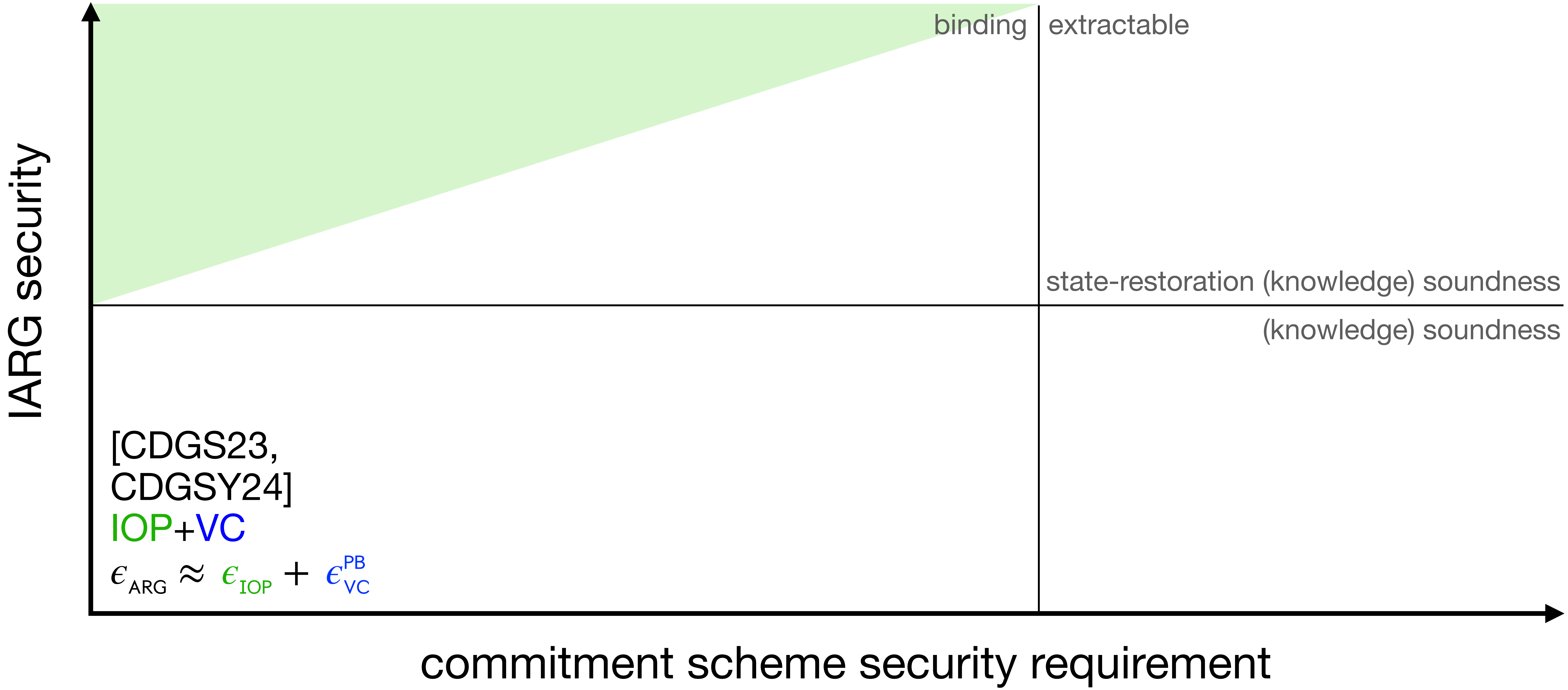
Previous standard-model analyses



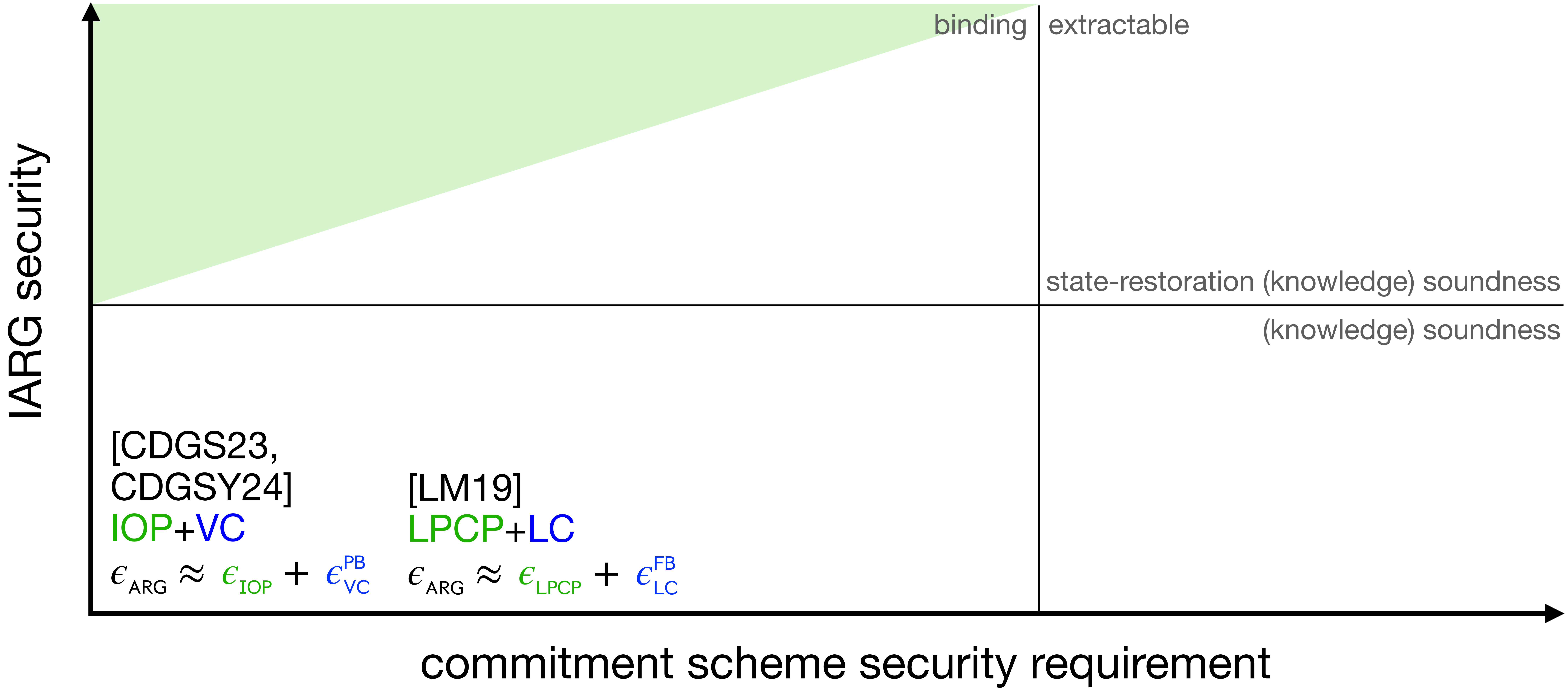
Previous standard-model analyses



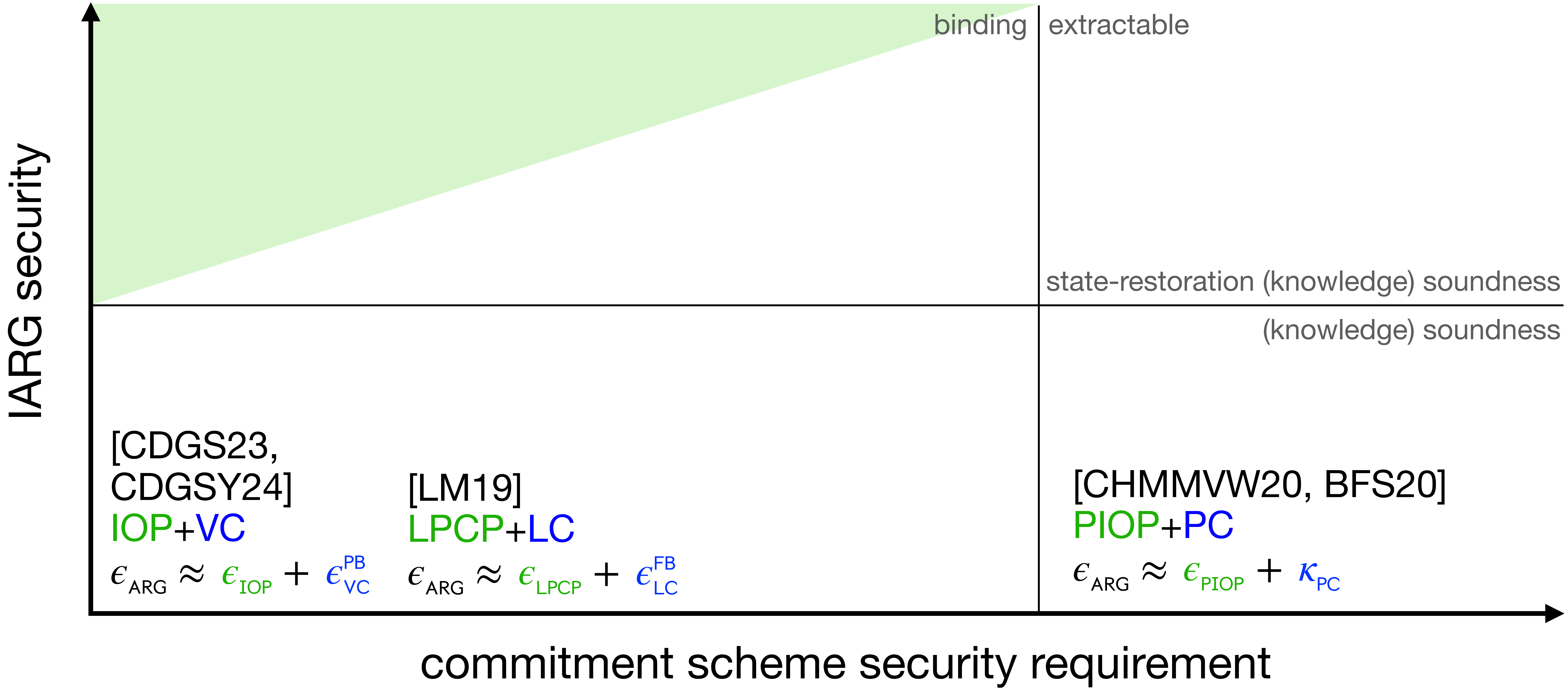
Previous standard-model analyses



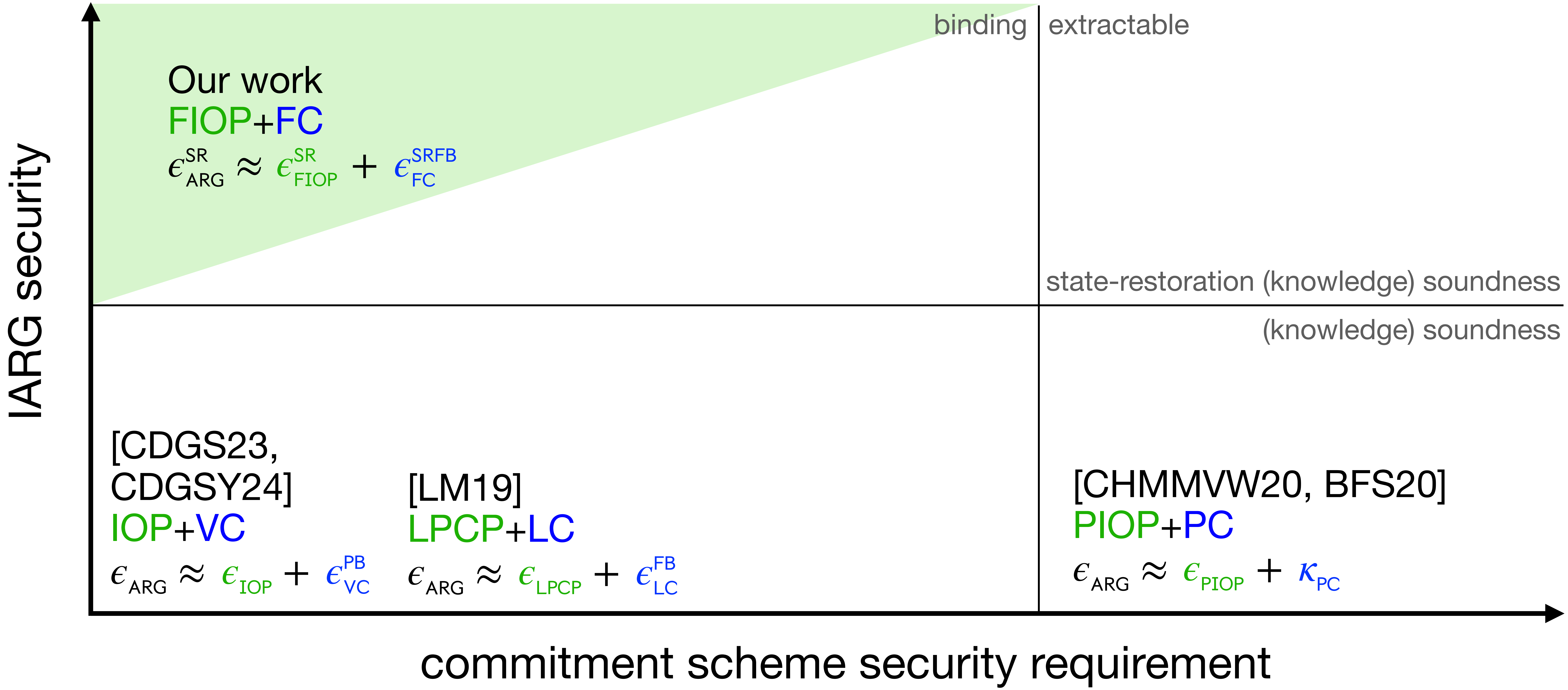
Previous standard-model analyses



Previous standard-model analyses



Previous standard-model analyses



Our results

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

Our results

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

FIOP is (knowledge) sound

\+ FC is function binding

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

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

Our results

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

FIOP is state-restoration (knowledge) sound

\+ FC is state-restoration function binding

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

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

Our results

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

FIOP is state-restoration (knowledge) sound

+ FC is state-restoration function binding

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

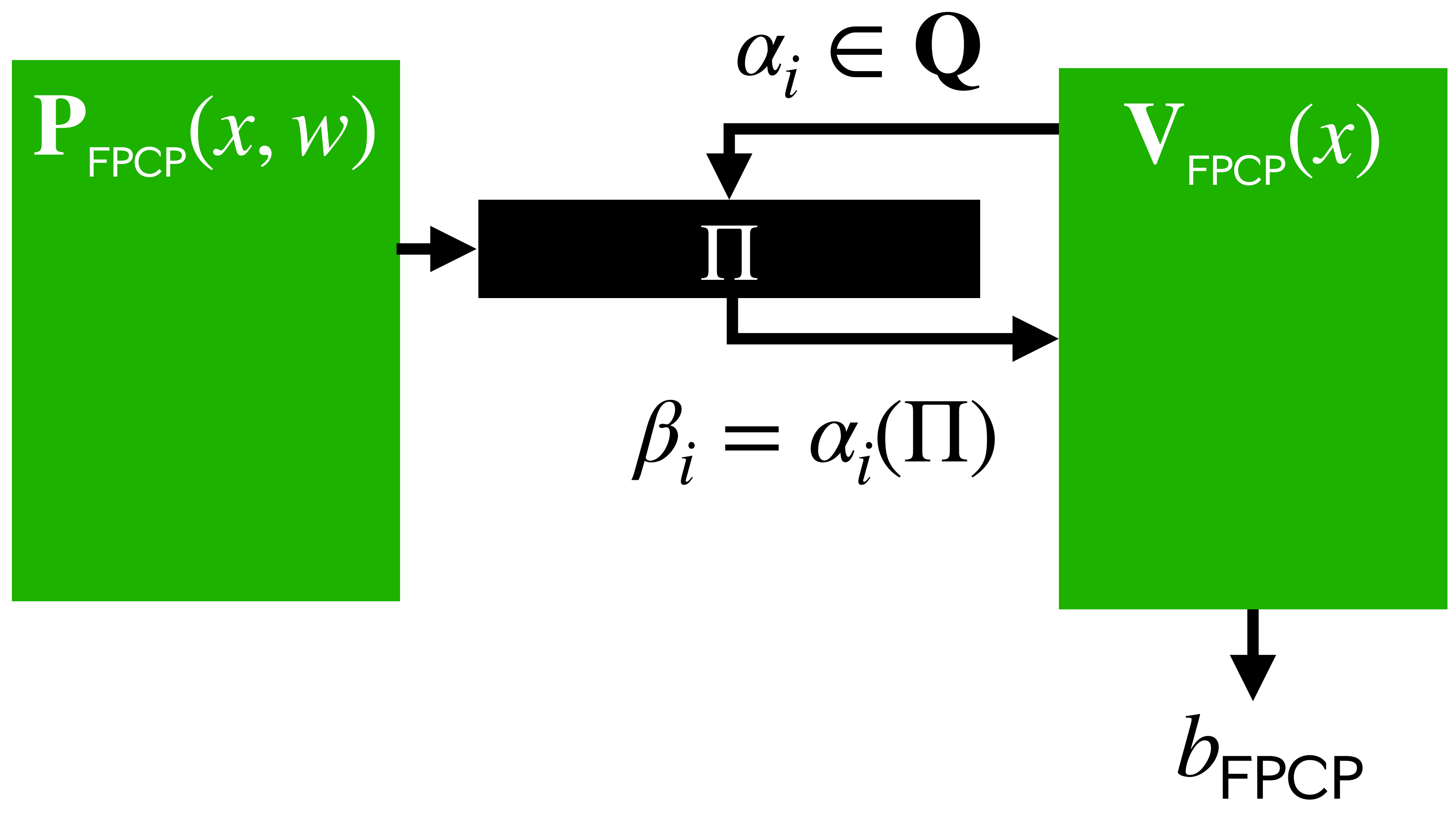
$$\epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(\mathsf{k}, \ell, t) \leq \epsilon^{\mathsf{SR}}_{\mathsf{FIOP}}(\mathsf{k}, \ell) + \epsilon^{\mathsf{SR}}_{\mathsf{FC}}(t \cdot \mathsf{k} \cdot \mathsf{N} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}(\ell, \mathsf{q}, \mathsf{N})$$

Application: Plonk = FS[Funky[PlonkIOP, linearized KZG]]
is a SNARK in the ROM
from the SDH assumption (previously: from ARSDH+SplitRSDH)

Warm-up:
the Funky protocol (for FPCPs and non-interactive FCs)
and its security

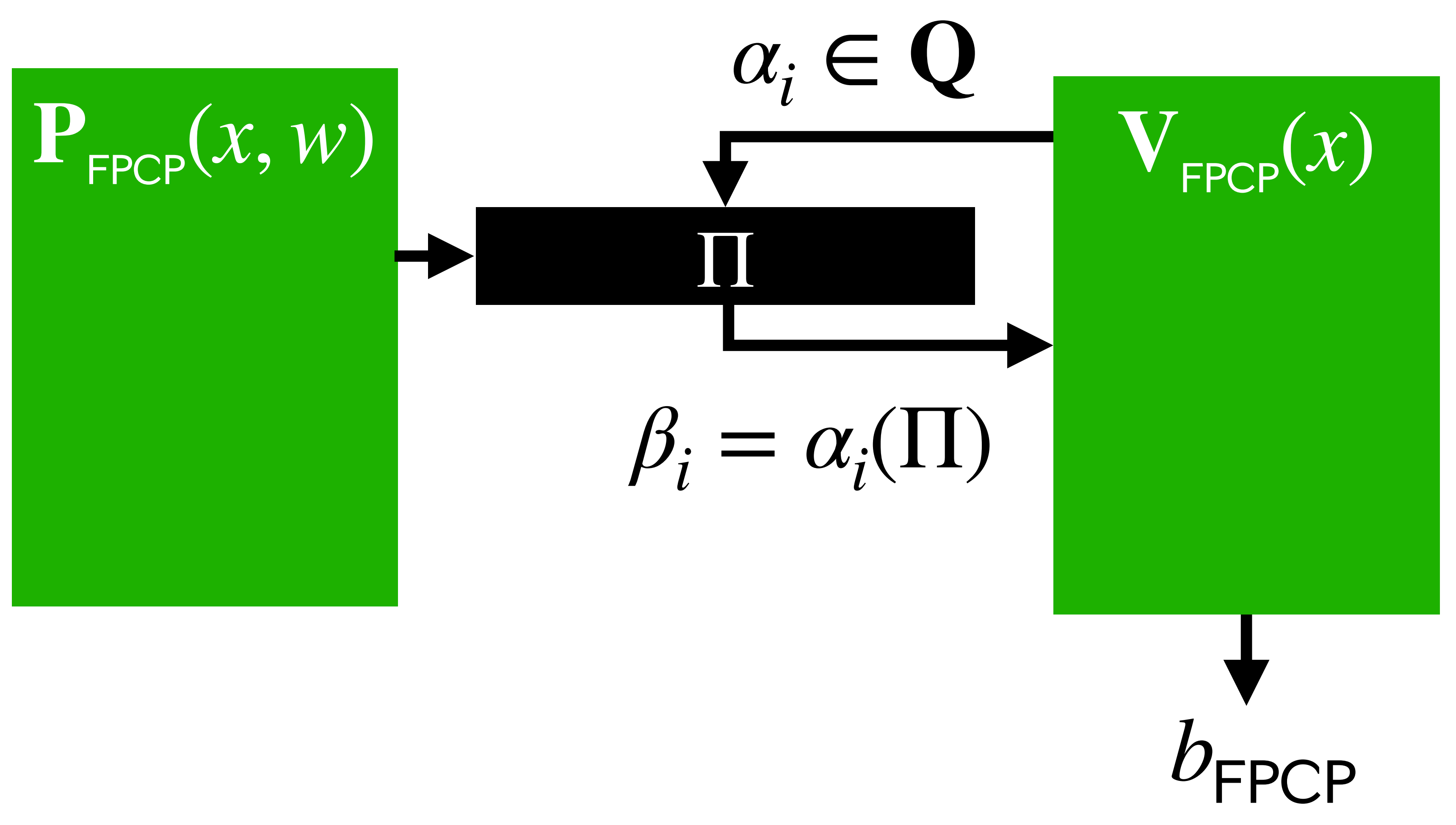
Building blocks

Functional
Probabilistically Checkable Proof
for $\mathbf{Q}$

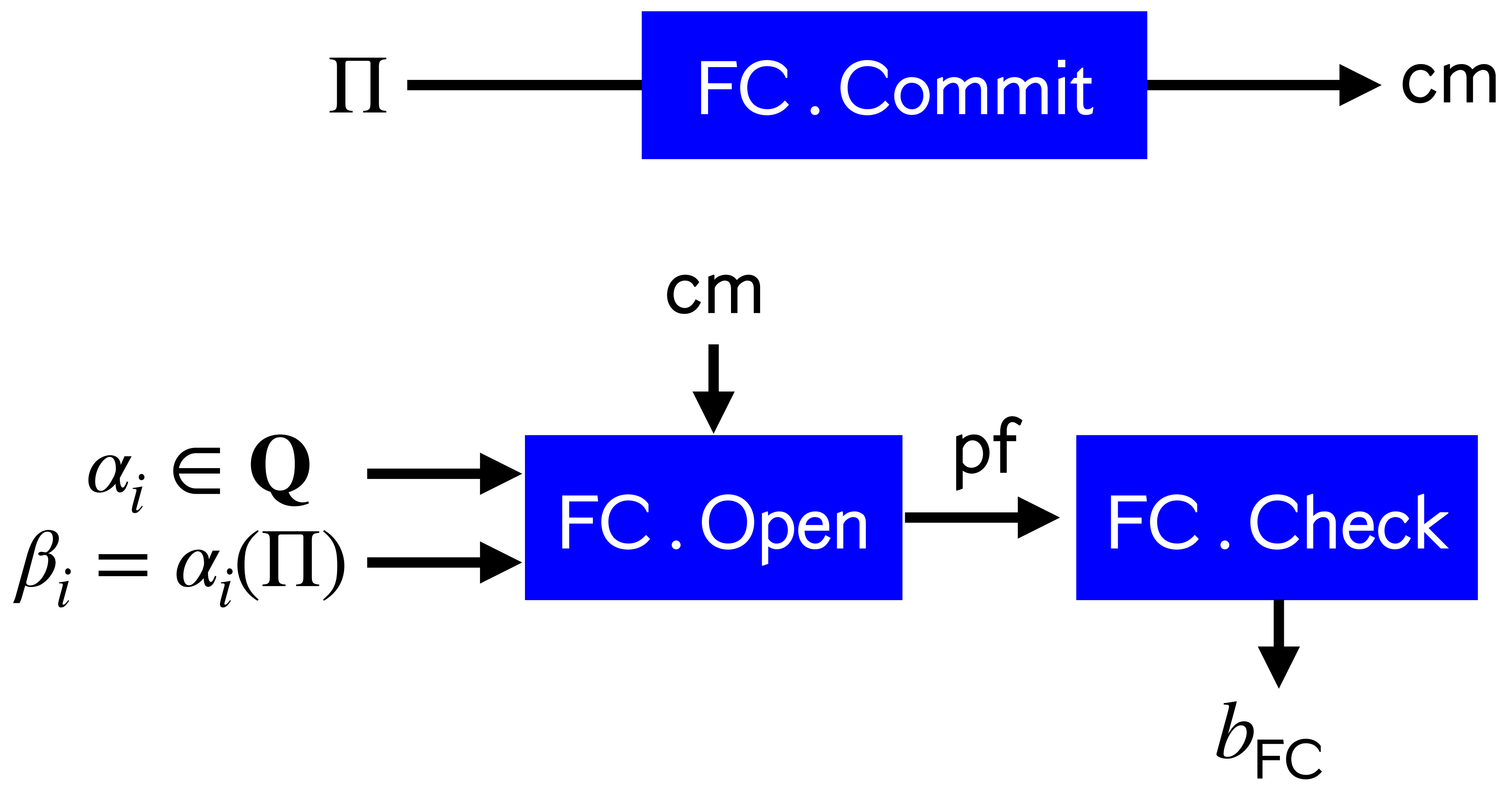


Building blocks

Functional
Probabilistically Checkable Proof
for $\mathbf{Q}$



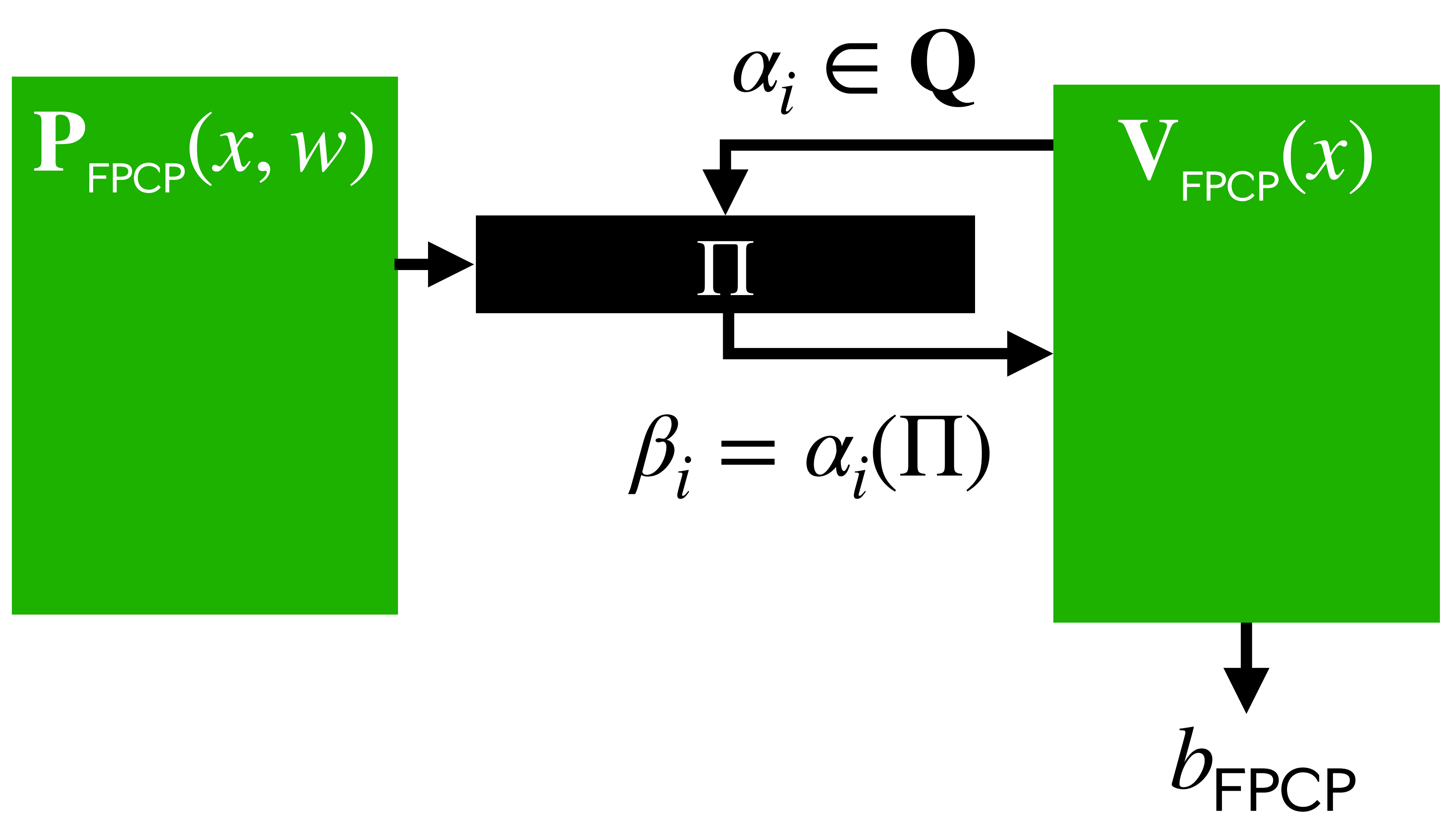
Functional
Commitment
for $\mathbf{Q}$



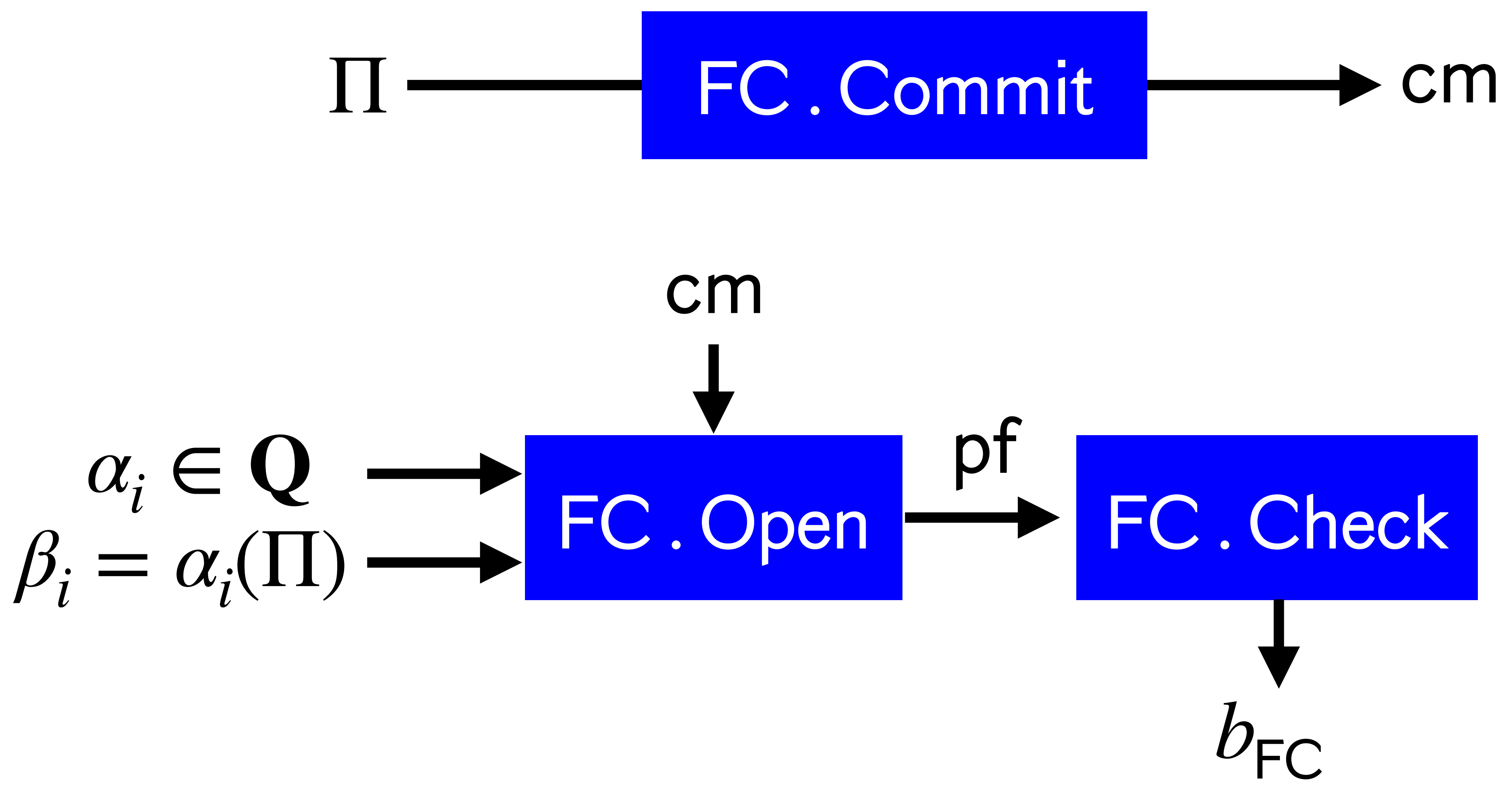
Building blocks

Security: (knowledge) soundness

Functional
Probabilistically Checkable Proof
for $\mathbf{Q}$



Functional
Commitment
for $\mathbf{Q}$



Which security property for FC?

Which security property for FC?

Vector Commitments position binding: $\Pr\left[\begin{array}{l} \beta_1 \neq \beta_2 \\ \wedge\ \forall i : \mathsf{FC}.\mathsf{Check}(\mathsf{pp}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 \end{array} \middle| \ (\mathsf{cm}, \alpha, \beta_1, \mathsf{pf}_1, \beta_2, \mathsf{pf}_2) \leftarrow A(\mathsf{pp}) \right] \leq \epsilon$

Which security property for FC?

Vector Commitments	position binding:	$\Pr\left[\begin{array}{l} \beta_1 \neq \beta_2 \\ \wedge\ \forall i : \mathsf{FC}.\mathsf{Check}(\mathsf{pp}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 \end{array} \;\middle|\; (\mathsf{cm}, \alpha, \beta_1, \mathsf{pf}_1, \beta_2, \mathsf{pf}_2) \leftarrow A(\mathsf{pp}) \right] \leq \epsilon$
Linear Commitments	function binding:	$\Pr\left[\begin{array}{l} \nexists \Pi : \forall i : \langle \alpha_i, \Pi \rangle = \beta_i \\ \wedge\ \forall i : \mathsf{FC}.\mathsf{Check}(\mathsf{pp}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 \end{array} \;\middle|\; (\mathsf{cm}, (\alpha_i, \beta_i, \mathsf{pf}_i)_{i \in [n]}) \leftarrow A(\mathsf{pp}) \right] \leq \epsilon$

Which security property for FC?

Vector Commitments	position binding:	$\Pr\left[\begin{array}{l}\beta_1 \neq \beta_2 \\ \wedge\ \forall i : \mathsf{FC}.\mathsf{Check}(\mathsf{pp}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1\end{array}\ \middle|\ (\mathsf{cm}, \alpha, \beta_1, \mathsf{pf}_1, \beta_2, \mathsf{pf}_2) \leftarrow A(\mathsf{pp})\right] \leq \epsilon$
Linear Commitments	function binding:	$\Pr\left[\begin{array}{l}\nexists \Pi : \forall i : \langle \alpha_i, \Pi \rangle = \beta_i \\ \wedge\ \forall i : \mathsf{FC}.\mathsf{Check}(\mathsf{pp}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1\end{array}\ \middle|\ (\mathsf{cm}, (\alpha_i, \beta_i, \mathsf{pf}_i)_{i \in [n]}) \leftarrow A(\mathsf{pp})\right] \leq \epsilon$
Polynomial Commitments	binding? strong correctness? interpolation binding? extractability? [KZG10] [AJMMS23] [CHM+20, BFS20]	

Which security property for FC?

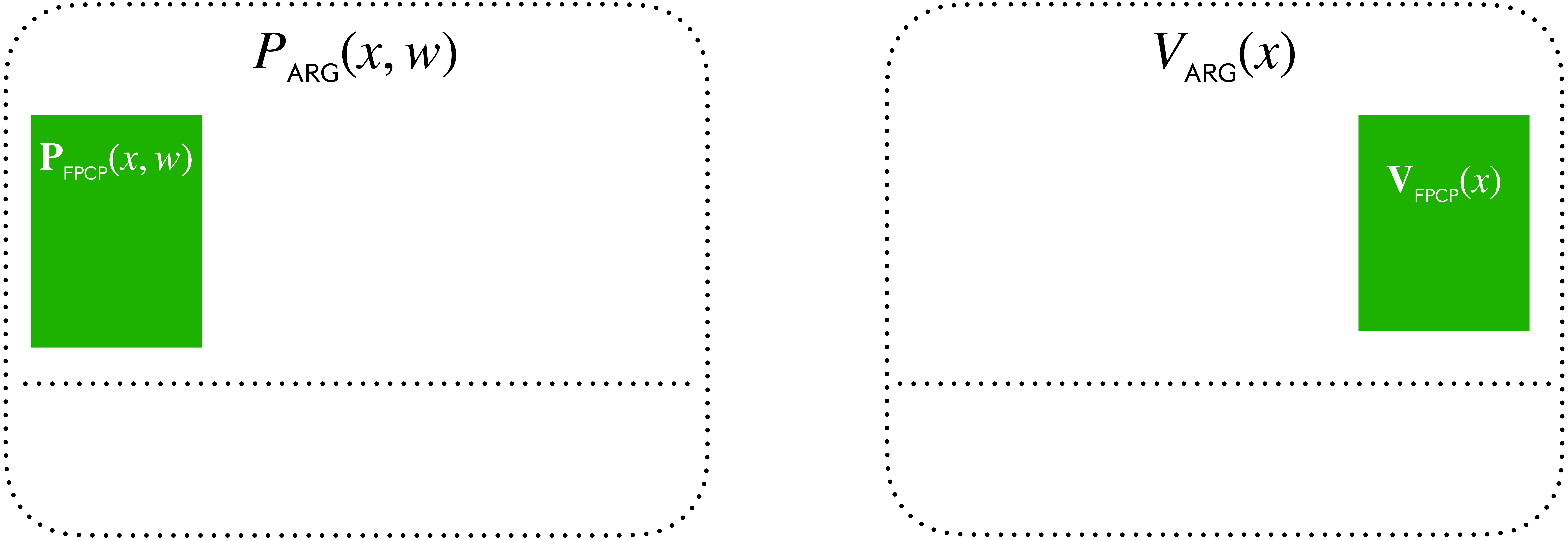
Vector Commitments	position binding:	$\Pr\left[\begin{array}{l}\beta_1 \neq \beta_2 \\ \wedge\ \forall i : \mathsf{FC.Check}(\mathsf{pp}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1\end{array} \middle\vert\ (\mathsf{cm}, \alpha, \beta_1, \mathsf{pf}_1, \beta_2, \mathsf{pf}_2) \leftarrow A(\mathsf{pp})\right] \leq \epsilon$
Linear Commitments	function binding:	$\Pr\left[\begin{array}{l}\nexists \Pi : \forall i : \langle \alpha_i, \Pi \rangle = \beta_i \\ \wedge\ \forall i : \mathsf{FC.Check}(\mathsf{pp}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1\end{array} \middle\vert\ (\mathsf{cm}, (\alpha_i, \beta_i, \mathsf{pf}_i)_{i \in [n]}) \leftarrow A(\mathsf{pp})\right] \leq \epsilon$
Polynomial Commitments	binding? strong correctness? interpolation binding? extractability? [KZG10] [AJMMS23] [CHM+20, BFS20]	
Functional Commitments	function binding:	$\Pr\left[\begin{array}{l}\nexists \Pi : \forall i : \alpha_i(\Pi) = \beta_i \\ \wedge\ \forall i : \mathsf{FC.Check}(\mathsf{pp}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1\end{array} \middle\vert\ (\mathsf{cm}, (\alpha_i, \beta_i, \mathsf{pf}_i)_{i \in [n]}) \leftarrow A(\mathsf{pp})\right] \leq \epsilon$

Which security property for FC?

Vector Commitments	position binding:	$\Pr\left[\begin{array}{l} \beta_1 \neq \beta_2 \\ \wedge\ \forall i : \mathsf{FC}.\mathsf{Check}(\mathsf{pp}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 \end{array} \middle\vert\ (\mathsf{cm}, \alpha, \beta_1, \mathsf{pf}_1, \beta_2, \mathsf{pf}_2) \leftarrow A(\mathsf{pp}) \right] \leq \epsilon$
Linear Commitments	function binding:	$\Pr\left[\begin{array}{l} \nexists \Pi : \forall i : \langle \alpha_i, \Pi \rangle = \beta_i \\ \wedge\ \forall i : \mathsf{FC}.\mathsf{Check}(\mathsf{pp}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 \end{array} \middle\vert\ (\mathsf{cm}, (\alpha_i, \beta_i, \mathsf{pf}_i)_{i \in [n]}) \leftarrow A(\mathsf{pp}) \right] \leq \epsilon$
Polynomial Commitments	binding? strong correctness? interpolation binding? extractability? [KZG10] [AJMMS23] [CHM+20, BFS20]	
Functional Commitments	function binding:	$\Pr\left[\begin{array}{l} \nexists \Pi : \forall i : \alpha_i(\Pi) = \beta_i \\ \wedge\ \forall i : \mathsf{FC}.\mathsf{Check}(\mathsf{pp}, \mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 \end{array} \middle\vert\ (\mathsf{cm}, (\alpha_i, \beta_i, \mathsf{pf}_i)_{i \in [n]}) \leftarrow A(\mathsf{pp}) \right] \leq \epsilon$

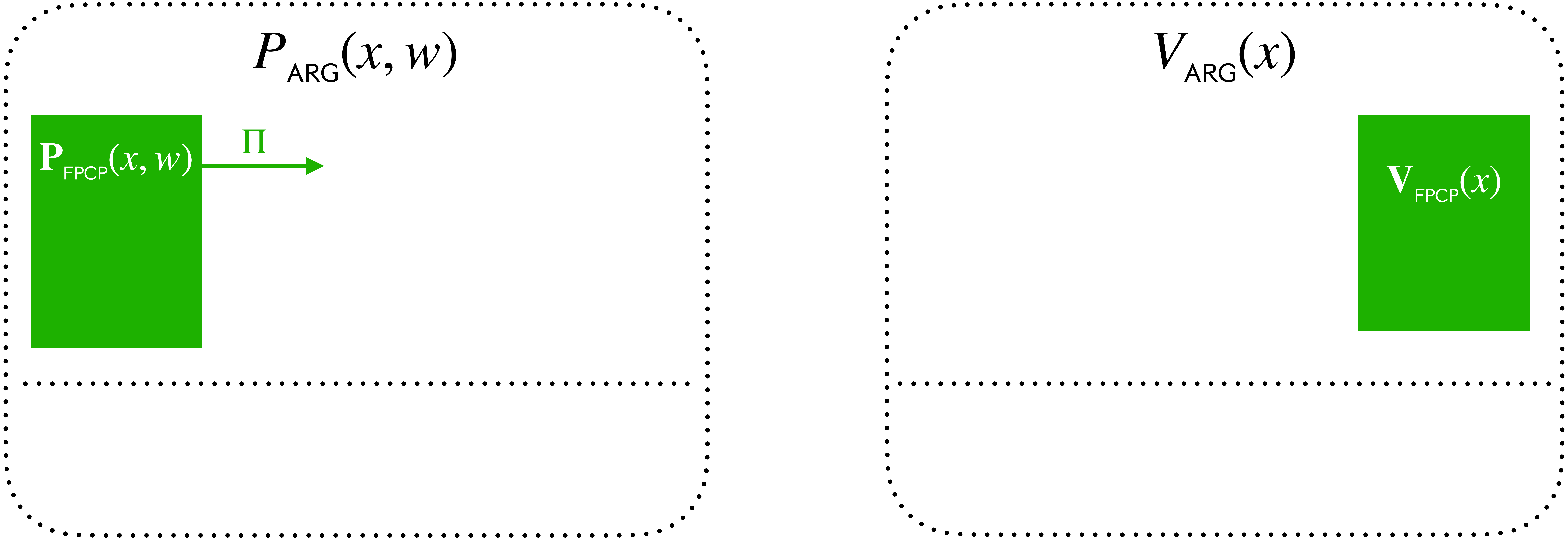
The Funky protocol

For FPCPs and non-interactive FCs



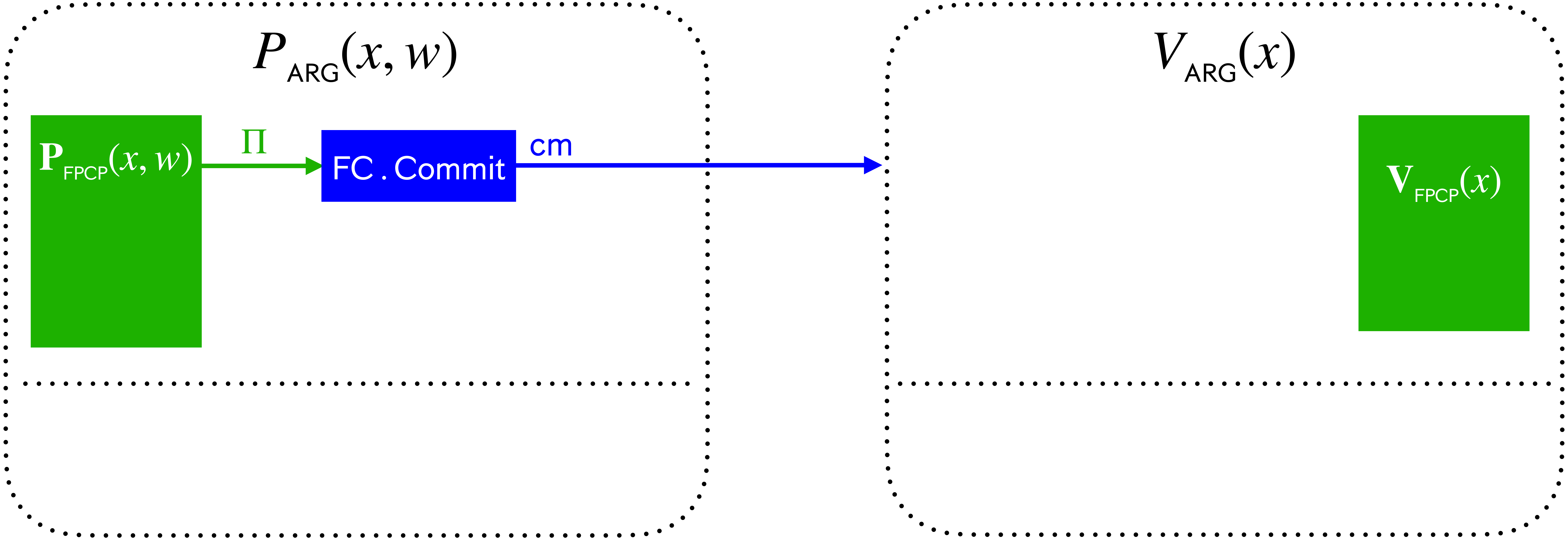
The Funky protocol

For FPCPs and non-interactive FCs



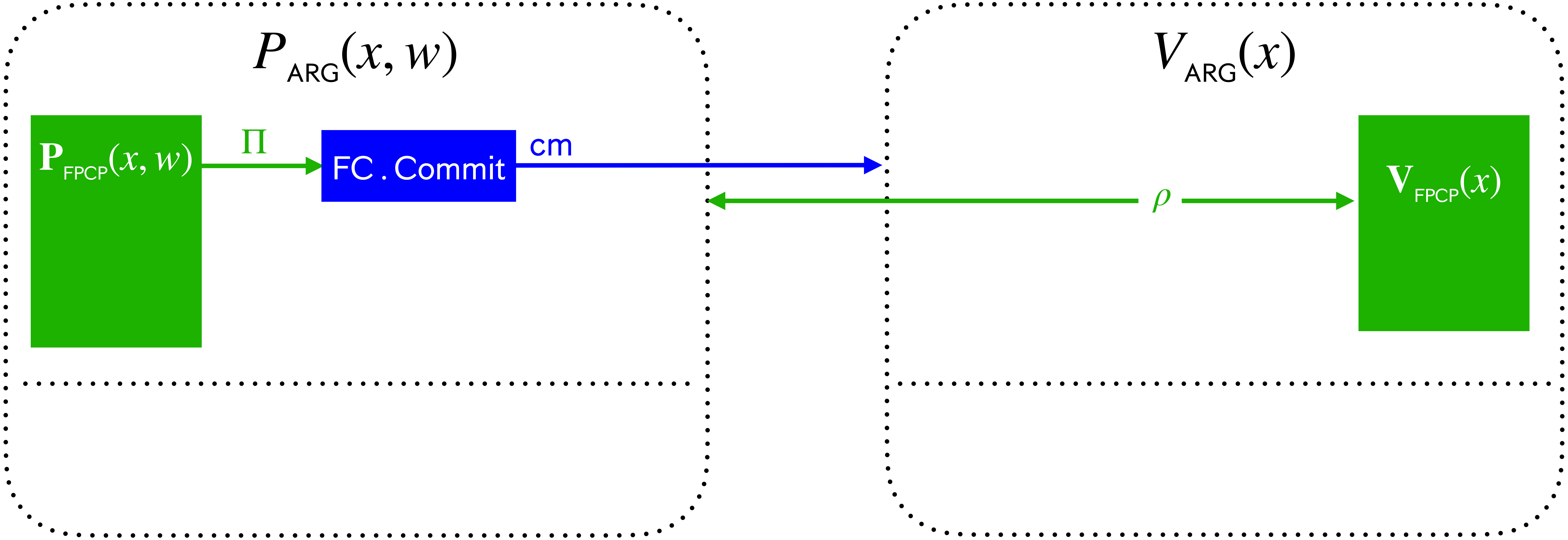
The Funky protocol

For FPCPs and non-interactive FCs



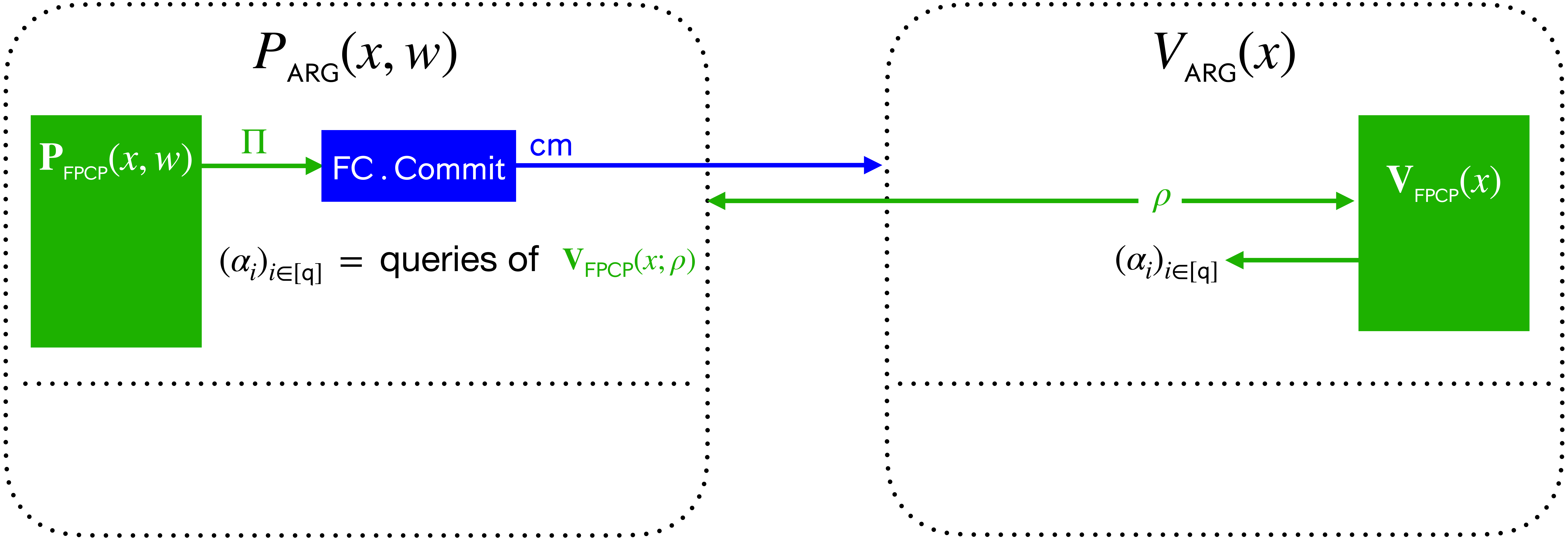
The Funky protocol

For FPCPs and non-interactive FCs



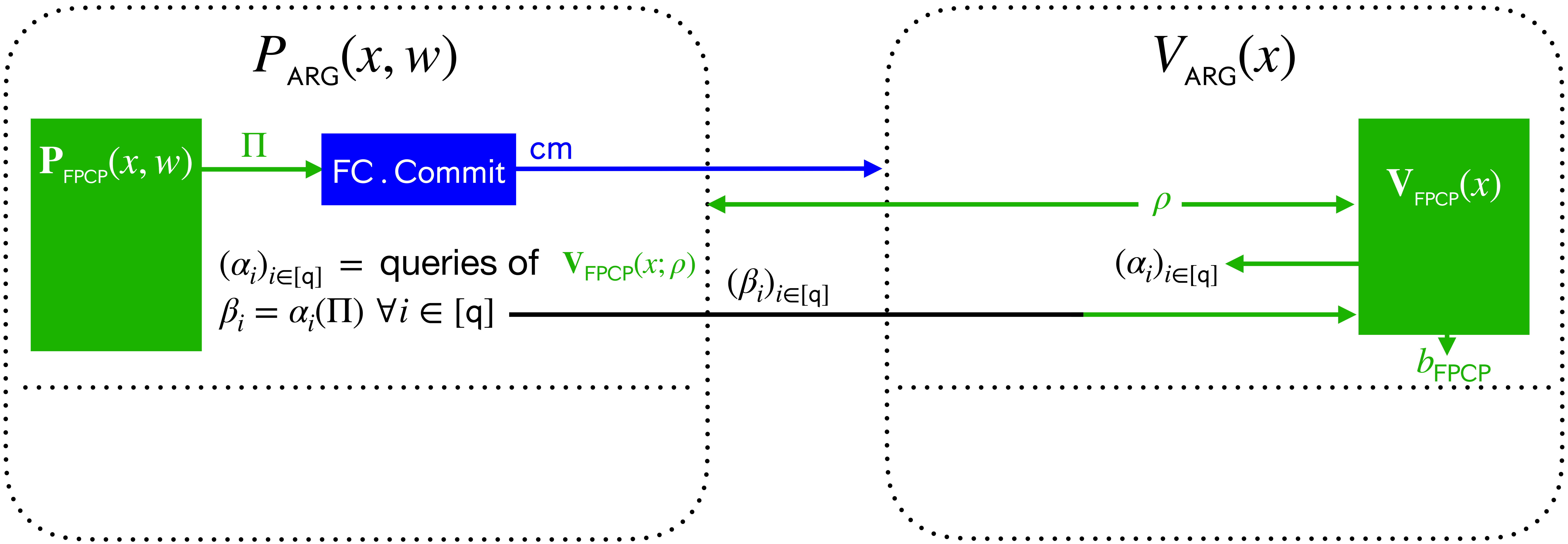
The Funky protocol

For FPCPs and non-interactive FCs



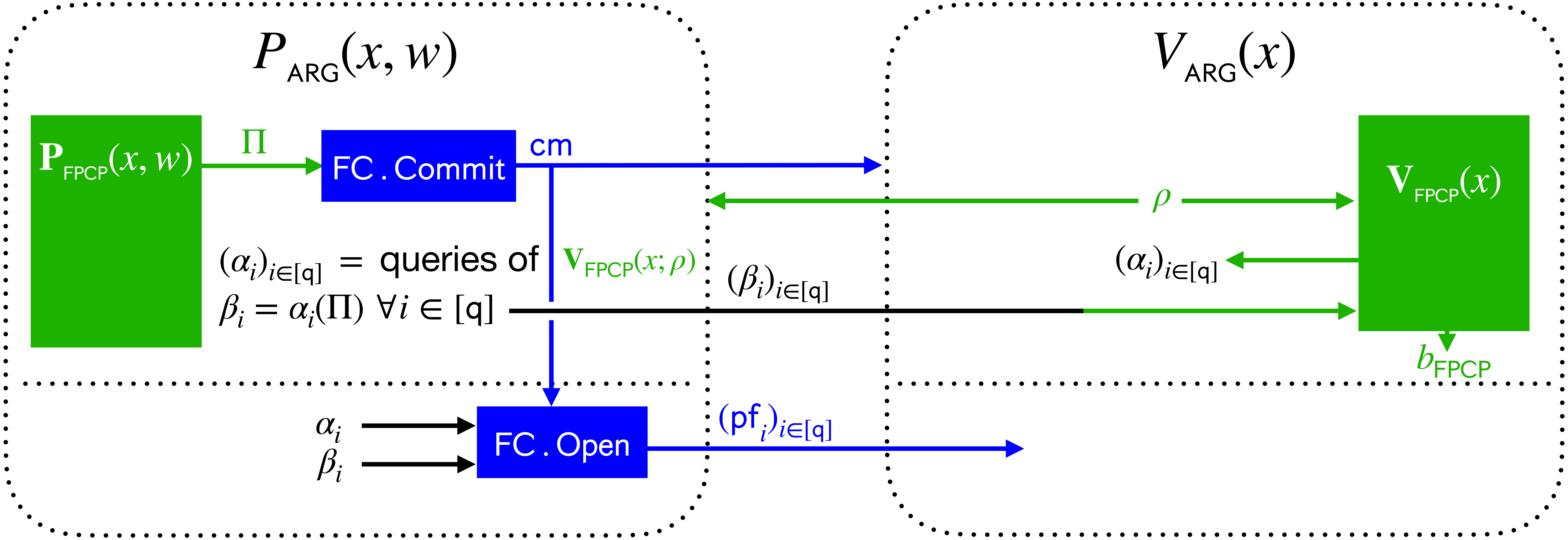
The Funky protocol

For FPCPs and non-interactive FCs



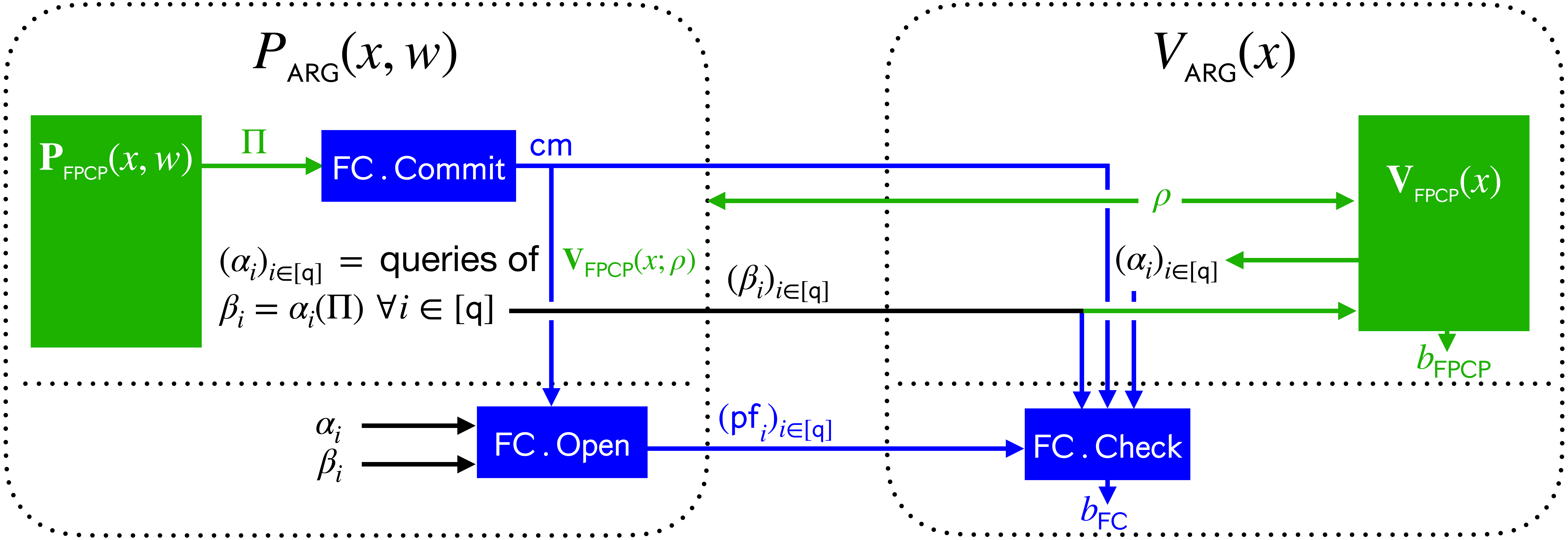
The Funky protocol

For FPCPs and non-interactive FCs



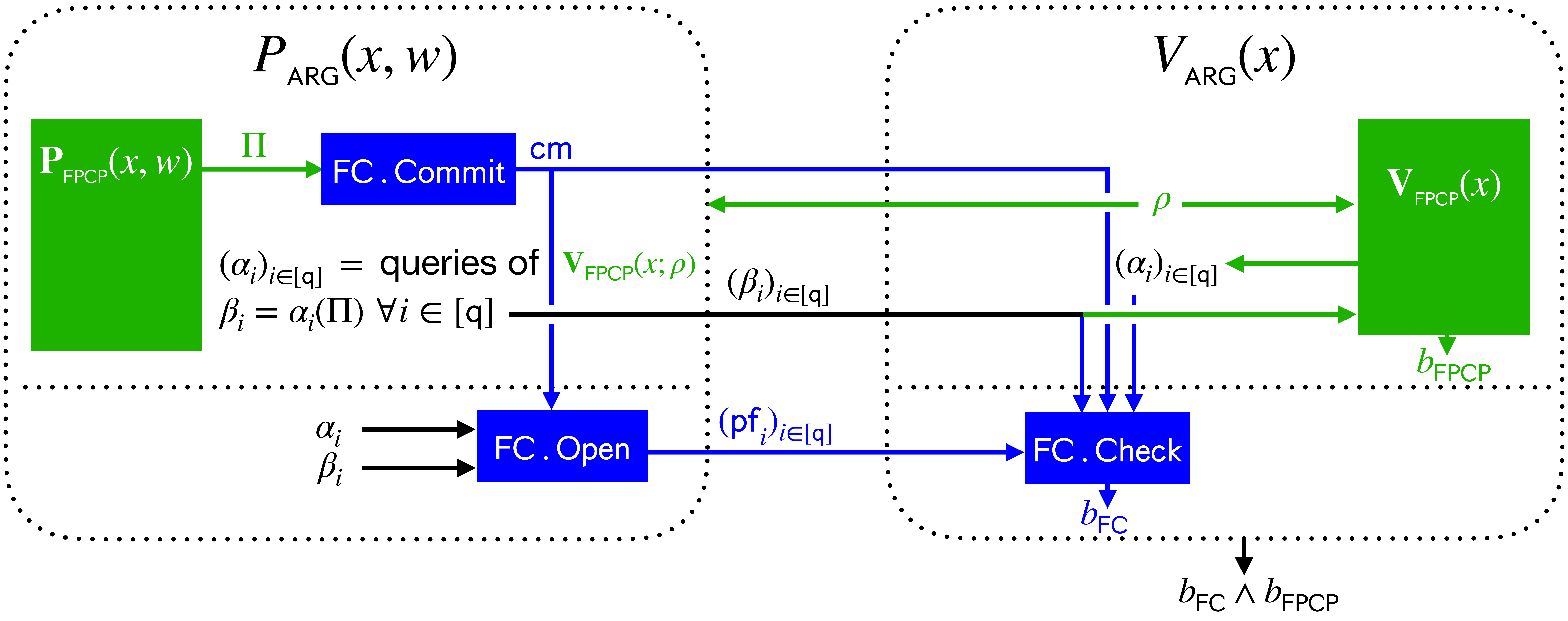
The Funky protocol

For FPCPs and non-interactive FCs



The Funky protocol

For FPCPs and non-interactive FCs



Security reduction for Funky[FPCP, FC]

Goal:

(for FPCPs and non-interactive FCs)

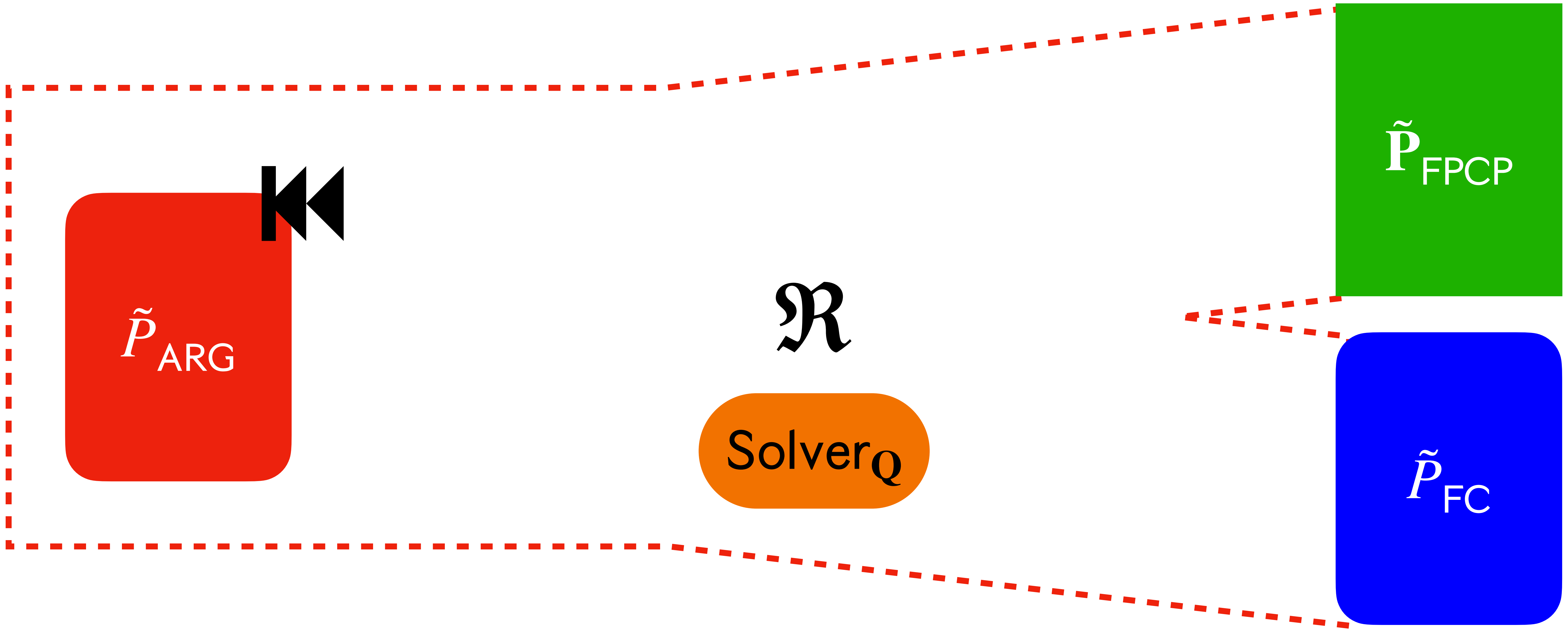
$$\epsilon_{\mathsf{ARG}}(\lambda, \ell, \mathsf{q}, t) \leq \epsilon_{\mathsf{FPCP}}(\ell, \mathsf{q}) + \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t \cdot \mathsf{N} + t_{\mathbf{Q}}) + \epsilon_{\mathbf{Q}}(\ell, \mathsf{N})$$

Security reduction for Funky[FPCP, FC]

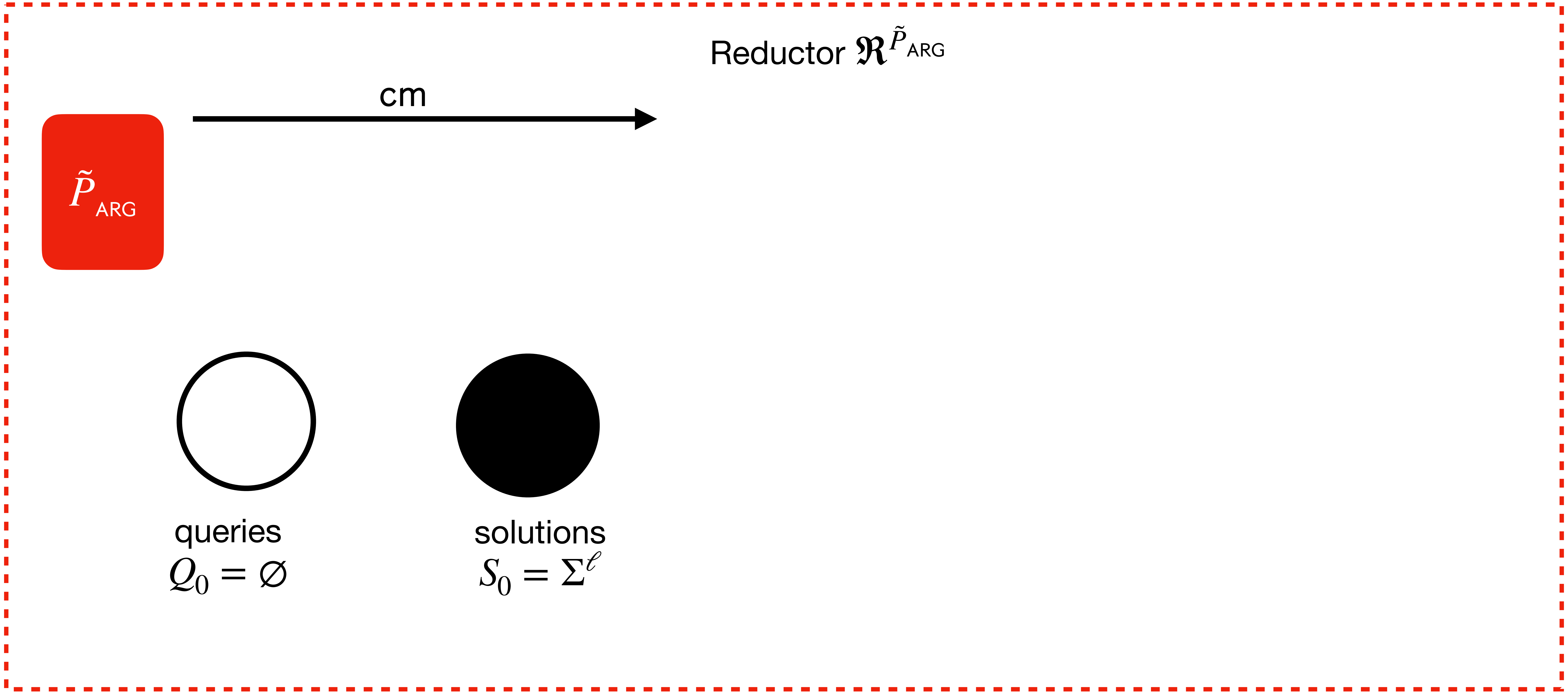
Goal:

(for FPCPs and non-interactive FCs)

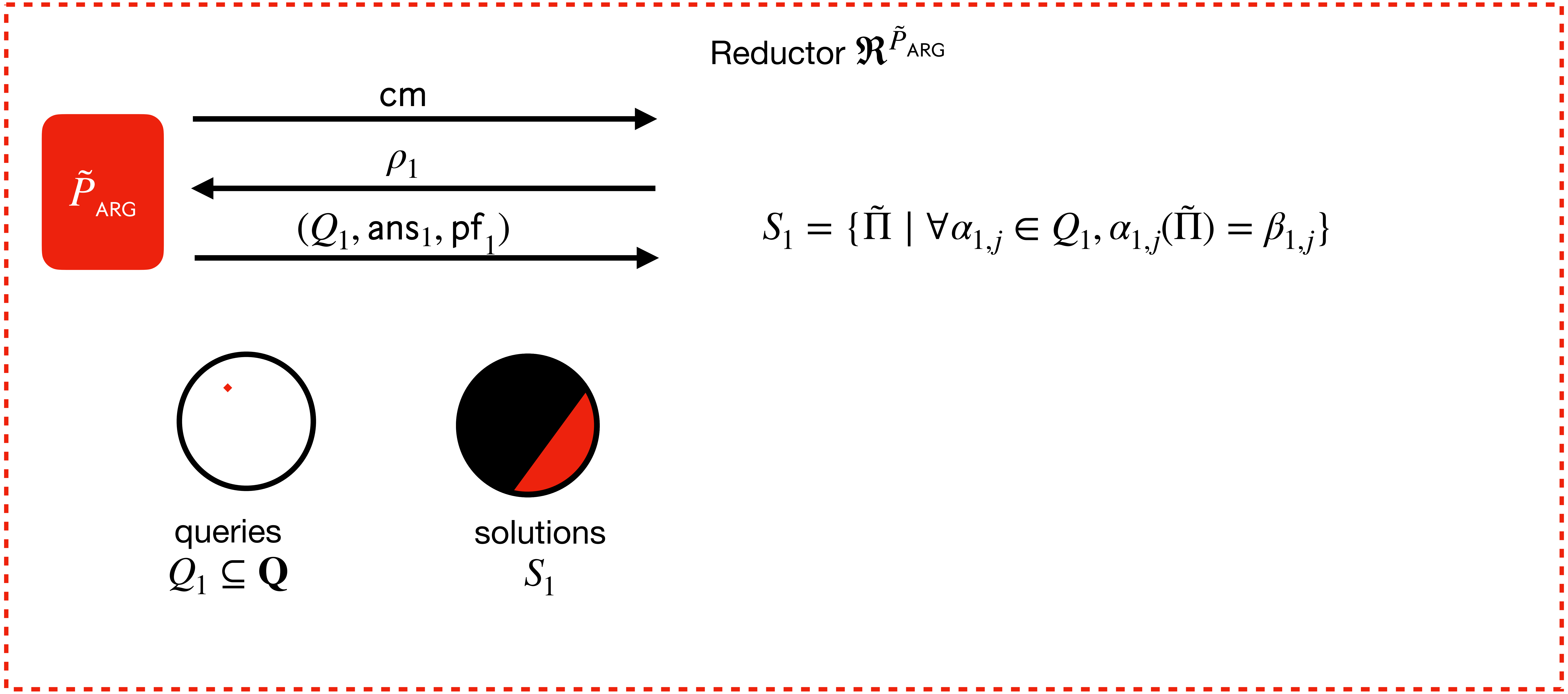
$$\epsilon_{\mathsf{ARG}}(\lambda, \ell, \mathsf{q}, t) \leq \epsilon_{\mathsf{FPCP}}(\ell, \mathsf{q}) + \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t \cdot \mathsf{N} + t_{\mathbf{Q}}) + \epsilon_{\mathbf{Q}}(\ell, \mathsf{N})$$



Security reduction for Funky[FPCP, FC]

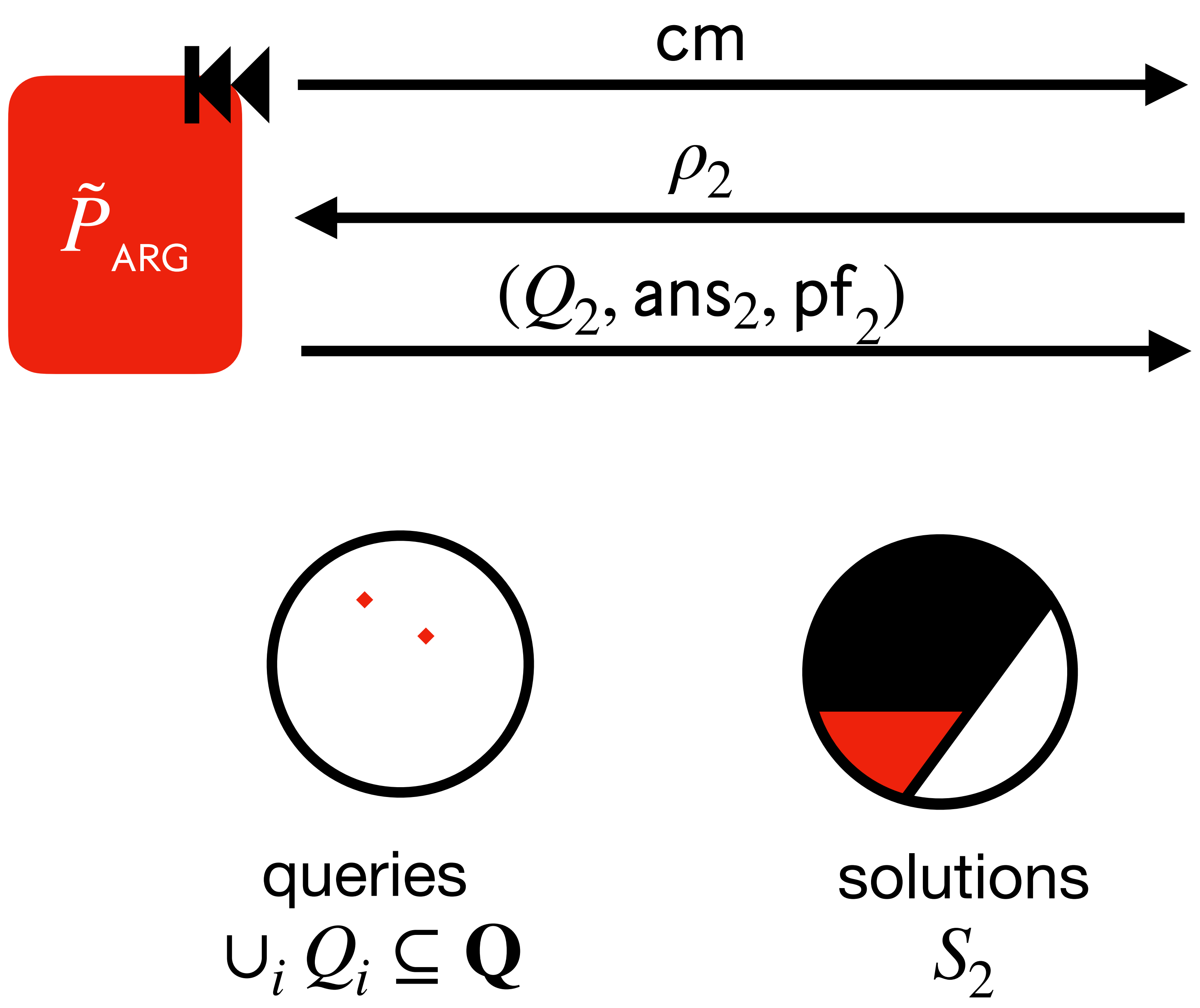


Security reduction for Funky[FPCP, FC]



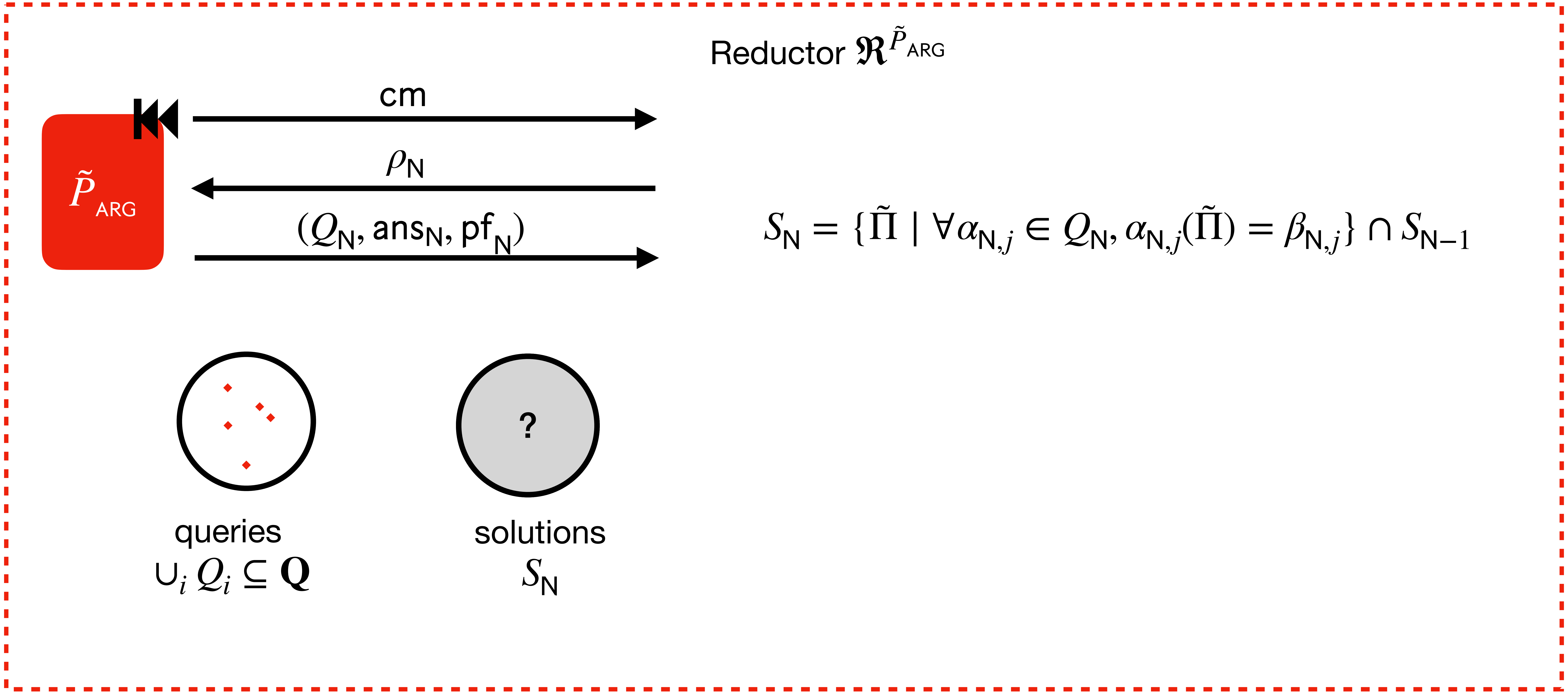
Security reduction for Funky[FPCP, FC]

Reductor $\mathfrak{R}^{\tilde{P}_{\text{ARG}}}$



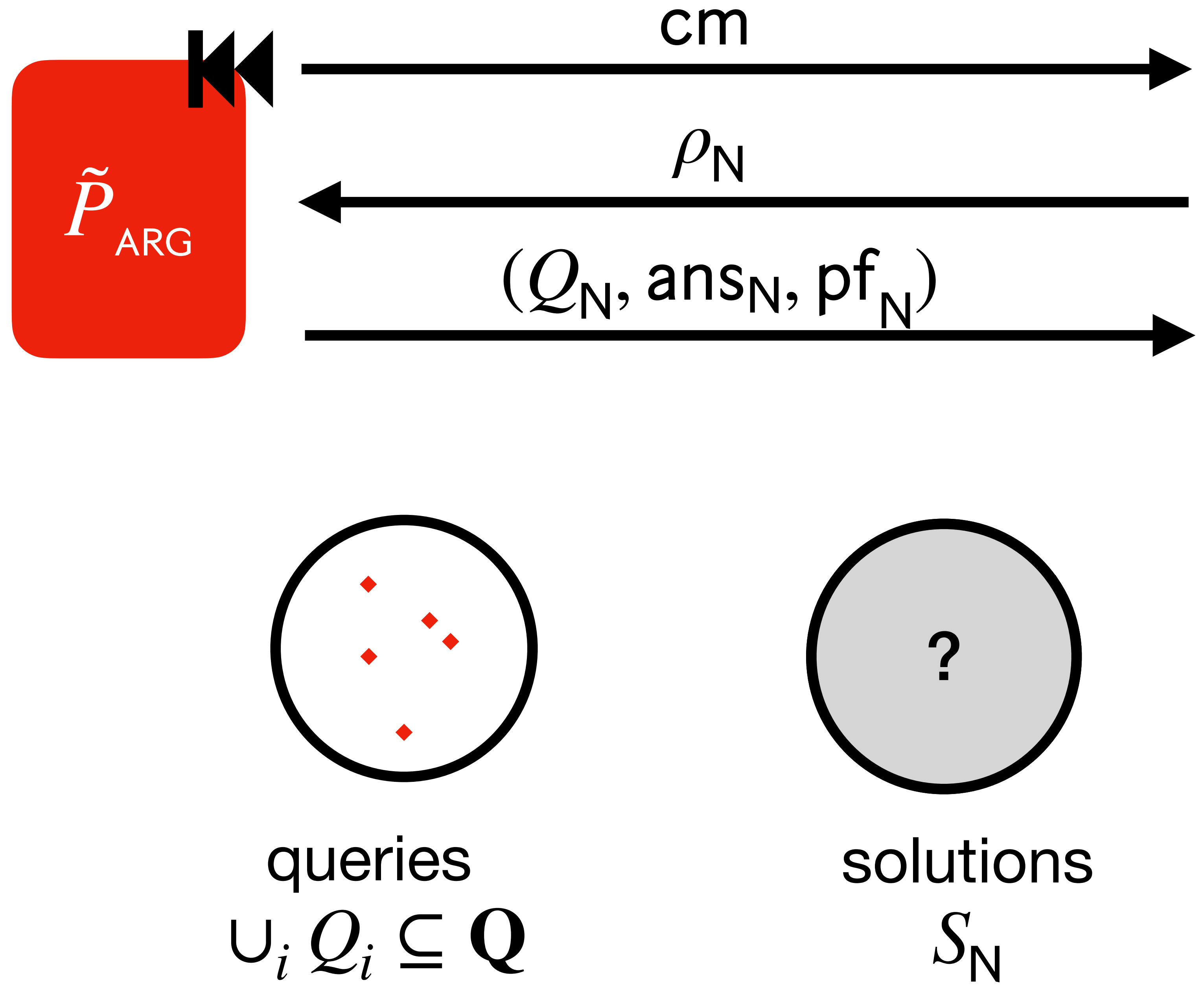
$$S_2 = \{\tilde{\Pi} \mid \forall \alpha_{2,j} \in Q_2, \alpha_{2,j}(\tilde{\Pi}) = \beta_{2,j}\} \cap S_1$$

Security reduction for Funky[FPCP, FC]



Security reduction for Funky[FPCP, FC]

Reductor $\mathfrak{R}^{\tilde{P}_{\mathsf{ARG}}}$



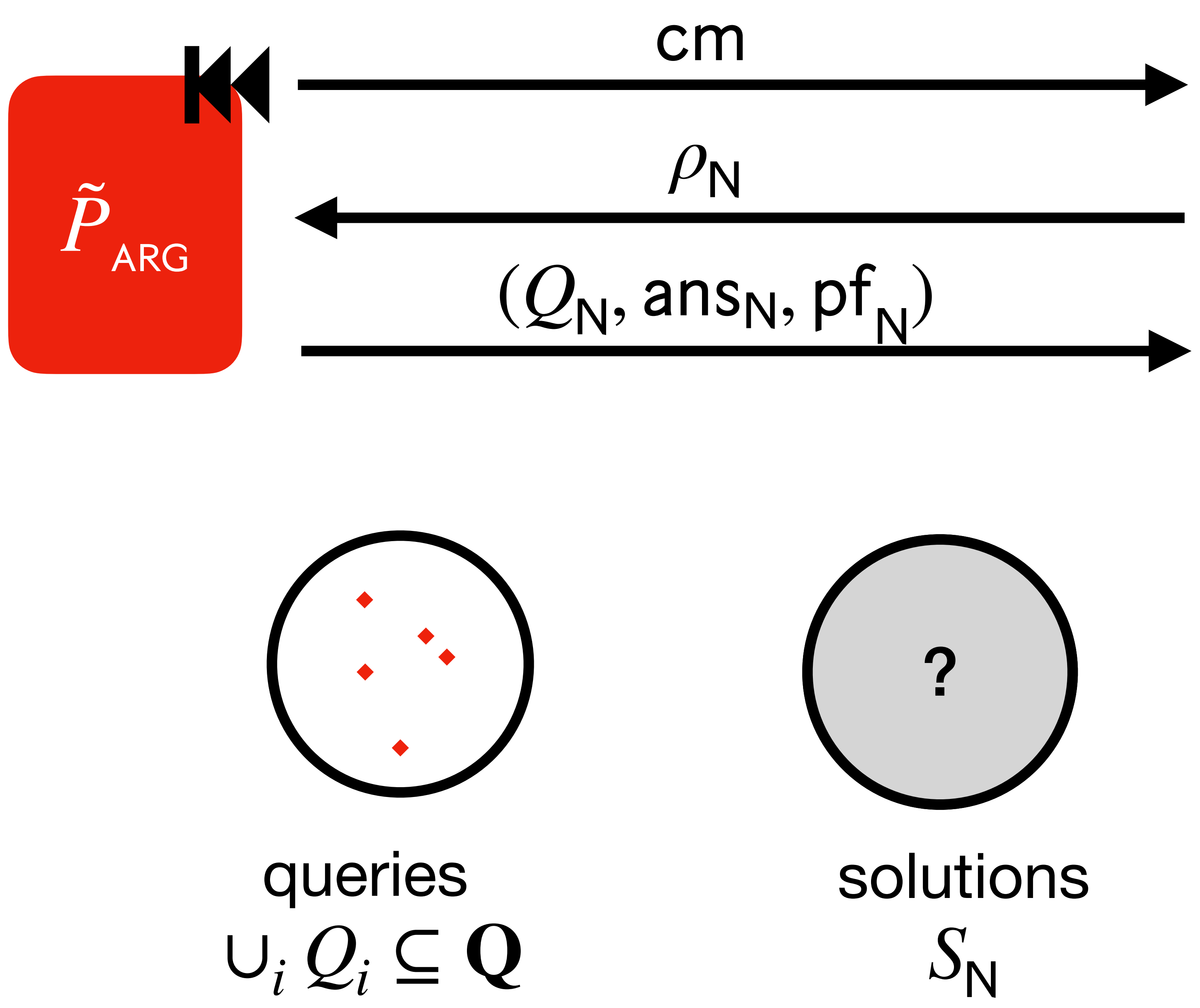
$S_{\mathsf{N}} = \{\tilde{\Pi} \mid \forall \alpha_{\mathsf{N},j} \in Q_{\mathsf{N}}, \alpha_{\mathsf{N},j}(\tilde{\Pi}) = \beta_{\mathsf{N},j}\} \cap S_{\mathsf{N}-1}$

Output $\tilde{\Pi} \leftarrow \mathsf{Solver}_{\mathbf{Q}}((\alpha_{i,j}, \beta_{i,j})_{i\in[\mathsf{N}], j\in[\mathsf{q}]})$

Outputs $\tilde{\Pi} \in S_{\mathsf{N}}$

Security reduction for Funky[FPCP, FC]

Reductor $\mathfrak{R}^{\tilde{P}_{\mathsf{ARG}}}$



$S_{\mathsf{N}} = \{\tilde{\Pi} \mid \forall \alpha_{\mathsf{N},j} \in Q_{\mathsf{N}}, \alpha_{\mathsf{N},j}(\tilde{\Pi}) = \beta_{\mathsf{N},j}\} \cap S_{\mathsf{N}-1}$

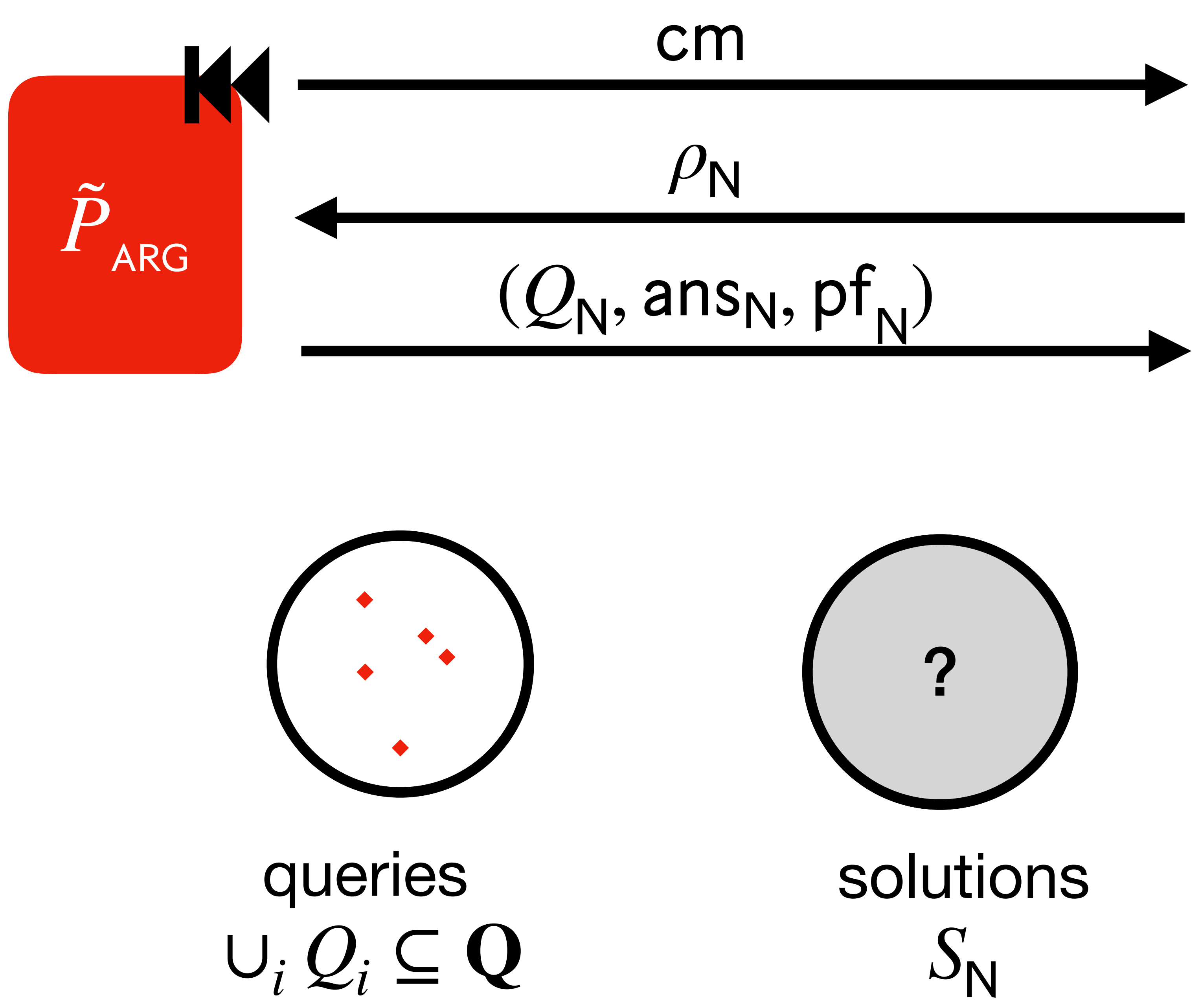
Output $\tilde{\Pi} \leftarrow \mathsf{Solver}_{\mathbf{Q}}((\alpha_{i,j}, \beta_{i,j})_{i \in [\mathsf{N}], j \in [\mathsf{q}]})$

Outputs $\tilde{\Pi} \in S_{\mathsf{N}}$

$S_{\mathsf{N}+1} \neq \emptyset \wedge S_{\mathsf{N}+1} = S_{\mathsf{N}}$ (some solutions)
→ FPCP verifier accepts $\tilde{\Pi}$

Security reduction for Funky[FPCP, FC]

Reductor $\mathfrak{R}^{\tilde{P}_{\mathsf{ARG}}}$



$S_{\mathsf{N}} = \{\tilde{\Pi} \mid \forall \alpha_{\mathsf{N},j} \in Q_{\mathsf{N}}, \alpha_{\mathsf{N},j}(\tilde{\Pi}) = \beta_{\mathsf{N},j}\} \cap S_{\mathsf{N}-1}$

Output $\tilde{\Pi} \leftarrow \mathsf{Solver}_{\mathbf{Q}}((\alpha_{i,j}, \beta_{i,j})_{i \in [\mathsf{N}], j \in [\mathsf{q}]})$

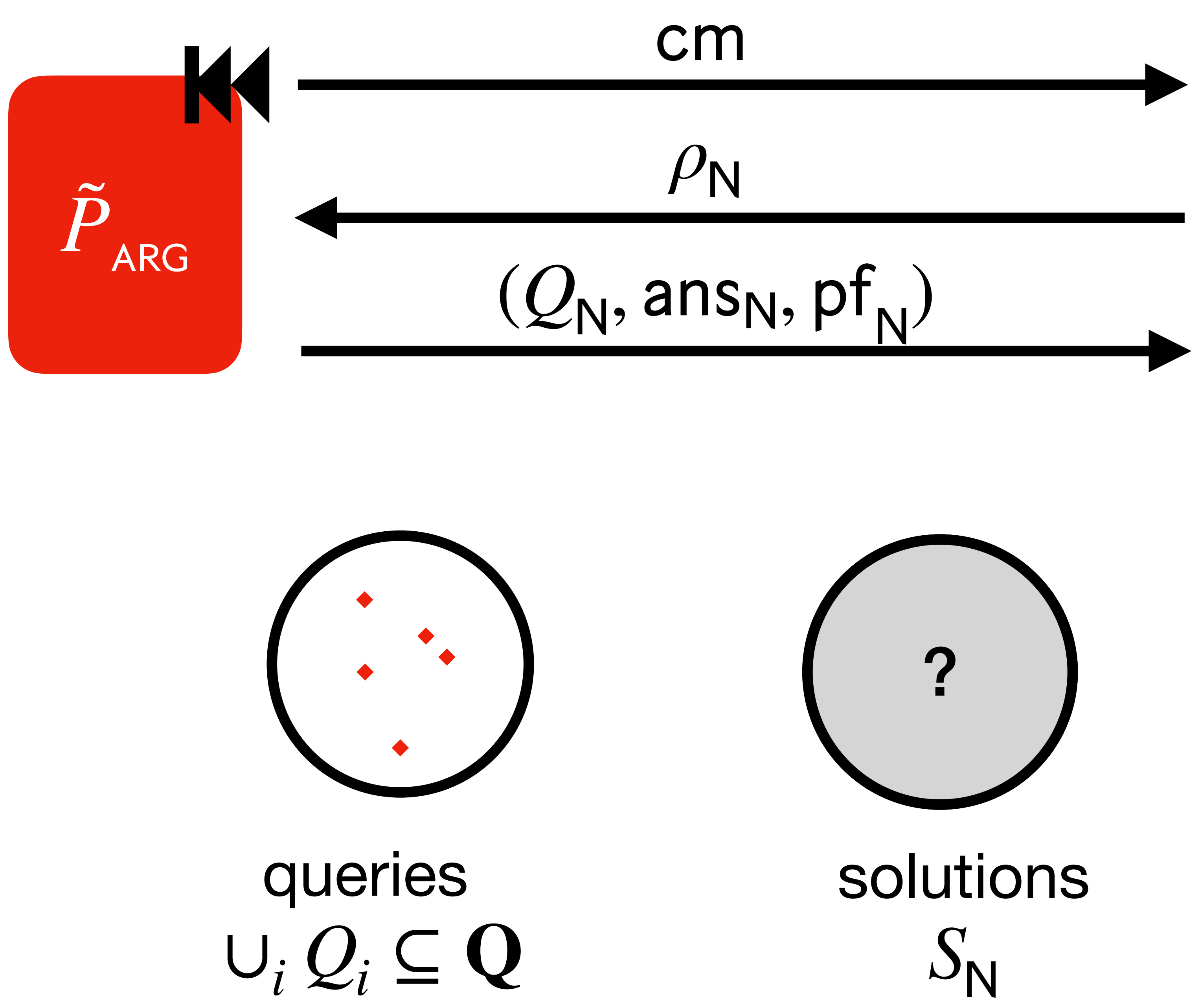
Outputs $\tilde{\Pi} \in S_{\mathsf{N}}$

$S_{\mathsf{N}+1} \neq \emptyset \wedge S_{\mathsf{N}+1} = S_{\mathsf{N}}$ (some solutions)
→ FPCP verifier accepts $\tilde{\Pi}$

$S_{\mathsf{N}+1} = \emptyset$ (no solution)
→ FC verifier accepts all FC proofs

Security reduction for Funky[FPCP, FC]

Reductor $\mathfrak{R}^{\tilde{P}_{\text{ARG}}}$



$S_{\text{N}} = \{\tilde{\Pi} \mid \forall \alpha_{\text{N},j} \in Q_{\text{N}}, \alpha_{\text{N},j}(\tilde{\Pi}) = \beta_{\text{N},j}\} \cap S_{\text{N}-1}$

Output $\tilde{\Pi} \leftarrow \text{Solver}_{\mathbf{Q}}((\alpha_{i,j}, \beta_{i,j})_{i\in[\text{N}],j\in[\text{q}]})$

Outputs $\tilde{\Pi} \in S_{\text{N}}$

$S_{\text{N}+1} \neq \varnothing \wedge S_{\text{N}+1} = S_{\text{N}}$ (some solutions)
→ FPCP verifier accepts $\tilde{\Pi}$

$S_{\text{N}+1} = \varnothing$ (no solution)
→ FC verifier accepts all FC proofs

$S_{\text{N}+1} \neq \varnothing \wedge S_{\text{N}+1} \neq S_{\text{N}}$
→ *tail error* of the query class $\mathbf{Q}$

Security reduction for Funky[FPCP, FC]

$\forall \tilde{P} : \forall x \notin L(R)$:

$\Pr\left[\langle \tilde{P}, V(x) \rangle = 1\right]$

Security reduction for Funky[FPCP, FC]

$\forall \tilde{P} : \forall x \notin L(R)$:

Produced by the reductor $\mathfrak{R}^{\tilde{P}_{\text{ARG}}}$

$$\Pr\left[\langle \tilde{P}, V(x)\rangle = 1\right] \le \Pr\left[\begin{array}{l}\text{Sample } \rho \\ \text{FPCP verifier accepts: } \mathbf{V}^{\tilde{\Pi}}(x;\rho) = 1 \\ \text{ARG verifier accepts: } V(x;\rho;Q,\text{ans},\text{pf})\rangle = 1\end{array}\right]$$

Produced by a t_{ARG}-time adversary $\tilde{P}_{\text{ARG}}$ given ρ

$$+ \Pr\left[\begin{array}{l}\text{Sample } \rho \\ \text{FPCP verifier rejects: } \mathbf{V}^{\tilde{\Pi}}(x;\rho) \neq 1 \\ \text{ARG verifier accepts: } V(x;\rho;Q,\text{ans},\text{pf})\rangle = 1\end{array}\right]$$

Security reduction for Funky[FPCP, FC]

$\forall \tilde{P} : \forall x \notin L(R)$:

$$\Pr\left[\langle \tilde{P}, V(x)\rangle = 1\right] \leq \Pr\begin{bmatrix}\text{Sample } \rho \\ \text{FPCP verifier accepts: } \mathbf{V}^{\tilde{\Pi}}(x;\rho) = 1 \\ \text{ARG verifier accepts: } V(x;\rho;Q,\text{ans},\text{pf}))\rangle = 1\end{bmatrix}$$

FPCP soundness

$\leq \epsilon_{\text{FPCP}}(\ell, \text{q})$

$$+ \Pr\begin{bmatrix}\text{Sample } \rho \\ \text{FPCP verifier rejects: } \mathbf{V}^{\tilde{\Pi}}(x;\rho) \neq 1 \\ \text{ARG verifier accepts: } V(x;\rho;Q,\text{ans},\text{pf}))\rangle = 1\end{bmatrix}$$

Security reduction for Funky[FPCP, FC]

$\forall \tilde{P} : \forall x \notin L(R)$:

$$\Pr\left[\langle \tilde{P}, V(x)\rangle = 1\right] \le \Pr\left[\begin{array}{l}\text{Sample } \rho \\ \text{FPCP verifier accepts: } \mathbf{V}^{\tilde{\Pi}}(x;\rho) = 1 \\ \text{ARG verifier accepts: } V(x;\rho;Q,\text{ans},\text{pf})\rangle = 1\end{array}\right]$$

FPCP soundness
$\le \epsilon_{\mathsf{FPCP}}(\ell, \mathsf{q})$

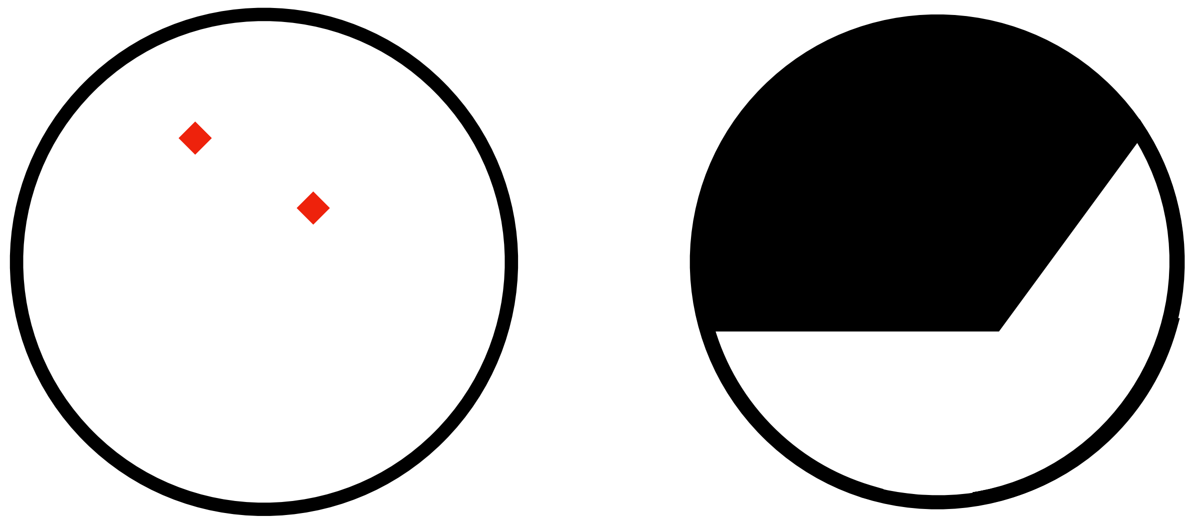
$$+ \Pr\left[\begin{array}{l}\text{Sample } \rho \\ \text{FPCP verifier rejects: } \mathbf{V}^{\tilde{\Pi}}(x;\rho) \neq 1 \\ \text{ARG verifier accepts: } V(x;\rho;Q,\text{ans},\text{pf})\rangle = 1\end{array}\right]$$

Security reduction lemma
$\le \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t_{\mathsf{FC}}) + \epsilon_{\mathbf{Q}}(\ell, \mathsf{q}, \mathsf{N})$

Security reduction lemma

queries $\cup_i Q_i$ solutions S_i

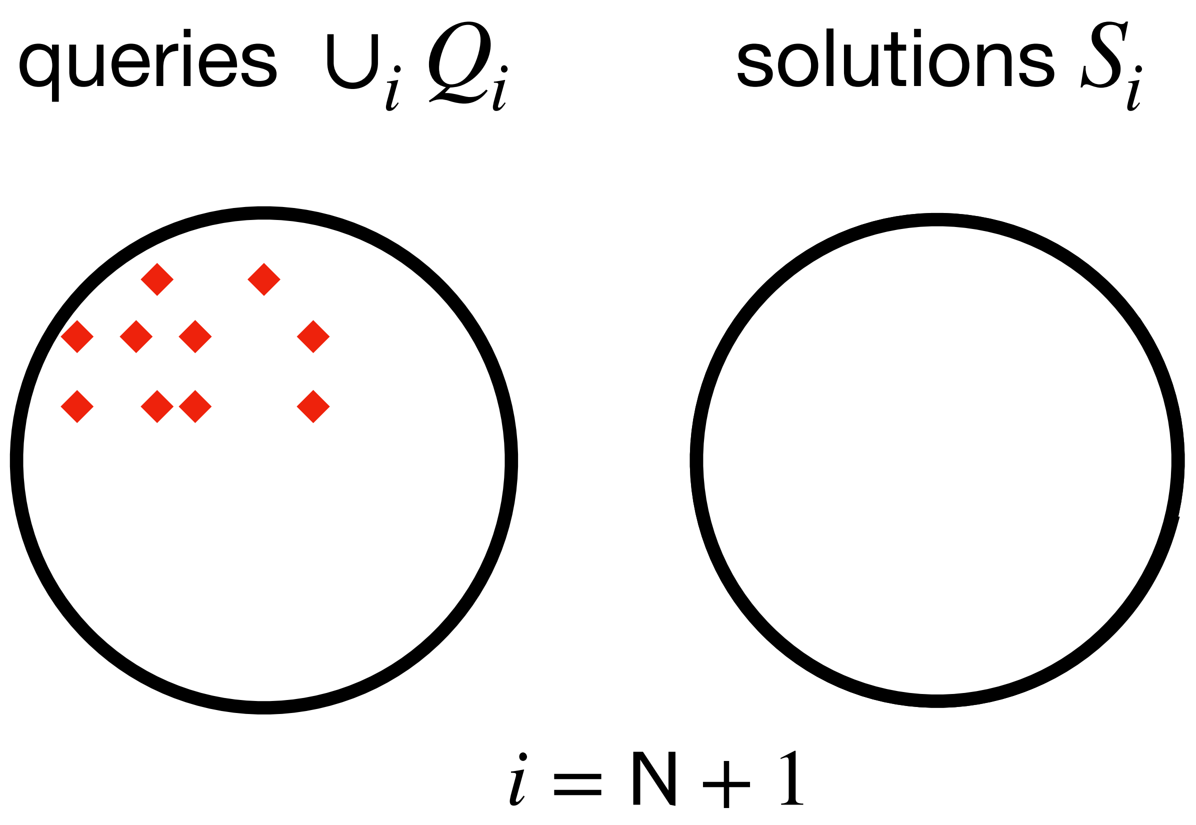
Case 1:



$i = 2$

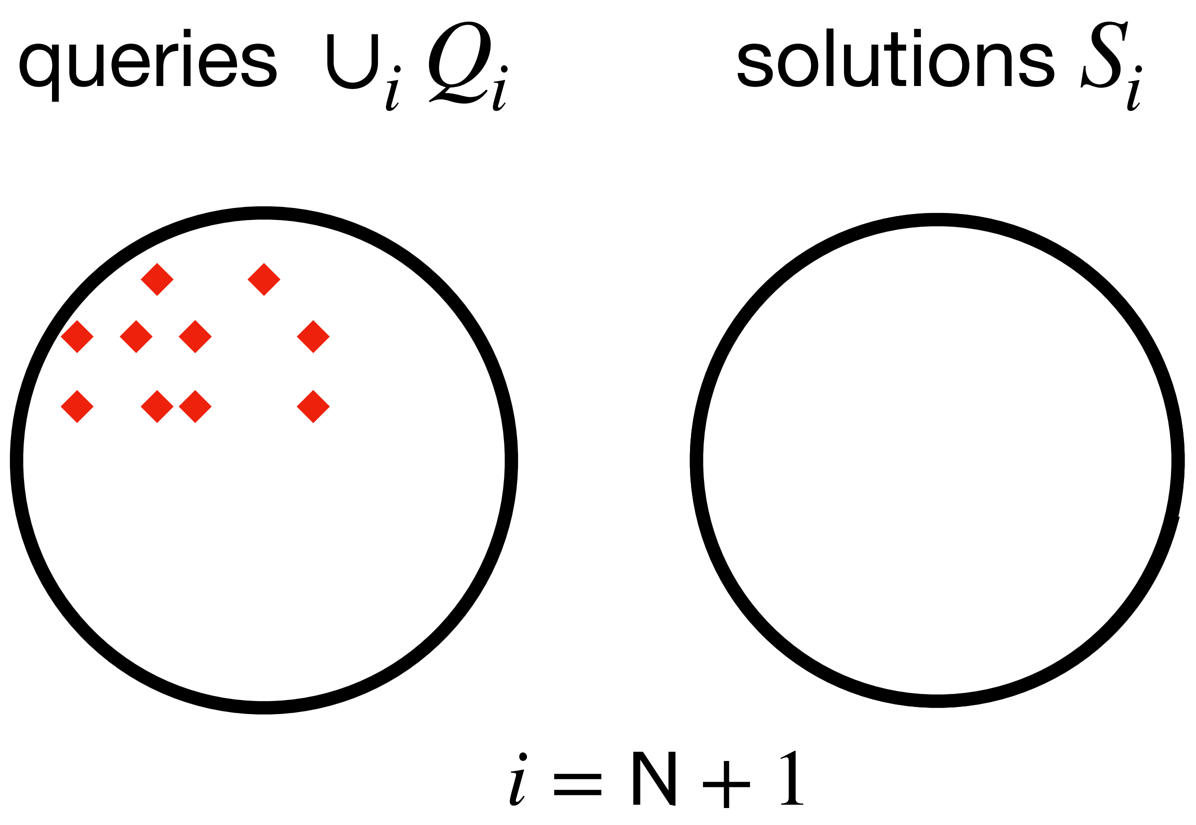
Security reduction lemma

Case 1:



Security reduction lemma

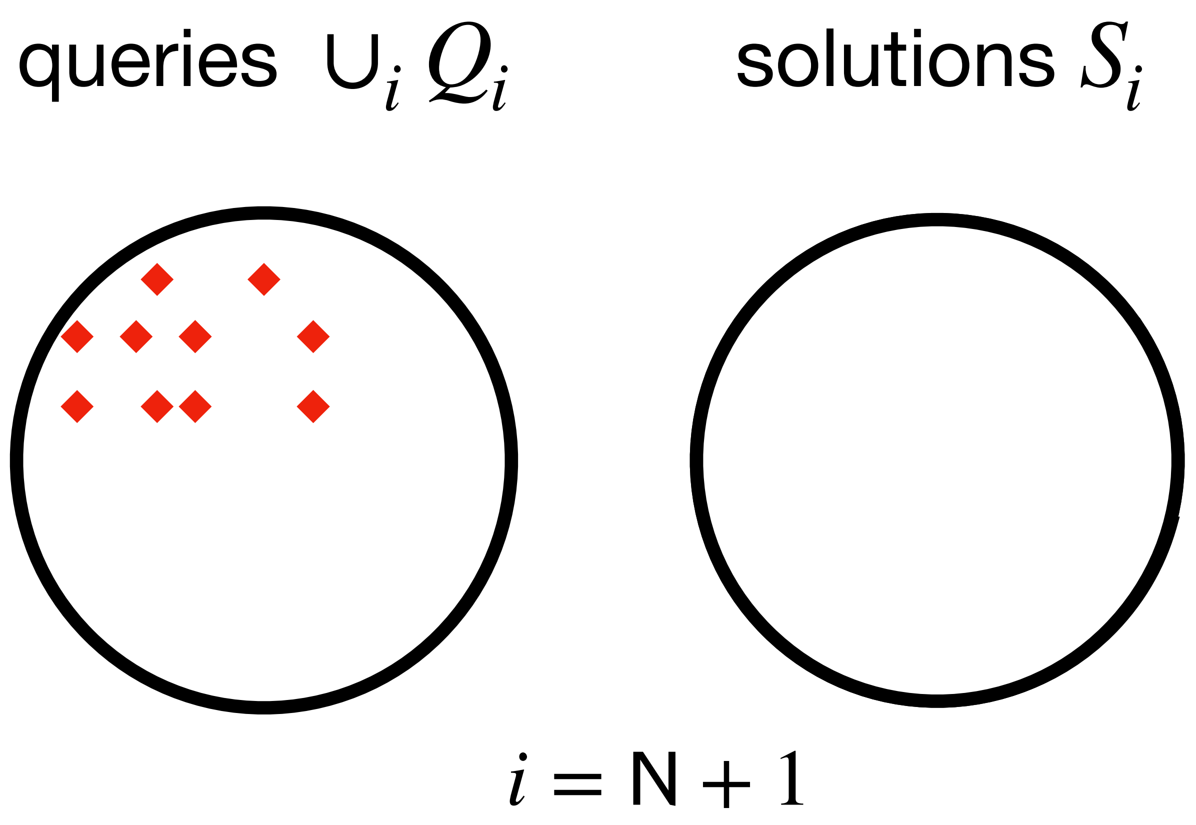
Case 1:



$$\left[\begin{array}{l} \nexists \tilde{\Pi} : \forall \alpha \in \cup_i Q_i : \alpha(\tilde{\Pi}) = \mathsf{ans}[\alpha] \\ \text{Yet all FC checks pass} \end{array}\right]$$

Security reduction lemma

Case 1:



$$\Pr\left[\begin{array}{l}\nexists \tilde{\Pi} : \forall \alpha \in \cup_i Q_i : \alpha(\tilde{\Pi}) = \mathsf{ans}[\alpha] \\ \text{Yet all FC checks pass}\end{array}\right] \leq \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t_{\mathsf{FC}})$$

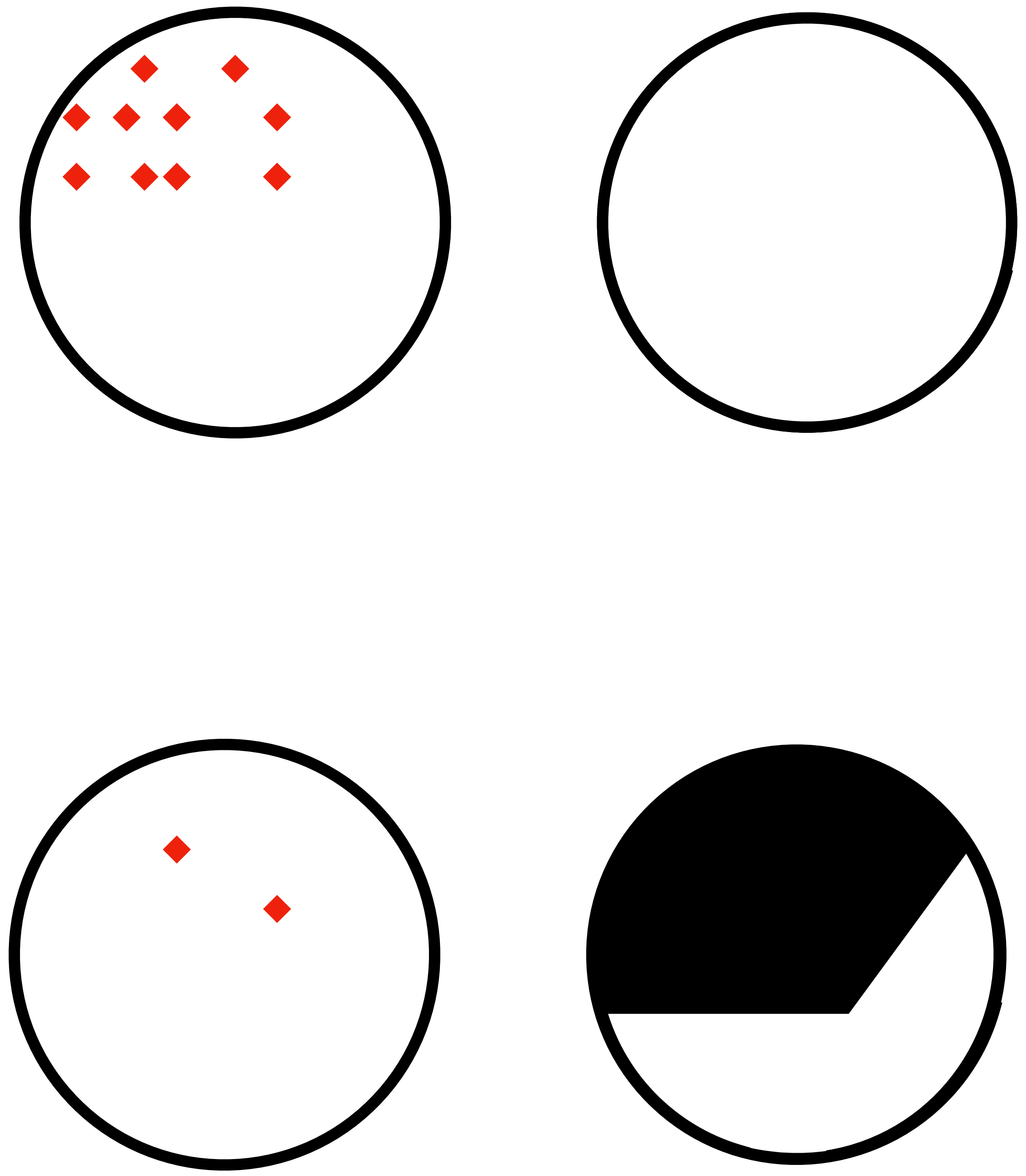
Function binding

Security reduction lemma

queries $\cup_i Q_i$ solutions S_i

Case 1:

Case 2:



$i = 2$

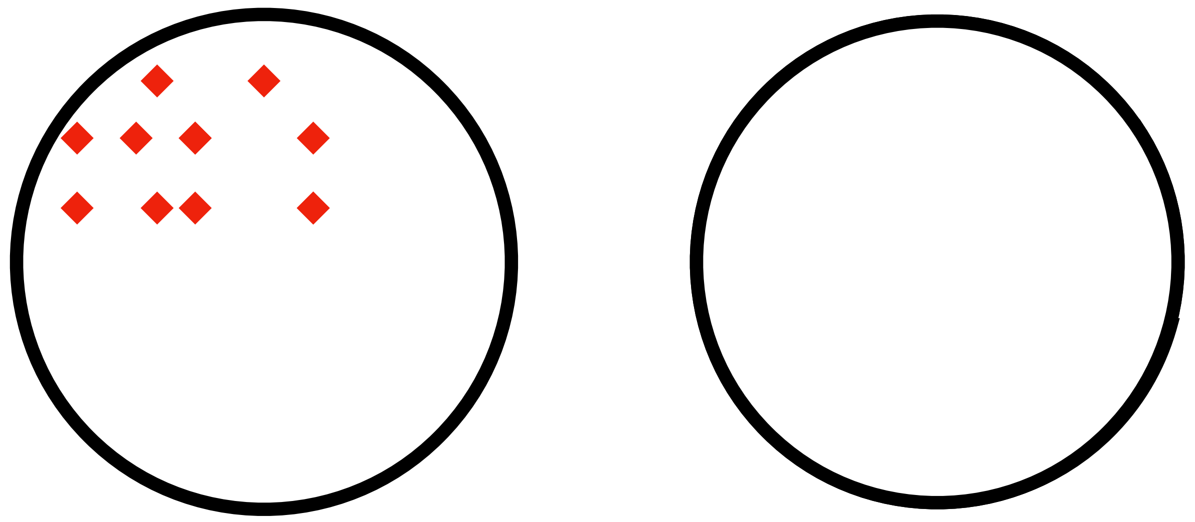
$$\Pr\left[\begin{array}{l}\nexists \tilde{\Pi} : \forall \alpha \in \cup_i Q_i : \alpha(\tilde{\Pi}) = \mathsf{ans}[\alpha] \\ \text{Yet all FC checks pass}\end{array}\right] \leq \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t_{\mathsf{FC}})$$

Function binding

Security reduction lemma

queries $\cup_i Q_i$ solutions S_i

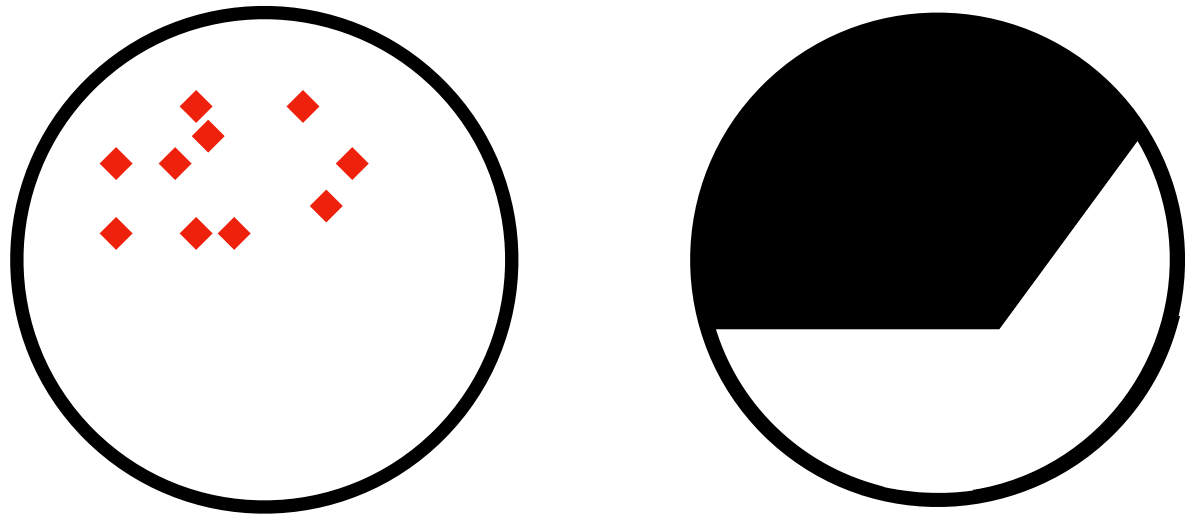
Case 1:



$$\Pr\left[\begin{array}{l}\nexists \tilde{\Pi} : \forall \alpha \in \cup_i Q_i : \alpha(\tilde{\Pi}) = \mathsf{ans}[\alpha] \\ \text{Yet all FC checks pass}\end{array}\right] \leq \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t_{\mathsf{FC}})$$

Function binding

Case 2:

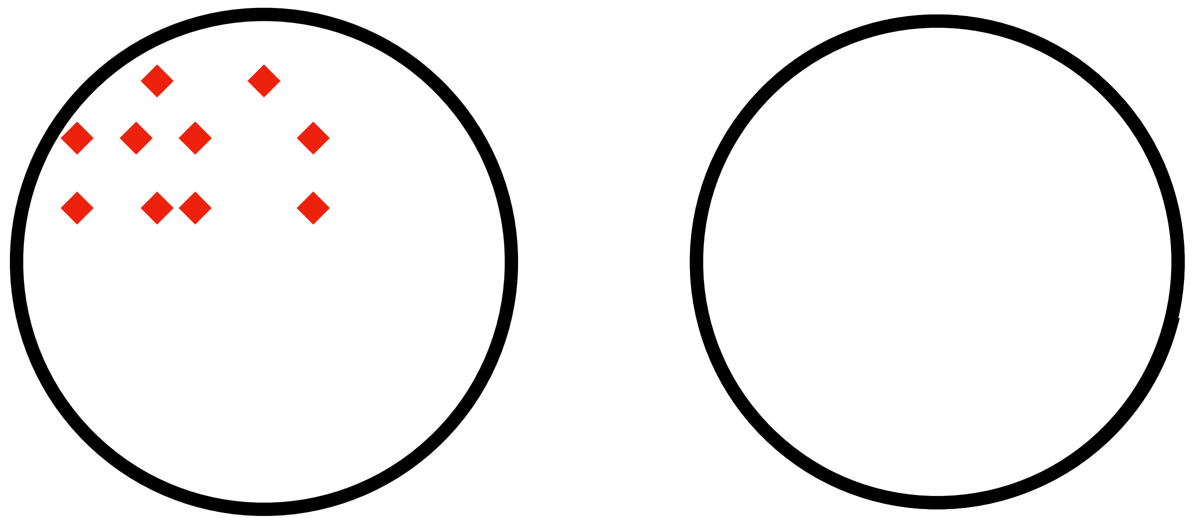


$i = \mathsf{N}$

Security reduction lemma

queries $\cup_i Q_i$ solutions S_i

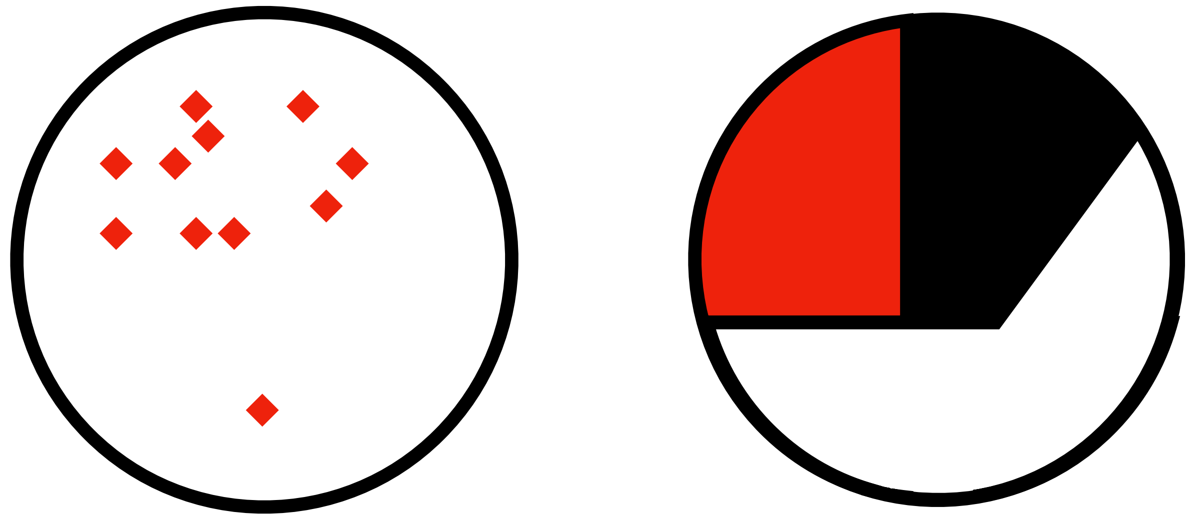
Case 1:



$$\Pr\left[\begin{array}{l}\nexists\tilde{\Pi} : \forall\alpha \in \cup_i Q_i : \alpha(\tilde{\Pi}) = \mathsf{ans}[\alpha] \\ \text{Yet all FC checks pass}\end{array}\right] \leq \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t_{\mathsf{FC}})$$

Function binding

Case 2:



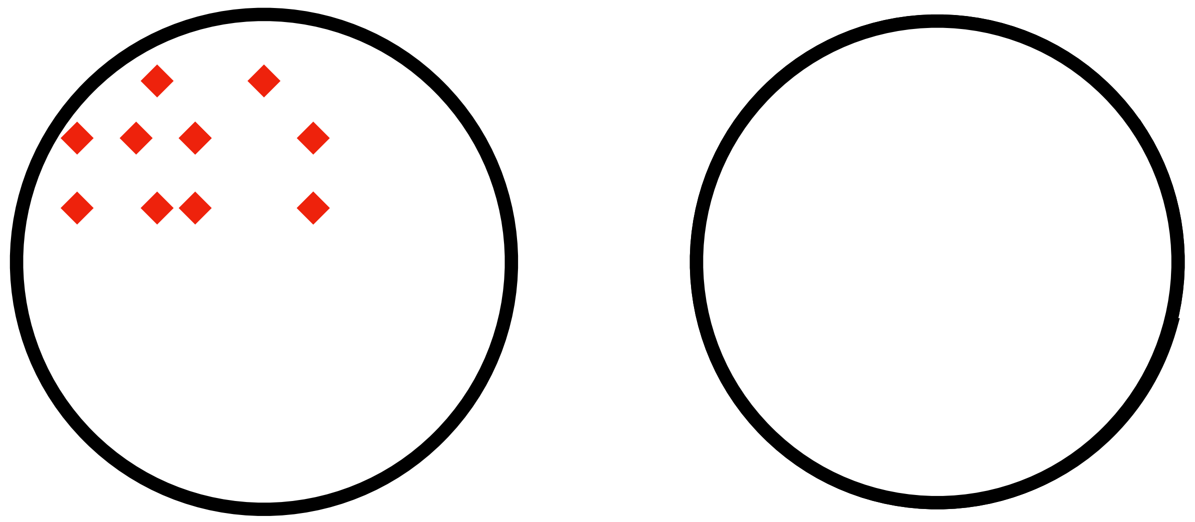
$i = \mathsf{N} + 1$

$$[S_{\mathsf{N}+1} \neq \emptyset \wedge S_{\mathsf{N}+1} \neq S_{\mathsf{N}}]$$

Security reduction lemma

queries $\cup_i Q_i$ solutions S_i

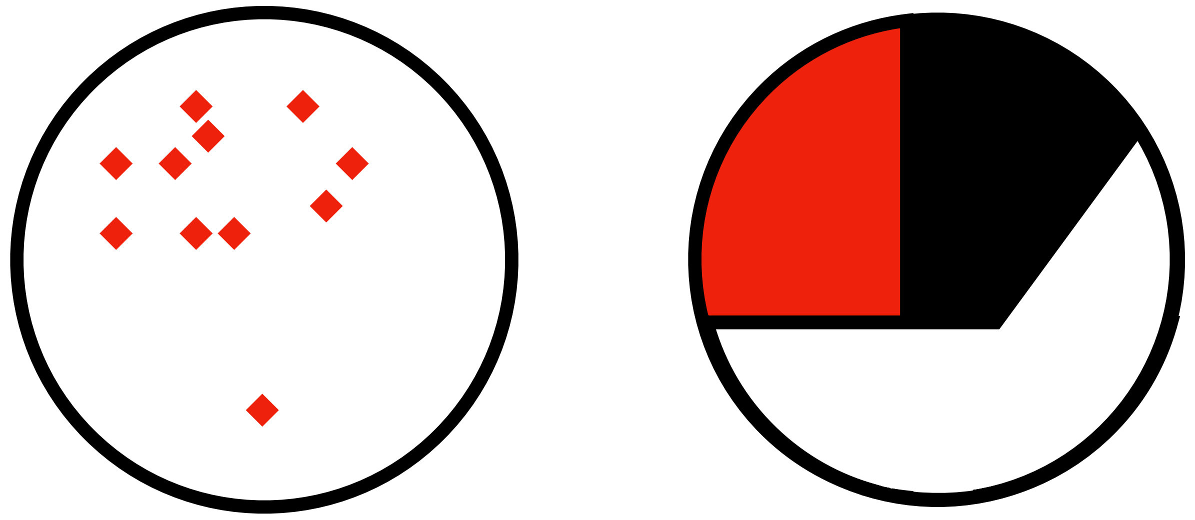
Case 1:



$$\Pr\left[\begin{array}{l} \nexists \tilde{\Pi} : \forall \alpha \in \cup_i Q_i : \alpha(\tilde{\Pi}) = \mathsf{ans}[\alpha] \\ \text{Yet all FC checks pass} \end{array}\right] \le \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t_{\mathsf{FC}})$$

Function binding

Case 2:



$i = \mathsf{N} + 1$

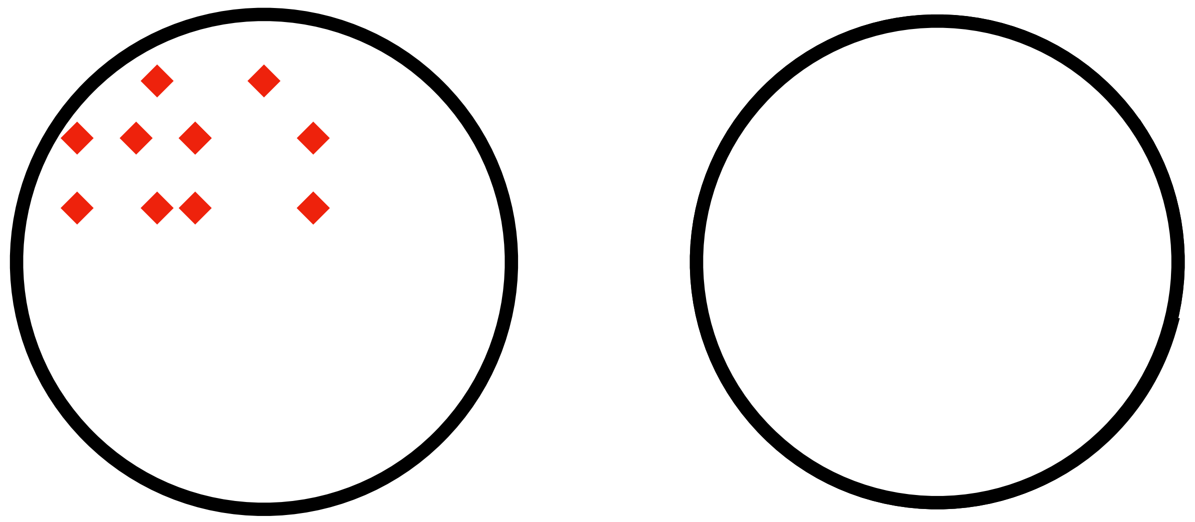
$$\Pr\left[S_{\mathsf{N}+1} \neq \emptyset \wedge S_{\mathsf{N}+1} \neq S_{\mathsf{N}}\right] \le \epsilon_{\mathbf{Q}}(\ell, \mathsf{N})$$

Tail error

Security reduction lemma

queries $\cup_i Q_i$ solutions S_i

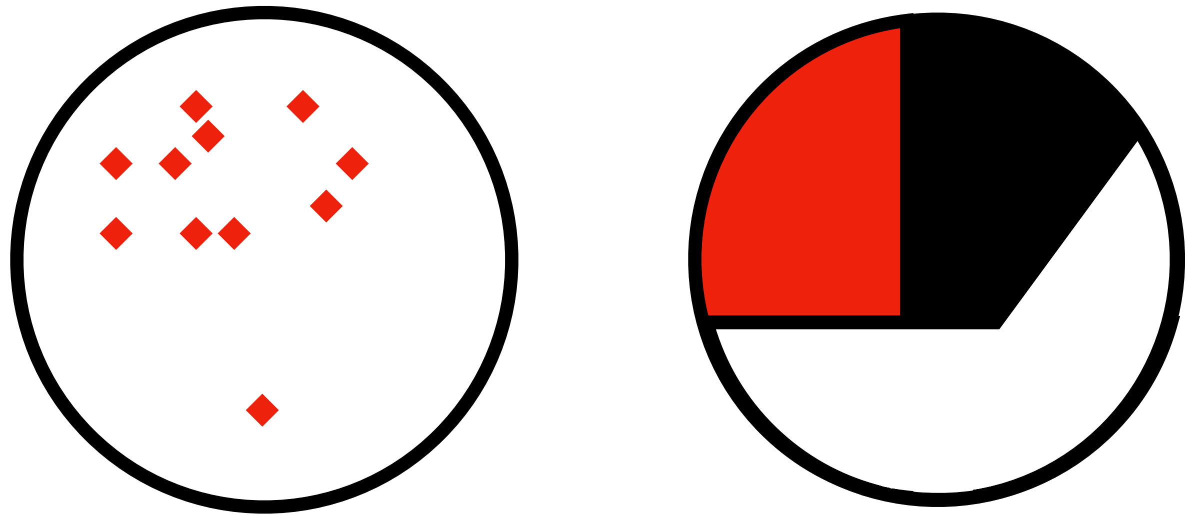
Case 1:



$$\Pr\left[\begin{array}{l}\nexists \tilde{\Pi} : \forall \alpha \in \cup_i Q_i : \alpha(\tilde{\Pi}) = \mathsf{ans}[\alpha] \\ \text{Yet all FC checks pass}\end{array}\right] \leq \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t_{\mathsf{FC}})$$

Function binding

Case 2:



$i = \mathsf{N} + 1$

depends on query class, but independent of FC!

$$\Pr\left[S_{\mathsf{N}+1} \neq \emptyset \wedge S_{\mathsf{N}+1} \neq S_{\mathsf{N}}\right] \leq \epsilon_{\mathbf{Q}}(\ell, \mathsf{N})$$

Tail error

Security reduction for Funky[FPCP, FC]

$$\epsilon_{\mathsf{ARG}}(\lambda, \ell, \mathsf{q}, t) \leq \epsilon_{\mathsf{FPCP}}(\ell, \mathsf{q}) + \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t \cdot \mathsf{N} + t_{\mathbf{Q}}) + \epsilon_{\mathbf{Q}}(\ell, \mathsf{N})$$

Security reduction for Funky[FPCP, FC]

$\leq \ell / \mathsf{N}$
for $\mathbf{Q}$ of interest

$$\epsilon_{\mathsf{ARG}}(\lambda, \ell, \mathsf{q}, t) \leq \epsilon_{\mathsf{FPCP}}(\ell, \mathsf{q}) + \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t \cdot \mathsf{N} + t_{\mathbf{Q}}) + \epsilon_{\mathbf{Q}}(\ell, \mathsf{N})$$

Security reduction for Funky[FPCP, FC]

poly$(\ell, \mathsf{q}, \mathsf{N})$
for $\mathbf{Q}$ of interest

$\leq \ell/\mathsf{N}$
for $\mathbf{Q}$ of interest

$$\epsilon_{\mathsf{ARG}}(\lambda, \ell, \mathsf{q}, t) \leq \epsilon_{\mathsf{FPCP}}(\ell, \mathsf{q}) + \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t \cdot \mathsf{N} + t_{\mathbf{Q}}) + \epsilon_{\mathbf{Q}}(\ell, \mathsf{N})$$

Security reduction for Funky[FPCP, FC]

poly$(\ell, \mathsf{q}, \mathsf{N})$ for $\mathbf{Q}$ of interest

$\leq \ell/\mathsf{N}$ for $\mathbf{Q}$ of interest

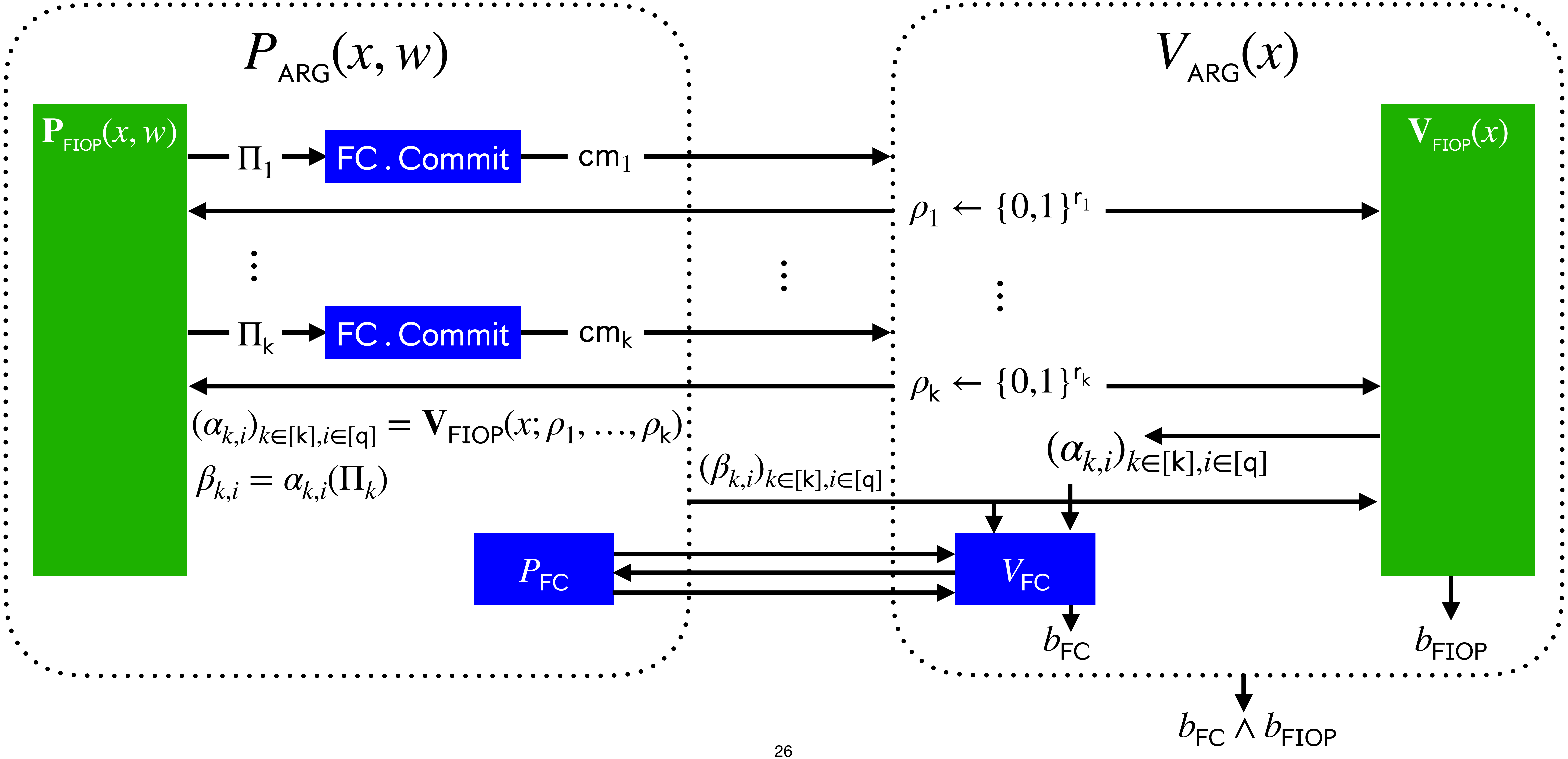
$$\epsilon_{\mathsf{ARG}}(\lambda, \ell, \mathsf{q}, t) \leq \epsilon_{\mathsf{FPCP}}(\ell, \mathsf{q}) + \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t \cdot \mathsf{N} + t_{\mathbf{Q}}) + \epsilon_{\mathbf{Q}}(\ell, \mathsf{N})$$

For $\mathbf{Q}$ of interest: $\leq \epsilon_{\mathsf{FPCP}}(\ell, \mathsf{q}) + \epsilon_{\mathsf{FC}}(\lambda, \ell, \mathsf{q}, t \cdot \ell/\epsilon + t_{\mathbf{Q}}) + \epsilon$

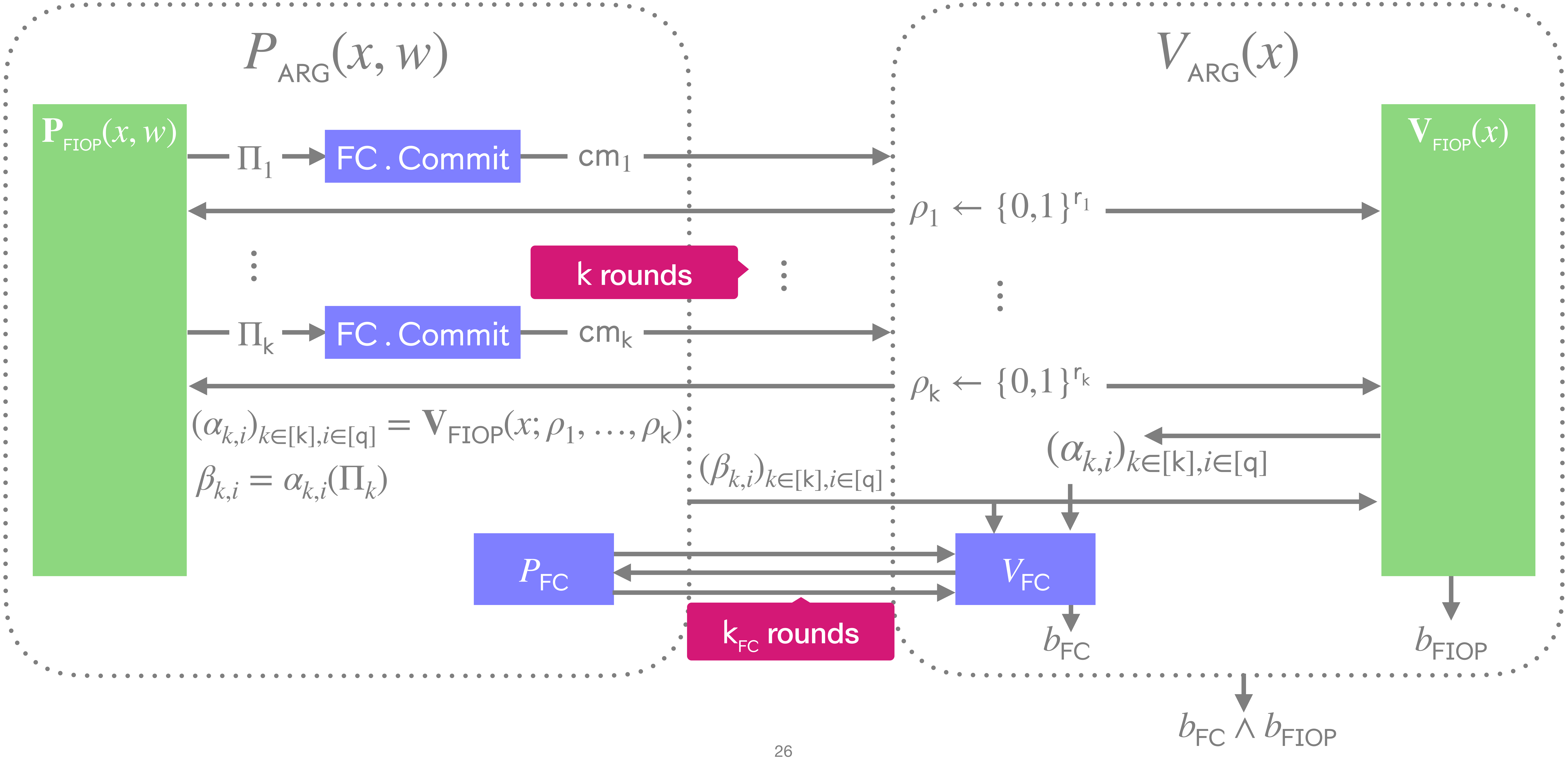
$\mathsf{N} := \ell/\epsilon$

Fiat–Shamir security of Funky in the ROM

The Funky protocol



The Funky protocol



State-restoration security of Funky[FIOP, FC]

State-restoration security of Funky[FIOP, FC]

Analogous for FIOP:

$$\epsilon_{\mathsf{ARG}}(t) \leq \epsilon_{\mathsf{FIOP}} + \epsilon_{\mathsf{FC}}(t \cdot \mathsf{kN} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}$$

State-restoration security of Funky[FIOP, FC]

Analogous for FIOP:

$$\epsilon_{\mathsf{ARG}}(t) \leq \epsilon_{\mathsf{FIOP}} + \epsilon_{\mathsf{FC}}(t \cdot \mathsf{kN} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}$$

RO queries

SR moves

Generically: $\epsilon_{\mathsf{NARG}}(m, t) \leq \epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(m, t)$

State-restoration security of Funky[FIOP, FC]

Analogous for FIOP:

$$\epsilon_{\mathsf{ARG}}(t) \leq \epsilon_{\mathsf{FIOP}} + \epsilon_{\mathsf{FC}}(t \cdot \mathsf{kN} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}$$

RO queries

SR moves

Generically:

$$\epsilon_{\mathsf{NARG}}(m, t) \leq \epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(m, t)$$
$$\leq (\epsilon_{\mathsf{FIOP}} + \epsilon_{\mathsf{FC}}(t \cdot \mathsf{kN} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}) \cdot (m + 1)^{\mathsf{k}}$$

State-restoration security of Funky[FIOP, FC]

Analogous for FIOP:

$$\epsilon_{\mathsf{ARG}}(t) \leq \epsilon_{\mathsf{FIOP}} + \epsilon_{\mathsf{FC}}(t \cdot \mathsf{kN} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}$$

RO queries

SR moves

k might be superconstant!

Generically:
$$\epsilon_{\mathsf{NARG}}(m, t) \leq \epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(m, t)$$
$$\leq (\epsilon_{\mathsf{FIOP}} + \epsilon_{\mathsf{FC}}(t \cdot \mathsf{kN} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}) \cdot (m + 1)^{\mathsf{k}}$$

State-restoration security of Funky[FIOP, FC]

Analogous for FIOP:

$$\epsilon_{\mathsf{ARG}}(t) \leq \epsilon_{\mathsf{FIOP}} + \epsilon_{\mathsf{FC}}(t \cdot \mathsf{kN} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}$$

RO queries

SR moves

k might be superconstant!

Generically:

$$\begin{aligned}\epsilon_{\mathsf{NARG}}(m, t) &\leq \epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(m, t) \\ &\leq (\epsilon_{\mathsf{FIOP}} + \epsilon_{\mathsf{FC}}(t \cdot \mathsf{kN} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}) \cdot (m + 1)^{\mathsf{k}}\end{aligned}$$

Theorem:

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

State-restoration security of Funky[FIOP, FC]

Analogous for FIOP:

$$\epsilon_{\mathsf{ARG}}(t) \leq \epsilon_{\mathsf{FIOP}} + \epsilon_{\mathsf{FC}}(t \cdot \mathsf{kN} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}$$

RO queries

SR moves

k might be superconstant!

Generically:

$$\epsilon_{\mathsf{NARG}}(m, t) \leq \epsilon^{\mathsf{SR}}_{\mathsf{ARG}}(m, t)$$
$$\leq (\epsilon_{\mathsf{FIOP}} + \epsilon_{\mathsf{FC}}(t \cdot \mathsf{kN} + t_{\mathbf{Q}} \cdot \mathsf{k}) + \mathsf{k} \cdot \epsilon_{\mathbf{Q}}) \cdot (m + 1)^{\mathsf{k}}$$

Theorem:

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

$\leq \epsilon_{\mathsf{FIOP}} \cdot \mathsf{poly}(m + \mathsf{k})$
for FIOPs of interest

$\leq \epsilon_{\mathsf{FC}} \cdot \mathsf{poly}(m \cdot \mathsf{kN})$
for FCs of interest

Application:
Plonk is a SNARK in the ROM from the SDH assumption

Plonk’s security

iPlonk = Funky[PlonkIOP, linKZG]
Plonk = FS[iPlonk]

Plonk's security

constant-round FIOP

iPlonk = Funky[PlonkIOP, linKZG]

Plonk = FS[iPlonk]

Plonk's security

constant-round FIOP

one-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Plonk = FS[iPlonk]

Plonk's security

Query class: evaluations of structured polynomials $\mathbf{Q}_{\text{poly}}{}^{*}$

constant-round FIOP

one-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Plonk = FS[iPlonk]

Plonk’s security

Query class: evaluations of structured polynomials $\mathbb{Q}_{\mathrm{poly}}{}^{*}$

constant-round FIOP

one-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Plonk = FS[iPlonk]

Not special sound!
(assuming DLog [LPS24b])

Plonk's security

Query class: evaluations of structured polynomials $\mathbf{Q}_{\text{poly}}{}^{*}$

constant-round FIOP

one-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Plonk = FS[iPlonk]

Plonk's security

Query class: evaluations of structured polynomials $\mathbf{Q}_{\text{poly}}{}^{*}$

constant-round FIOP

one-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Security in idealized models

Knowledge-sound in AGM [GWC19]

Plonk's security

Query class: evaluations of structured polynomials $\mathbf{Q}_{\text{poly}}{}^{*}$

constant-round FIOP

one-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Security in idealized models

Knowledge-sound in AGM [GWC19]

Security in standard model

Special-sound from new ARSDH and SplitRSDH assumptions [LPS24b]

Plonk's security

Query class: evaluations of structured polynomials $\mathbf{Q}_{\text{poly}}{}^{*}$

constant-round FIOP

one-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Security in idealized models

Knowledge-sound in AGM [GWC19]

Security in standard model

Special-sound from new ARSDH and SplitRSDH assumptions [LPS24b]

Corollary:

iPlonk is state-restoration (knowledge) sound
assuming Strong Diffie-Hellman

Plonk's security

Query class: evaluations of structured polynomials $\mathbf{Q}_{\text{poly}}{}^{*}$

constant-round FIOP

constant-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Corollary:

iPlonk is state-restoration (knowledge) sound
assuming SDH

Plonk's security

Query class: evaluations of structured polynomials $\mathbf{Q}_{\text{poly}}{}^{*}$

constant-round FIOP

constant-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Corollary:

iPlonk is state-restoration (knowledge) sound
assuming SDH

Lemma:

PlonkIOP is state-restoration (knowledge) sound

Plonk's security

Query class: evaluations of structured polynomials $\mathbf{Q}_{\text{poly}}{}^{*}$

constant-round FIOP

constant-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Corollary:

iPlonk is state-restoration (knowledge) sound *assuming SDH*

Lemma:

PlonkIOP is state-restoration (knowledge) sound

Lemma:

linKZG is state-restoration function binding *assuming SDH*

Plonk's security

Query class: evaluations of structured polynomials $\mathbf{Q}_{\text{poly}^*}$

constant-round FIOP

constant-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Corollary:

iPlonk is state-restoration (knowledge) sound *assuming SDH*

Lemma:

PlonkIOP is state-restoration (knowledge) sound

Lemma:

linKZG is state-restoration function binding *assuming SDH*

Lemma:

$\mathbf{Q}_{\text{poly}^*}$ has good tail error and solver time

Plonk's security

Query class: evaluations of structured polynomials $\mathbf{Q}_{\text{poly}^*}$

constant-round FIOP

constant-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Corollary:

iPlonk is state-restoration (knowledge) sound *assuming SDH*

Lemma:

PlonkIOP is state-restoration (knowledge) sound

Lemma:

linKZG is state-restoration function binding *assuming SDH*

Lemma:

$\mathbf{Q}_{\text{poly}^*}$ has good tail error and solver time

Plonk's security

Query class: evaluations of structured polynomials $\mathbf{Q}_{\text{poly}}{}^{*}$

constant-round FIOP

constant-round FC

iPlonk = Funky[PlonkIOP, linKZG]

Corollary:

iPlonk is state-restoration (knowledge) sound *assuming SDH*

Lemma:

PlonkIOP is state-restoration (knowledge) sound

Lemma:

linKZG is state-restoration function binding *assuming SDH*

Lemma:

$\mathbf{Q}_{\text{poly}}{}^{*}$ has good tail error and solver time

linKZG is state-restoration function binding

linKZG = lin[batchKZG] is state-restoration function binding

linKZG is state-restoration function binding

linKZG = lin[batchKZG] is state-restoration function binding

$\boxed{\text{Lemma: lin[bPC] SR-FB} \Leftarrow \text{bPC SR-FB for any homomorphic bPC}}$

$\Leftarrow$ batchKZG = batch[KZG] is state-restoration function binding

linKZG is state-restoration function binding

linKZG = lin[batchKZG] is state-restoration function binding

Lemma: lin[bPC] SR-FB $\Leftarrow$ bPC SR-FB for any homomorphic bPC

$\Leftarrow$ batchKZG = batch[KZG] is state-restoration function binding

Lemma: batch[PC] SR-FB $\Leftarrow$ PC SR-FB for any homomorphic PC

$\Leftarrow$ KZG is state-restoration function binding

linKZG is state-restoration function binding

linKZG = lin[batchKZG] is state-restoration function binding

Lemma: lin[bPC] SR-FB $\Leftarrow$ bPC SR-FB for any homomorphic bPC

$\Leftarrow$ batchKZG = batch[KZG] is state-restoration function binding

Lemma: batch[PC] SR-FB $\Leftarrow$ PC SR-FB for any homomorphic PC

$\Leftarrow$ KZG is state-restoration function binding

$\Leftarrow$ KZG is function binding (KZG is non-interactive)

linKZG is state-restoration function binding

linKZG = lin[batchKZG] is state-restoration function binding

Lemma: lin[bPC] SR-FB ⇐ bPC SR-FB for any homomorphic bPC

⇐ batchKZG = batch[KZG] is state-restoration function binding

Lemma: batch[PC] SR-FB ⇐ PC SR-FB for any homomorphic PC

⇐ KZG is state-restoration function binding

⇐ KZG is function binding (KZG is non-interactive)

Lemma: PC FB ⇐ PC EB for any homomorphic PC

⇐ KZG is evaluation binding

linKZG is state-restoration function binding

linKZG = lin[batchKZG] is state-restoration function binding

$\boxed{\text{Lemma: lin[bPC] SR-FB} \Leftarrow \text{bPC SR-FB for any homomorphic bPC}}$

$\Leftarrow$ batchKZG = batch[KZG] is state-restoration function binding

$\boxed{\text{Lemma: batch[PC] SR-FB} \Leftarrow \text{PC SR-FB for any homomorphic PC}}$

$\Leftarrow$ KZG is state-restoration function binding

$\Leftarrow$ KZG is function binding (KZG is non-interactive)

$\boxed{\text{Lemma: PC FB} \Leftarrow \text{PC EB for any homomorphic PC}}$

$\Leftarrow$ KZG is evaluation binding

$\Leftarrow$ *Strong Diffie-Hellman* [KZG10]

Our results

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

FIOP is state-restoration (knowledge) sound

+ FC is state-restoration function binding

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

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

Application: Plonk = FS[Funky[PlonkIOP, linearized KZG]]

is a SNARK in the ROM

from the SDH assumption (previously: from ARSDH+SplitRSDH)

On the Fiat–Shamir Security of Succinct Arguments from Functional Commitments

Alessandro Chiesa, Ziyi Guan, Christian Knabenhans, Zihan Yu

EPFL

Tail error and solving time

	Messages	Answers		Tail error $\epsilon_{\mathbf{Q}}(\ell, \mathrm{N})$	Solving time $t_{\mathbf{Q}}(\ell, \mathrm{q}, \mathrm{N})$

Tail error and solving time

	Messages	Answers		Tail error $\epsilon_{\mathbf{Q}}(\ell, \mathrm{N})$	Solving time $t_{\mathbf{Q}}(\ell, \mathrm{q}, \mathrm{N})$
Point queries	$\Pi \in \mathbb{F}^{\ell}$	$\beta = \Pi[\alpha]$	fill	ℓ / N	$O(\mathrm{qN})$

Tail error and solving time

	Messages	Answers		Tail error $\epsilon_Q(\ell, N)$	Solving time $t_Q(\ell, q, N)$
Point queries	$\Pi \in \mathbb{F}^\ell$	$\beta = \Pi[\alpha]$	fill	ℓ / N	$O(qN)$
Linear queries	$\Pi \in \mathbb{F}^\ell$	$\beta = \sum_{i \in [\ell]} \Pi[i] \cdot \alpha[i]$	Gaussian elimination	ℓ / N	$O(qN\ell^2)$

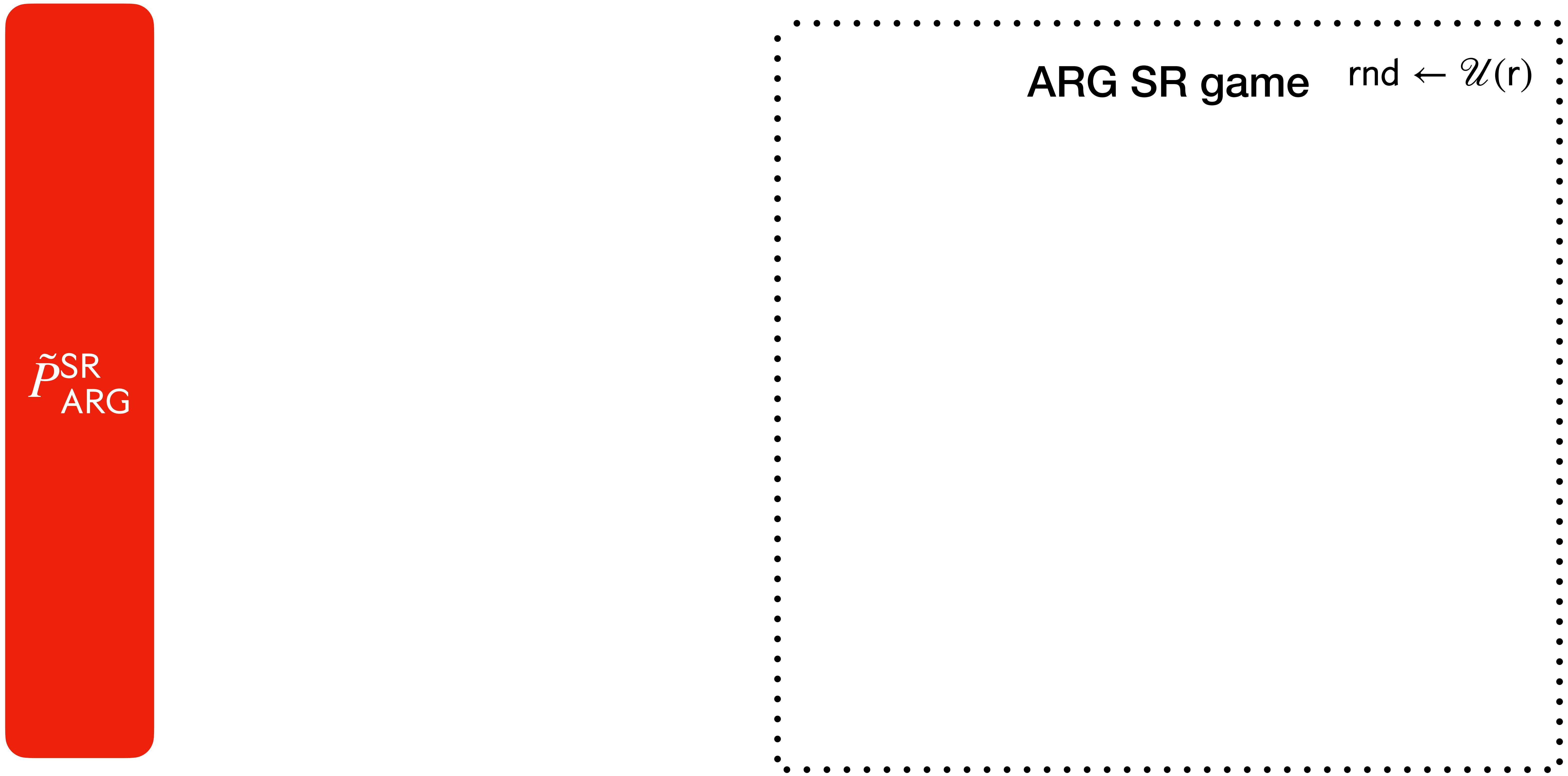
Tail error and solving time

	Messages	Answers		Tail error $\epsilon_Q(\ell, N)$	Solving time $t_Q(\ell, q, N)$
Point queries	$\Pi \in \mathbb{F}^{\ell}$	$\beta = \Pi[\alpha]$	fill	ℓ/N	$O(qN)$
Linear queries	$\Pi \in \mathbb{F}^{\ell}$	$\beta = \sum_{i \in [\ell]} \Pi[i] \cdot \alpha[i]$	Gaussian elimination	ℓ/N	$O(qN\ell^2)$
Evaluation queries on polynomials	$\Pi \in \mathbb{F}[X]^{<\ell}$	$\beta = \Pi(\alpha)$	interpolation	ℓ/N	$O(\ell^2 + qN\ell)$

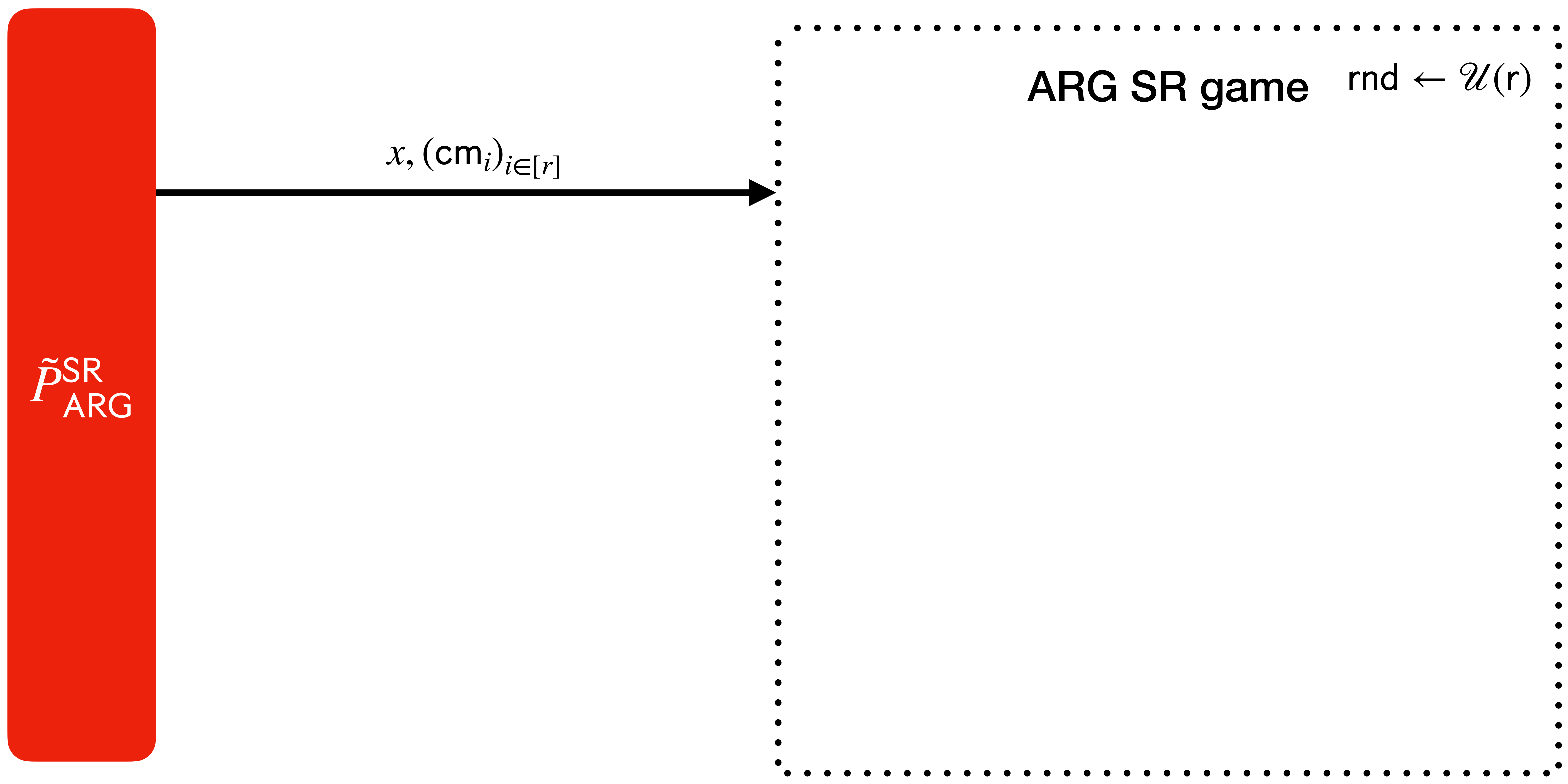
Tail error and solving time

	Messages	Answers		Tail error $\epsilon_{\mathbf{Q}}(\ell, \mathsf{N})$	Solving time $t_{\mathbf{Q}}(\ell, \mathsf{q}, \mathsf{N})$
Point queries	$\Pi \in \mathbb{F}^{\ell}$	$\beta = \Pi[\alpha]$	fill	ℓ/N	$O(\mathsf{qN})$
Linear queries	$\Pi \in \mathbb{F}^{\ell}$	$\beta = \sum_{i\in[\ell]} \Pi[i] \cdot \alpha[i]$	Gaussian elimination	ℓ/N	$O(\mathsf{qN}\ell^2)$
Evaluation queries on polynomials	$\Pi \in \mathbb{F}[X]^{<\ell}$	$\beta = \Pi(\alpha)$	interpolation	ℓ/N	$O(\ell^2 + \mathsf{qN}\ell)$
Evaluation queries on structured polynomials	$\Pi \in (\mathbb{F}[X]^{\leq \mathsf{D}})^{m+n} = (f_1, \ldots, f_m, g_1, \ldots, g_n)$	$\beta = \sum_{k\in[n]} h_k(f_1(\alpha), \cdots, f_m(\alpha)) \cdot g_k(\alpha)$		ℓ/N $\ell = (m+n)(\mathsf{D}+1)$	$O(\mathsf{qN}\ell^2)$

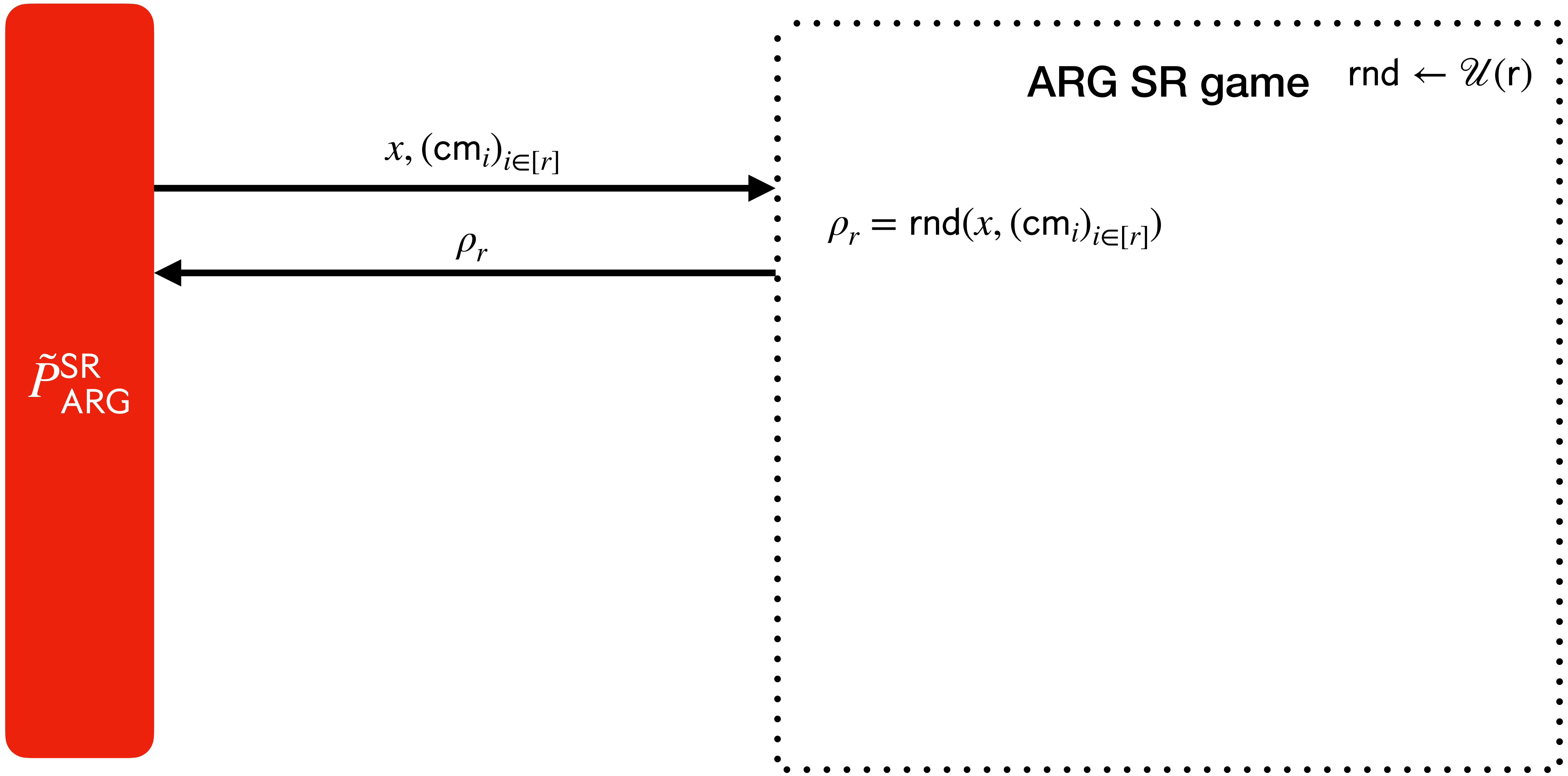
Security reduction for Funky[FIOP, FC]



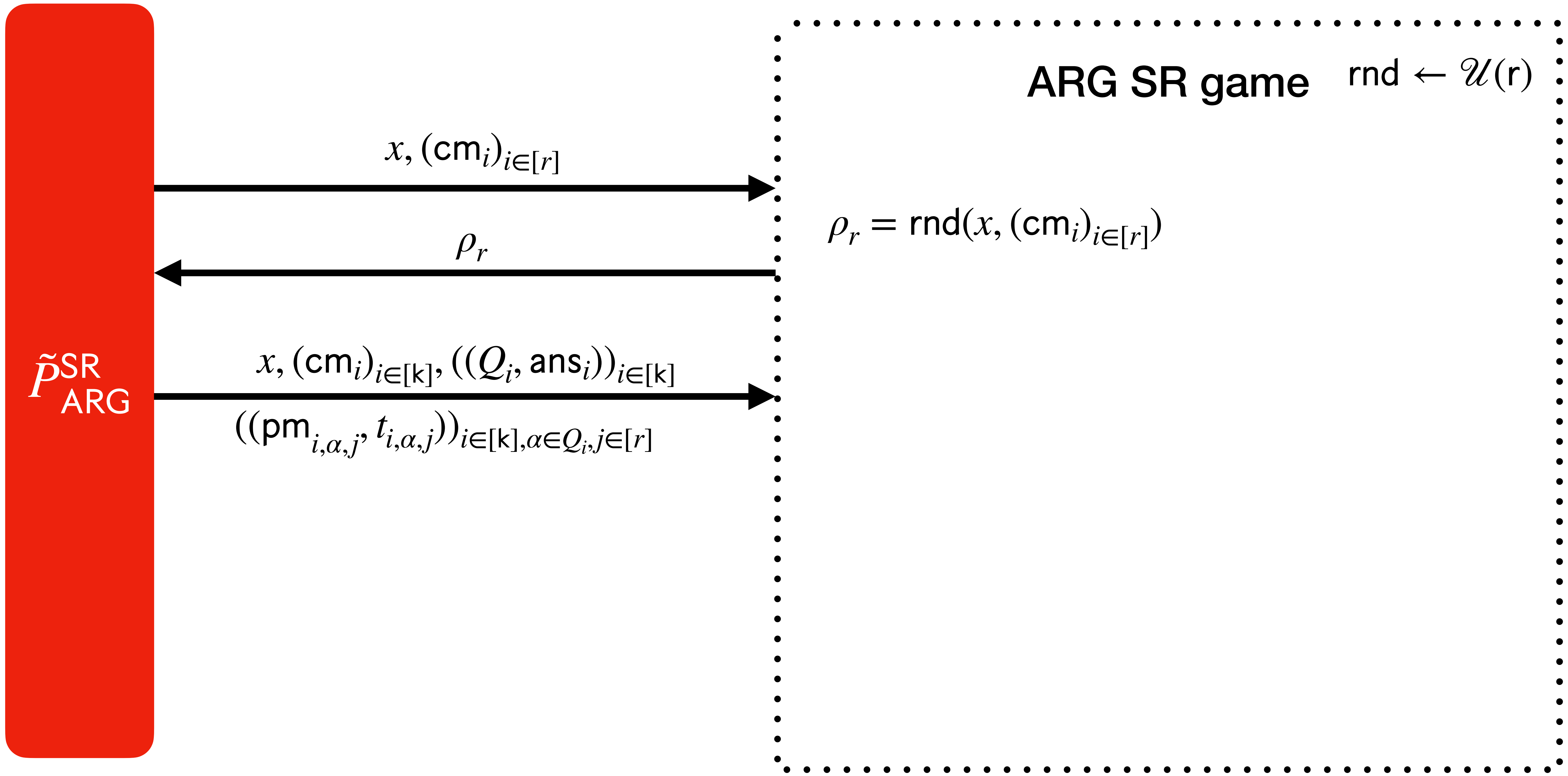
Security reduction for Funky[FIOP, FC]



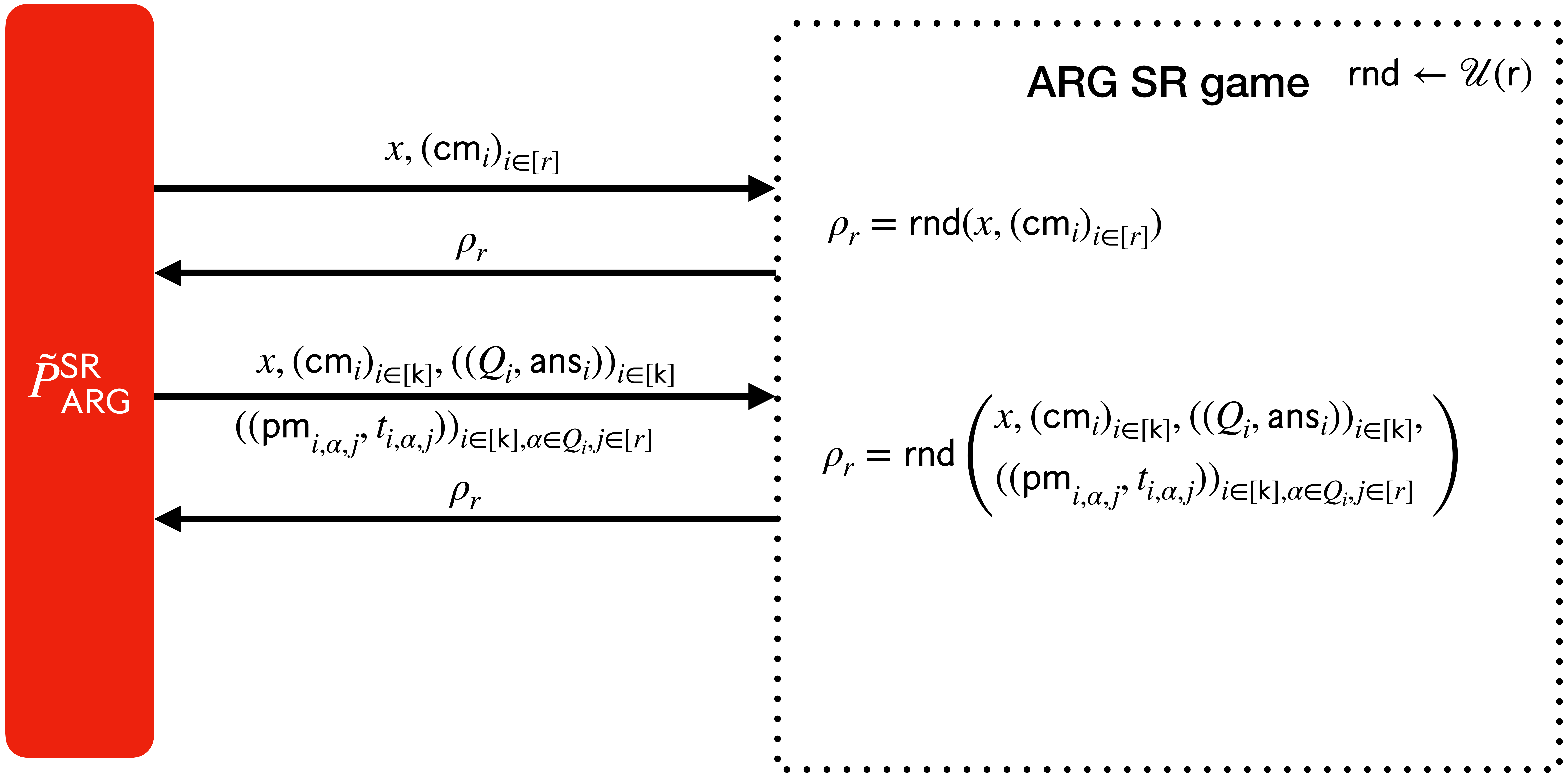
Security reduction for Funky[FIOP, FC]



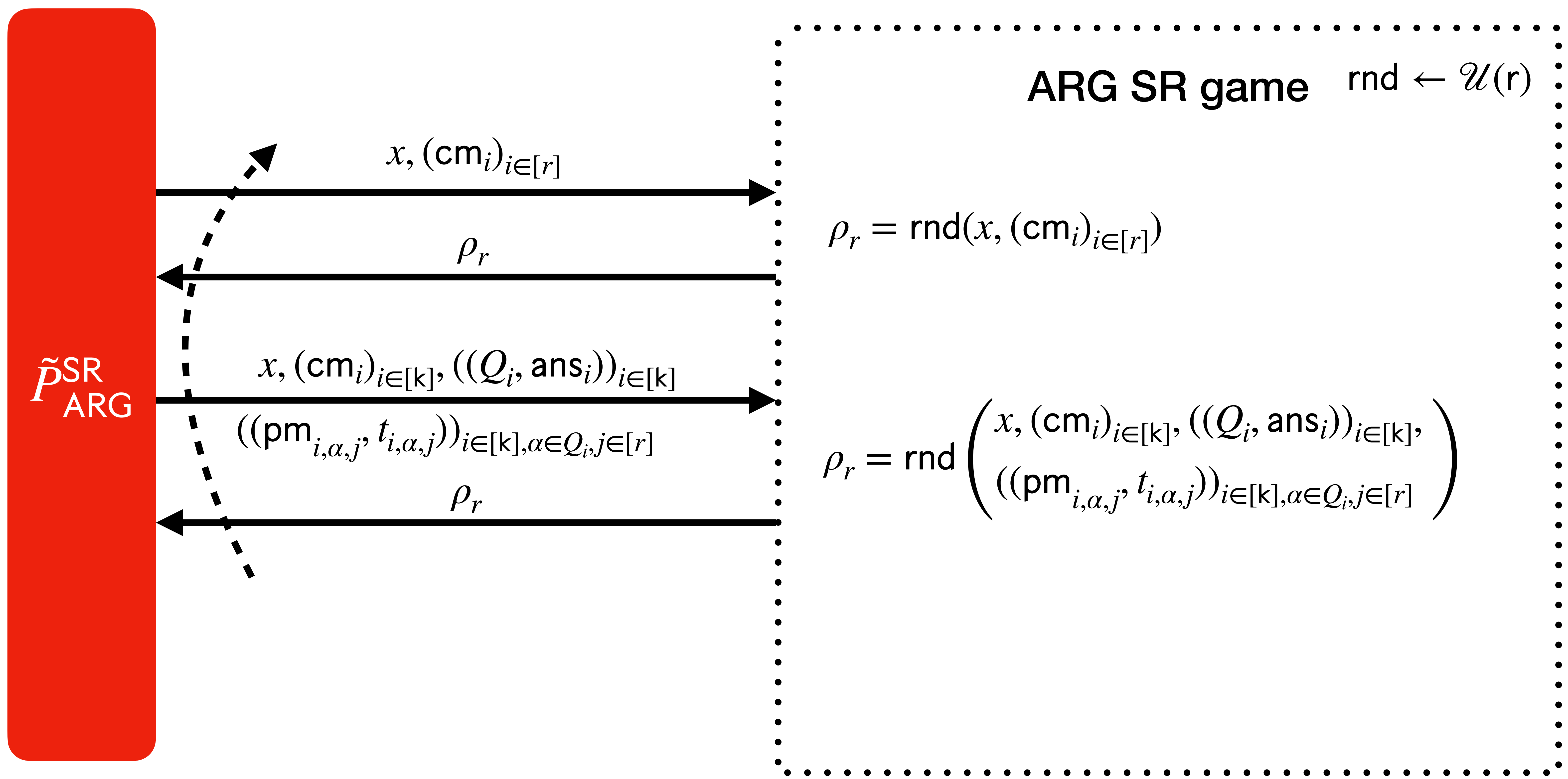
Security reduction for Funky[FIOP, FC]



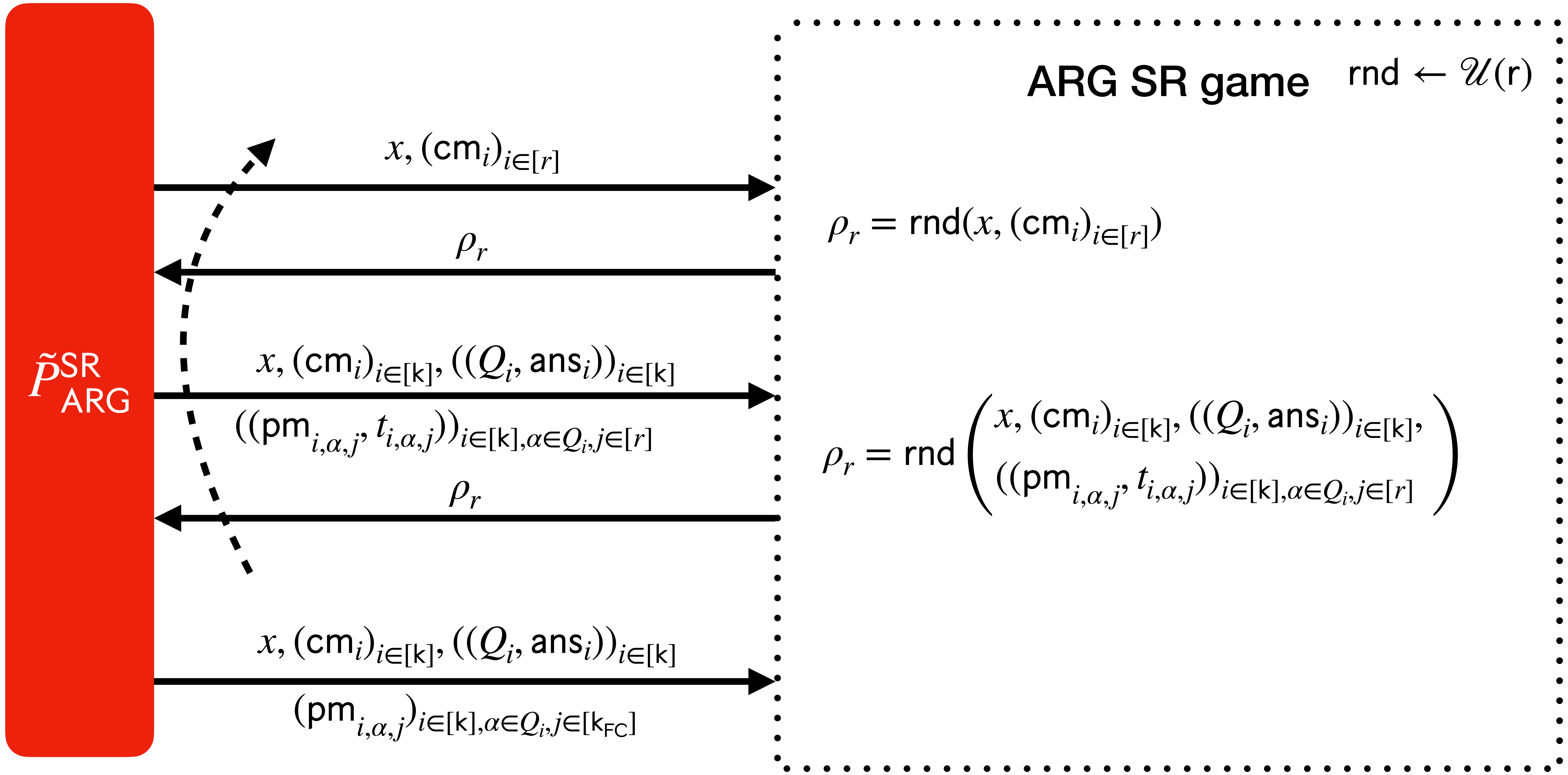
Security reduction for Funky[FIOP, FC]



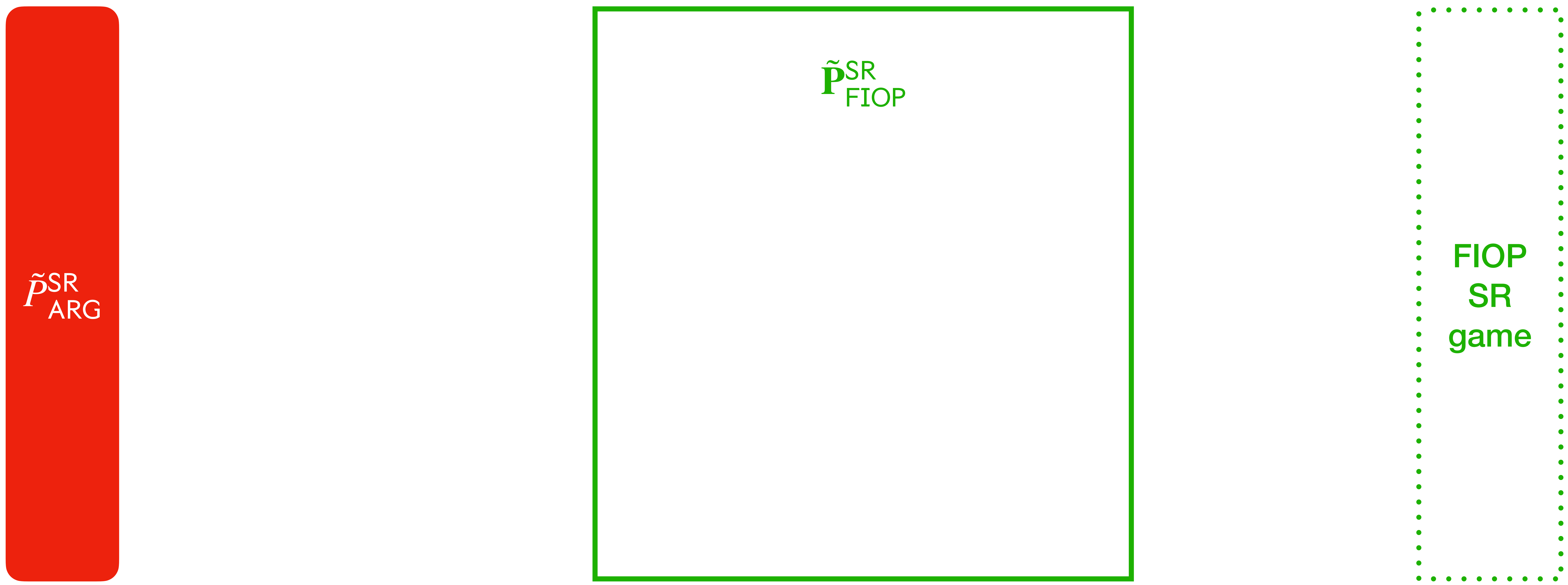
Security reduction for Funky[FIOP, FC]



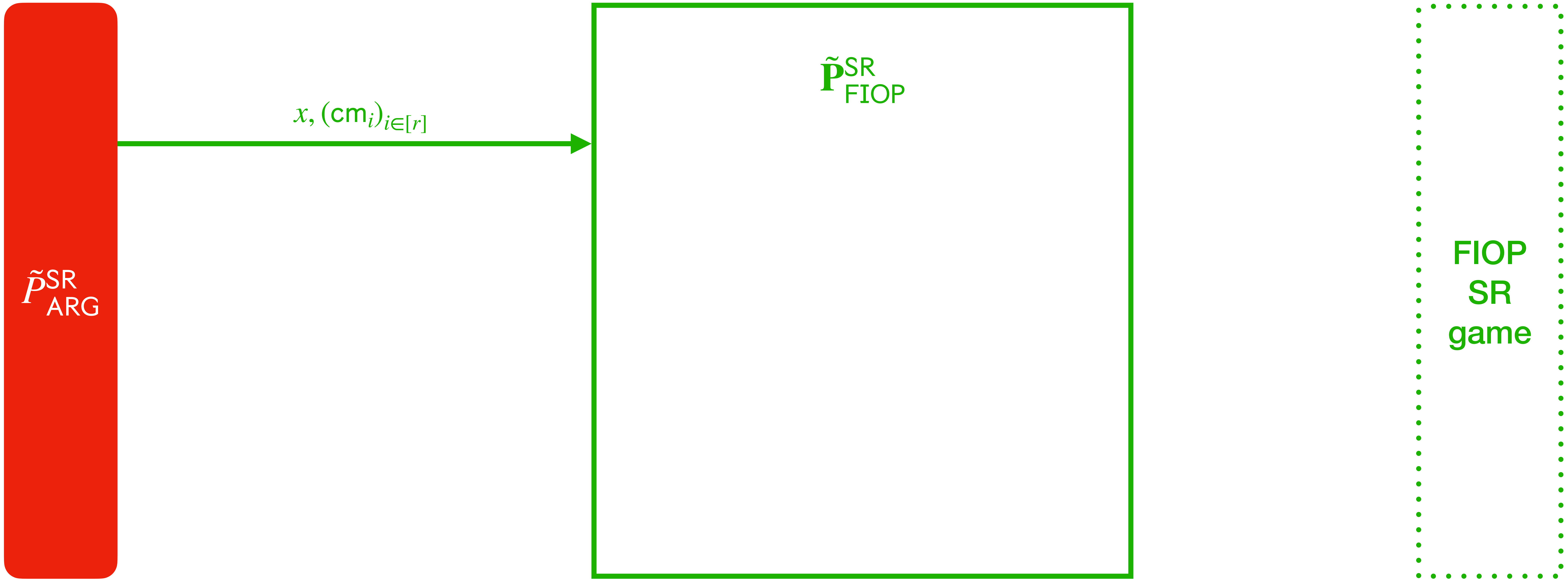
Security reduction for Funky[FIOP, FC]



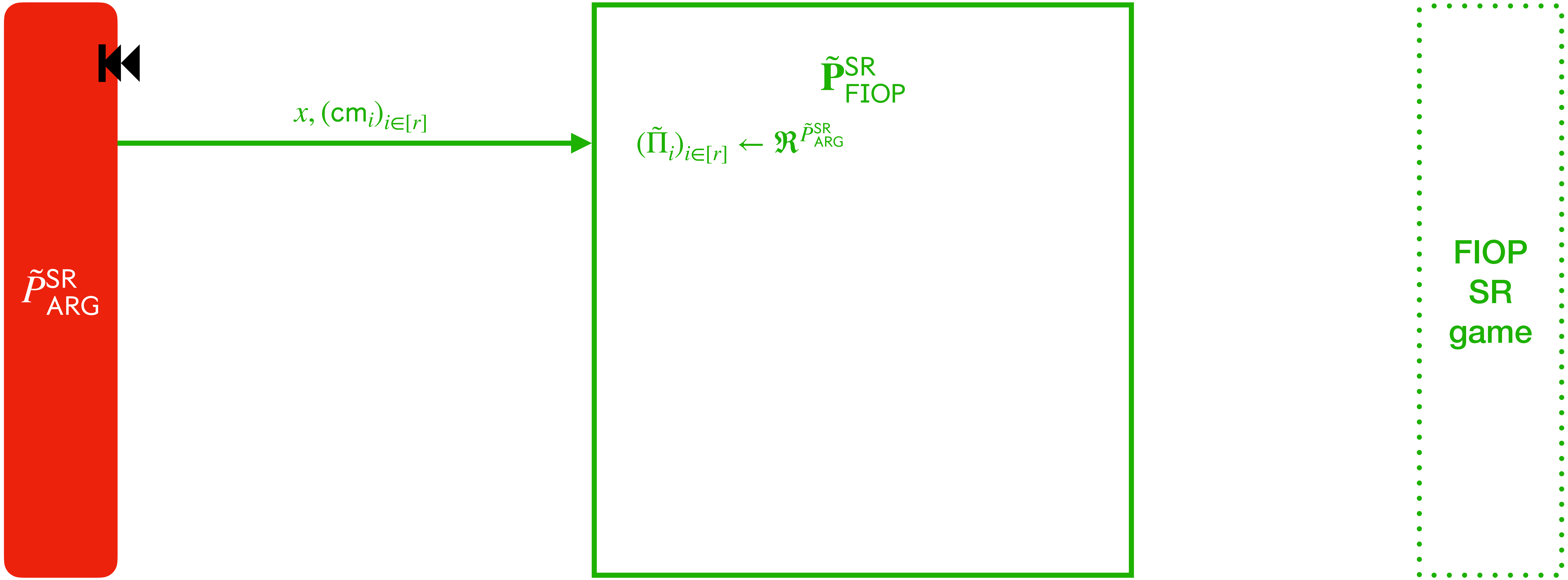
Security reduction for Funky[FIOP, FC]



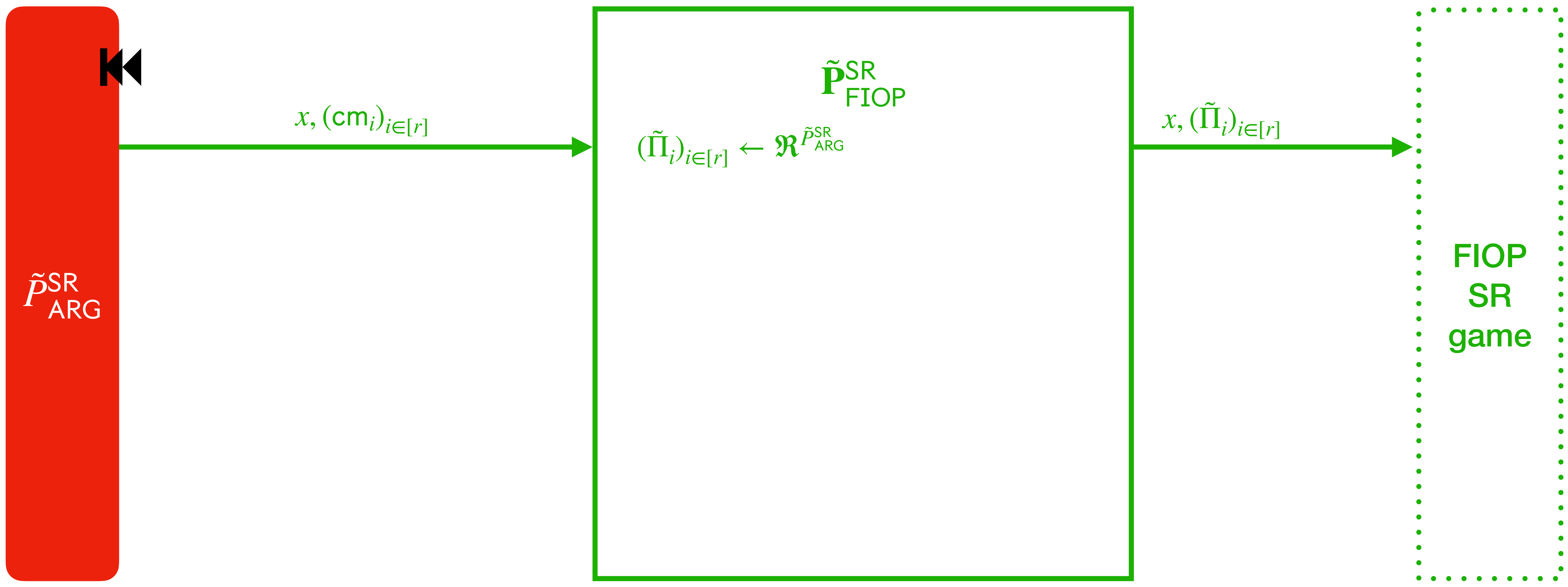
Security reduction for Funky[FIOP, FC]



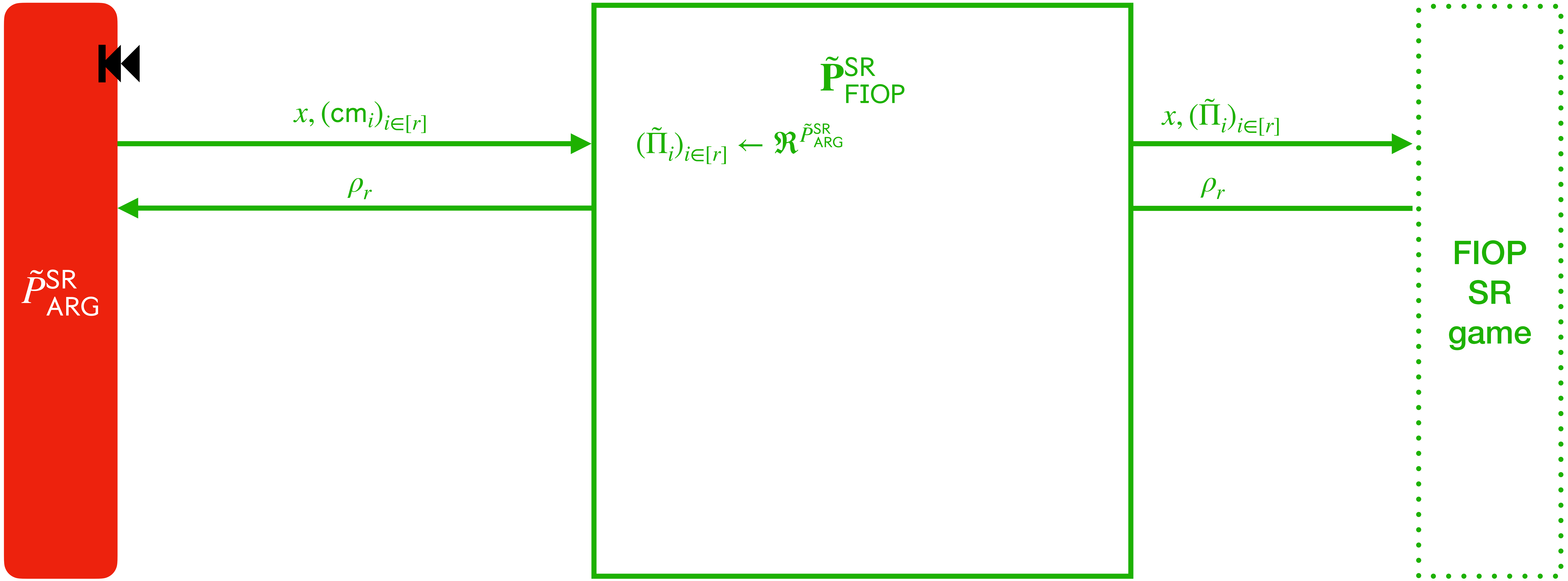
Security reduction for Funky[FIOP, FC]



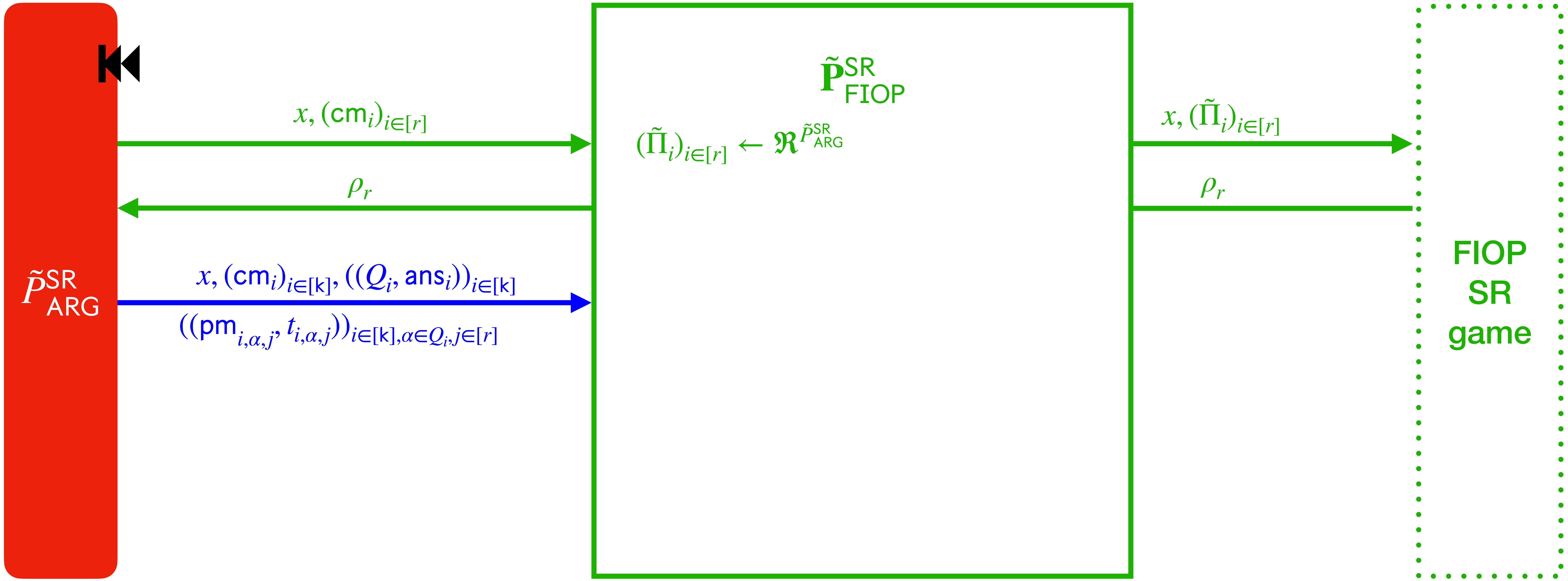
Security reduction for Funky[FIOP, FC]



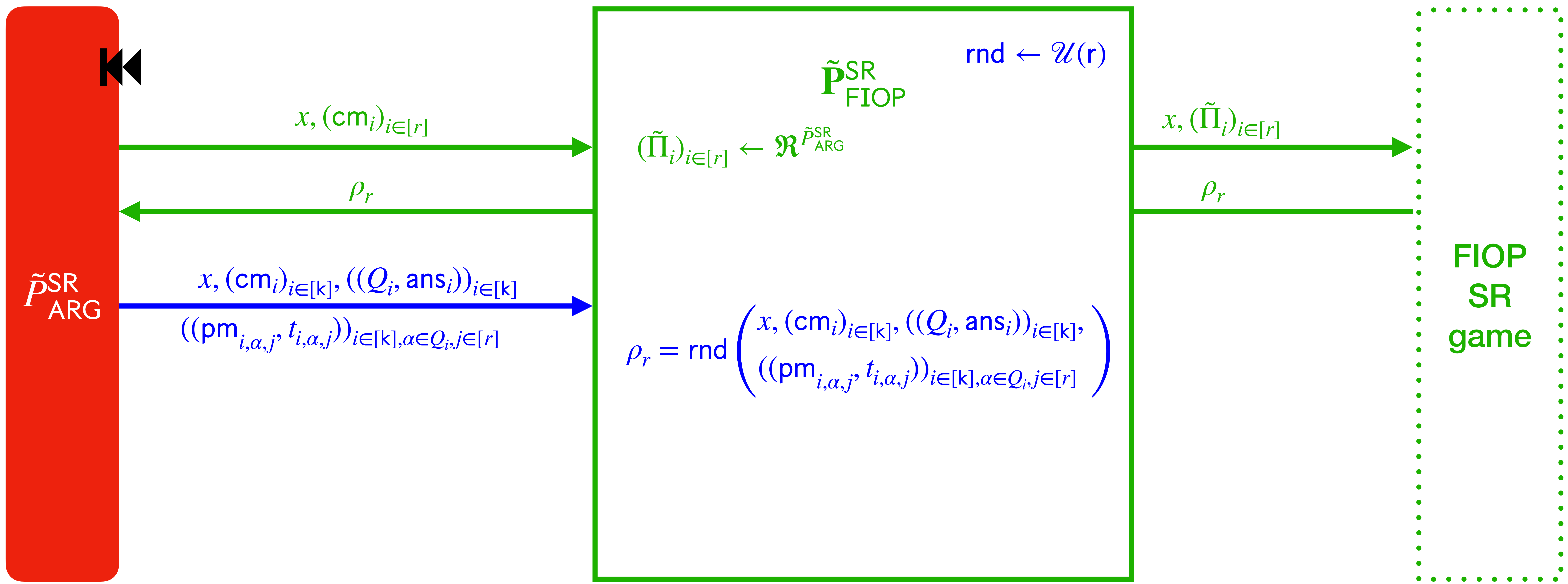
Security reduction for Funky[FIOP, FC]



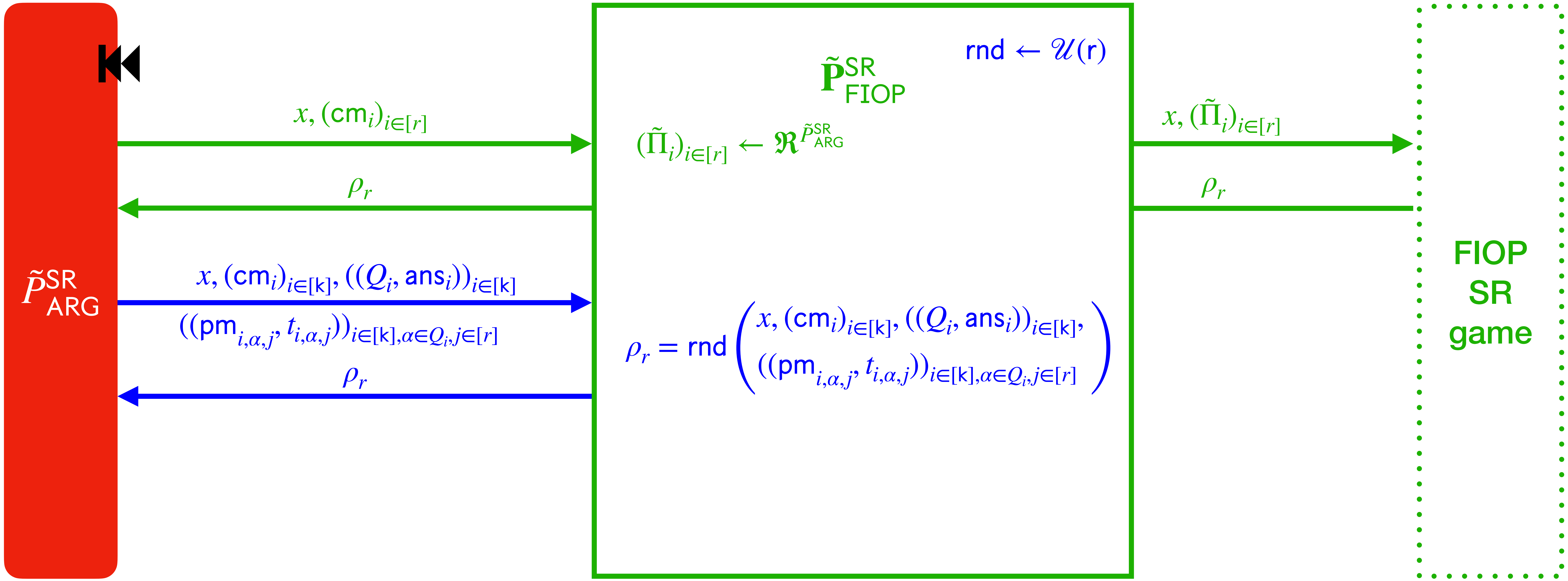
Security reduction for Funky[FIOP, FC]



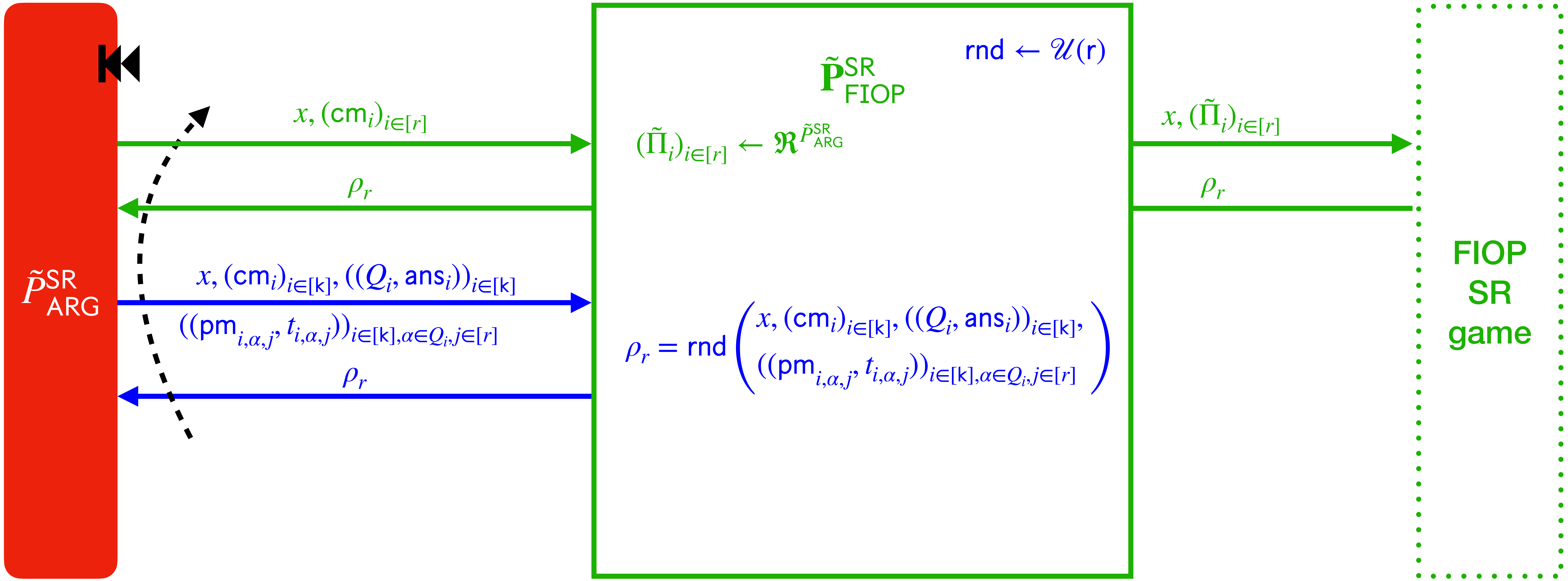
Security reduction for Funky[FIOP, FC]



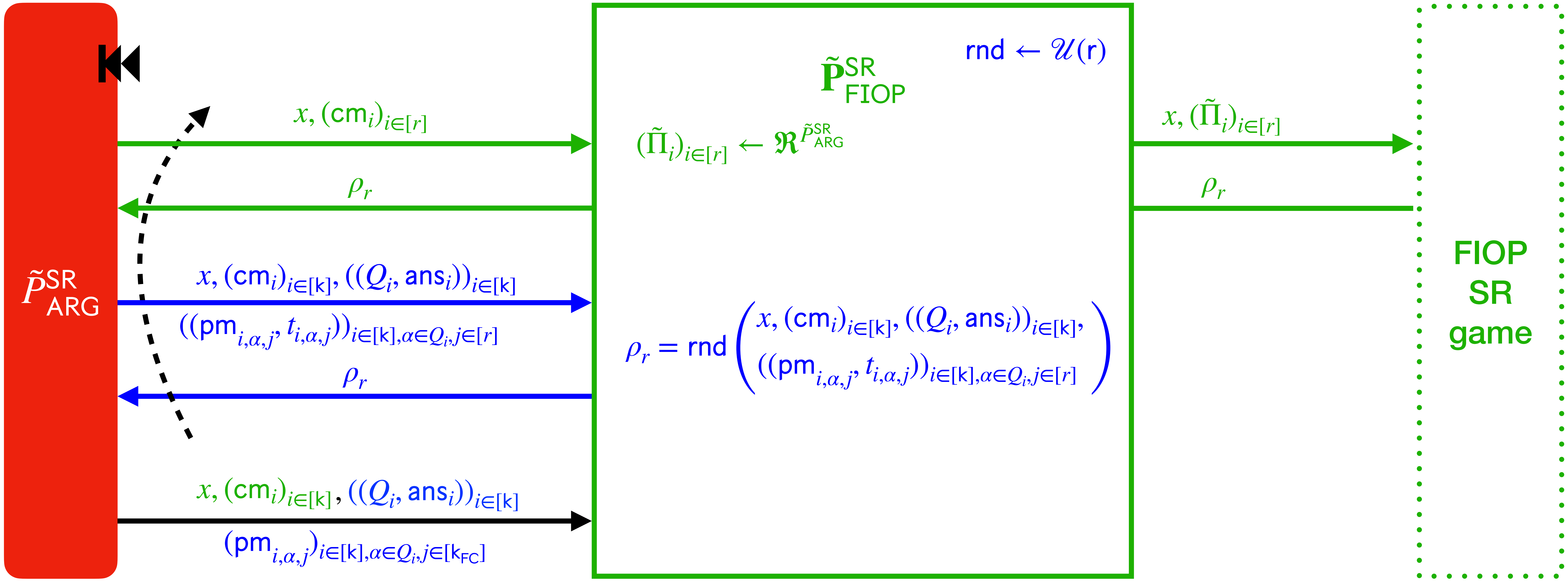
Security reduction for Funky[FIOP, FC]



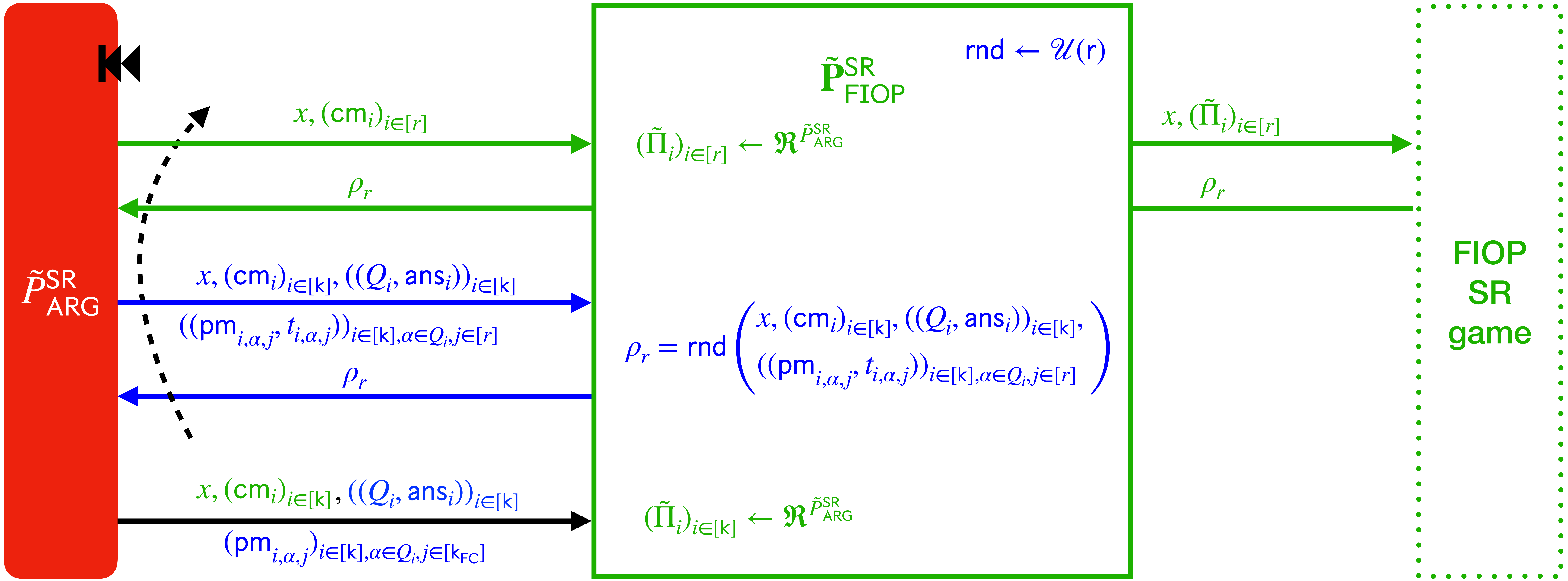
Security reduction for Funky[FIOP, FC]



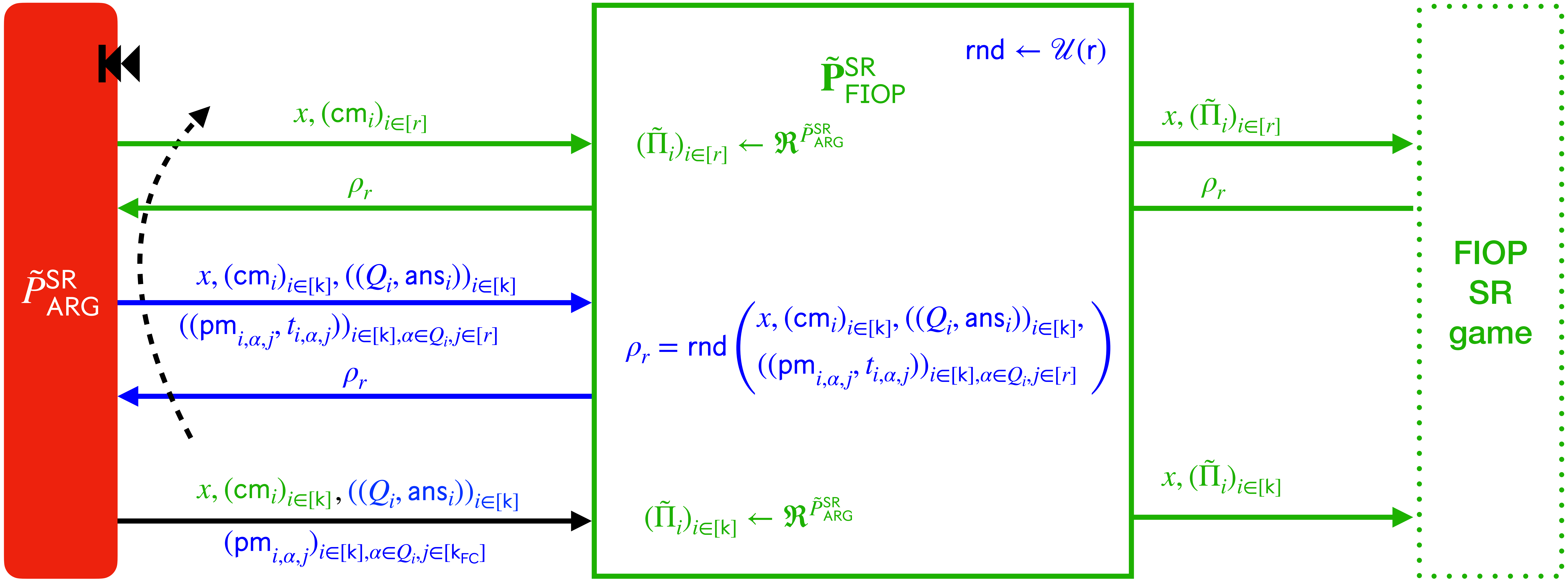
Security reduction for Funky[FIOP, FC]



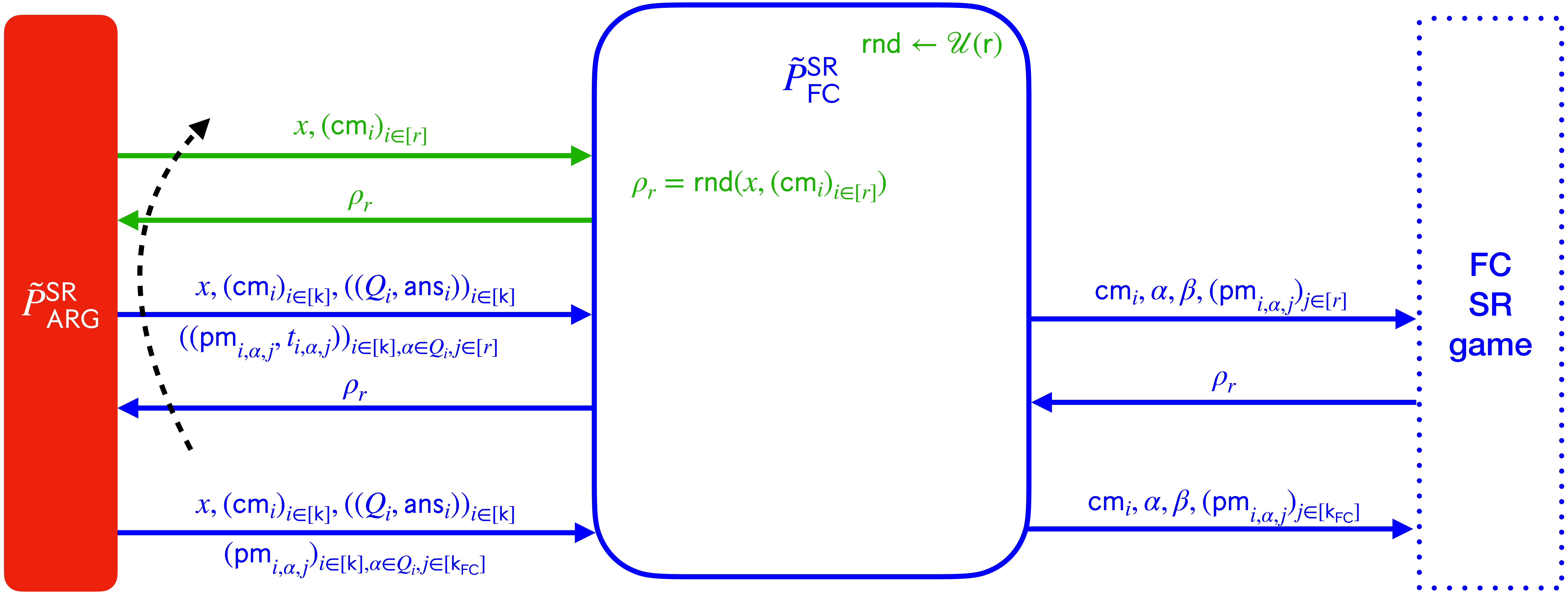
Security reduction for Funky[FIOP, FC]



Security reduction for Funky[FIOP, FC]



Security reduction for Funky[FIOP, FC]



linKZG = Lin[batchKZG]

linKZG = Lin[batchKZG]

public

$\in \mathbb{F}[X]^{\leq D}$

$\in \mathbb{F}[X]^{\leq D}$

$$\Pi(X) = \sum_{k \in [n]} h_k(f_1(X), \ldots, f_m(X)) \cdot g_k(X)$$

linKZG = Lin[batchKZG]

public $\in \mathbb{F}[X]^{\leq D}$ $\in \mathbb{F}[X]^{\leq D}$

$$\Pi(X) = \sum_{k\in[n]} h_k(f_1(X), \ldots, f_m(X)) \cdot g_k(X)$$

$$\mathsf{Commit}(\Pi) = \mathsf{cm}_{f_1}, \ldots, \mathsf{cm}_{f_m}, \mathsf{cm}_{g_1}, \ldots, \mathsf{cm}_{g_n}$$

linKZG = Lin[batchKZG]

public $\quad \in \mathbb{F}[X]^{\leq D} \quad \in \mathbb{F}[X]^{\leq D}$

$$\Pi(X) = \sum_{k\in[n]} h_k(f_1(X), \ldots, f_m(X)) \cdot g_k(X)$$

$$\text{Commit}(\Pi) = \text{cm}_{f_1}, \ldots, \text{cm}_{f_m}, \text{cm}_{g_1}, \ldots, \text{cm}_{g_n}$$

$$P_{\text{lin}}(\text{cm}, \alpha, \beta; \Pi)$$

$$P_{\text{batch}}$$

$$\left(\begin{bmatrix} \text{cm}_{f_1} \\ \vdots \\ \text{cm}_{f_m} \\ \sum_k h_k(f_1(\alpha), \ldots, f_m(\alpha)) \cdot \text{cm}_{g_k} \end{bmatrix}, \alpha, \begin{bmatrix} f_1(\alpha) \\ \vdots \\ f_m(\alpha) \\ \beta \end{bmatrix}, \begin{bmatrix} f_1 \\ \vdots \\ f_m \\ \sum_{k\in[m]} h_k(f_1(\alpha), \ldots, f_m(\alpha)) \cdot g_k(X) \end{bmatrix} \right)$$

linKZG = Lin[batchKZG]

public $\quad \in \mathbb{F}[X]^{\leq D} \quad \in \mathbb{F}[X]^{\leq D}$

$$\Pi(X) = \sum_{k \in [n]} h_k(f_1(X), \ldots, f_m(X)) \cdot g_k(X)$$

$$\text{Commit}(\Pi) = \mathsf{cm}_{f_1}, \ldots, \mathsf{cm}_{f_m}, \mathsf{cm}_{g_1}, \ldots, \mathsf{cm}_{g_n}$$

$$P_{\text{lin}}(\mathsf{cm}, \alpha, \beta; \Pi)$$

$$P_{\text{batch}}$$

$$\left(\begin{bmatrix} \mathsf{cm}_{f_1} \\ \vdots \\ \mathsf{cm}_{f_m} \\ \sum_k h_k(f_1(\alpha), \ldots, f_m(\alpha)) \cdot \mathsf{cm}_{g_k}) \end{bmatrix}, \alpha, \begin{bmatrix} f_1(\alpha) \\ \vdots \\ f_m(\alpha) \\ \beta \end{bmatrix}, \begin{bmatrix} f_1 \\ \vdots \\ f_m \\ \sum_{k \in [m]} h_k(f_1(\alpha), \ldots, f_m(\alpha)) \cdot g_k(X) \end{bmatrix} \right)$$

constant $\in \mathbb{F}$ $\quad$ Linearized polynomial $\in \mathbb{F}[X]^{\leq D}$

linKZG = Lin[batchKZG]

public $\quad \in \mathbb{F}[X]^{\leq D} \quad \in \mathbb{F}[X]^{\leq D}$

$$\Pi(X) = \sum_{k\in[n]} h_k(f_1(X), \ldots, f_m(X)) \cdot g_k(X)$$

$$\text{Commit}(\Pi) = \mathsf{cm}_{f_1}, \ldots, \mathsf{cm}_{f_m}, \mathsf{cm}_{g_1}, \ldots, \mathsf{cm}_{g_n}$$

$P_{\text{lin}}(\mathsf{cm}, \alpha, \beta; \Pi)$

P_{batch}

$$\left(\begin{bmatrix} \mathsf{cm}_{f_1} \\ \vdots \\ \mathsf{cm}_{f_m} \\ \sum_k h_k(f_1(\alpha), \ldots, f_m(\alpha)) \cdot \mathsf{cm}_{g_k} \end{bmatrix}, \alpha, \begin{bmatrix} f_1(\alpha) \\ \vdots \\ f_m(\alpha) \\ \beta \end{bmatrix}, \begin{bmatrix} f_1 \\ \vdots \\ f_m \\ \sum_{k\in[m]} h_k(f_1(\alpha), \ldots, f_m(\alpha)) \cdot g_k(X) \end{bmatrix} \right)$$

constant $\in \mathbb{F}$

Linearized polynomial $\in \mathbb{F}[X]^{\leq D}$

$V_{\text{lin}}(\mathsf{cm}, \alpha, \beta)$

V_{batch}

$$\left(\begin{bmatrix} \mathsf{cm}_{f_1} \\ \vdots \\ \mathsf{cm}_{f_m} \\ \sum_k h_k(f_1(\alpha), \ldots, f_m(\alpha)) \cdot \mathsf{cm}_{g_k} \end{bmatrix}, \alpha, \begin{bmatrix} f_1(\alpha) \\ \vdots \\ f_m(\alpha) \\ \beta \end{bmatrix} \right)$$

linKZG = Lin[batchKZG]

public $\quad \in \mathbb{F}[X]^{\leq \mathsf{D}} \quad \in \mathbb{F}[X]^{\leq \mathsf{D}}$

$$\Pi(X) = \sum_{k\in[n]} h_k(f_1(X), \ldots, f_m(X)) \cdot g_k(X)$$

Degree: $\mathsf{D}_h \cdot \mathsf{D} + \mathsf{D}$

$$\mathsf{Commit}(\Pi) = \mathsf{cm}_{f_1}, \ldots, \mathsf{cm}_{f_m}, \mathsf{cm}_{g_1}, \ldots, \mathsf{cm}_{g_n}$$

$P_{\text{lin}}(\mathsf{cm}, \alpha, \beta; \Pi)$

Batch size: $m+1$

Degree: D

P_{batch}

$$\left(\begin{bmatrix} \mathsf{cm}_{f_1} \\ \vdots \\ \mathsf{cm}_{f_m} \\ \sum_k \underbrace{h_k(f_1(\alpha), \ldots, f_m(\alpha))}_{\text{constant } \in \mathbb{F}} \cdot \mathsf{cm}_{g_k}) \end{bmatrix}, \alpha, \begin{bmatrix} f_1(\alpha) \\ \vdots \\ f_m(\alpha) \\ \beta \end{bmatrix}, \begin{bmatrix} f_1 \\ \vdots \\ f_m \\ \underbrace{\sum_{k\in[m]} h_k(f_1(\alpha), \ldots, f_m(\alpha)) \cdot g_k(X)}_{\text{Linearized polynomial } \in \mathbb{F}[X]^{\leq \mathsf{D}}} \end{bmatrix}\right)$$

$V_{\text{lin}}(\mathsf{cm}, \alpha, \beta)$

V_{batch}

$$\left(\begin{bmatrix} \mathsf{cm}_{f_1} \\ \vdots \\ \mathsf{cm}_{f_m} \\ \sum_k h_k(f_1(\alpha), \ldots, f_m(\alpha)) \cdot \mathsf{cm}_{g_k}) \end{bmatrix}, \alpha, \begin{bmatrix} f_1(\alpha) \\ \vdots \\ f_m(\alpha) \\ \beta \end{bmatrix}\right)$$

constant $\in \mathbb{F}$

Linearized polynomial $\in \mathbb{F}[X]^{\leq \mathsf{D}}$

KZG function binding $\Leftarrow$ evaluation binding

Goal: given A_{FB}, construct A_{EB}

KZG function binding $\Leftarrow$ evaluation binding

Goal: given A_{FB}, construct A_{EB}

Proof sketch:

$\mathsf{cm}, ((\alpha_i, \beta_i, \mathsf{pf}_i))_{i \in [n]} \leftarrow A_{\mathsf{FB}}$ s.t. $\begin{cases} \forall i : \mathsf{Check}(\mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 & (1) \\ \nexists \Pi \in \mathbb{F}[X]^{\leq \mathsf{D}} : \forall i : \Pi(\alpha_i) = \beta_i & (2) \end{cases}$

KZG function binding $\Leftarrow$ evaluation binding

Goal: given A_{FB}, construct A_{EB}

Proof sketch:

$$\mathsf{cm}, ((\alpha_i, \beta_i, \mathsf{pf}_i))_{i\in[n]} \leftarrow A_{\mathsf{FB}} \text{ s.t. } \begin{cases} \forall i : \mathsf{Check}(\mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 & (1) \\ \nexists \Pi \in \mathbb{F}[X]^{\leq \mathsf{D}} : \forall i : \Pi(\alpha_i) = \beta_i & (2) \end{cases}$$

If $\exists i, j : \alpha_i = \alpha_j \wedge \beta_i \neq \beta_j$: output $(\mathsf{cm}, \alpha_i, \beta_i, \beta_j, \mathsf{pf}_i, \mathsf{pf}_j)$

KZG function binding $\Leftarrow$ evaluation binding

Goal: given A_{FB}, construct A_{EB}

Proof sketch:

$$\mathsf{cm}, ((\alpha_i, \beta_i, \mathsf{pf}_i))_{i \in [n]} \leftarrow A_{\mathsf{FB}} \text{ s.t. } \begin{cases} \forall i : \mathsf{Check}(\mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 & (1) \\ \nexists \Pi \in \mathbb{F}[X]^{\leq \mathsf{D}} : \forall i : \Pi(\alpha_i) = \beta_i & (2) \end{cases}$$

If $\exists i, j : \alpha_i = \alpha_j \wedge \beta_i \neq \beta_j$: output $(\mathsf{cm}, \alpha_i, \beta_i, \beta_j, \mathsf{pf}_i, \mathsf{pf}_j)$

Pick any i, choose any $\beta_i^* \neq \beta_i$

$\color{blue}{\text{KZG}}$ function binding $\Leftarrow$ evaluation binding

Goal: given A_{FB}, construct A_{EB}

Proof sketch:

$\mathsf{cm}, ((\alpha_i, \beta_i, \mathsf{pf}_i))_{i \in [n]} \leftarrow A_{\mathsf{FB}}$ s.t. $\begin{cases} \forall i : \mathsf{Check}(\mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 & (1) \\ \nexists \Pi \in \mathbb{F}[X]^{\leq \mathsf{D}} : \forall i : \Pi(\alpha_i) = \beta_i & (2) \end{cases}$

If $\exists i, j : \alpha_i = \alpha_j \wedge \beta_i \neq \beta_j$: output $(\mathsf{cm}, \alpha_i, \beta_i, \beta_j, \mathsf{pf}_i, \mathsf{pf}_j)$

Pick any i, choose any $\beta_i^* \neq \beta_i$, find $\vec{r}$ s.t. $\begin{bmatrix} 1 & 1 & 1 \\ \alpha_{i_1} & \alpha_{i_2} & \alpha_{i_3} \\ \beta_{i_1} & \beta_{i_2} & \beta_{i_3} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \alpha_i \\ \beta_i^* \end{bmatrix}$

invertible, as otherwise there is a linear function Π s.t. $\forall i : \Pi(\alpha_i) = \beta_i$

KZG function binding $\Leftarrow$ evaluation binding

Goal: given A_{FB}, construct A_{EB}

Proof sketch:

$\mathsf{cm}, ((\alpha_i, \beta_i, \mathsf{pf}_i))_{i\in[n]} \leftarrow A_{\mathsf{FB}}$ s.t. $\begin{cases} \forall i : \mathsf{Check}(\mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 & (1) \\ \nexists \Pi \in \mathbb{F}[X]^{\leq D} : \forall i : \Pi(\alpha_i) = \beta_i & (2) \end{cases}$

If $\exists i, j : \alpha_i = \alpha_j \wedge \beta_i \neq \beta_j$: output $(\mathsf{cm}, \alpha_i, \beta_i, \beta_j, \mathsf{pf}_i, \mathsf{pf}_j)$

Pick any i, choose any $\beta_i^* \neq \beta_i$, find $\vec{r}$ s.t. $\begin{bmatrix} 1 & 1 & 1 \\ \alpha_{i_1} & \alpha_{i_2} & \alpha_{i_3} \\ \beta_{i_1} & \beta_{i_2} & \beta_{i_3} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \alpha_i \\ \beta_i^* \end{bmatrix}$

invertible, as otherwise there is a linear function Π s.t. $\forall i : \Pi(\alpha_i) = \beta_i$

Homomorphism: $(\forall j : \mathsf{Check}(\mathsf{cm}, \alpha_{i_j}, \beta_{i_j}, \mathsf{pf}_{i_j}) = 1) \Rightarrow \mathsf{Check}(\mathsf{cm}, \sum_j r_j \alpha_{i_j}, \sum_j r_j \beta_{i_j}, \sum_j r_j \mathsf{pf}_{i_j}) = 1$

$\textcolor{blue}{\text{KZG}}$ function binding $\Leftarrow$ evaluation binding

Goal: given A_{FB}, construct A_{EB}

Proof sketch:

$$\text{cm}, ((\alpha_i, \beta_i, \text{pf}_i))_{i\in[n]} \leftarrow A_{\text{FB}} \text{ s.t. } \begin{cases} \forall i : \text{Check}(\text{cm}, \alpha_i, \beta_i, \text{pf}_i) = 1 & (1) \\ \nexists \Pi \in \mathbb{F}[X]^{\leq D} : \forall i : \Pi(\alpha_i) = \beta_i & (2) \end{cases}$$

If $\exists i, j : \alpha_i = \alpha_j \wedge \beta_i \neq \beta_j$: output $(\text{cm}, \alpha_i, \beta_i, \beta_j, \text{pf}_i, \text{pf}_j)$

Pick any i, choose any $\beta_i^* \neq \beta_i$, find $\vec{r}$ s.t. $\begin{bmatrix} 1 & 1 & 1 \\ \alpha_{i_1} & \alpha_{i_2} & \alpha_{i_3} \\ \beta_{i_1} & \beta_{i_2} & \beta_{i_3} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \alpha_i \\ \beta_i^* \end{bmatrix}$

invertible, as otherwise there is a linear function Π s.t. $\forall i : \Pi(\alpha_i) = \beta_i$

Homomorphism: $(\forall j : \text{Check}(\text{cm}, \alpha_{i_j}, \beta_{i_j}, \text{pf}_{i_j}) = 1) \Rightarrow \text{Check}(\text{cm}, \sum_j r_j \alpha_{i_j}, \sum_j r_j \beta_{i_j}, \sum_j r_j \text{pf}_{i_j}) = 1$

$$\Rightarrow \text{Check}(\text{cm}, \alpha_i, \beta_i^*, \textstyle\sum_j r_j \text{pf}_{i_j}) = \text{Check}(\sum_j r_j \text{cm}, \sum_j r_j \alpha_{i_j}, \sum_j r_j \beta_{i_j}, \sum_j r_j \text{pf}_{i_j}) = 1$$

KZG function binding $\Leftarrow$ evaluation binding

Goal: given A_{FB}, construct A_{EB}

Proof sketch:

$$\mathsf{cm}, ((\alpha_i, \beta_i, \mathsf{pf}_i))_{i\in[n]} \leftarrow A_{\mathsf{FB}} \text{ s.t. } \begin{cases} \forall i : \mathsf{Check}(\mathsf{cm}, \alpha_i, \beta_i, \mathsf{pf}_i) = 1 & (1) \\ \nexists \Pi \in \mathbb{F}[X]^{\leq D} : \forall i : \Pi(\alpha_i) = \beta_i & (2) \end{cases}$$

If $\exists i, j : \alpha_i = \alpha_j \wedge \beta_i \neq \beta_j$: output $(\mathsf{cm}, \alpha_i, \beta_i, \beta_j, \mathsf{pf}_i, \mathsf{pf}_j)$

Pick any i, choose any $\beta_i^* \neq \beta_i$, find $\vec{r}$ s.t. $\begin{bmatrix} 1 & 1 & 1 \\ \alpha_{i_1} & \alpha_{i_2} & \alpha_{i_3} \\ \beta_{i_1} & \beta_{i_2} & \beta_{i_3} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \alpha_i \\ \beta_i^* \end{bmatrix}$

invertible, as otherwise there is a linear function Π s.t. $\forall i : \Pi(\alpha_i) = \beta_i$

Homomorphism: $(\forall j : \mathsf{Check}(\mathsf{cm}, \alpha_{i_j}, \beta_{i_j}, \mathsf{pf}_{i_j}) = 1) \Rightarrow \mathsf{Check}(\mathsf{cm}, \sum_j r_j \alpha_{i_j}, \sum_j r_j \beta_{i_j}, \sum_j r_j \mathsf{pf}_{i_j}) = 1$

$$\Rightarrow \mathsf{Check}(\mathsf{cm}, \alpha_i, \beta_i^*, \underbrace{\textstyle\sum_j r_j \mathsf{pf}_{i_j}}) = \mathsf{Check}(\textstyle\sum_j r_j \mathsf{cm}, \sum_j r_j \alpha_{i_j}, \sum_j r_j \beta_{i_j}, \sum_j r_j \mathsf{pf}_{i_j}) = 1$$

Output $(\mathsf{cm}, \alpha_i, \beta_i, \beta_i^*, \mathsf{pf}_i, \mathsf{pf}_i^*)$