Non-Interactive Batch Arguments

and more

Arka Rai Choudhuri

University of California, Berkeley

Abhishek Jain

Johns Hopkins University

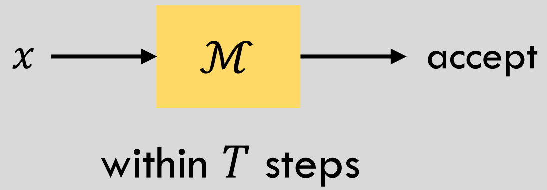
Zhengzhong Jin

Johns Hopkins University

Succinct Non-Interactive Arguments (SNARGs)

$\mathcal{M}$, x

$\mathcal{M}$, x



Succinct Non-Interactive Arguments (SNARGs)

$\mathcal{M}$, x

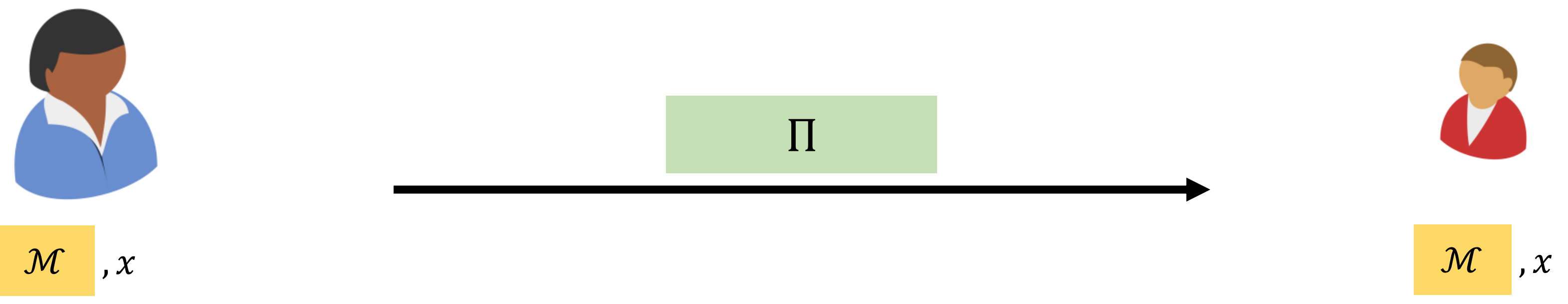
$\mathcal{M}$, x

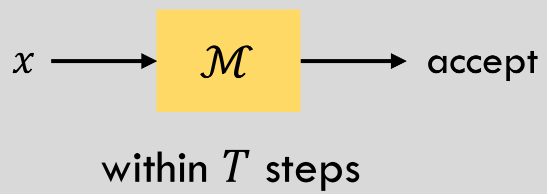
$x \longrightarrow \mathcal{M} \longrightarrow$ accept

within T steps

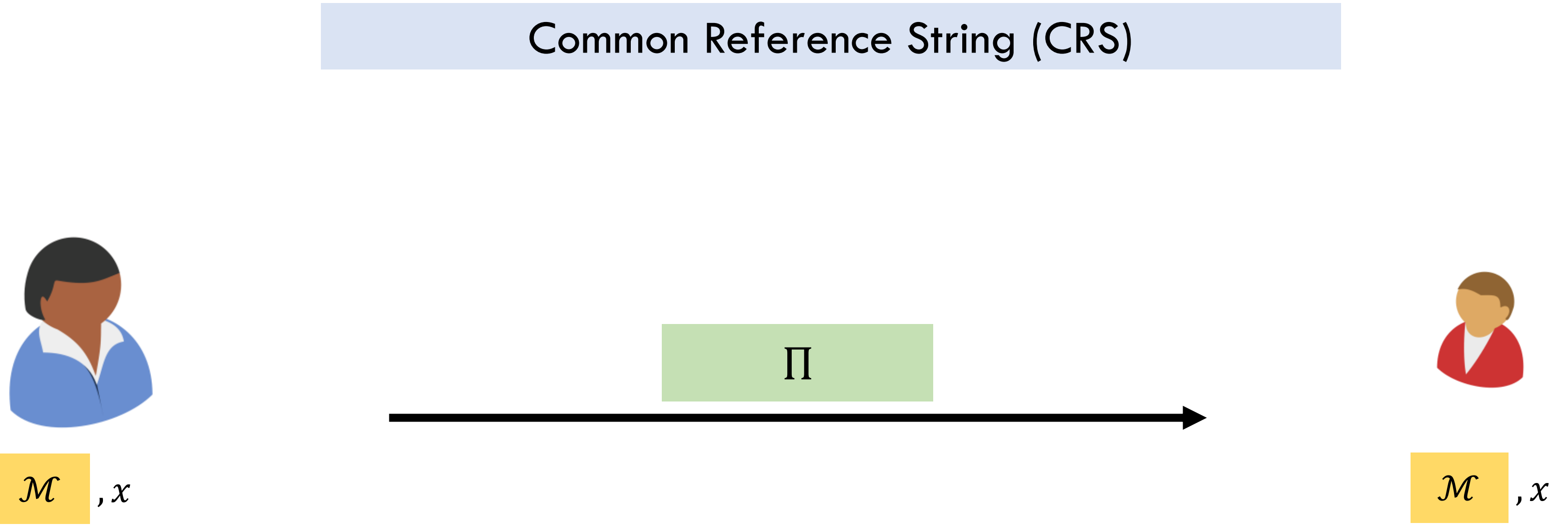
wants to delegate computation to

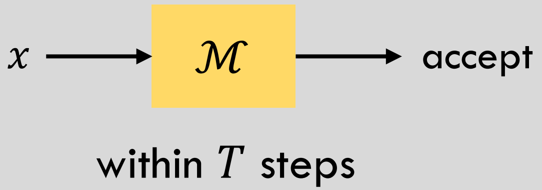
Succinct Non-Interactive Arguments (SNARGs)





Succinct Non-Interactive Arguments (SNARGs)





Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

Π

$\mathcal{M}$, x

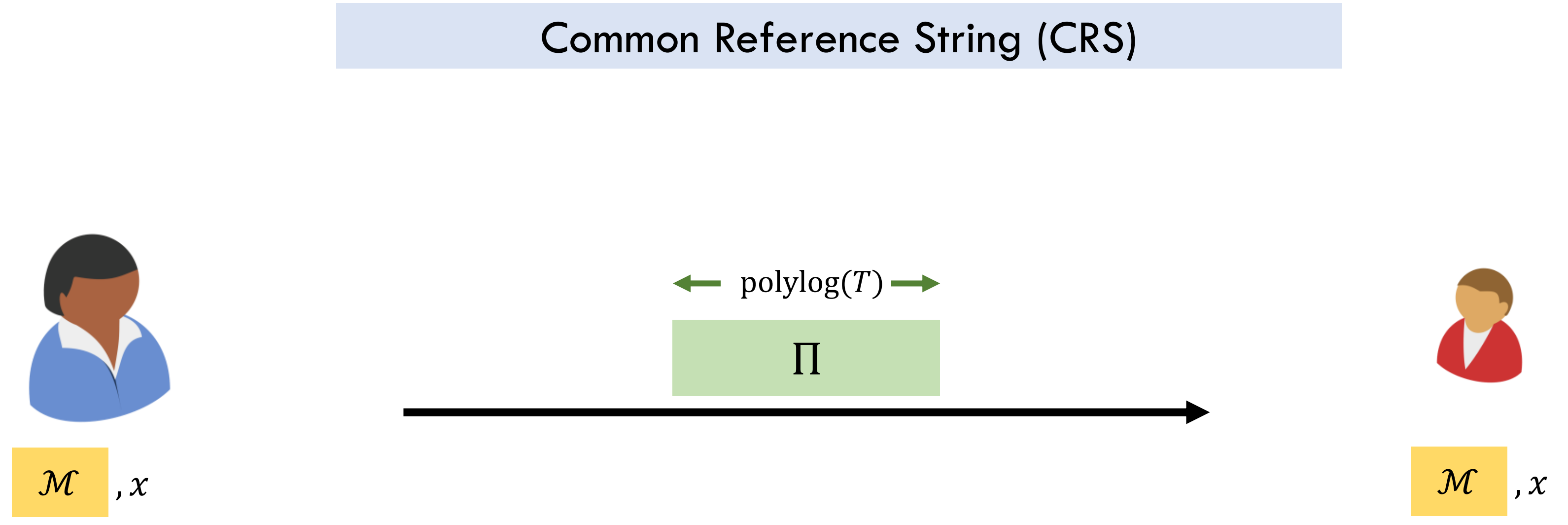
$\mathcal{M}$, x

Π is publicly verifiable

$x \longrightarrow \mathcal{M} \longrightarrow$ accept

within T steps

Succinct Non-Interactive Arguments (SNARGs)



Verifier running time: polylog(T)

Π is publicly verifiable

$x \rightarrow \mathcal{M} \rightarrow$ accept

within T steps

Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

$\mathcal{M}$, x

← polylog(T) →

Π

$\mathcal{M}$, x

Verifier running time: polylog(T)

Π is publicly verifiable

x → $\mathcal{M}$ → accept

within T steps

No PPT  can produce accepting Π if

x → $\mathcal{M}$ ✗→ accept

within T steps

Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

$\mathcal{M}$, x

$\leftarrow$ polylog(T) $\rightarrow$

Π

$\mathcal{M}$, x

Verifier running time: polylog(T)

Π is publicly verifiable

$x \longrightarrow \mathcal{M} \longrightarrow$ accept

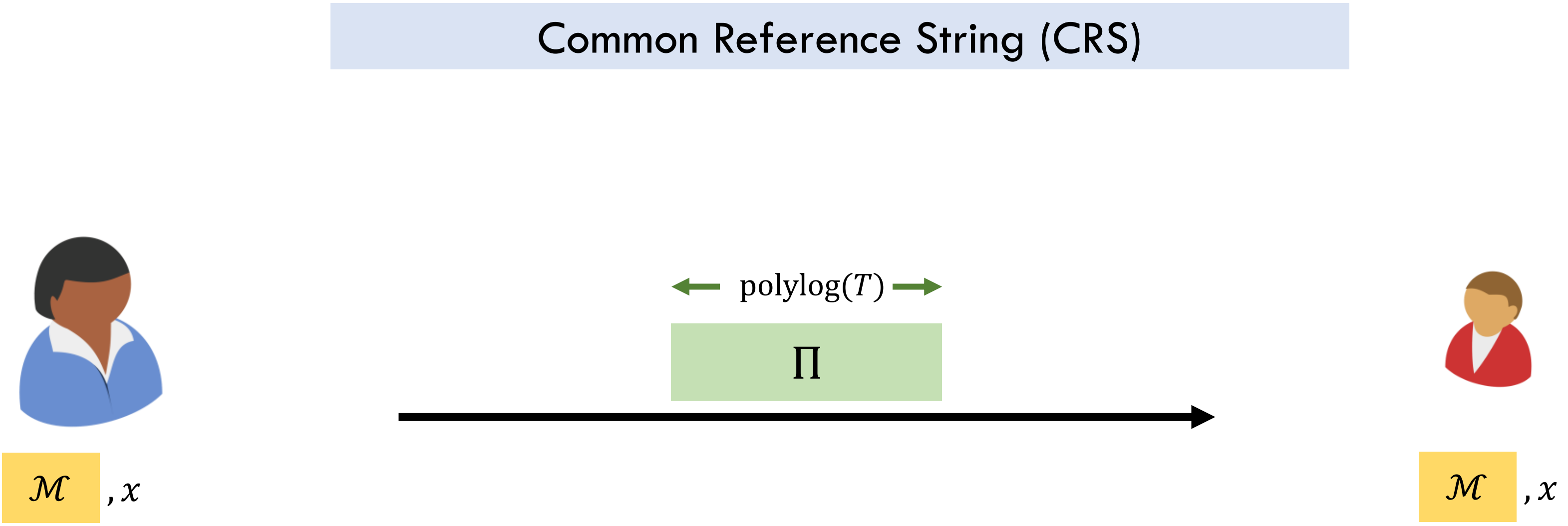
within T steps

No PPT  can produce accepting x, Π if

$x \longrightarrow \mathcal{M} \not\longrightarrow$ accept

within T steps

Succinct Non-Interactive Arguments (SNARGs)



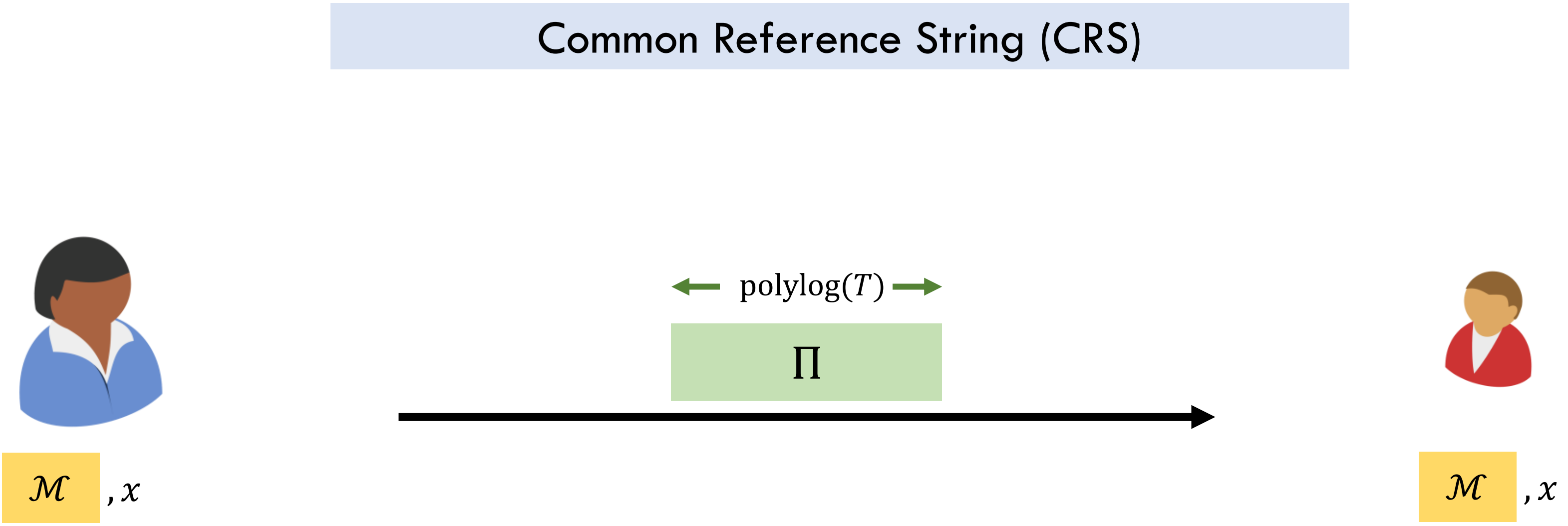
Verifier running time: polylog(T)

Π is publicly verifiable

What kind of computation can we hope to delegate based on standard assumptions?

x

Succinct Non-Interactive Arguments (SNARGs)



Verifier running time: polylog(T)

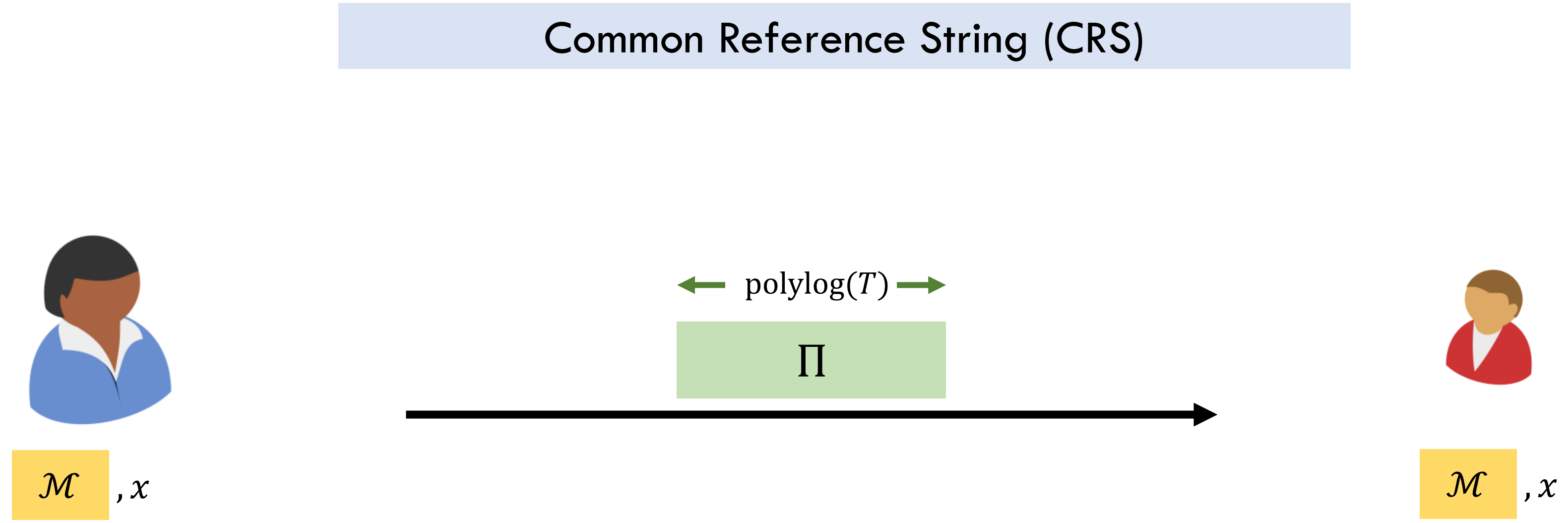
Π is publicly verifiable

What kind of computation can we hope to delegate based on standard assumptions?

- Nondeterministic polynomial-time computation (NP)? Unlikely! [Gentry-Wichs'11]

x

Succinct Non-Interactive Arguments (SNARGs)



Verifier running time: polylog(T)

Π is publicly verifiable

What kind of computation can we hope to delegate based on standard assumptions?

- Nondeterministic polynomial-time computation (NP)? Unlikely! [Gentry-Wichs'11]
- Deterministic polynomial-time computation (P)?

x

Prior Works

Prior Works

Non-falsifiable assumptions/ Random oracle model

[Micali'94, Groth'10, Lipmaa'12, Damgård-Faust-Hazay'12, Gennaro-Gentry-Parno-Raykova'13, Bitansky-Chiesa-Ishai-Ostrovsky-Paneth'13, Bitansky-Canetti-Chiesa-Tromer'13, Bitanksy-Canetti-Chiesa-Goldwasser-Lin-Rubinstein-Tromer'17]

Some works can delegate NP

Prior Works

Non-falsifiable assumptions/ Random oracle model

[Micali'94, Groth'10, Lipmaa'12, Damgård-Faust-Hazay'12, Gennaro-Gentry-Parno-Raykova'13, Bitansky-Chiesa-Ishai-Ostrovsky-Paneth'13, Bitansky-Canetti-Chiesa-Tromer'13, Bitanksy-Canetti-Chiesa-Goldwasser-Lin-Rubinstein-Tromer'17]

Some works can delegate NP

"Less standard" assumptions

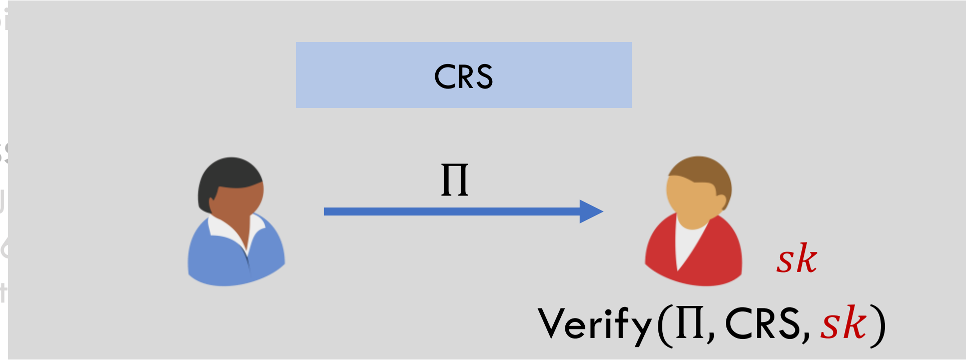
[Canetti-Holmgren-Jain-Vaikuntanathan'15, Koppula-Lewko-Waters'15, Bitansky-Garg-Lin-Pass-Telang'15, Canetti-Holmgren'16, Ananth-Chen-Chung-Lin-Lin'16, Chen-Chow-Chung-Lai-Lin-Zhou'16, Paneth-Rothblum'17, Canetti-Chen-Holmgren-Lombardi-Rothblum-Rothblum-Wichs'19, Kalai-Paneth-Yang'19]

Delegation for P

Prior Works

Non-falsifiable assumptions/ Random oracle model

[Micali'94, Groth'10, Lipmaa'12, Damgård-Faust-Hazay'12, Gennaro-Gentry-Parno-Raykova'13, Bitansky-Chiesa-Ishai-Ostrovsky-Paneth'13, Bitansky-Canetti-Chiesa-Tromer'13, Bitanksy-Canetti-Chiesa-Goldwasser-Lin-Rubi



Some works can delegate NP

"Less standard" as

[Canetti-Holmgren-J ... rg-Lin-Pass-Telang'15, Canetti-Holmgren'1 ... '16, Paneth-Rothblum'17, Canett ... Paneth-Yang'19]

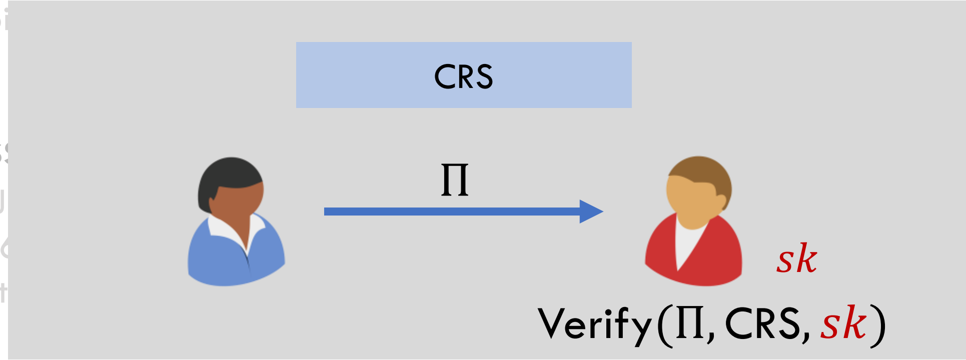
Delegation for P

Designated Verifier (standard assumptions)

Prior Works

Non-falsifiable assumptions/ Random oracle model

[Micali'94, Groth'10, Lipmaa'12, Damgård-Faust-Hazay'12, Gennaro-Gentry-Parno-Raykova'13, Bitansky-Chiesa-Ishai-Ostrovsky-Paneth'13, Bitansky-Canetti-Chiesa-Tromer'13, Bitanksy-Canetti-Chiesa-Goldwasser-Lin-Rubi



"Less standard" as

[Canetti-Holmgren-J ... rg-Lin-Pass-Telang'15, Canetti-Holmgren'1 ... '16, Paneth-Rothblum'17, Canett ... Paneth-Yang'19]

Designated Verifier (standard assumptions)

[Kalai-Raz-Rothblum'13, Kalai-Raz-Rothblum'14, Kalai-Paneth'16, Brakerski-Holgren-Kalai'17, Badrinarayanan-Kalai-Khurana-Sahai-Wichs'18, Holmgren-Rothblum'18, Brakerski-Kalai'20]

Some works can delegate NP

Delegation for P

Delegation for P

Do there exists SNARGs for P based on standard assumptions?

Previously best known: [Jawale-Kalai-Khurana-Zhang'21] for depth bounded computation based on sub-exponential hardness of LWE.

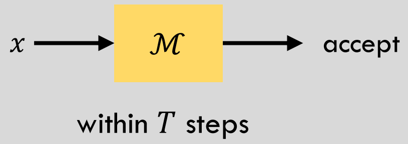
Builds on [Canetti-Chen-Holmgren-Lombardi-Rothblum-Rothblum-Wichs'19]

Our Result

Theorem

Assuming LWE there exists a SNARG for P where

$|\mathsf{CRS}|, |\Pi|, |$  $| = \mathrm{polylog}(T)$



LWE – Learning with Errors

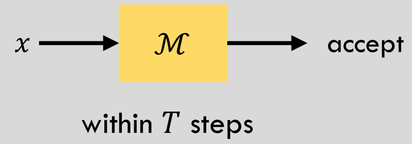
Our Result

Theorem

Assuming LWE there exists a SNARG for P where

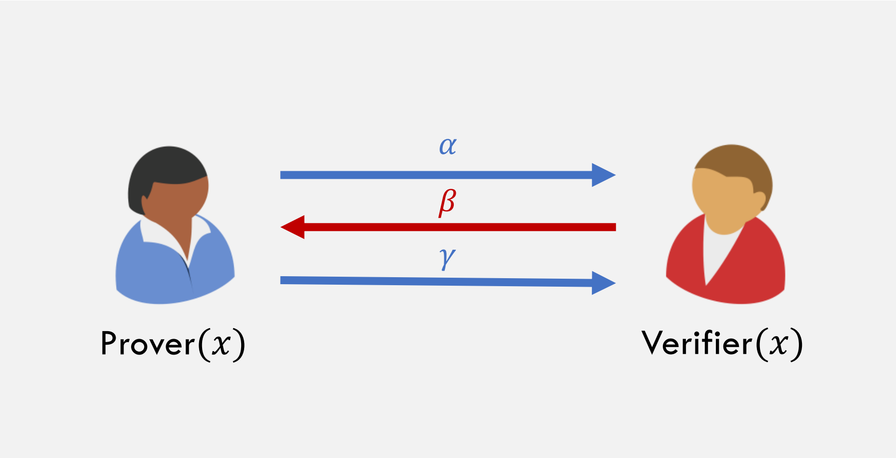
$|\mathsf{CRS}|, |\Pi|,$ |  | $= \mathrm{polylog}(T)$

LWE – Learning with Errors



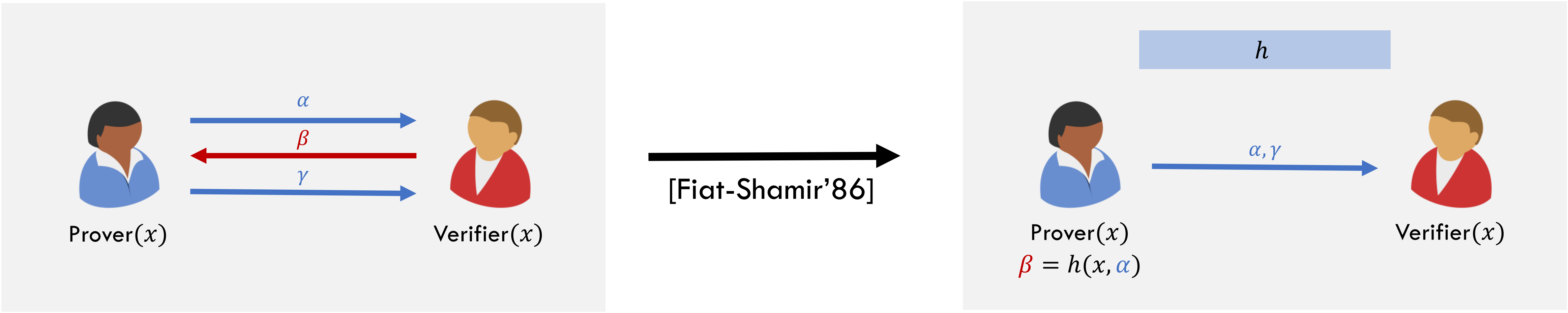
Tool: Fiat-Shamir (FS) Methodology

Fiat-Shamir (FS) Methodology



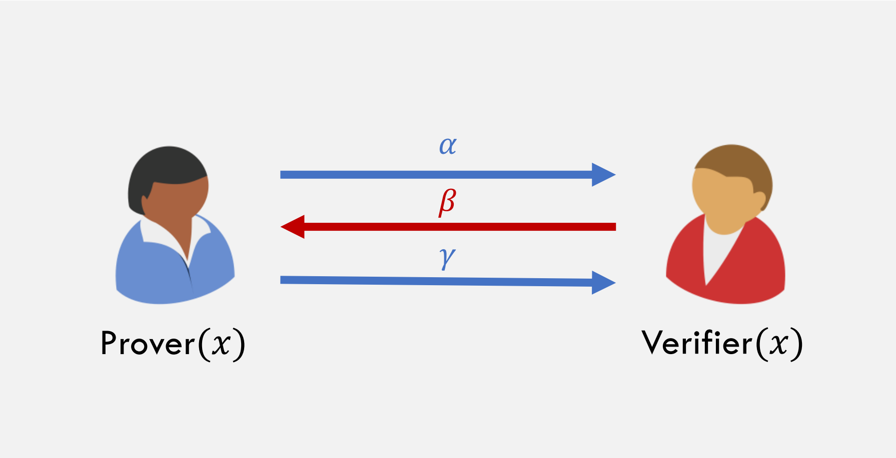
β is a random string

Fiat-Shamir (FS) Methodology



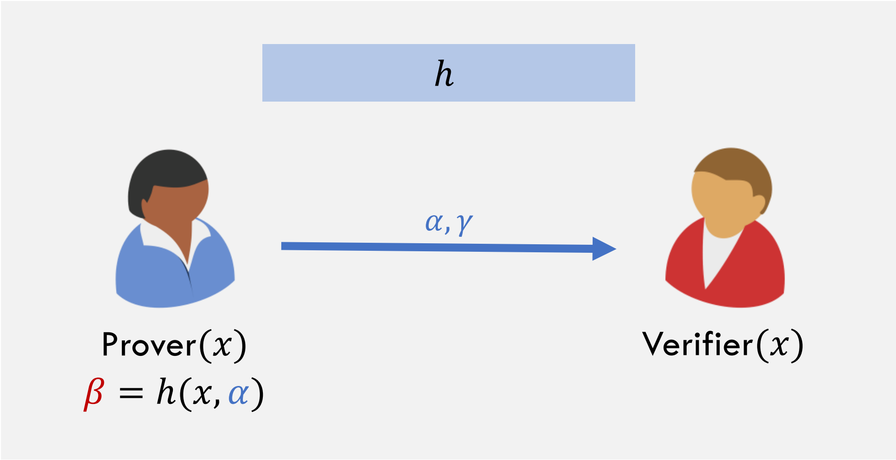
β is a random string

Fiat-Shamir (FS) Methodology



β is a random string

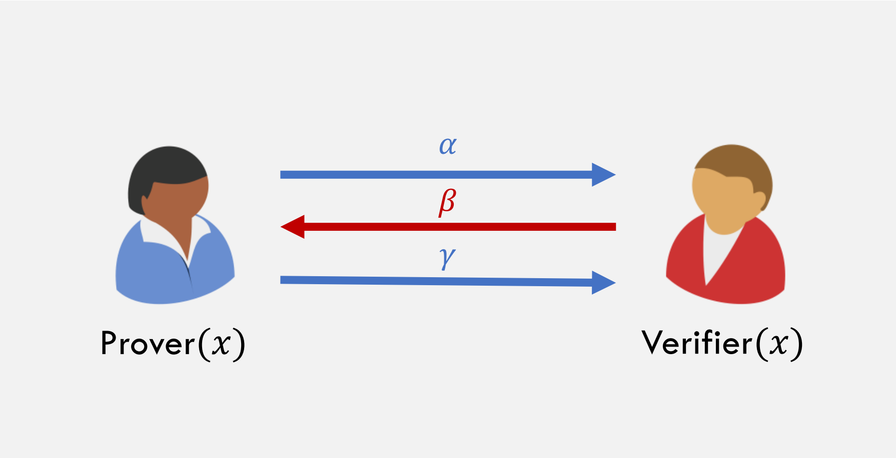
[Fiat-Shamir'86]



$\forall\, x \notin \mathcal{L}$

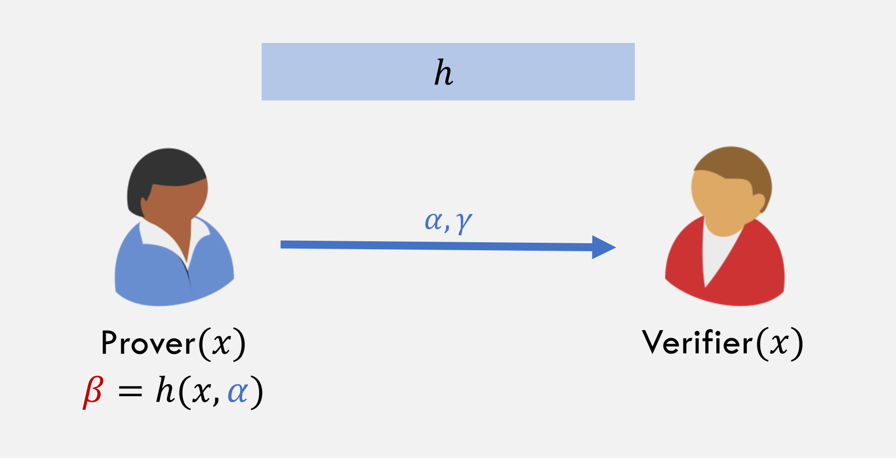
$$\mathrm{BAD}_{x,\alpha} = \{\beta \mid \exists \gamma \text{ s.t. Verifier accepts } (\alpha, \beta, \gamma)\}$$

Fiat-Shamir (FS) Methodology



β is a random string

[Fiat-Shamir'86]



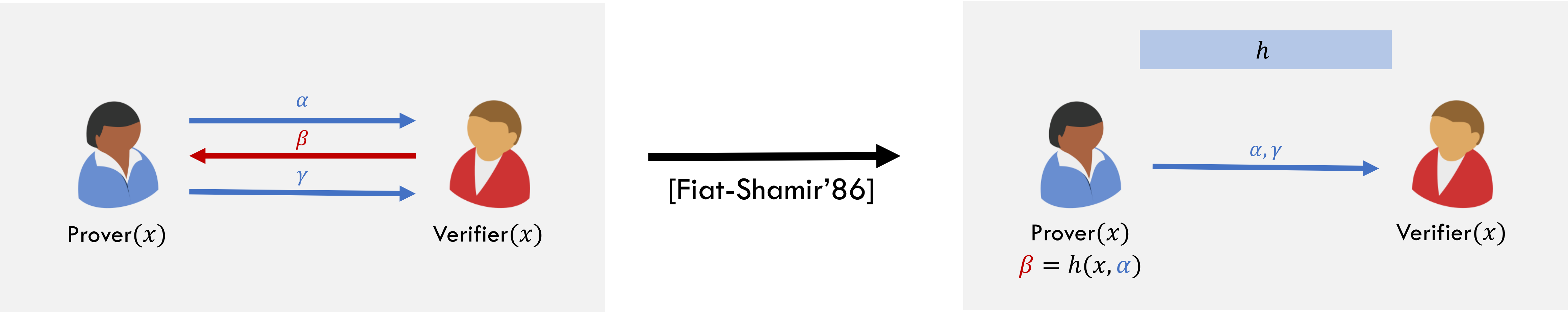
$\forall\, x \notin \mathcal{L}$

$\mathrm{BAD}_{x,\alpha} = \{\beta \mid \exists \gamma$ s.t. Verifier accepts $(\alpha, \beta, \gamma)\}$

If $x \notin \mathcal{L}$, no PPT  can find α such that

$$h(x, \alpha) \in \mathrm{BAD}_{x,\alpha}$$

Correlation Intractability [Canetti-Goldreich-Halevi'98]



β is a random string

$\forall\, x \notin \mathcal{L}$

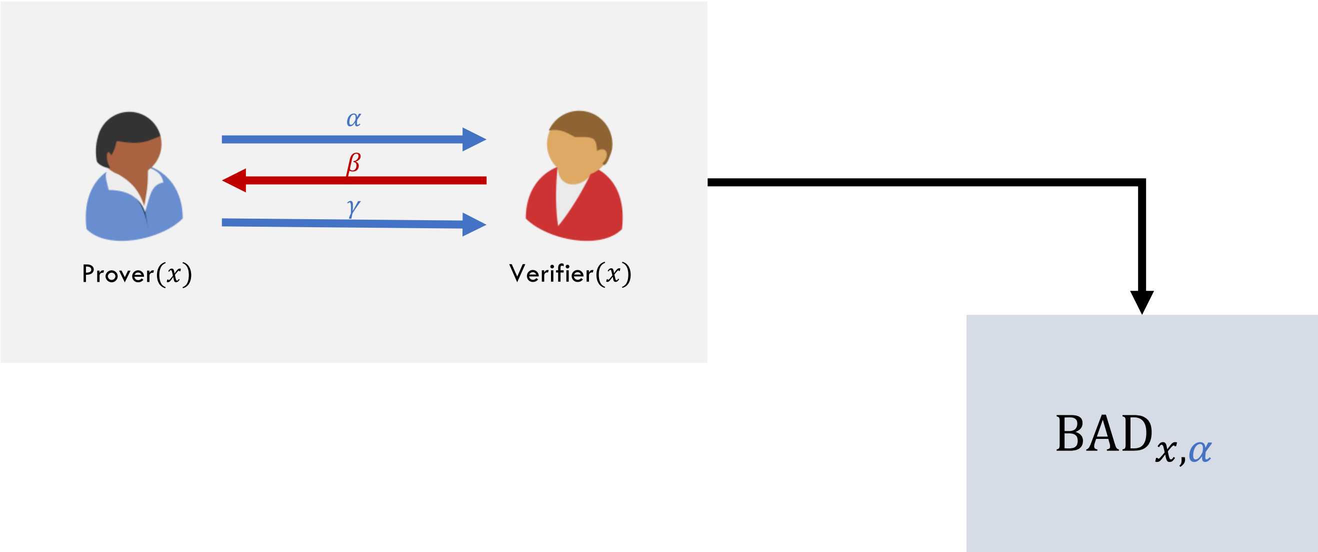
$\mathrm{BAD}_{x,\alpha} = \{\beta \mid \exists \gamma$ s.t. Verifier accepts $(\alpha, \beta, \gamma)\}$

If $x \notin \mathcal{L}$, no PPT  can find α such that

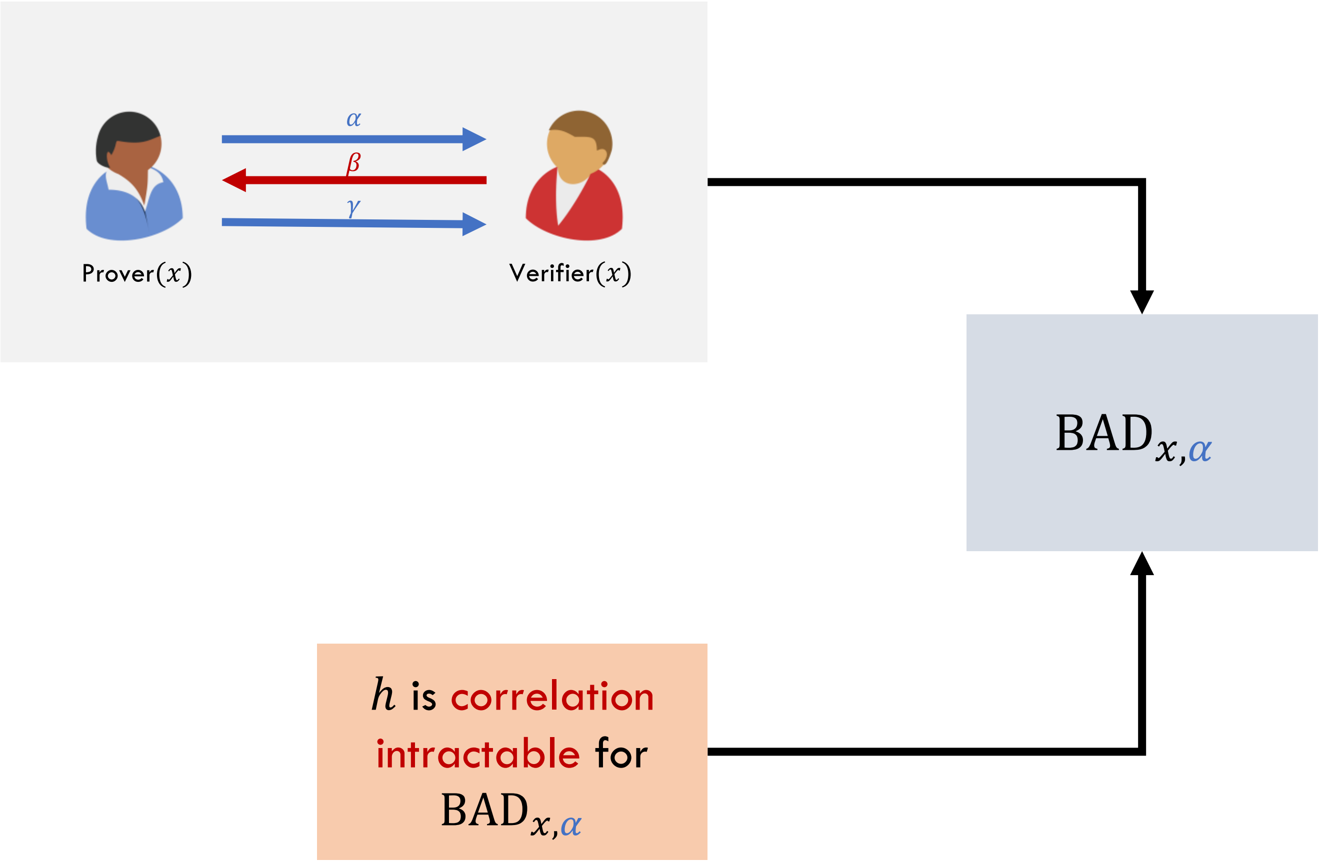
$$h(x, \alpha) \in \mathrm{BAD}_{x,\alpha}$$

h is **correlation intractable (CI)** for $\mathrm{BAD}_{x,\alpha}$

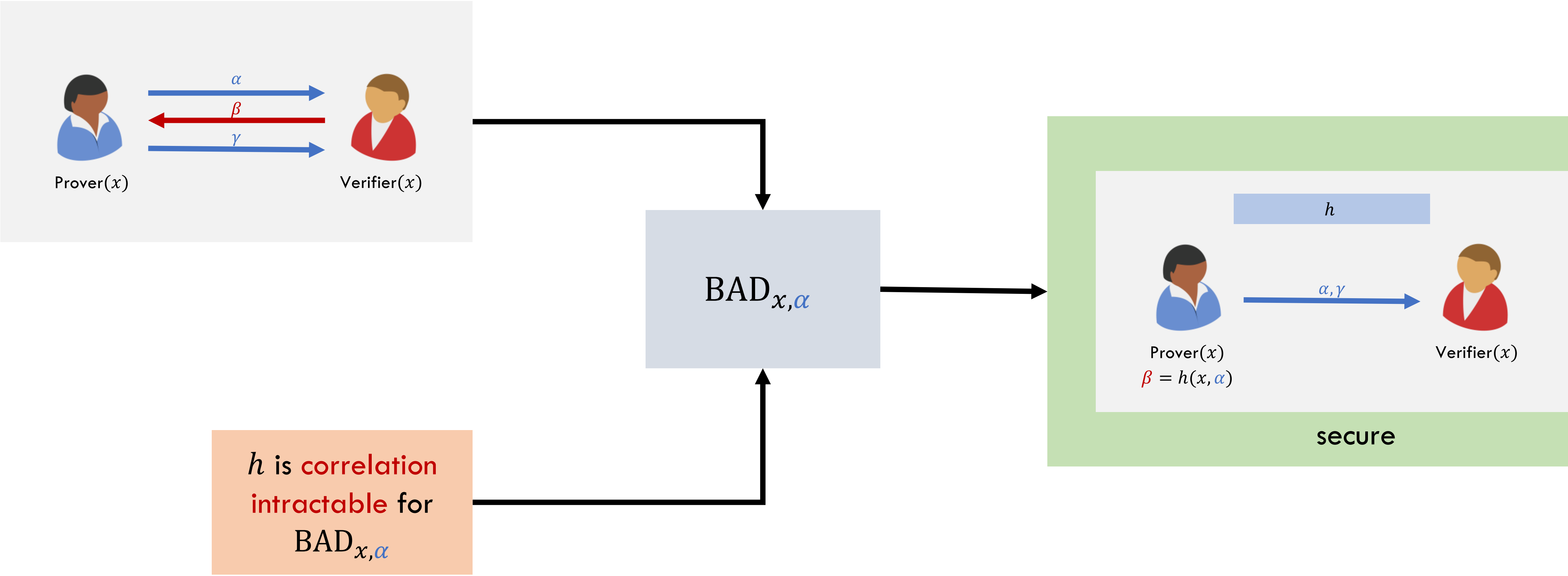
Instantiating the FS Transform



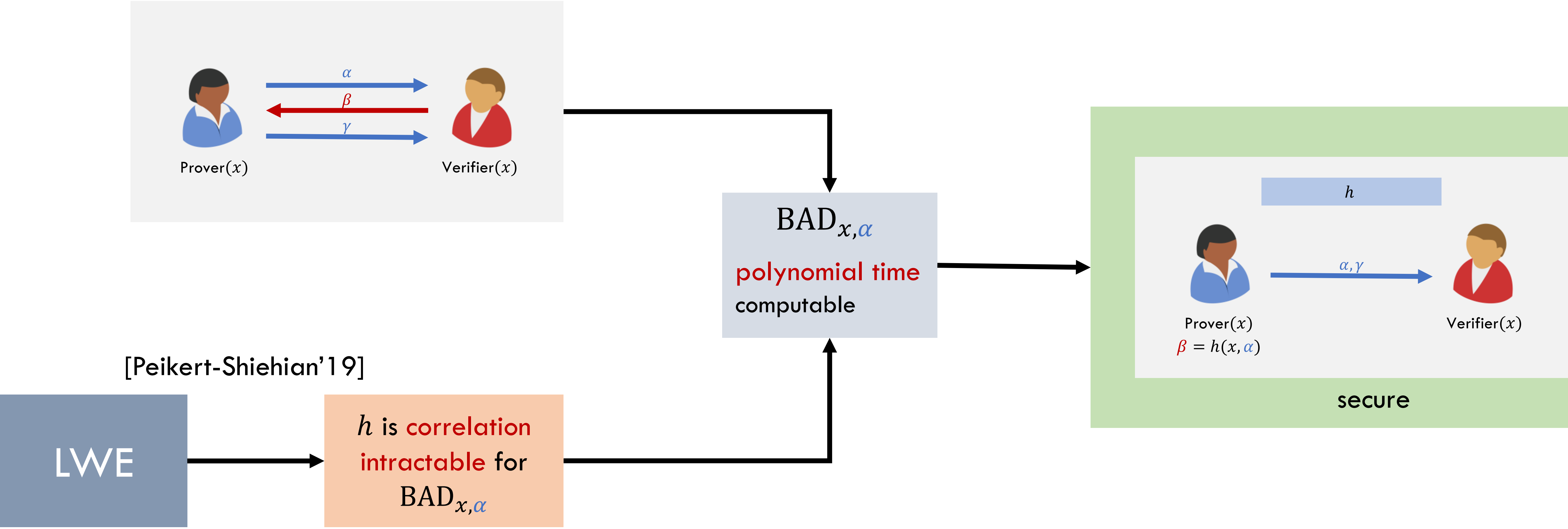
Instantiating the FS Transform



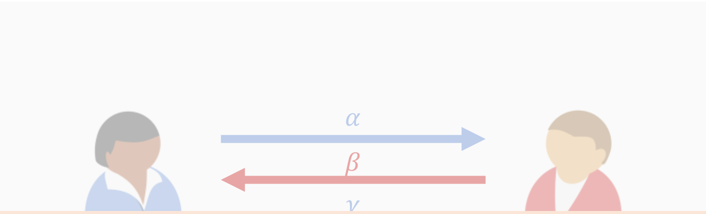
Instantiating the FS Transform



Instantiating the FS Transform



Instantiating the FS Transform



FS methodology is secure for certain protocols under a variety of assumptions (via correlation intractable hash functions)

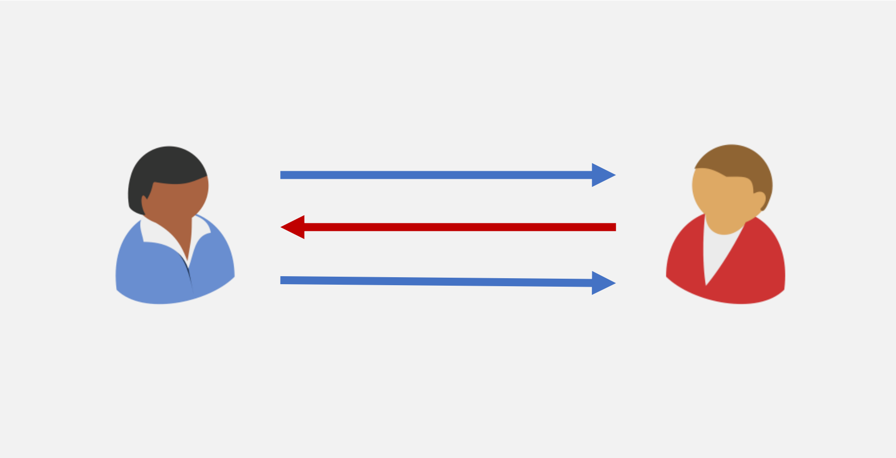
[Kalai-Rothblum-Rothblum'17, Canetti-Chen-Reyzin-Rothblum'18, Holmgren-Lombardi'18, Canetti-Chen-Holmgren-Lombardi-Rothblum-Rothblum-Wichs'19, Peikert-Sheihian'19, Brakerski-Koppula-Mour'20, Couteau-Katsumata-Ursu'20, Jain-Jin'21, Jawale-Kalai-Khurana-Zhang'21, Holmgren-Lombardi-Rothblum'21]

secure

LWE

h is correlation intractable for $\mathrm{BAD}_{x,\alpha}$

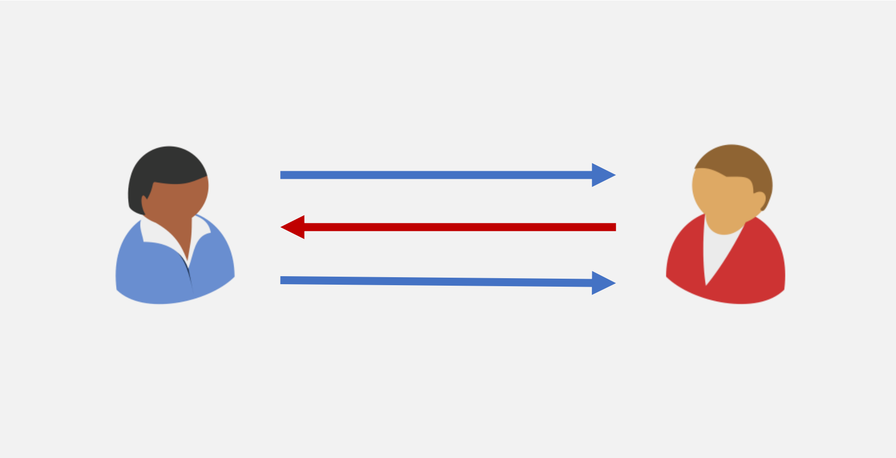
Fiat-Shamir (FS) Methodology



[Kilian'92]

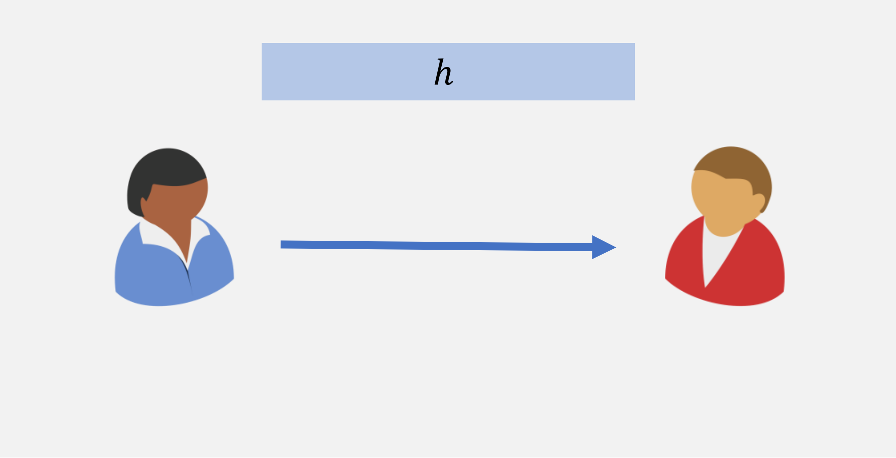
Succinct interactive arguments for all polynomial time computation.

Fiat-Shamir (FS) Methodology



[Kilian'92]

Instantiate with CIH

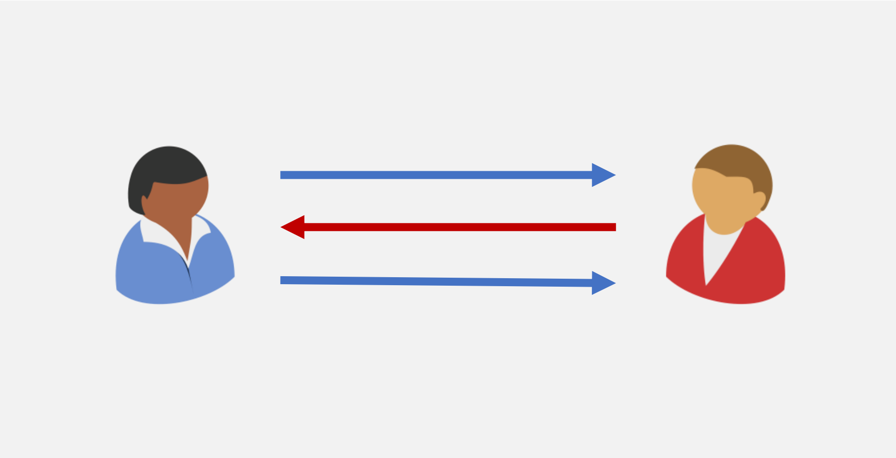


Succinct interactive arguments for all polynomial time computation.

[Bartusek-Bronfman-Holmgren-Ma-Rothblum'19]

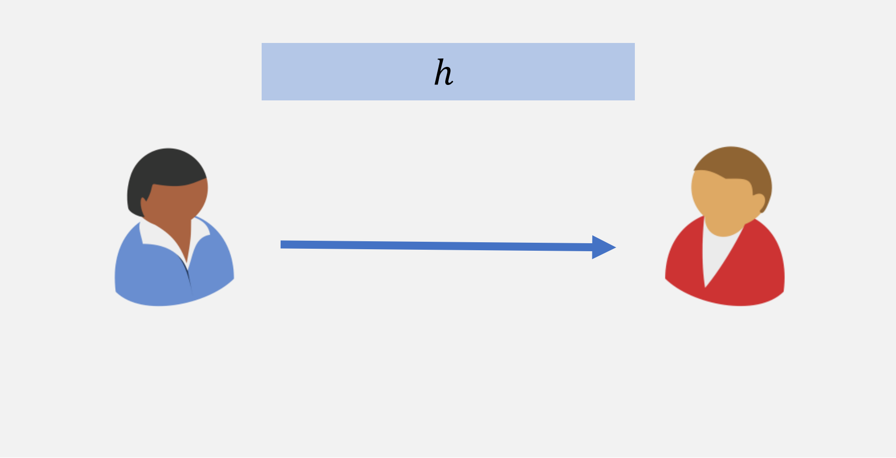
Instantiating hash function for Fiat-Shamir transformation of Kilian's protocol is hard.

Fiat-Shamir (FS) Methodology



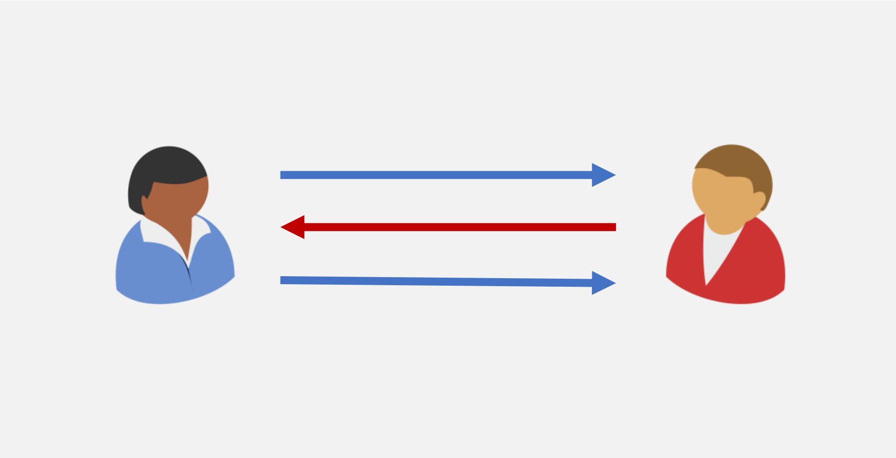
[Kilian'92]

Instantiate with CIH



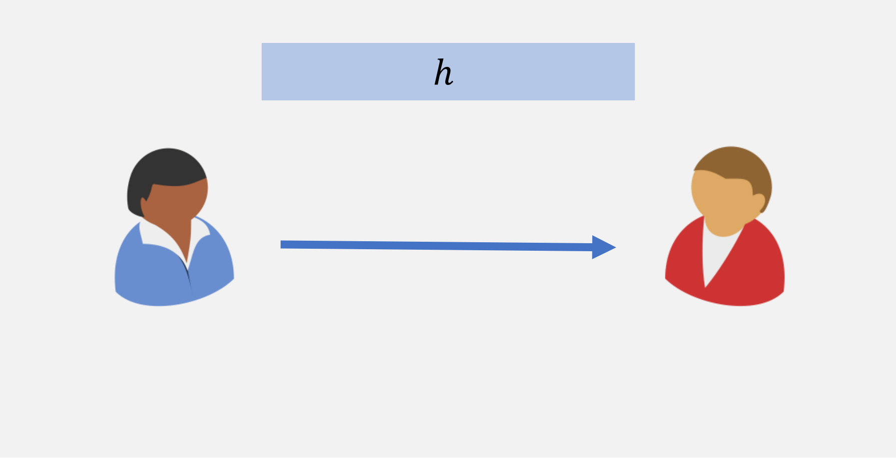
Succinct interactive arguments for all polynomial time computation.

Fiat-Shamir (FS) Methodology



[Kilian'92]

Instantiate with CIH

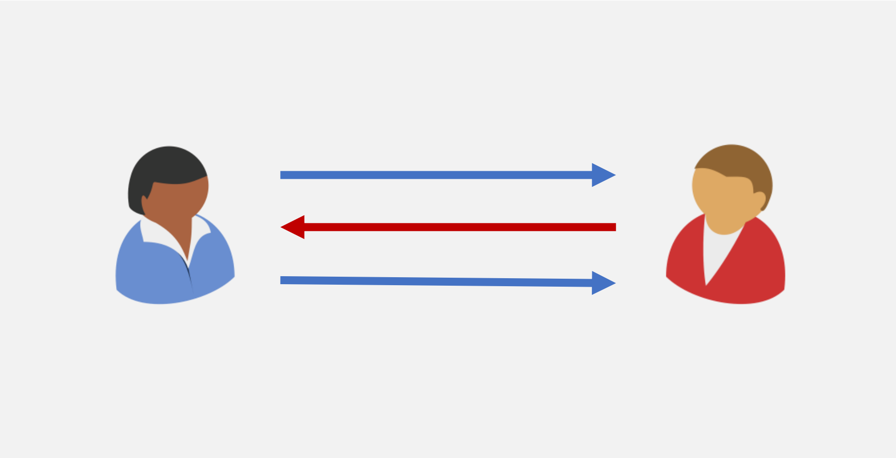


Succinct interactive arguments for all polynomial time computation.

Known instantiations of CI Hash for Fiat-Shamir transform are for proofs.

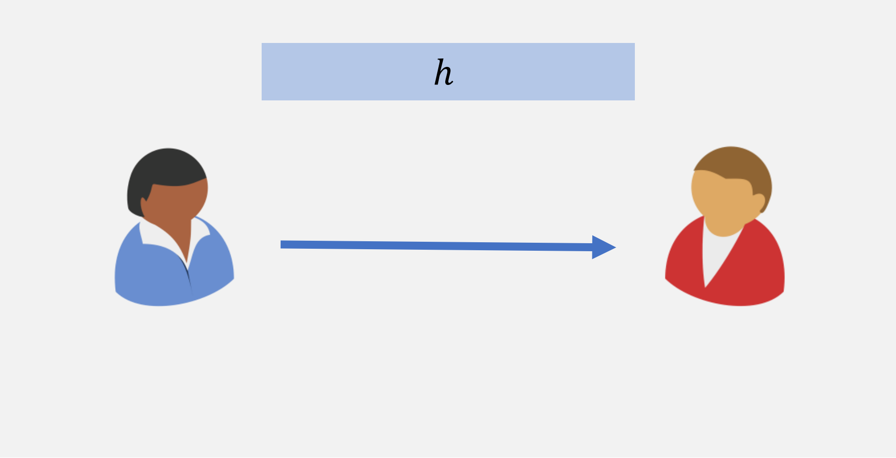
[Canetti-Sarkar-Wang'20] instantiate Fiat-Shamir transform for specific Sigma protocol that is an argument.

Fiat-Shamir (FS) Methodology



[Goldwasser-Kalai-Rothblum'08]

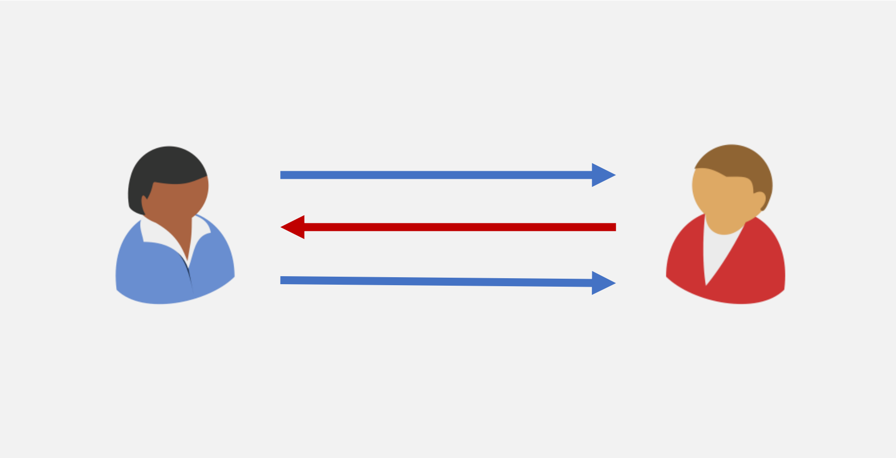
Instantiate with CIH



Succinct interactive proof for depth bounded computation.

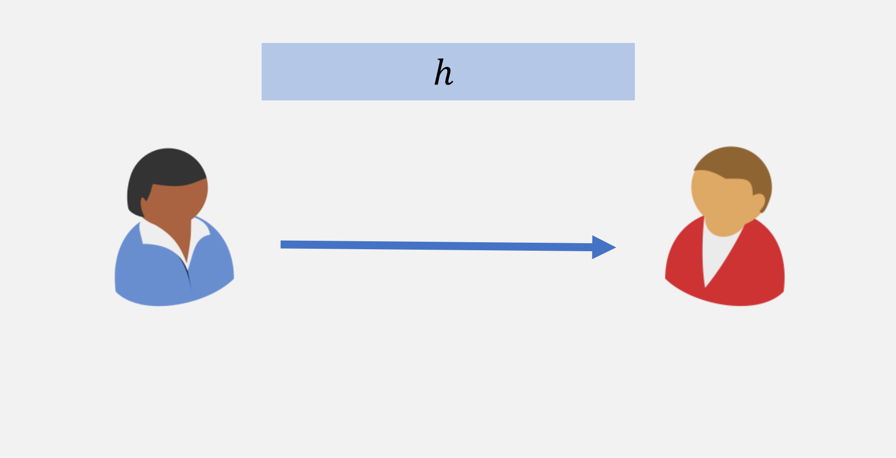
[Canetti-Chen-Holmgren-Lombardi-Rothblum-Rothblum-Wichs'19, Jawale-Kalai-Khurana-Zhang'21]

Fiat-Shamir (FS) Methodology



[Goldwasser-Kalai-Rothblum'08]

Instantiate with CIH

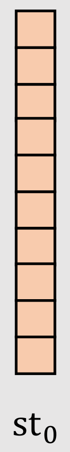


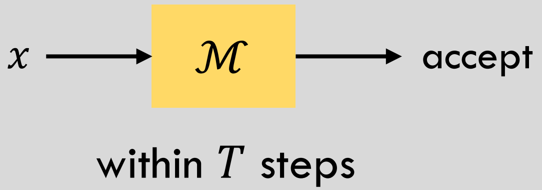
Succinct interactive proof for depth bounded computation.

[Canetti-Chen-Holmgren-Lombardi-Rothblum-Rothblum-Wichs'19, Jawale-Kalai-Khurana-Zhang'21]

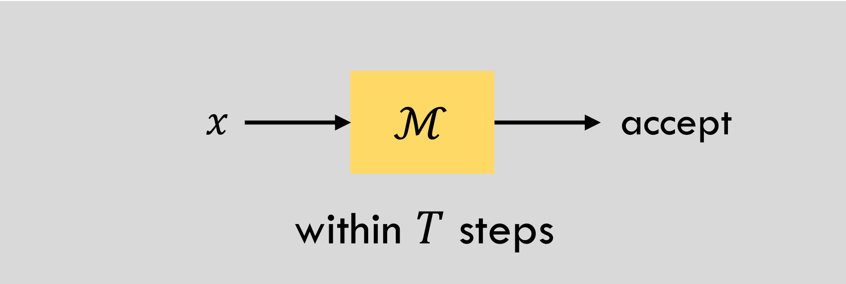
Interactive proofs for all polynomial time computation unlikely to exist.

Delegation via Batching [Reingold-Rothblum-Rothblum'16]

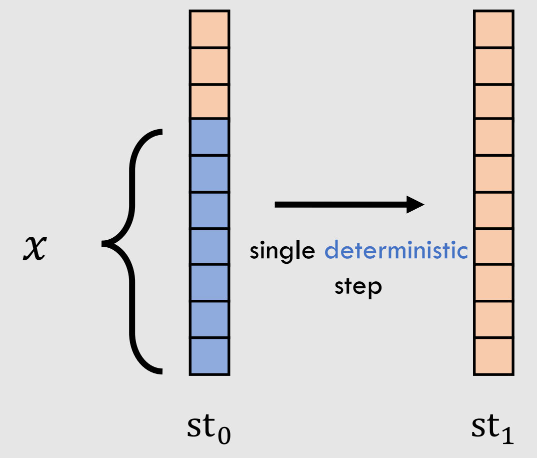


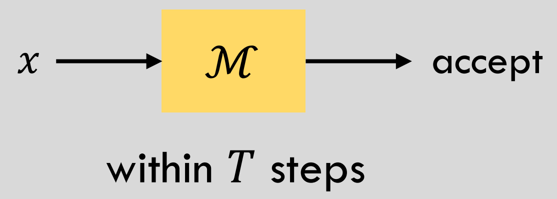


Delegation via Batching [Reingold-Rothblum-Rothblum'16]

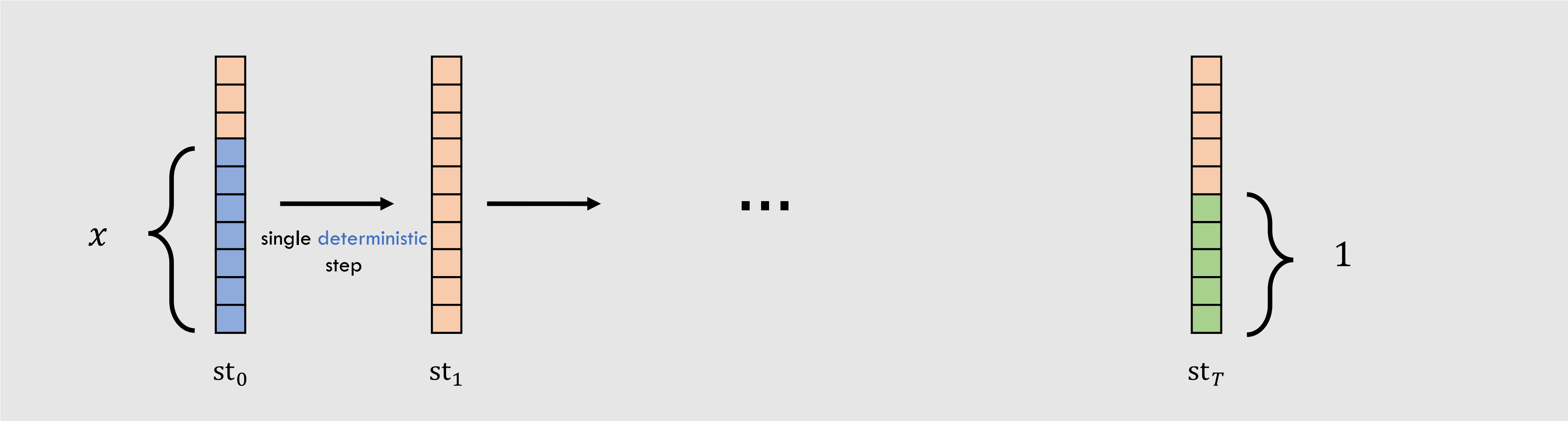


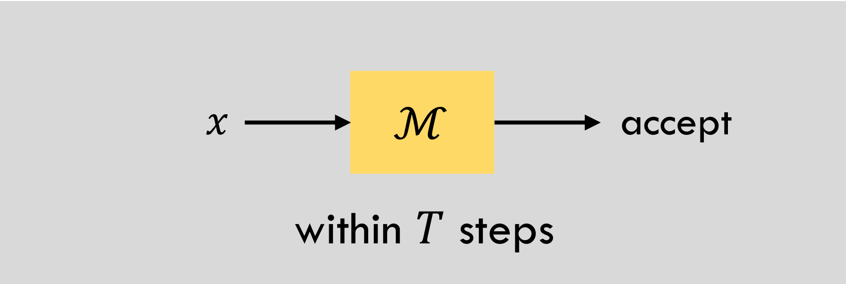
Delegation via Batching [Reingold-Rothblum-Rothblum'16]



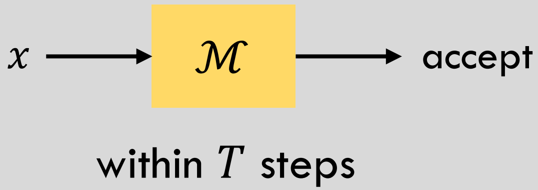


Delegation via Batching [Reingold-Rothblum-Rothblum'16]



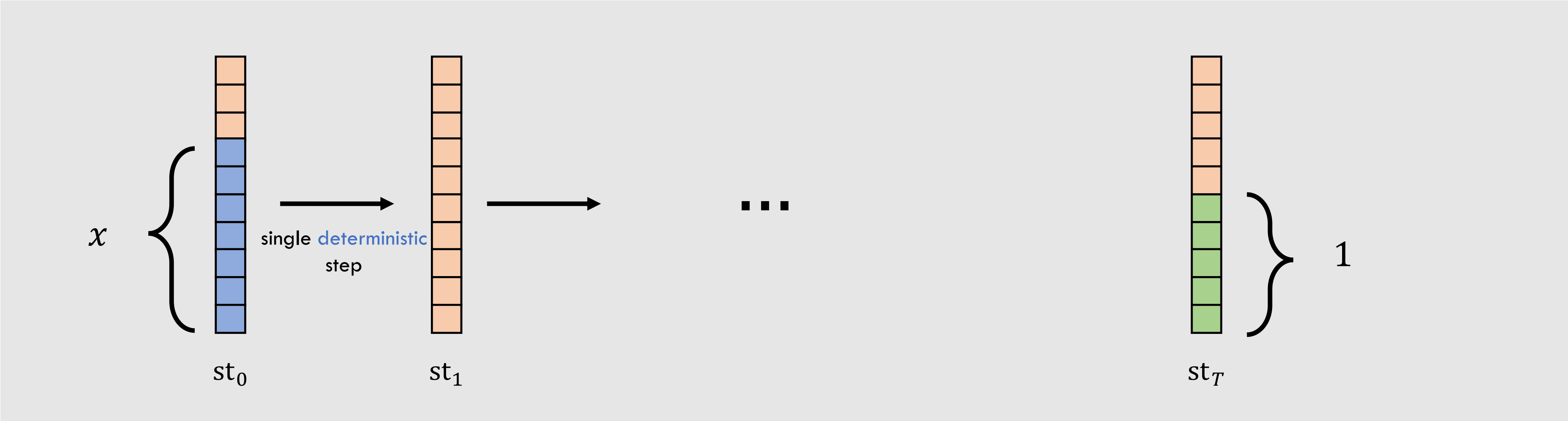


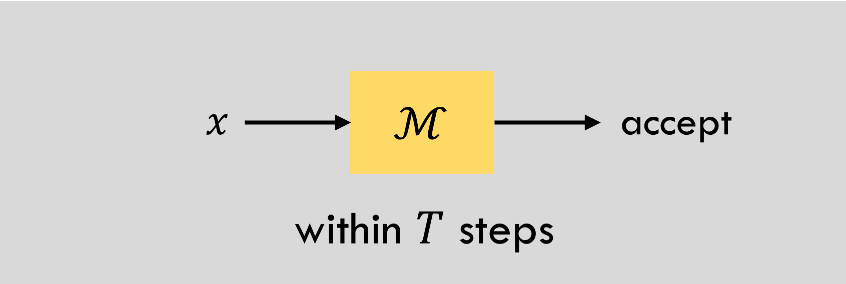
Delegation via Batching [Reingold-Rothblum-Rothblum'16]



This talk: Bounded space computation

Delegation via Batching [Reingold-Rothblum-Rothblum'16]





Prove for every $i \in [0, \ldots, T-1]$

$$\mathrm{st}_i \to \mathrm{st}_{i+1}$$

is the correct transition.

SNARGs for Batch NP

CRS

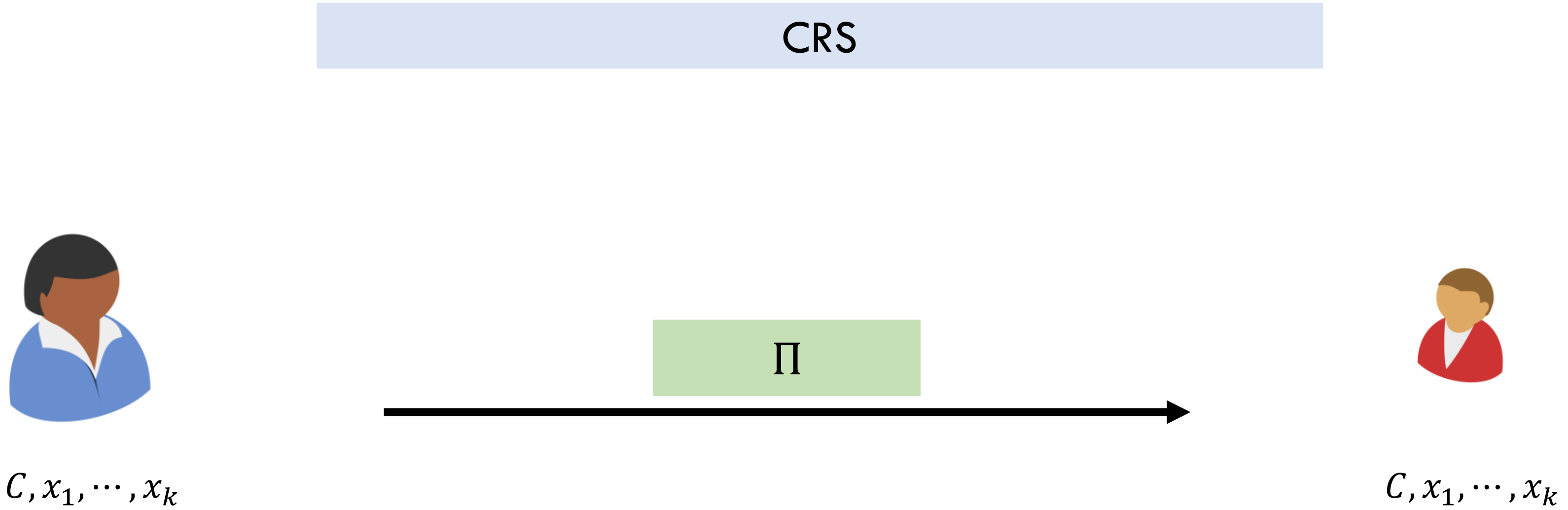
$C, x_1, \cdots, x_k$

$C, x_1, \cdots, x_k$

$\text{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$

SNARGs for Batch NP



Π is publicly verifiable

$$\text{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$$

$$\forall i \in [k], (C, x_i) \in \text{SAT}$$

SNARGs for Batch NP

CRS

$C, x_1, \cdots, x_k$

Π

$C, x_1, \cdots, x_k$

Π is publicly verifiable

$\text{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$

No PPT  can produce accepting Π if

$\exists i^* \in [k], (C, x_{i^*}) \notin \text{SAT}$

SNARGs for Batch NP

CRS

Π

$C, x_1, \cdots, x_k$

$C, x_1, \cdots, x_k$

Π is publicly verifiable

$\text{SAT} = \{(C, x) \mid \exists w \ s.t.\ C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$

SNARGs for Batch NP

CRS

$C, x_1, \cdots, x_k$

$w_1, \ldots, w_k$

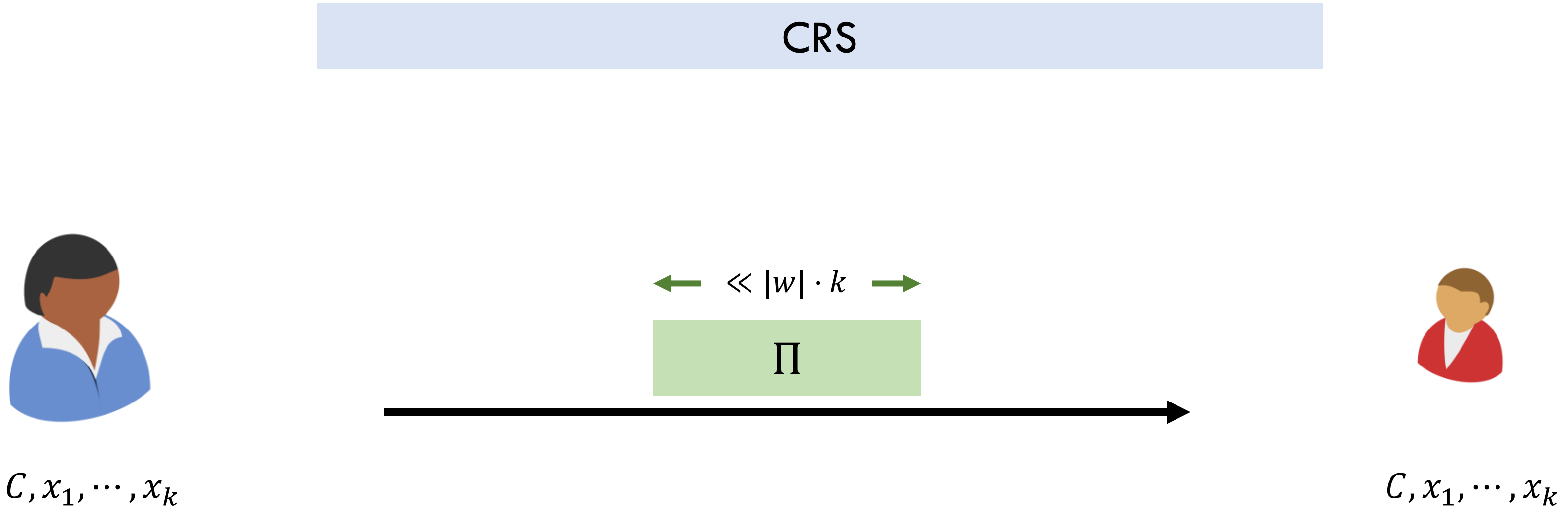
$C, x_1, \cdots, x_k$

Π is publicly verifiable

$\text{SAT} = \{(C, x) \mid \exists w \ s.t. \ C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$

SNARGs for Batch NP



Π is publicly verifiable

$\mathrm{SAT} = \{(C, x) \mid \exists w \ s.t. \ C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \mathrm{SAT}$

SNARGs for Batch NP

CRS

$C, x_1, \cdots, x_k$

$\ll |w| \cdot k$

Π

$C, x_1, \cdots, x_k$

Verifier running time:
$k \cdot |x| + |\Pi|$

Π is publicly verifiable

$\text{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$

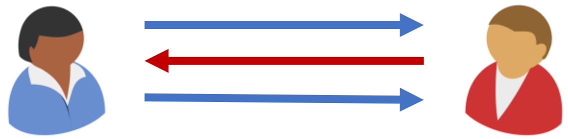
$\forall i \in [k], (C, x_i) \in \text{SAT}$

Prior Works

Prior Works

Interactive batch proofs for UP

[Reingold-Rothblum-Rothblum'16, Reingold-Rothblum-Rothblum'18, Rothblum-Rothblum'20]



UP – each statement has a unique witness.

Prior Works

Interactive batch proofs for UP

[Reingold-Rothblum-Rothblum’16, Reingold-Rothblum-Rothblum’18, Rothblum-Rothblum’20]

Succinct Non-interactive Arguments (SNARGs) for NP

[Micali’94, Damgård-Faust-Hazay’12, Bitansky-Canetti-Chiesa-Tromer’13, Bitanksy-Canetti-Chiesa-Goldwasser-Lin-Rubinstein-Tromer’16]

SNARGs

$|\Pi| \ll |w|$

$\mathrm{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \mathrm{SAT}$

Prior Works

Interactive batch proofs for UP

[Reingold-Rothblum-Rothblum'16, Reingold-Rothblum-Rothblum'18, Rothblum-Rothblum'20]

Succinct Non-interactive Arguments (SNARGs) for NP

[Micali'94, Damgård-Faust-Hazay'12, Bitansky-Canetti-Chiesa-Tromer'13, Bitanksy-Canetti-Chiesa-Goldwasser-Lin-Rubinstein-Tromer'16]

SNARGs

$|\Pi| \ll |w|$

$$\mathrm{SAT}^{\otimes k} = \{(C, x_1, \cdots, x_k) \mid \forall i \in [k], (C, x_i) \in \mathrm{SAT}\}$$

$$\mathrm{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$$

$\forall i \in [k], (C, x_i) \in \mathrm{SAT}$

Prior Works

Interactive batch proofs for UP

[Reingold-Rothblum-Rothblum'16, Reingold-Rothblum-Rothblum'18, Rothblum-Rothblum'20]

SNARGs for NP from Non-falsifiable assumptions/ Random oracle model

[Micali'94, Damgård-Faust-Hazay'12, Bitansky-Canetti-Chiesa-Tromer'13, Bitanksy-Canetti-Chiesa-Goldwasser-Lin-Rubinstein-Tromer'16]

SNARGs

$|\Pi| \ll |w|$

$$\mathrm{SAT}^{\otimes k} = \{(C, x_1, \cdots, x_k) \mid \forall i \in [k], (C, x_i) \in \mathrm{SAT}\}$$

$$\mathrm{SAT} = \{(C, x) \mid \exists w \ s.t. \ C(x, w) = 1\}$$

$\forall i \in [k], (C, x_i) \in \mathrm{SAT}$

Prior Works

Interactive batch proofs for UP

[Reingold-Rothblum-Rothblum'16, Reingold-Rothblum-Rothblum'18, Rothblum-Rothblum'20]

SNARGs for NP from Non-falsifiable assumptions/ Random oracle model

[Micali'94, Damgård-Faust-Hazay'12, Bitansky-Canetti-Chiesa-Tromer'13, Bitanksy-Canetti-Chiesa-Goldwasser-Lin-Rubinstein-Tromer'16]

Designated Verifier SNARGs for Batch NP

[Brakerski-Holmgren-Kalai'17, Brakerski-Kalai'20]

Prior Works

Interactive batch proofs for UP

[Reingold-Rothblum-Rothblum'16, Reingold-Rothblum-Rothblum'18, Rothblum-Rothblum'20]

SNARGs for NP from Non-falsifiable assumptions/ Random oracle model

[Micali'94, Damgård-Faust-Hazay'12, Bitansky-Canetti-Chiesa-Tromer'13, Bitanksy-Canetti-Chiesa-Goldwasser-Lin-Rubinstein-Tromer'16]

Designated Verifier from standard assumptions

[Brakerski-Holmgren-Kalai'17, Brakerski-Kalai'20]

Prior Works

Interactive batch proofs for UP

[Reingold-Rothblum-Rothblum'16, Reingold-Rothblum-Rothblum'18, Rothblum-Rothblum'20]

SNARGs for NP from Non-falsifiable assumptions/ Random oracle model

[Micali'94, Damgård-Faust-Hazay'12, Bitansky-Canetti-Chiesa-Tromer'13, Bitanksy-Canetti-Chiesa-Goldwasser-Lin-Rubinstein-Tromer'16]

Designated Verifier from standard assumptions

[Brakerski-Holmgren-Kalai'17, Brakerski-Kalai'20]

SNARGs for Batch NP from new non-standard assumption

[Kalai-Paneth-Yang'19]

Falsifiable assumption on groups with bilinear maps.

Do there exists SNARGs for Batch NP based on standard assumptions?

[Gentry-Wichs'11] lower bound doesn't apply.

Our Result

Theorem

There exists SNARGs for Batch NP

Assuming QR + sub-exp DDH

$$|\Pi| = \tilde{O}(|C| + \sqrt{k|C|})$$

[C-Jain-Jin'21a]

QR – Quadratic residuosity, LWE – Learning with Error, DDH – Decisional Diffie-Hellman

$\mathrm{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \mathrm{SAT}$

Our Result

Theorem

There exists SNARGs for Batch NP

Assuming LWE

$$|\Pi| = \mathrm{poly}(\log k, |C|)$$

[C-Jain-Jin'21b]

Assuming QR + sub-exp DDH

$$|\Pi| = \tilde{O}(|C| + \sqrt{k|C|})$$

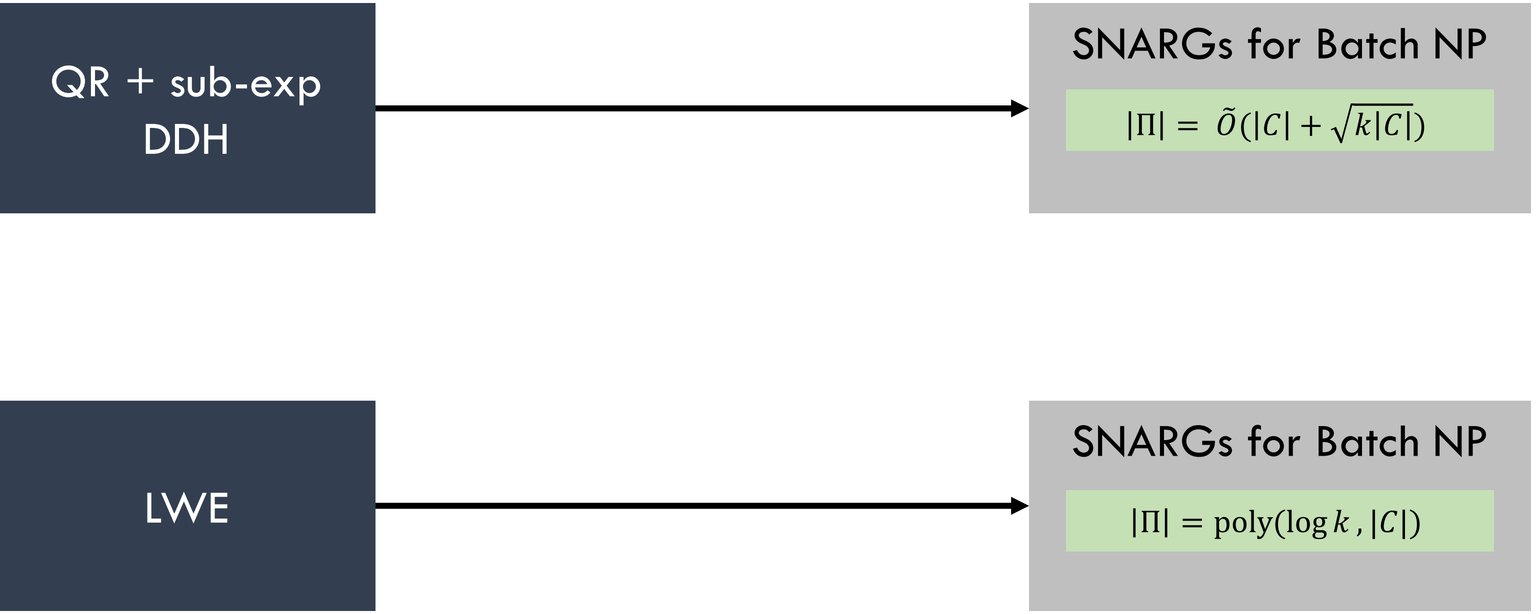
[C-Jain-Jin'21a]

QR – Quadratic residuosity, LWE – Learning with Error, DDH – Decisional Diffie-Hellman

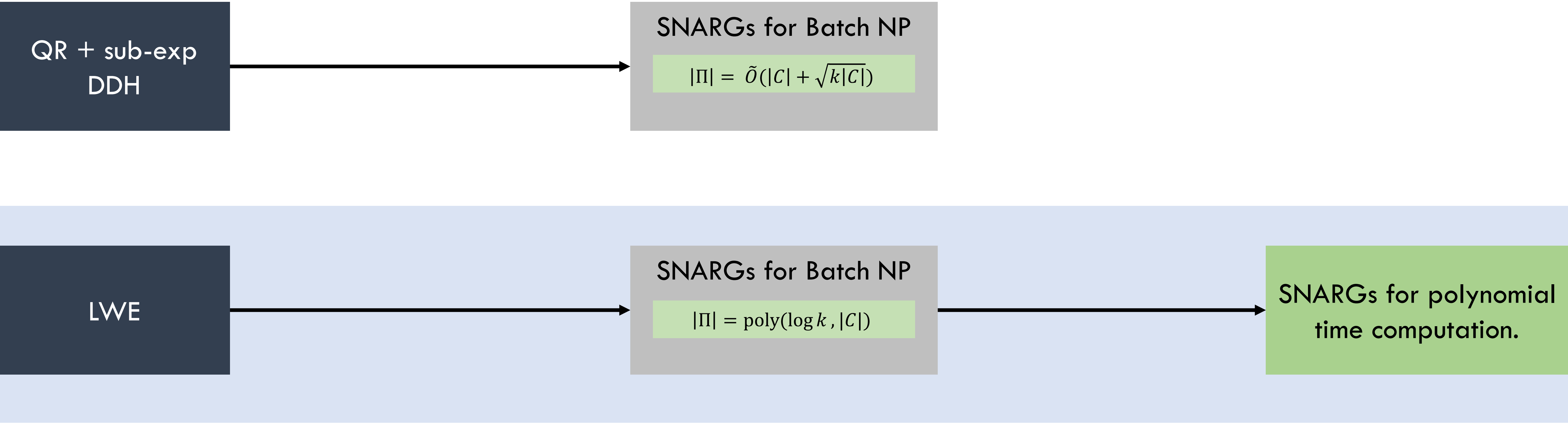
$\mathrm{SAT} = \{(C, x) \mid \exists w \ s.t. \ C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \mathrm{SAT}$

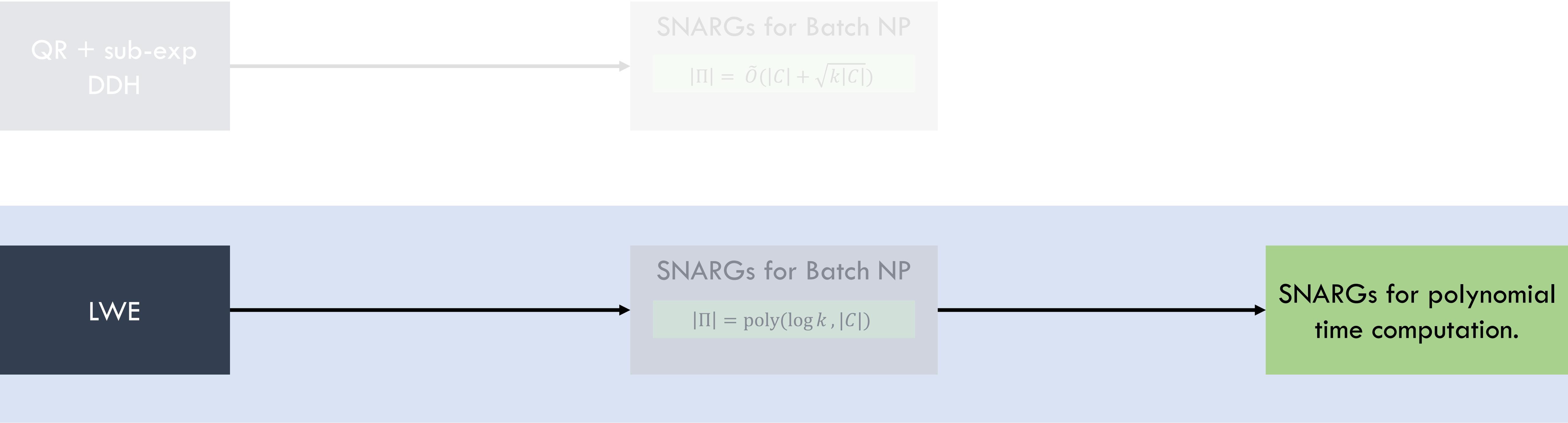
Results Overview



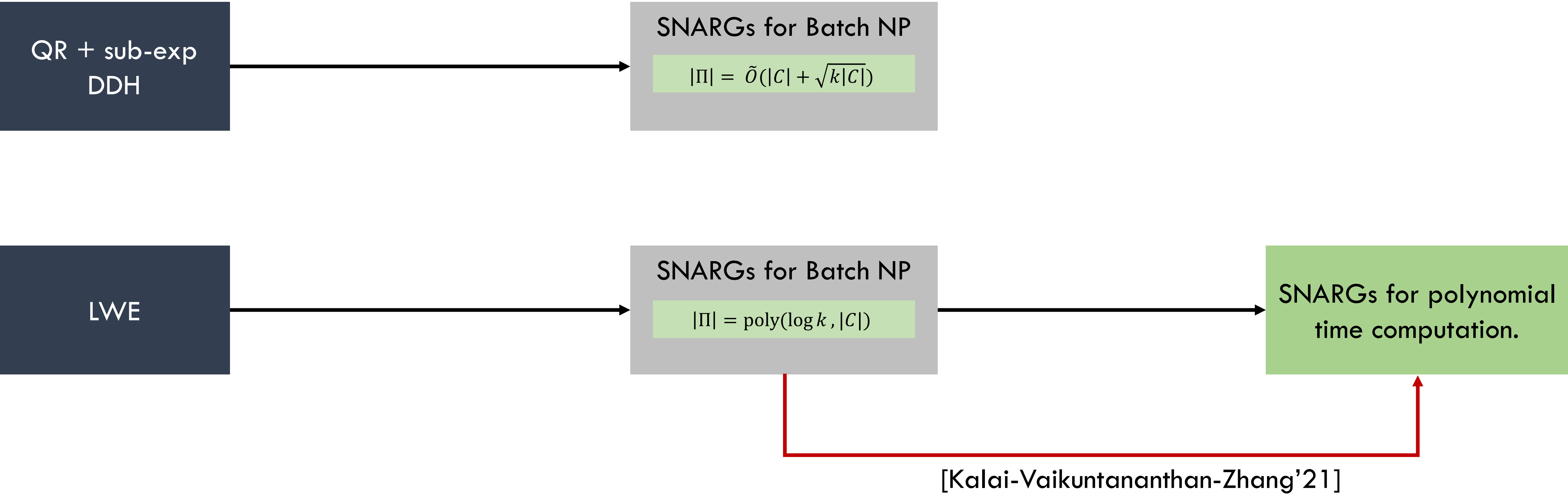
Results Overview



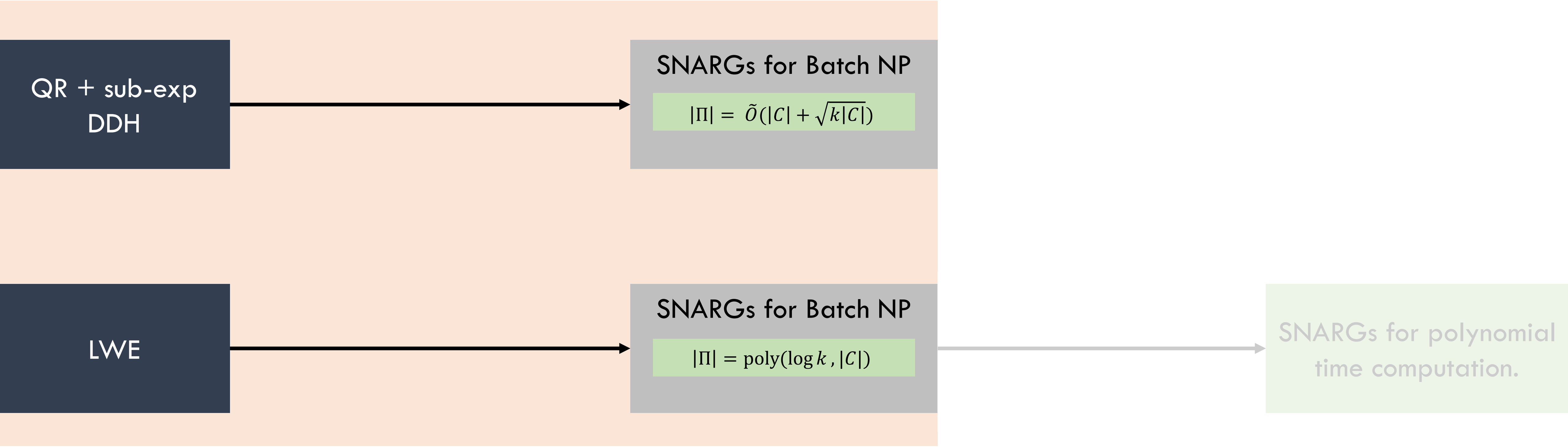
Results Overview



Results Overview

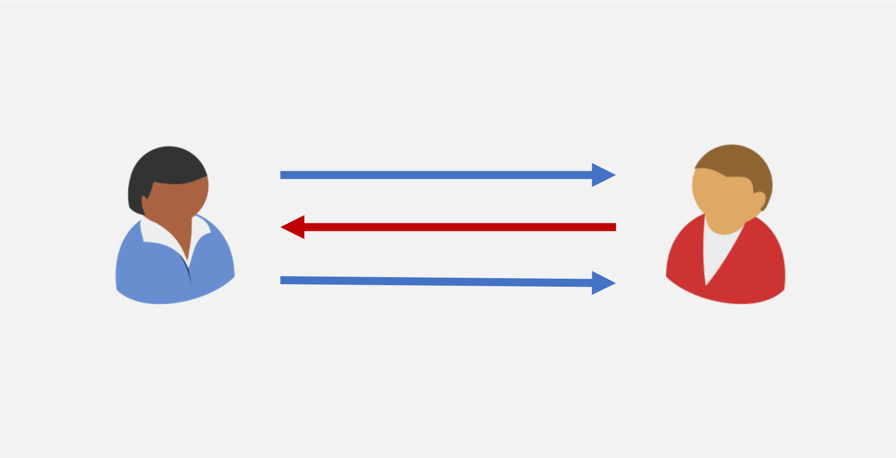


Results Overview

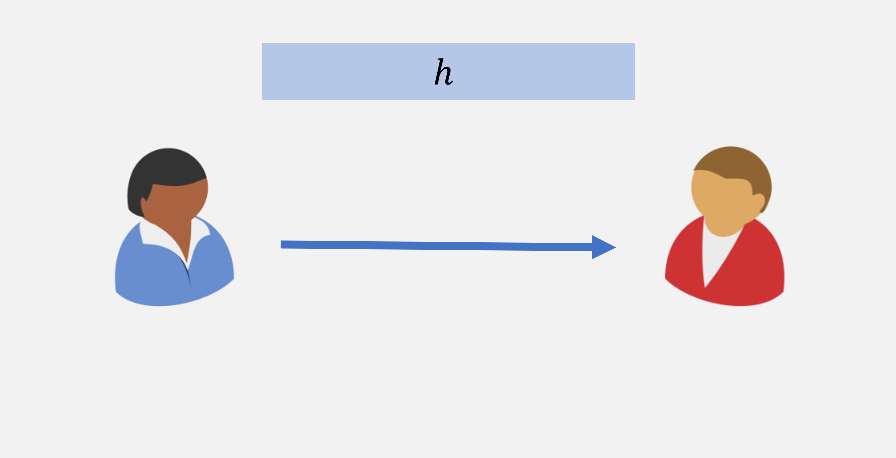


SNARGs for Batch NP

Fiat-Shamir (FS) Methodology

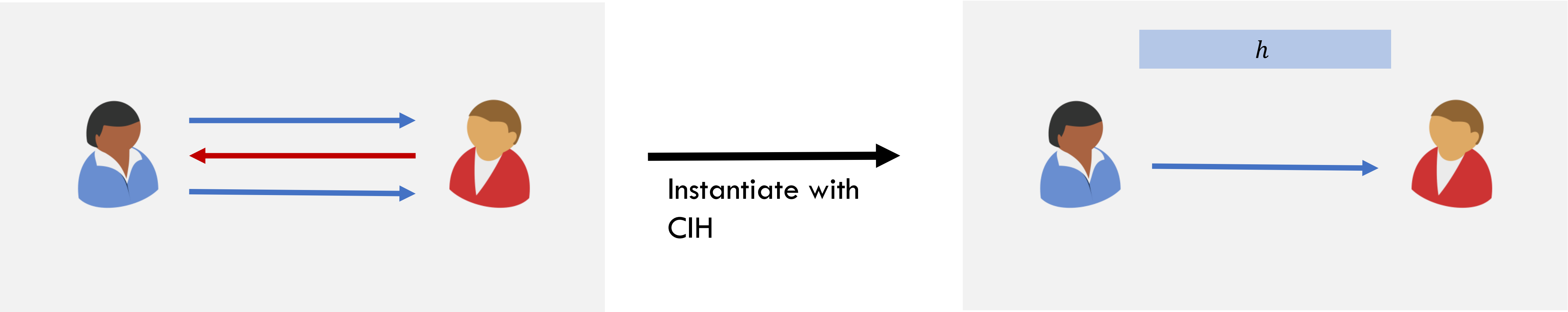


Instantiate with CIH



No known interactive proof for batch NP

Fiat-Shamir (FS) Methodology

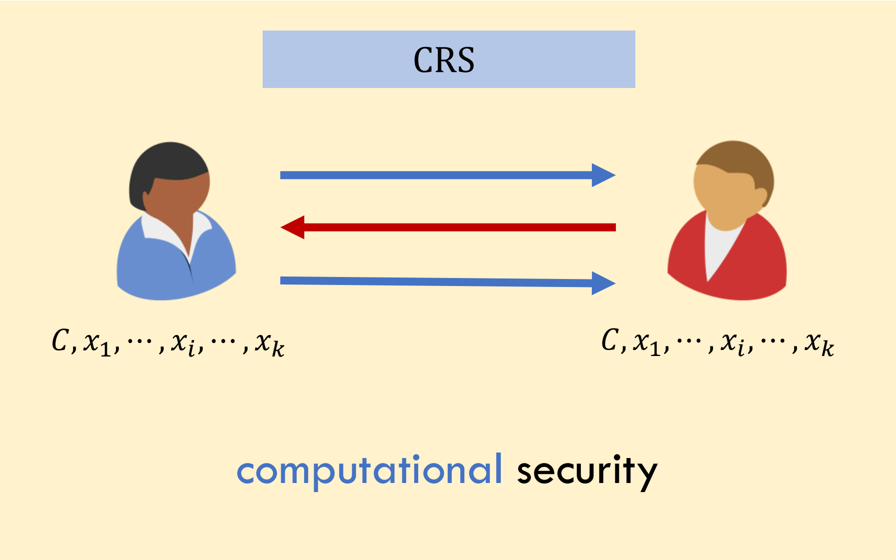


A Different Starting Point for Fiat-Shamir Methodology

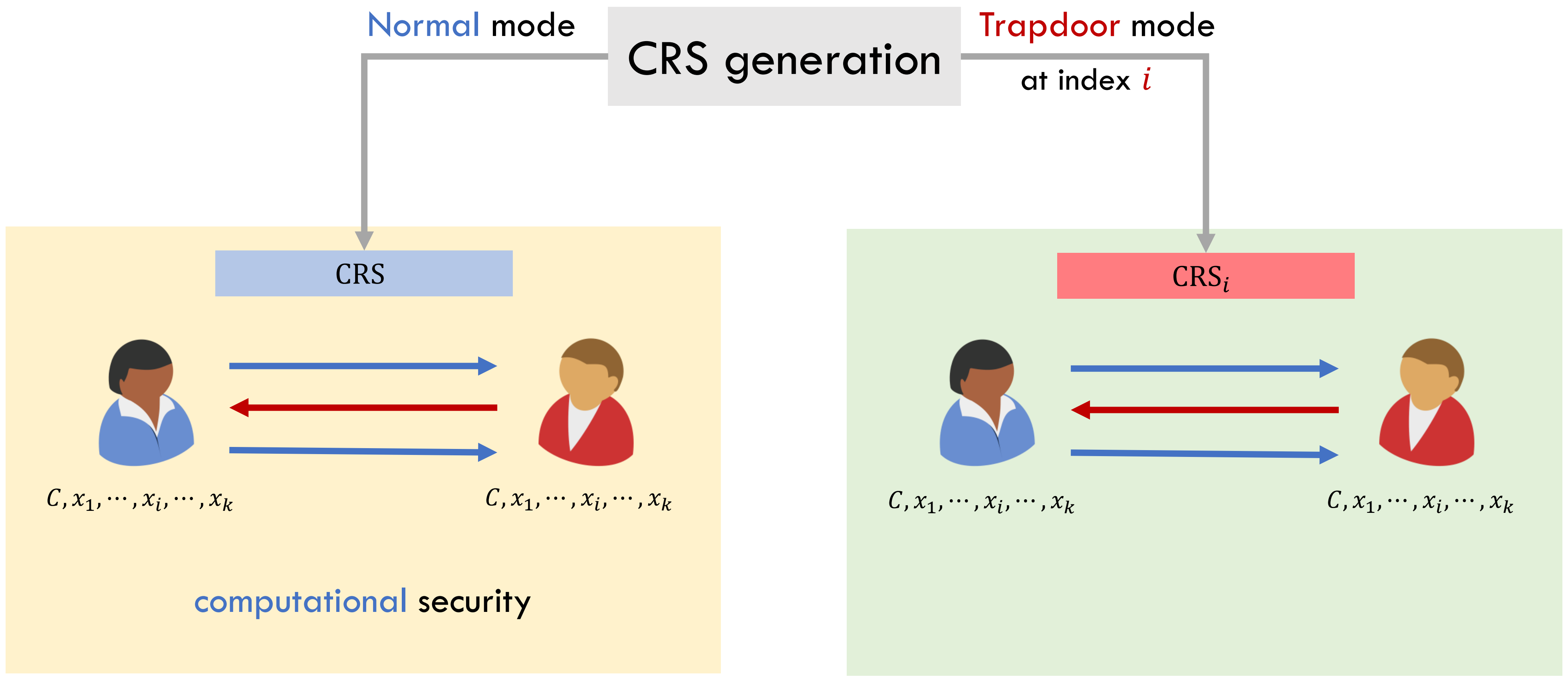
No known interactive proof for batch NP

Dual-Mode Interactive Batch Arguments

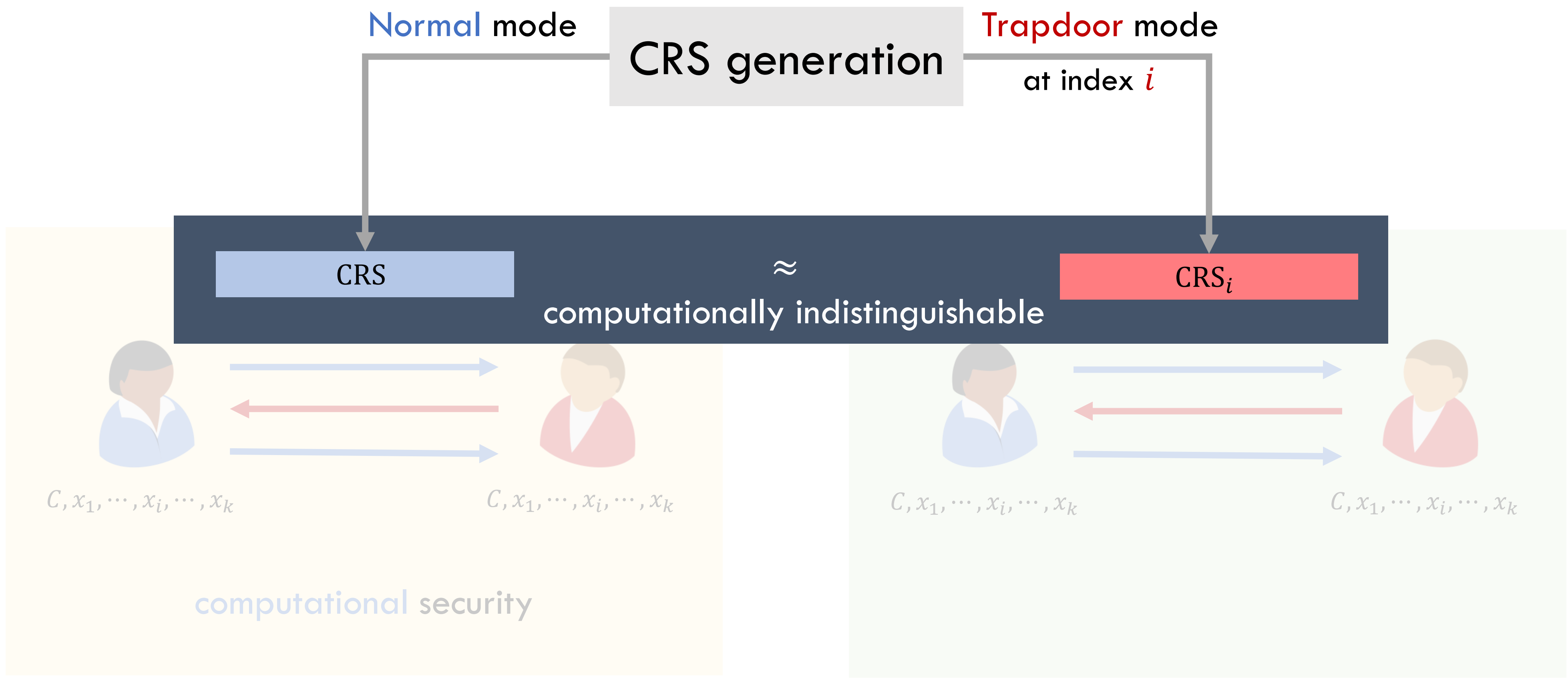
Dual-Mode Interactive Batch Arguments



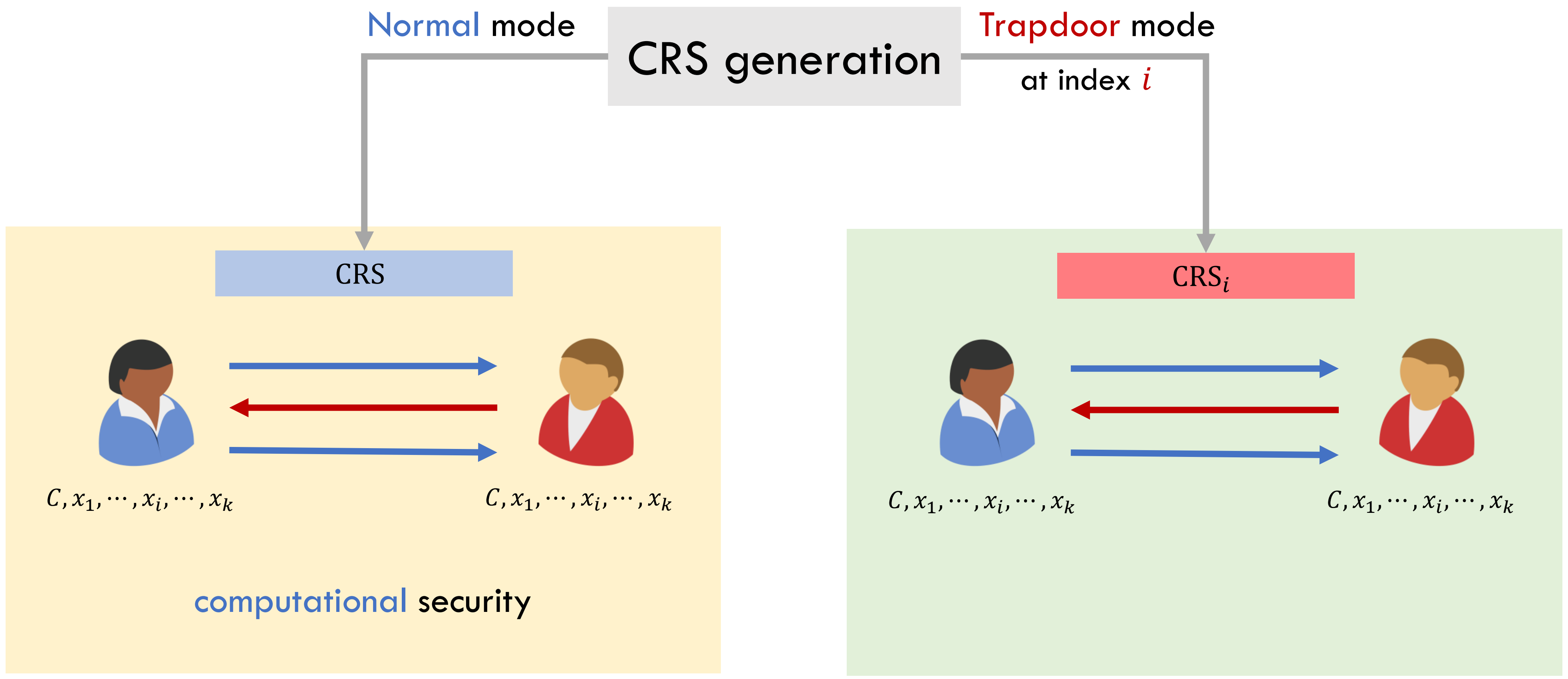
Dual-Mode Interactive Batch Arguments



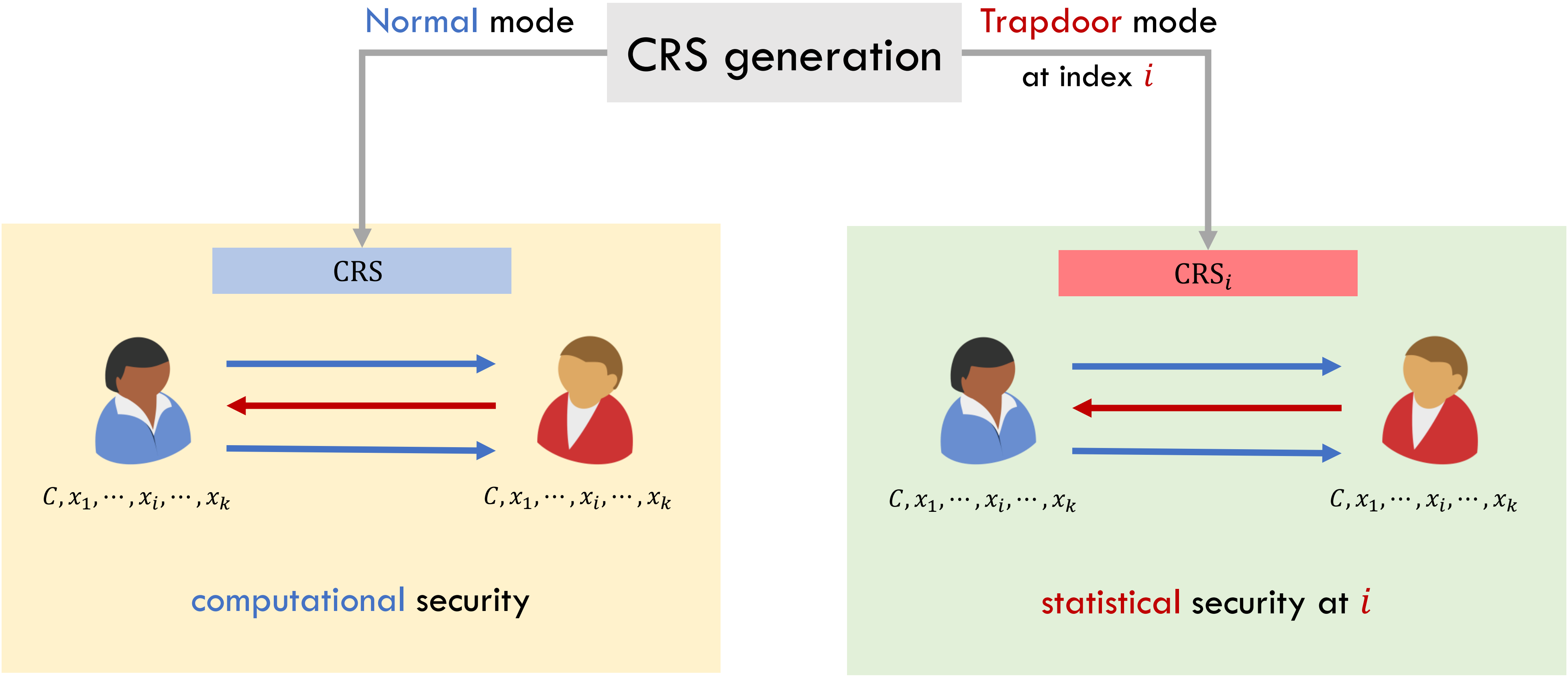
Dual-Mode Interactive Batch Arguments



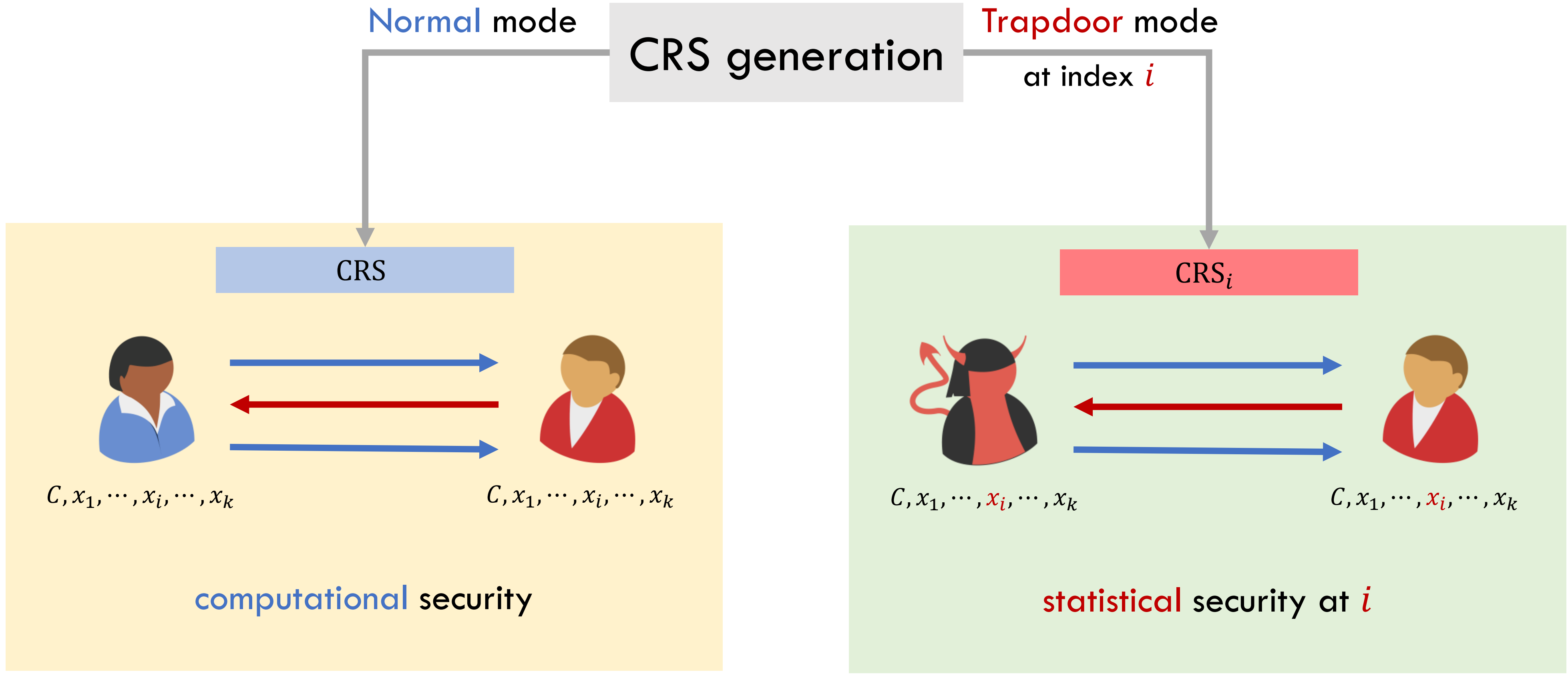
Dual-Mode Interactive Batch Arguments



Dual-Mode Interactive Batch Arguments

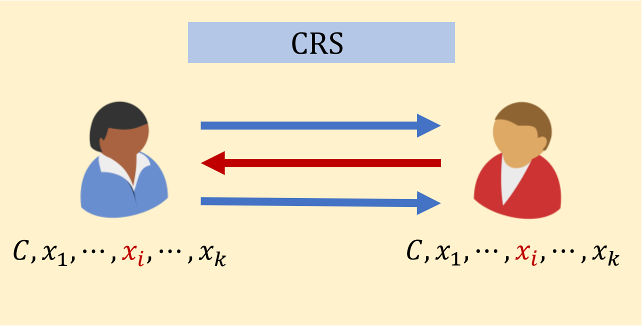


Dual-Mode Interactive Batch Arguments

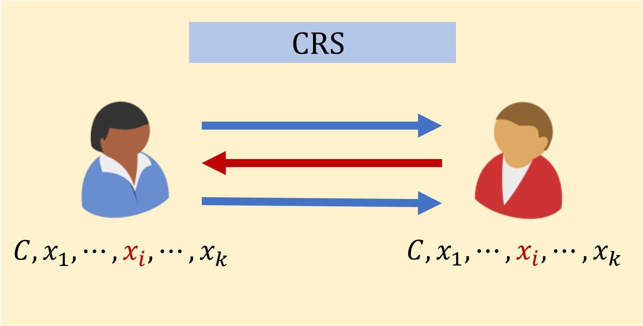


Even unbounded cannot make accept if $(C, x_i) \notin \mathrm{SAT}$

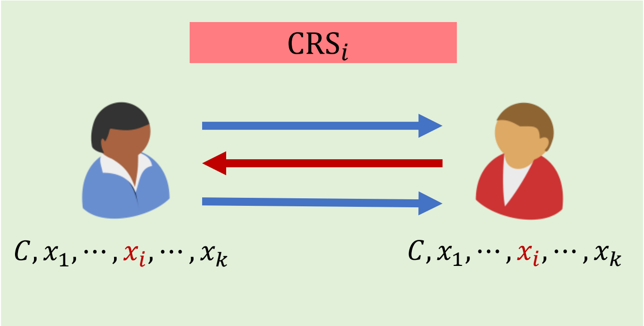
Security Intuition



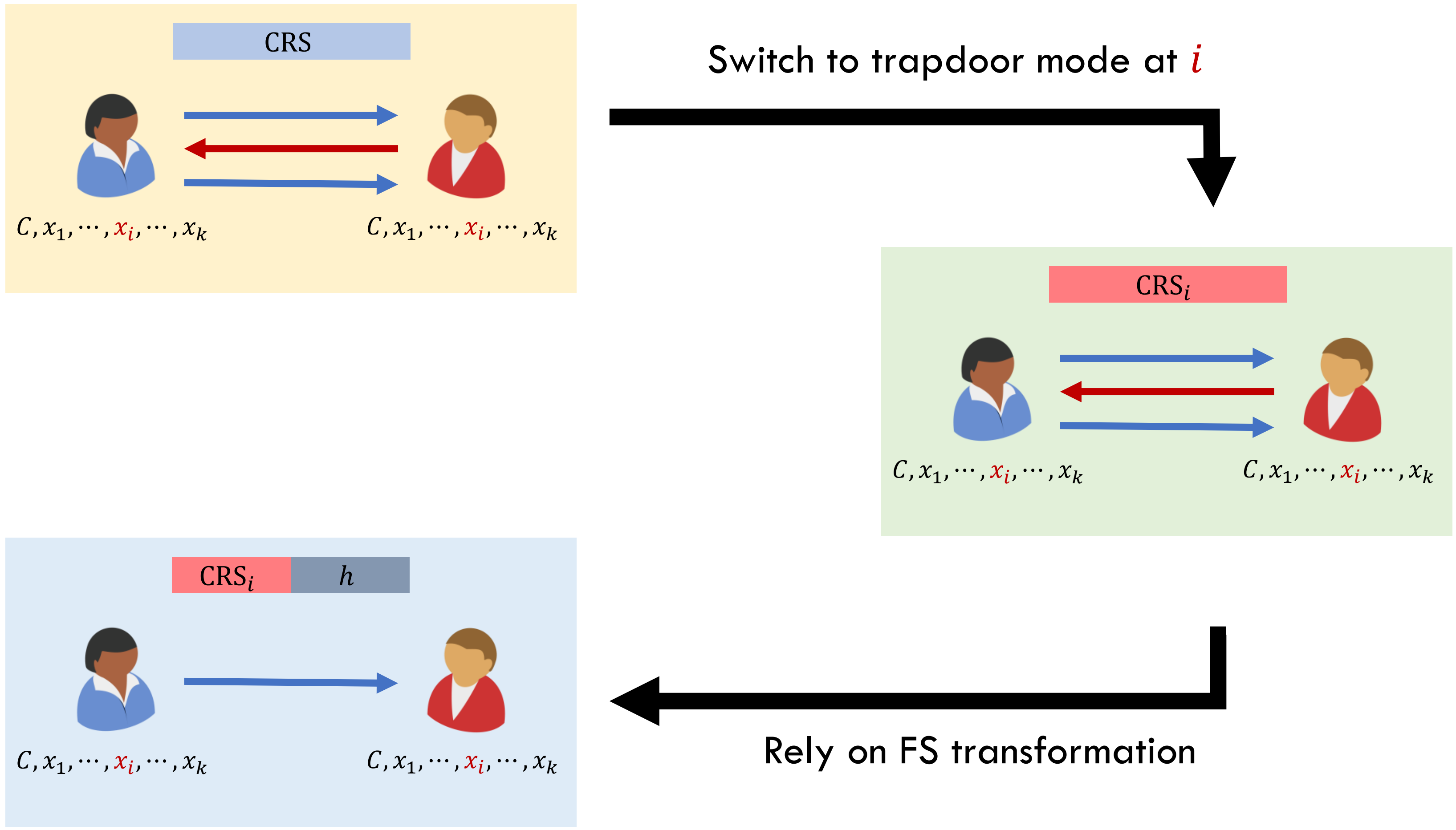
Security Intuition



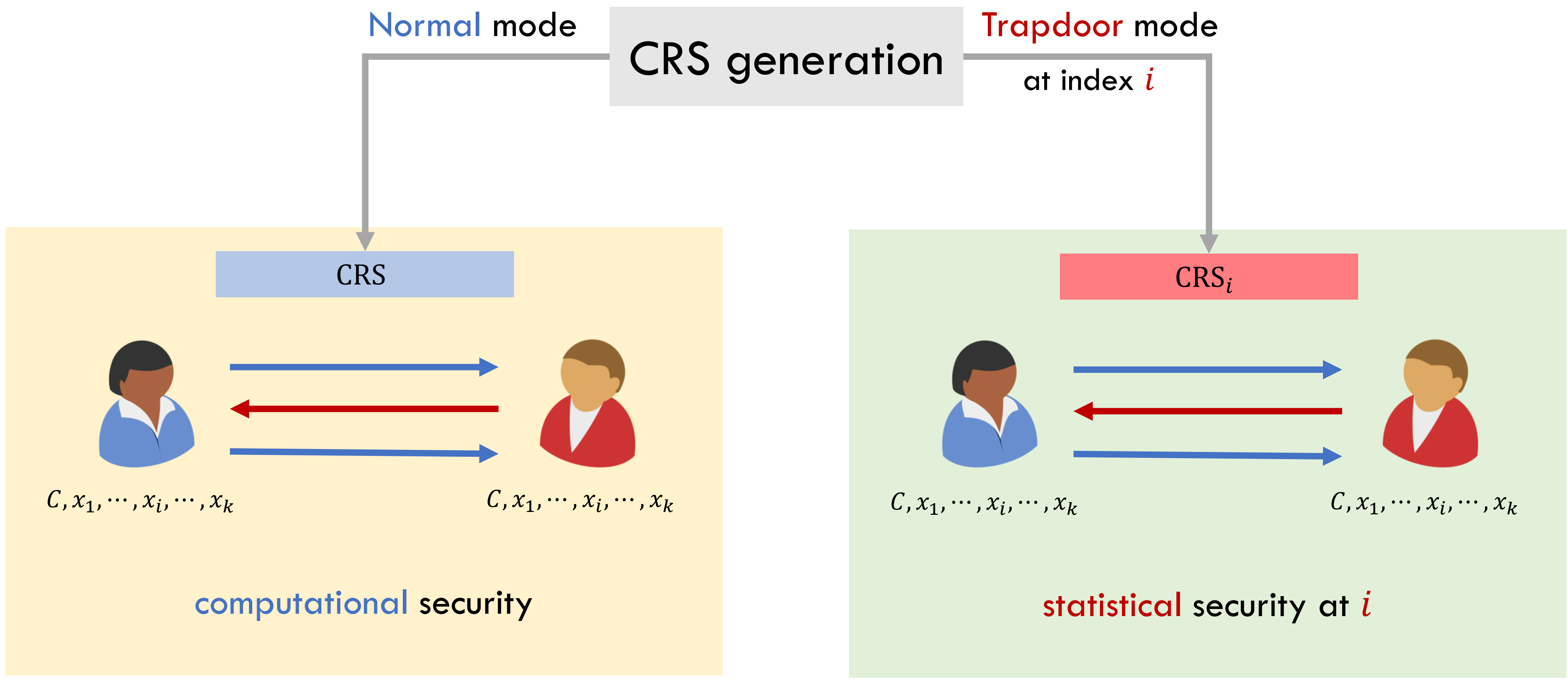
Switch to trapdoor mode at i



Security Intuition

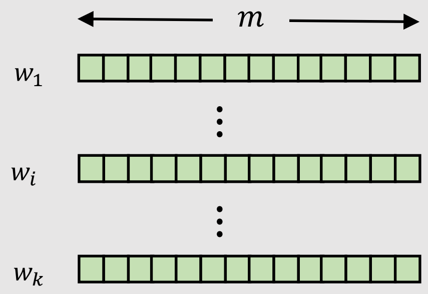


Dual-Mode Interactive Batch Arguments



Dual Mode Batch Argument

Protocol Template

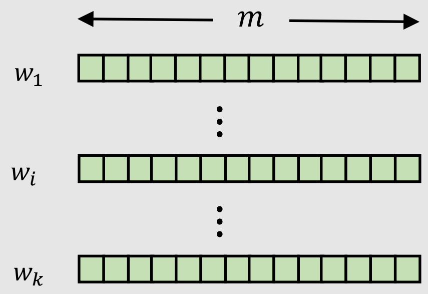


$\text{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$

Dual Mode Batch Argument

Protocol Template



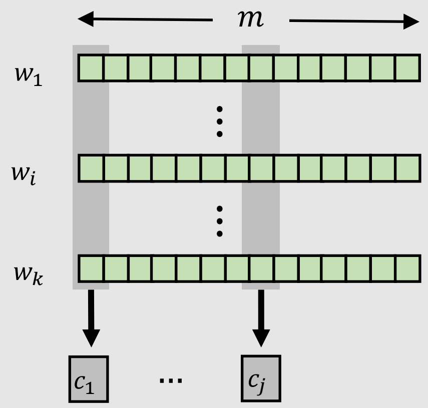
commitment key K

$\mathrm{SAT} = \{(C, x) \mid \exists w \ s.t. \ C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \mathrm{SAT}$

Dual Mode Batch Argument

Protocol Template



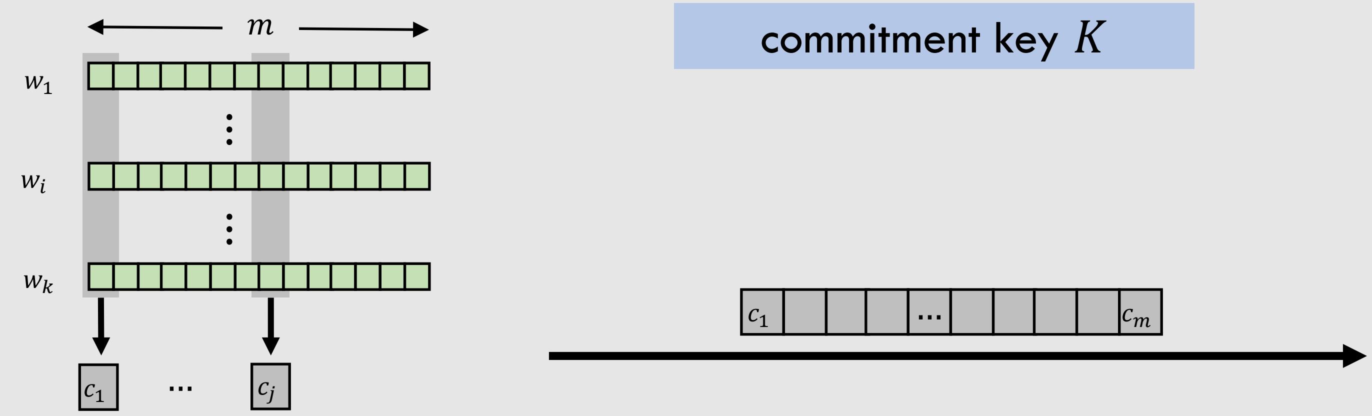
commitment key K

$\mathrm{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \mathrm{SAT}$

Dual Mode Batch Argument

Protocol Template



$\text{SAT} = \{(C, x) \mid \exists w \ s.t. \ C(x, w) = 1\}$

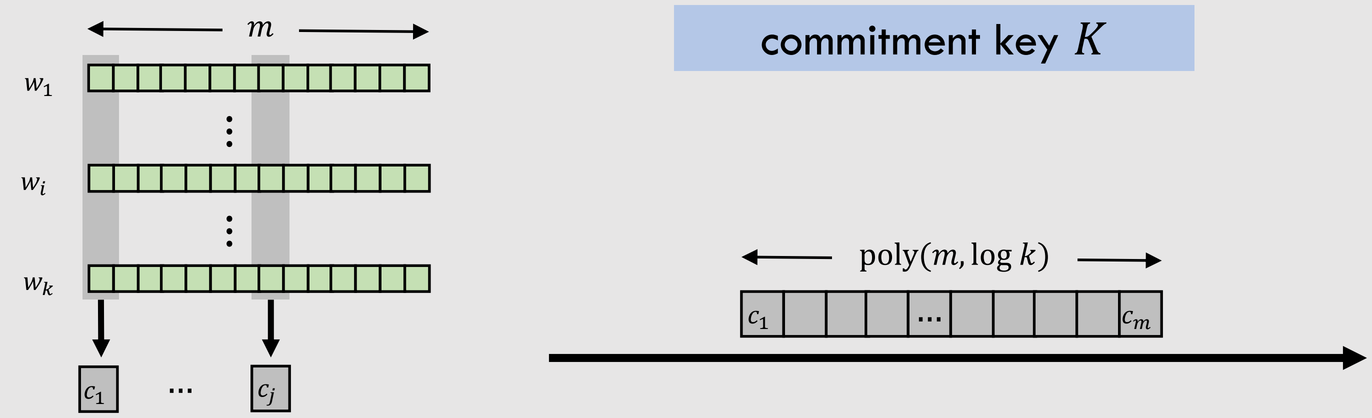
$\forall i \in [k], (C, x_i) \in \text{SAT}$

Dual Mode Batch Argument

Protocol Template

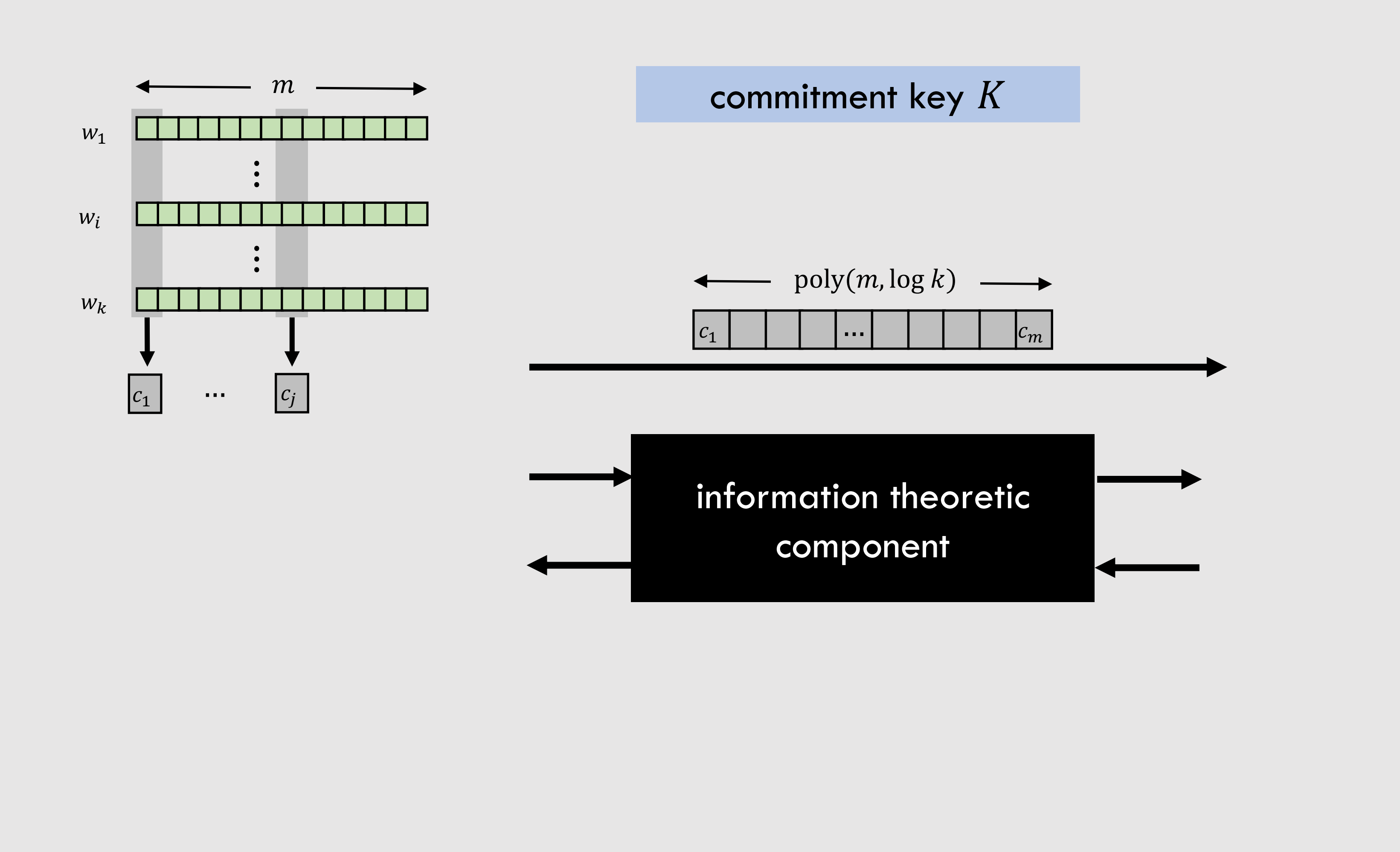
$\text{SAT} = \{(C, x) \mid \exists w \ s.t. \ C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$



Dual Mode Batch Argument

Protocol Template

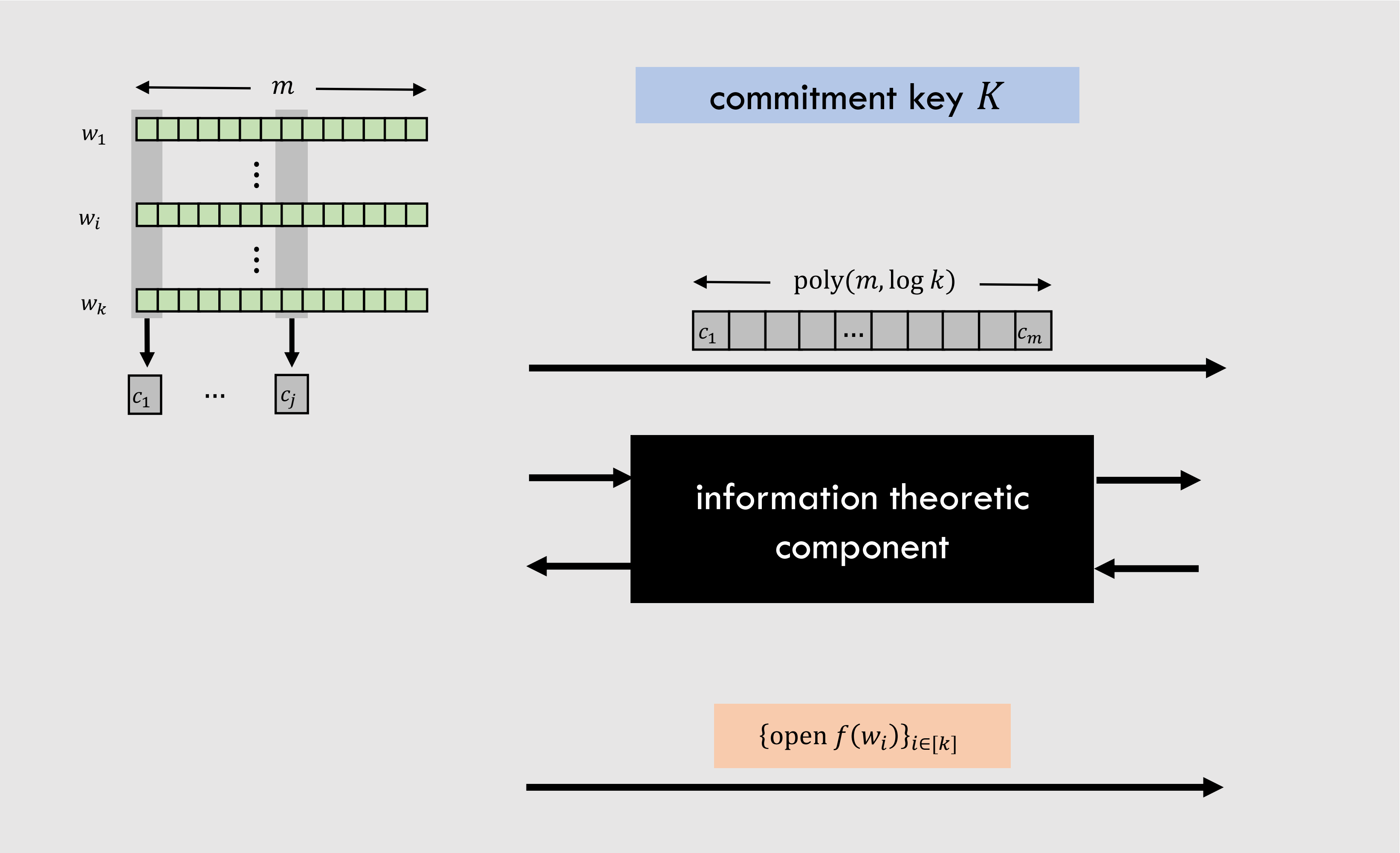


$\mathrm{SAT} = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \mathrm{SAT}$

Dual Mode Batch Argument

Protocol Template



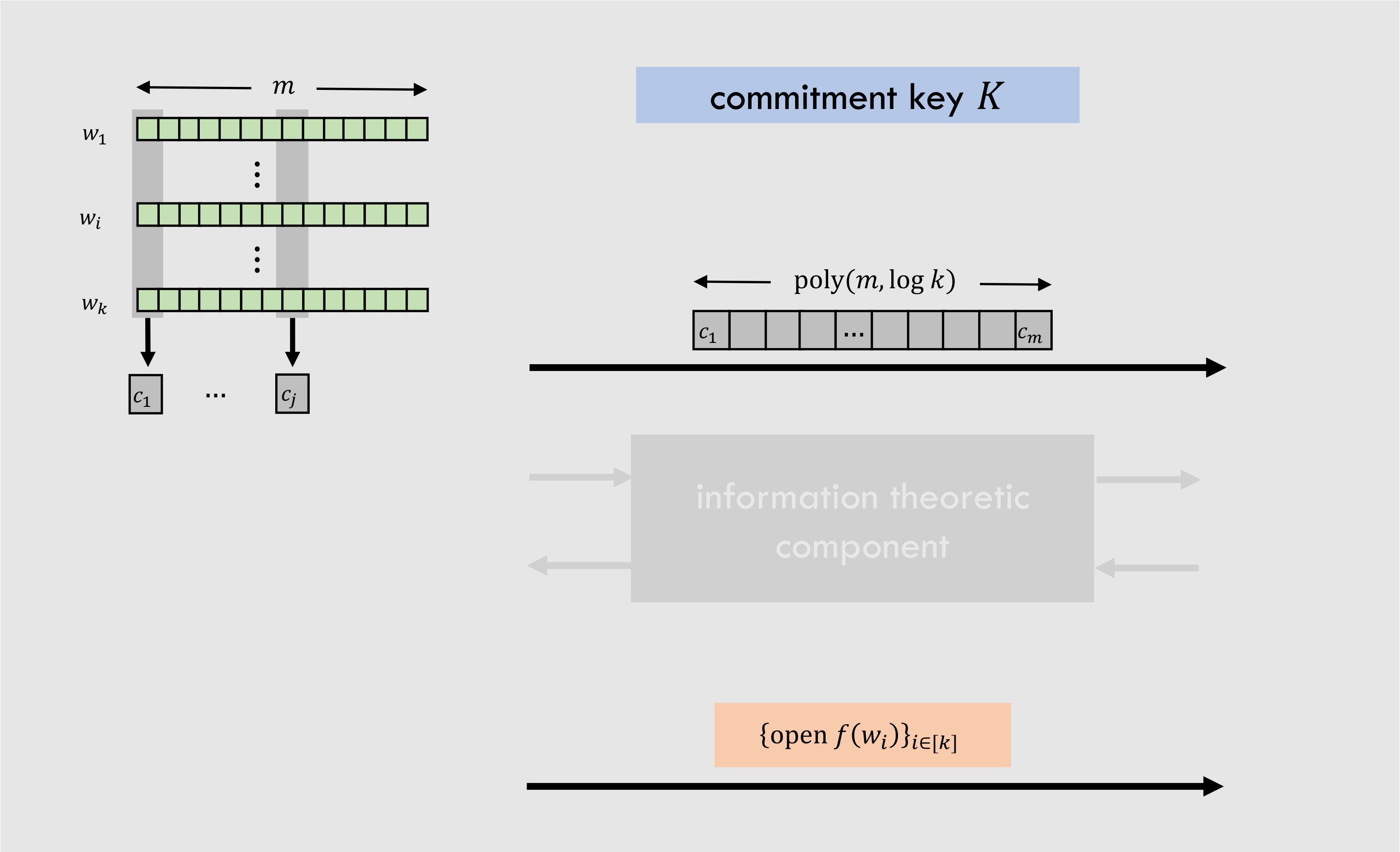
$\text{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$

f determined by the information theoretic component.

Dual Mode Batch Argument

Protocol Template

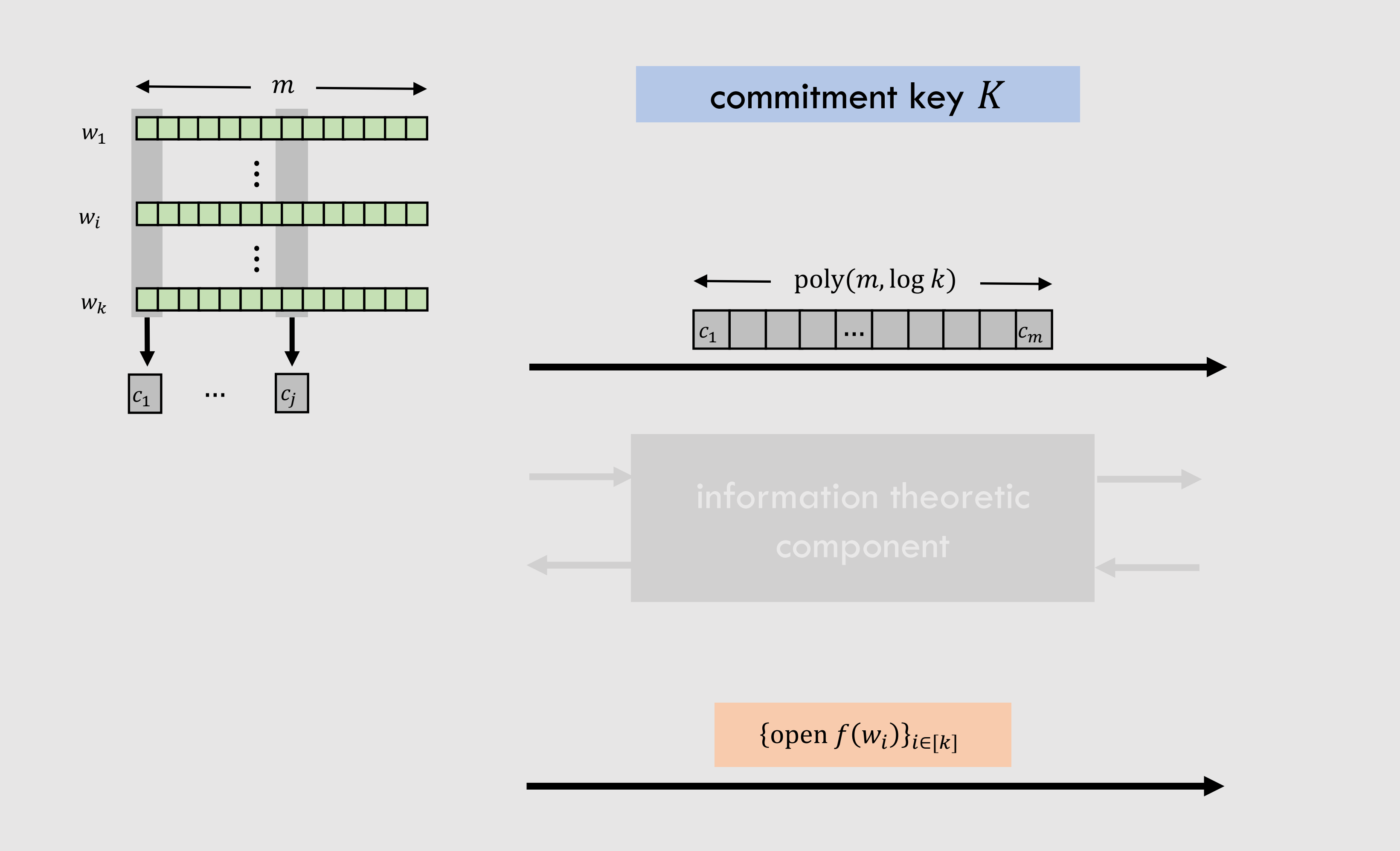


$\text{SAT} = \{(C, x) \mid \exists w \ s.t.\ C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$

Dual Mode Batch Argument

Protocol Template



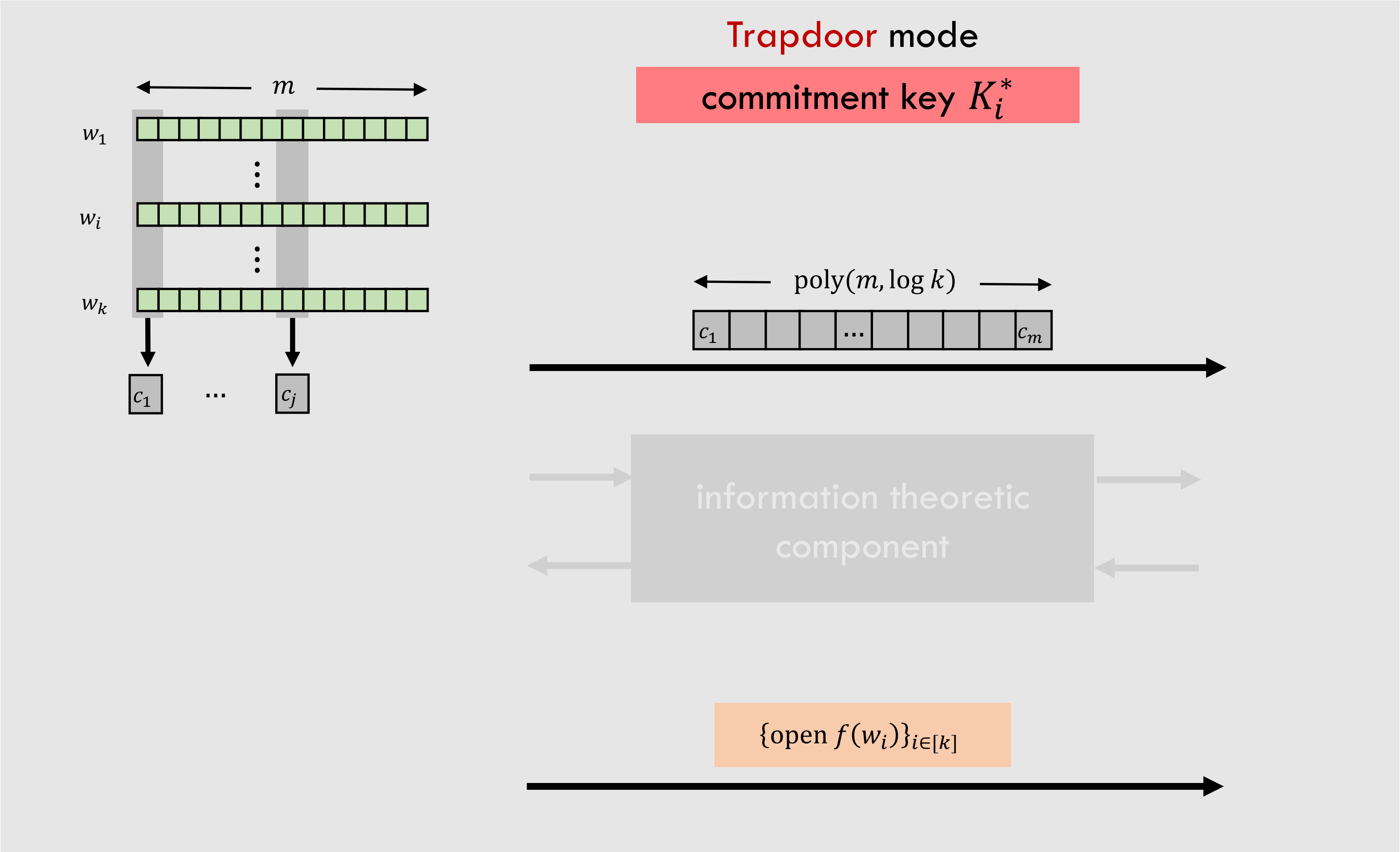
$\text{SAT} = \{(C, x) \mid \exists w \ s.t. \ C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$

Somewhere Statistically Binding (SSB) Commitment Scheme [Hubáček -Wichs'15]

Dual Mode Batch Argument

Protocol Template



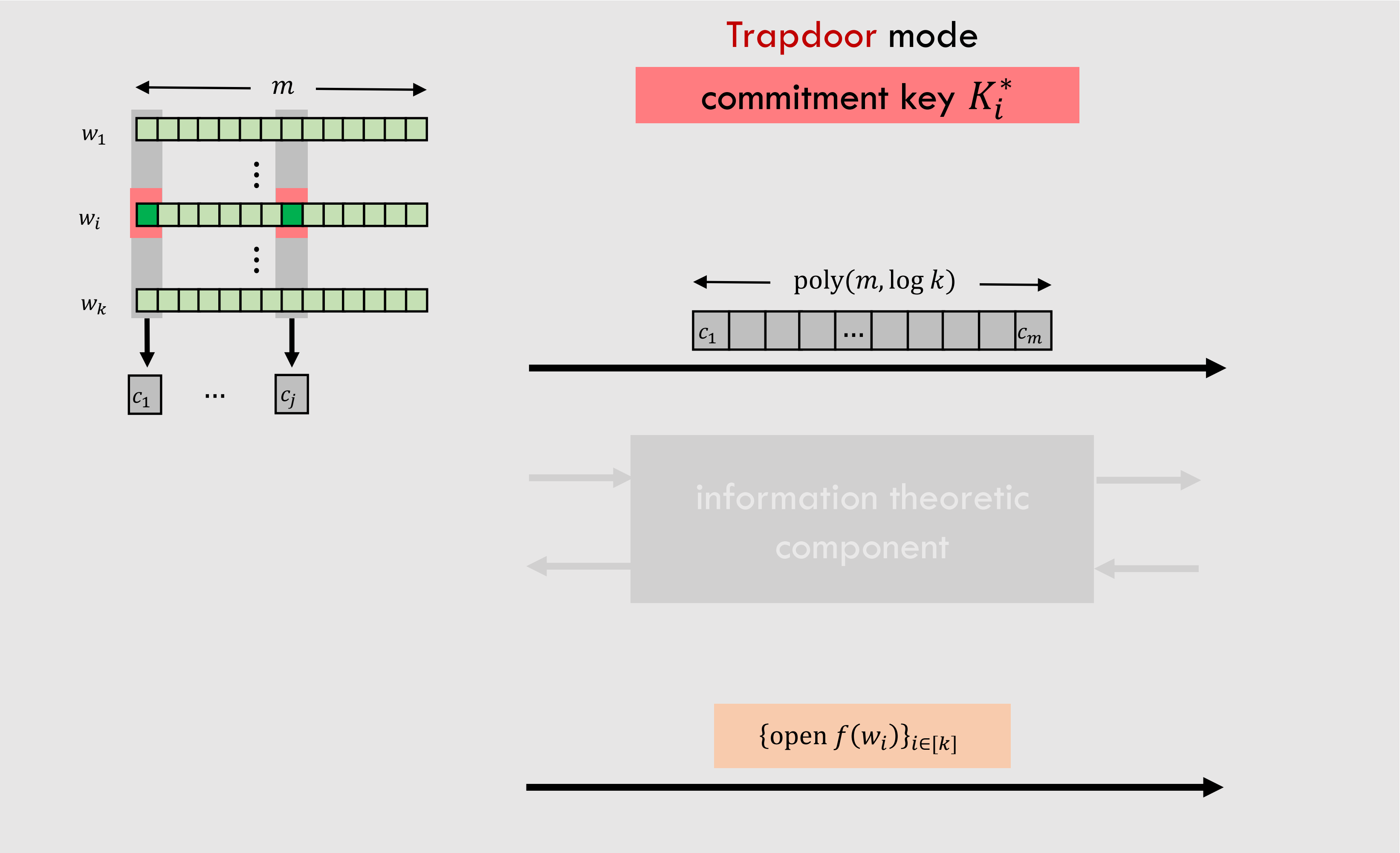
$\text{SAT} = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$

Somewhere Statistically Binding (SSB) Commitment Scheme

Dual Mode Batch Argument

Protocol Template



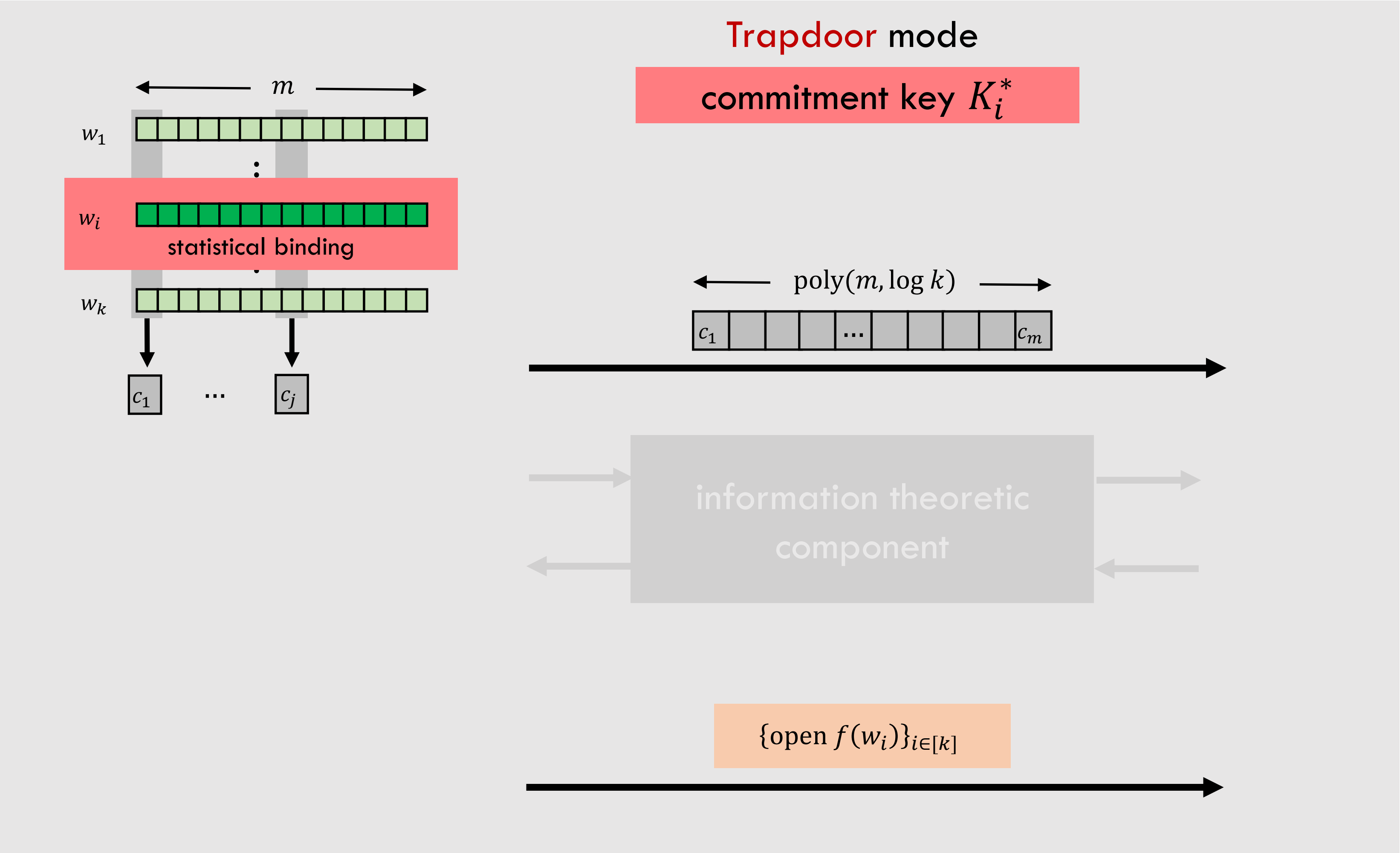
$\text{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$

Somewhere Statistically Binding (SSB) Commitment Scheme

Dual Mode Batch Argument

Protocol Template



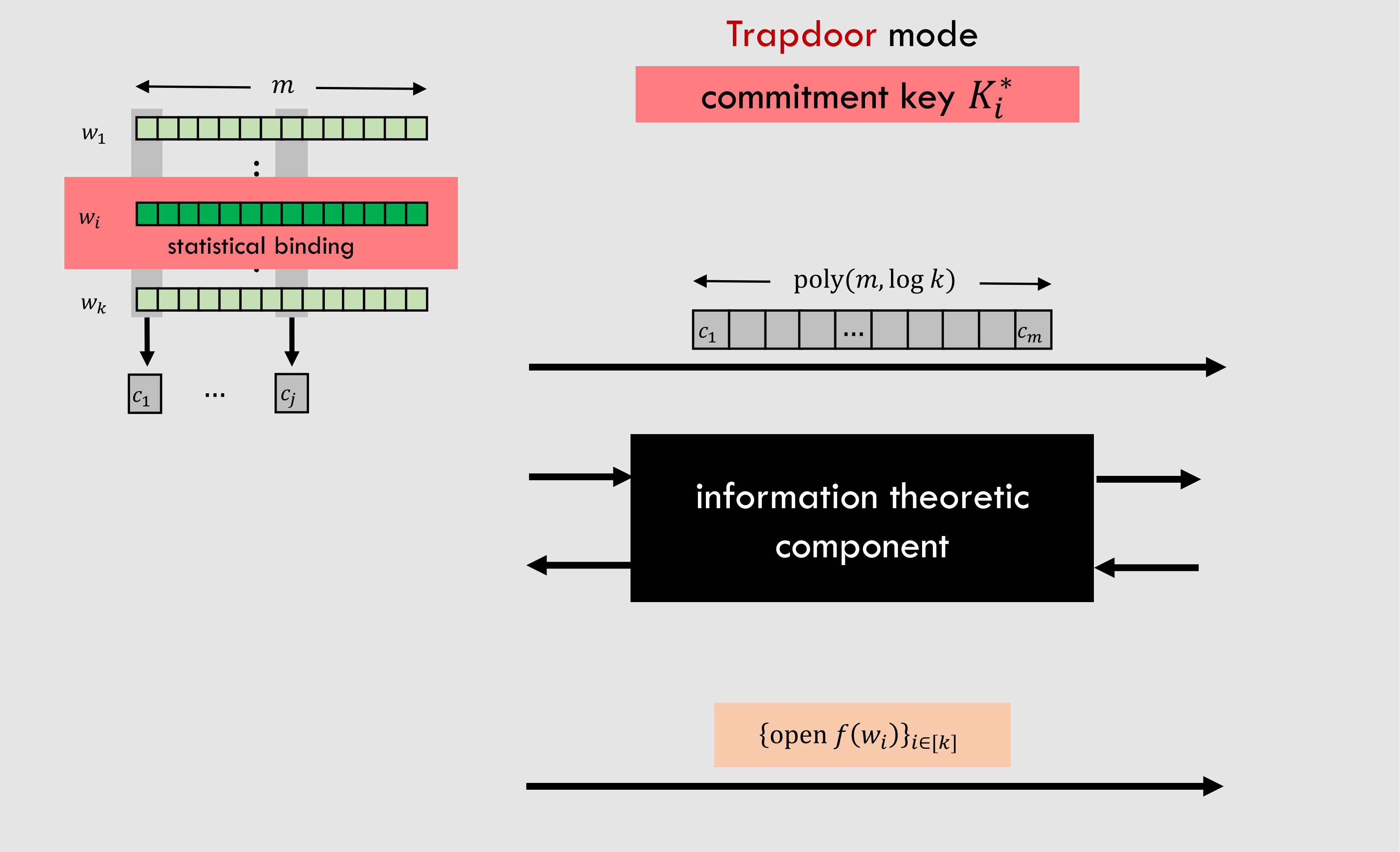
$\text{SAT} = \{(C, x) \mid \exists w \; s.t. \; C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \text{SAT}$

Somewhere Statistically Binding (SSB) Commitment Scheme

Dual Mode Batch Argument

Protocol Template



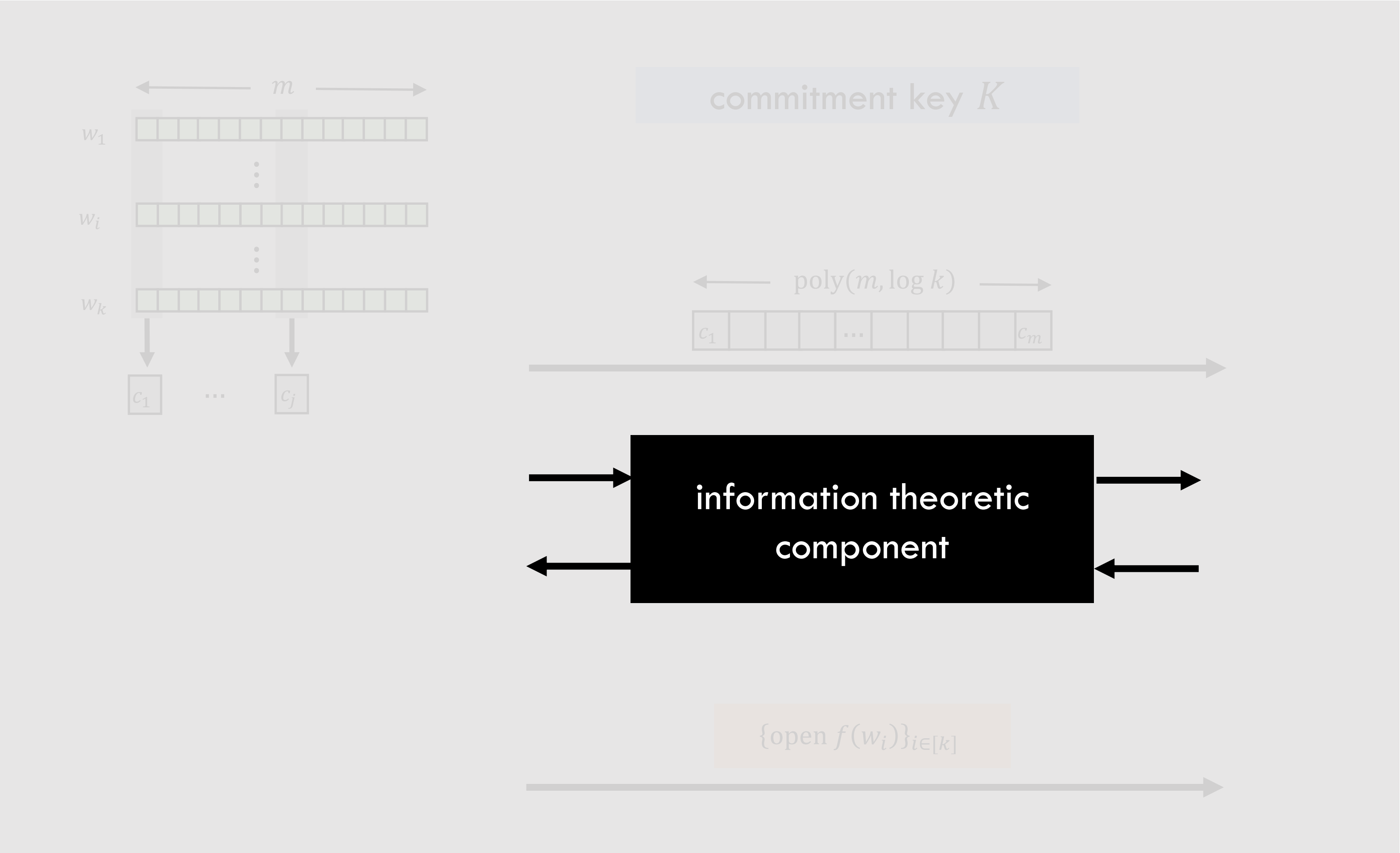
$\mathrm{SAT} = \{(C, x) \mid \exists w \ s.t. \ C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \mathrm{SAT}$

Somewhere Statistically Binding (SSB) Commitment Scheme

Dual Mode Batch Argument

Protocol Template



$\mathrm{SAT} = \{(C, x) \mid \exists w\ s.t.\ C(x, w) = 1\}$

$\forall i \in [k], (C, x_i) \in \mathrm{SAT}$

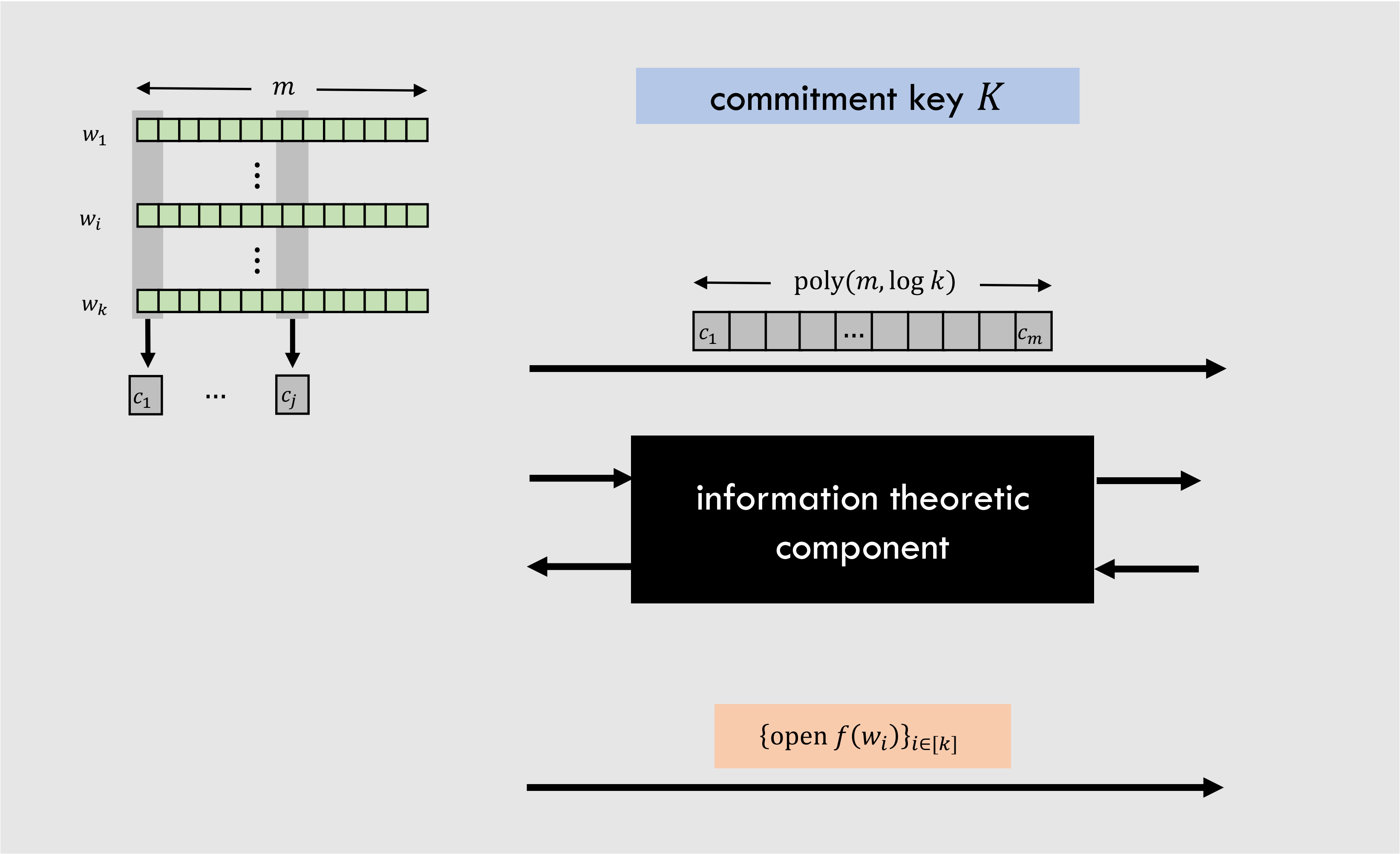
Somewhere Statistically Binding (SSB) Commitment Scheme

Needs to be Fiat-Shamir friendly.

Based on LWE/sub-exp DDH

Dual Mode Batch Argument

Protocol Template



$\text{SAT} = \{(C, x) \mid \exists w \text{ s.t. } C(x, w) = 1\}$

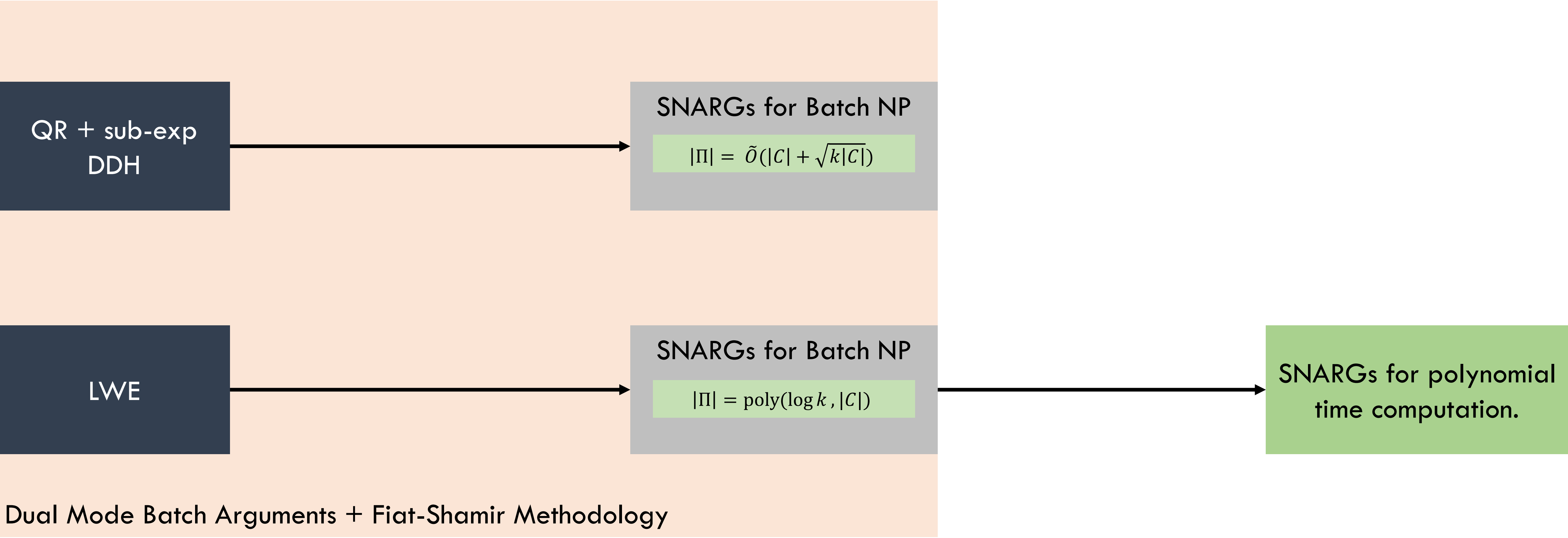
$\forall i \in [k], (C, x_i) \in \text{SAT}$

Somewhere Statistically Binding (SSB) Commitment Scheme

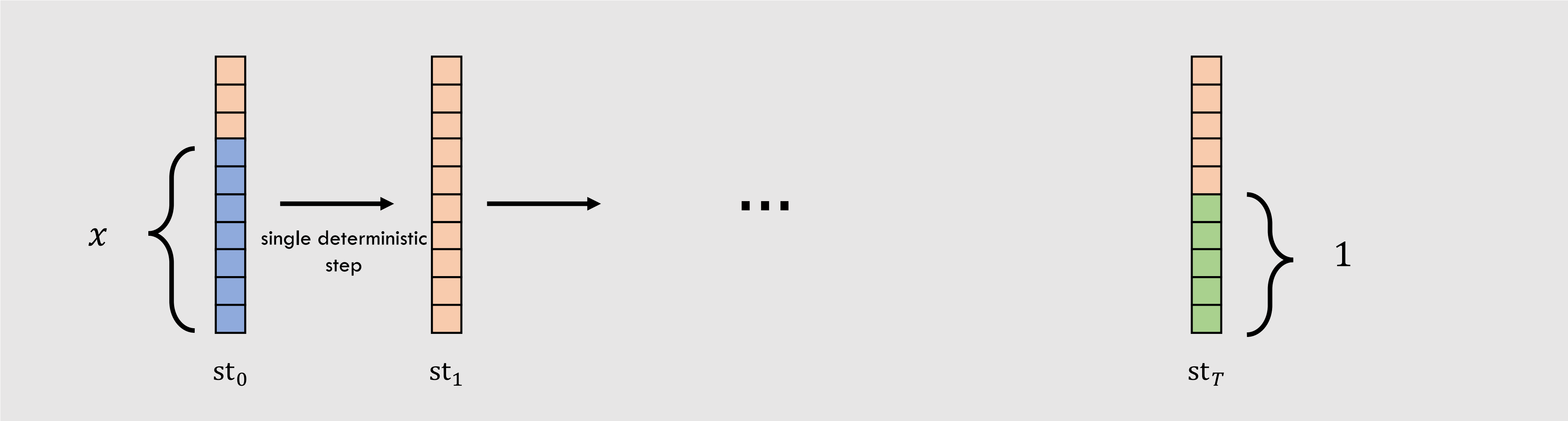
Needs to be Fiat-Shamir friendly.

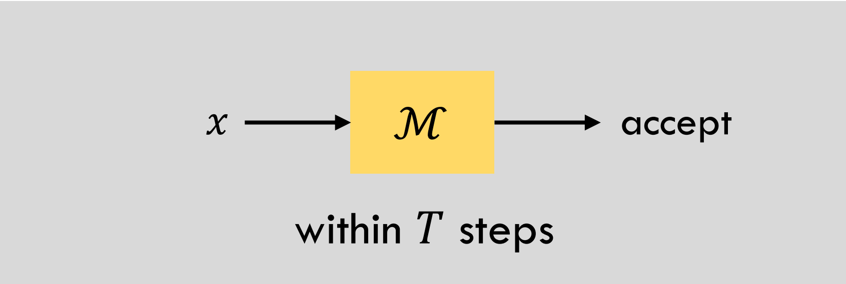
Based on LWE/sub-exp DDH

Results Overview



Delegation via Batching [Reingold-Rothblum-Rothblum'16]



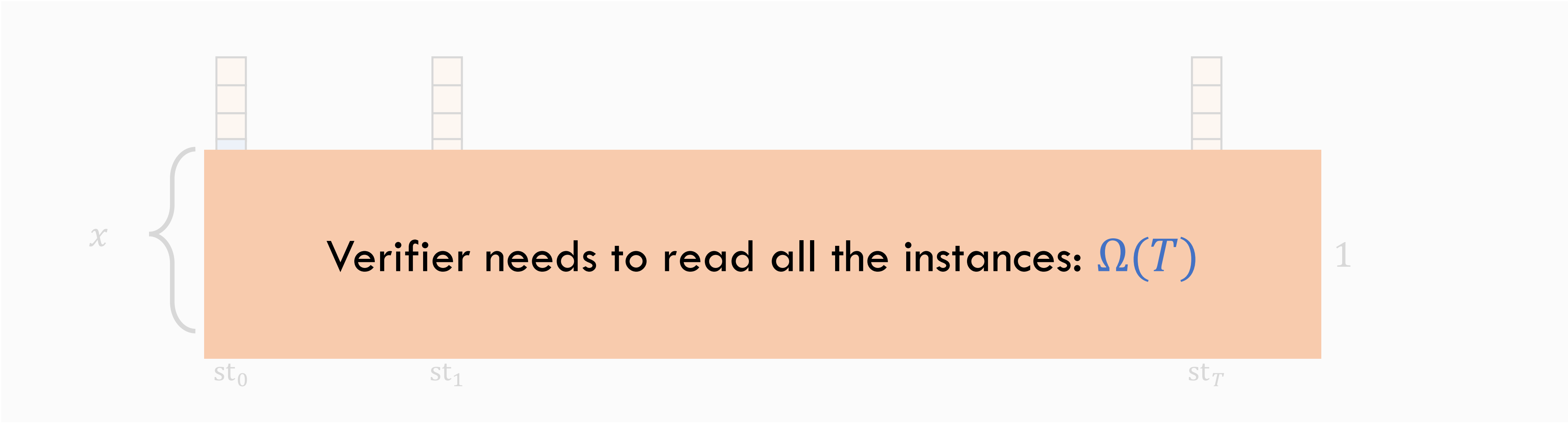


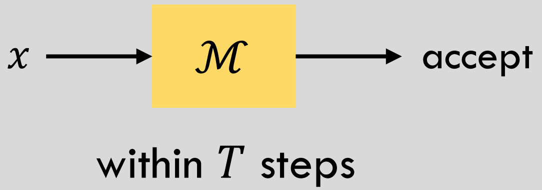
Prove for every $i \in [0, \ldots, T-1]$

$$\mathrm{st}_i \to \mathrm{st}_{i+1}$$

is the correct transition.

Delegation via Batching [Reingold-Rothblum-Rothblum'16]





Prove for every $i \in [0, \dots, T-1]$

$$\mathrm{st}_i \to \mathrm{st}_{i+1}$$

is the correct transition.

SNARGs for Batch Index

$$L_C = \{i \mid \exists w \; s.t. \; C(i, w) = 1\}$$

$$\forall i \in [k], i \in L_C$$

C, k

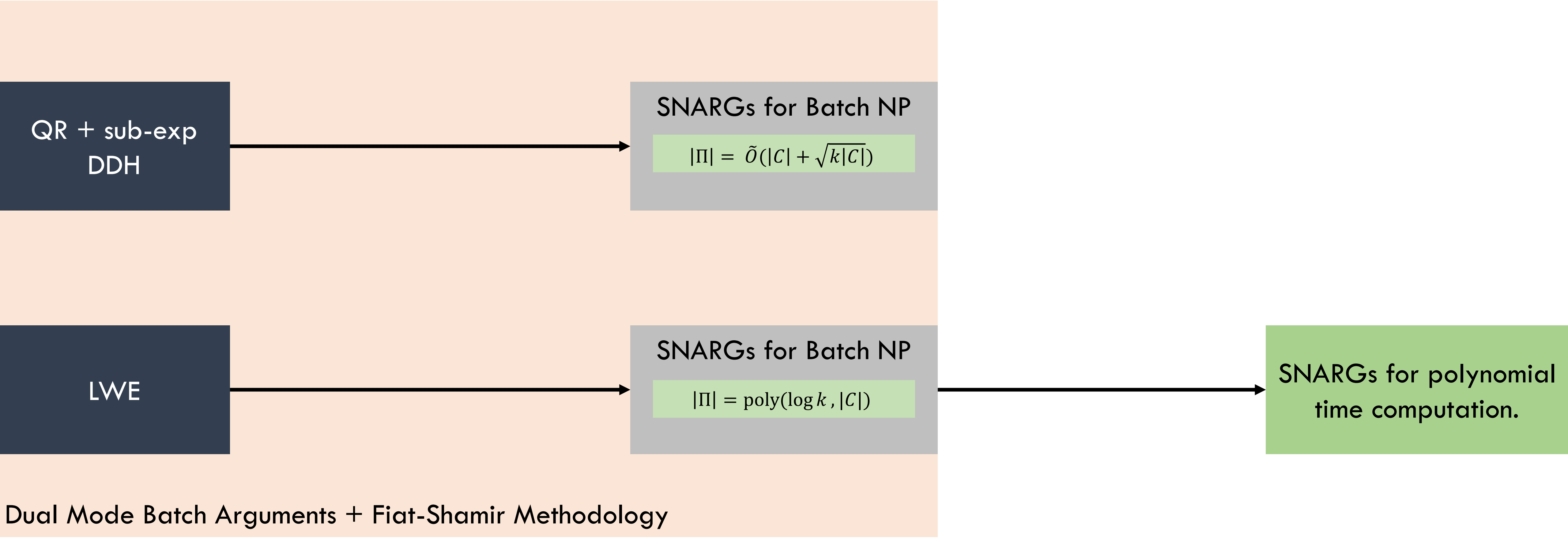
Π

C, k

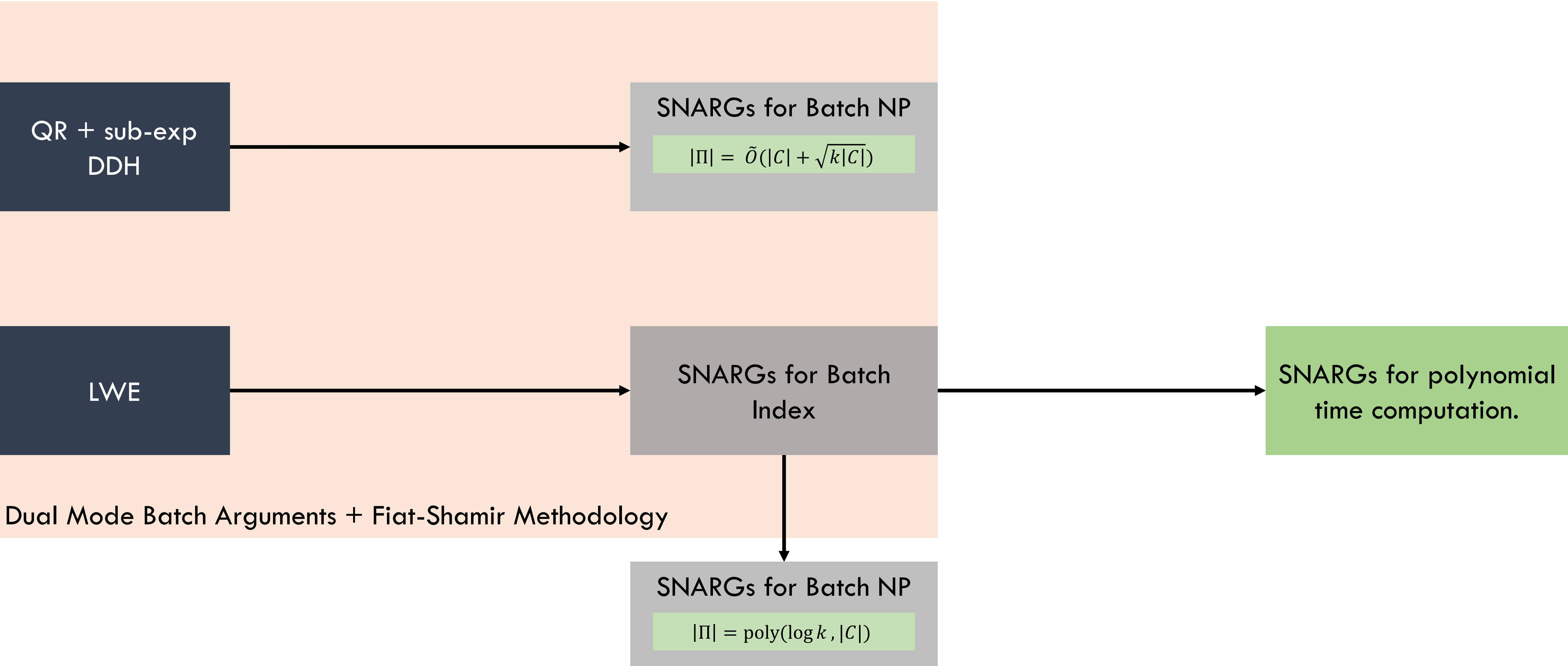
Verifier running time:

$$\mathrm{poly}(\log k, |C|)$$

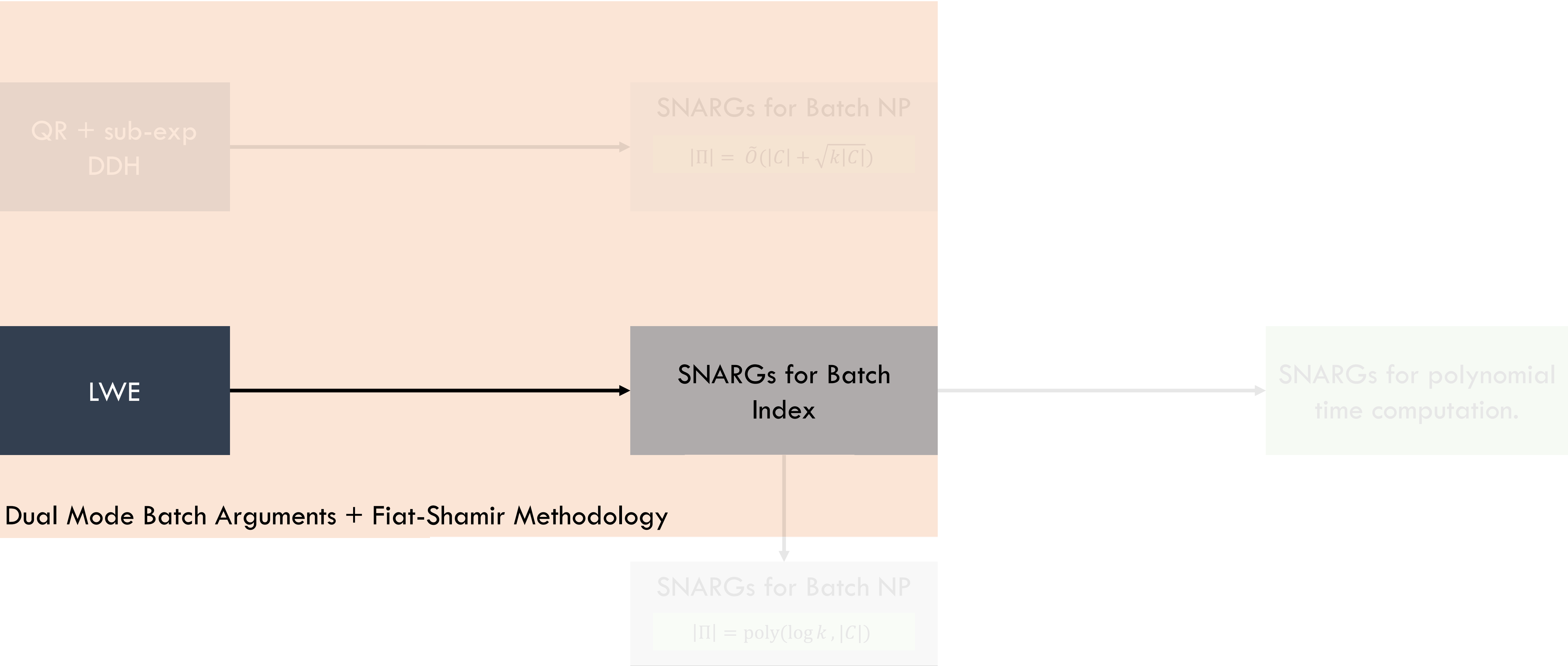
Results Overview



Results Overview



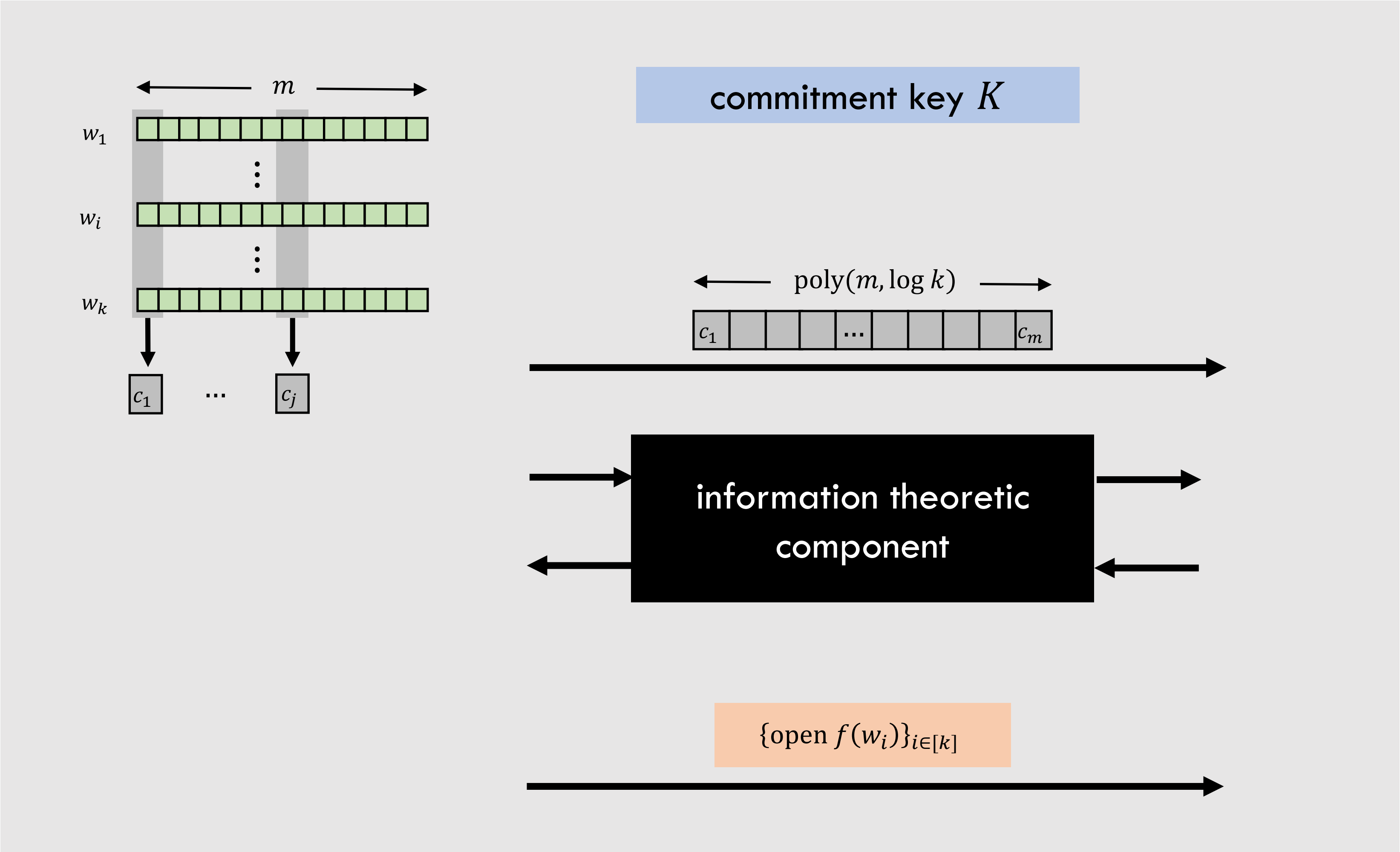
Results Overview



Dual Mode Argument for Batch Index

$L_C = \{i \mid \exists w \text{ s.t. } C(i,w) = 1\}$

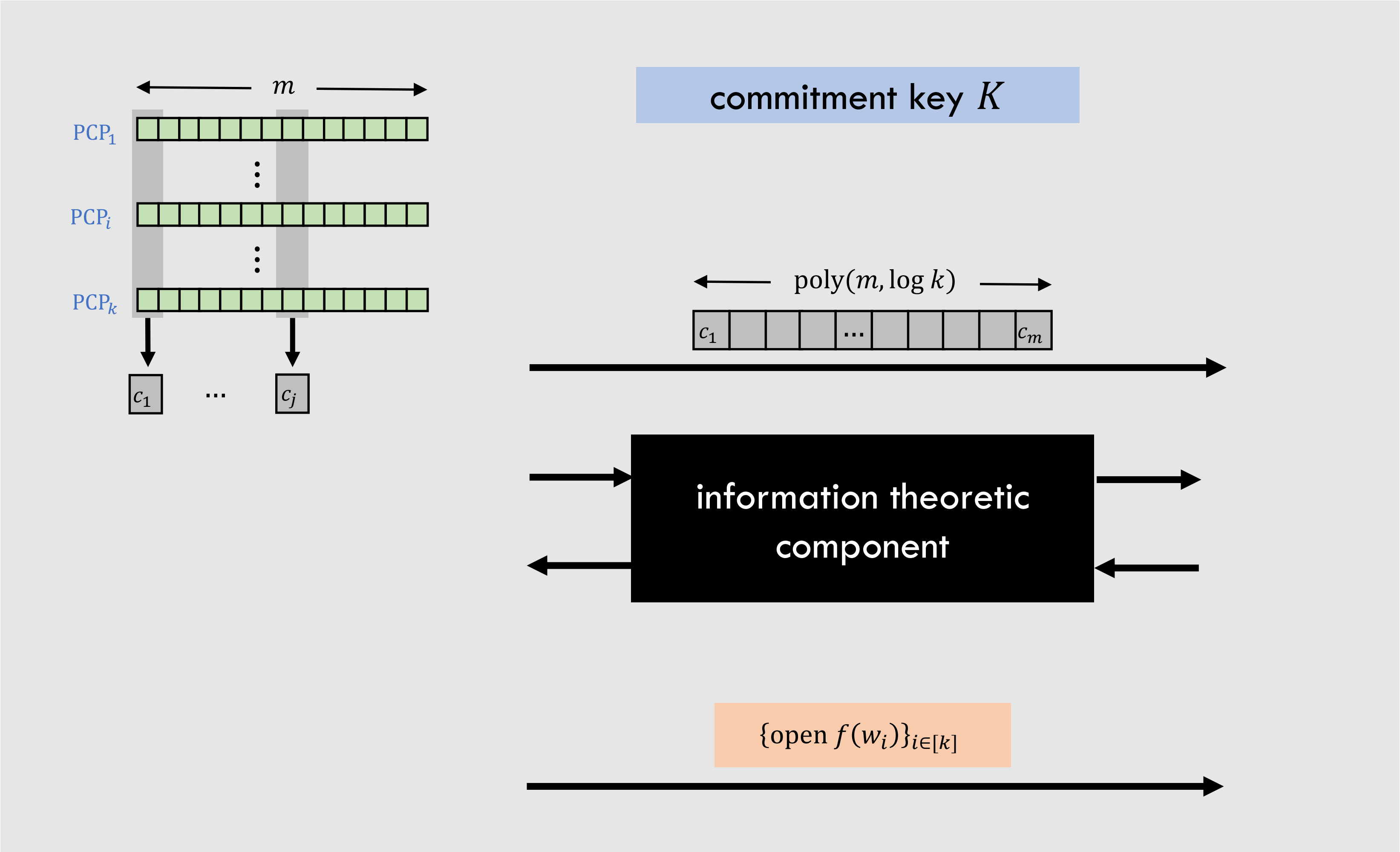
$\forall i \in [k], i \in L_C$



Dual Mode Argument for Batch Index

$L_C = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

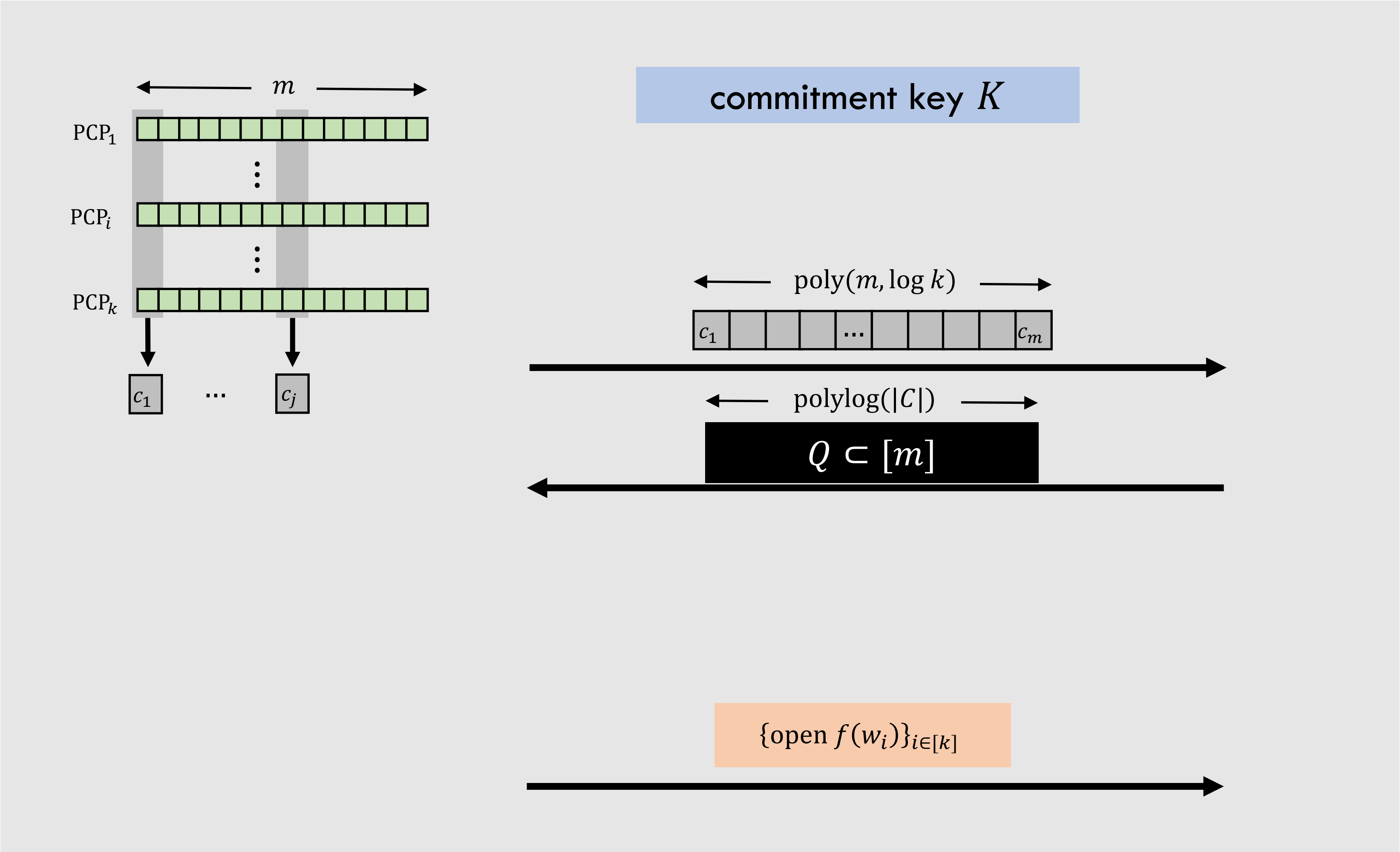
$\forall i \in [k], i \in L_C$



Dual Mode Argument for Batch Index

$L_C = \{i \mid \exists w \text{ s.t. } C(i,w) = 1\}$

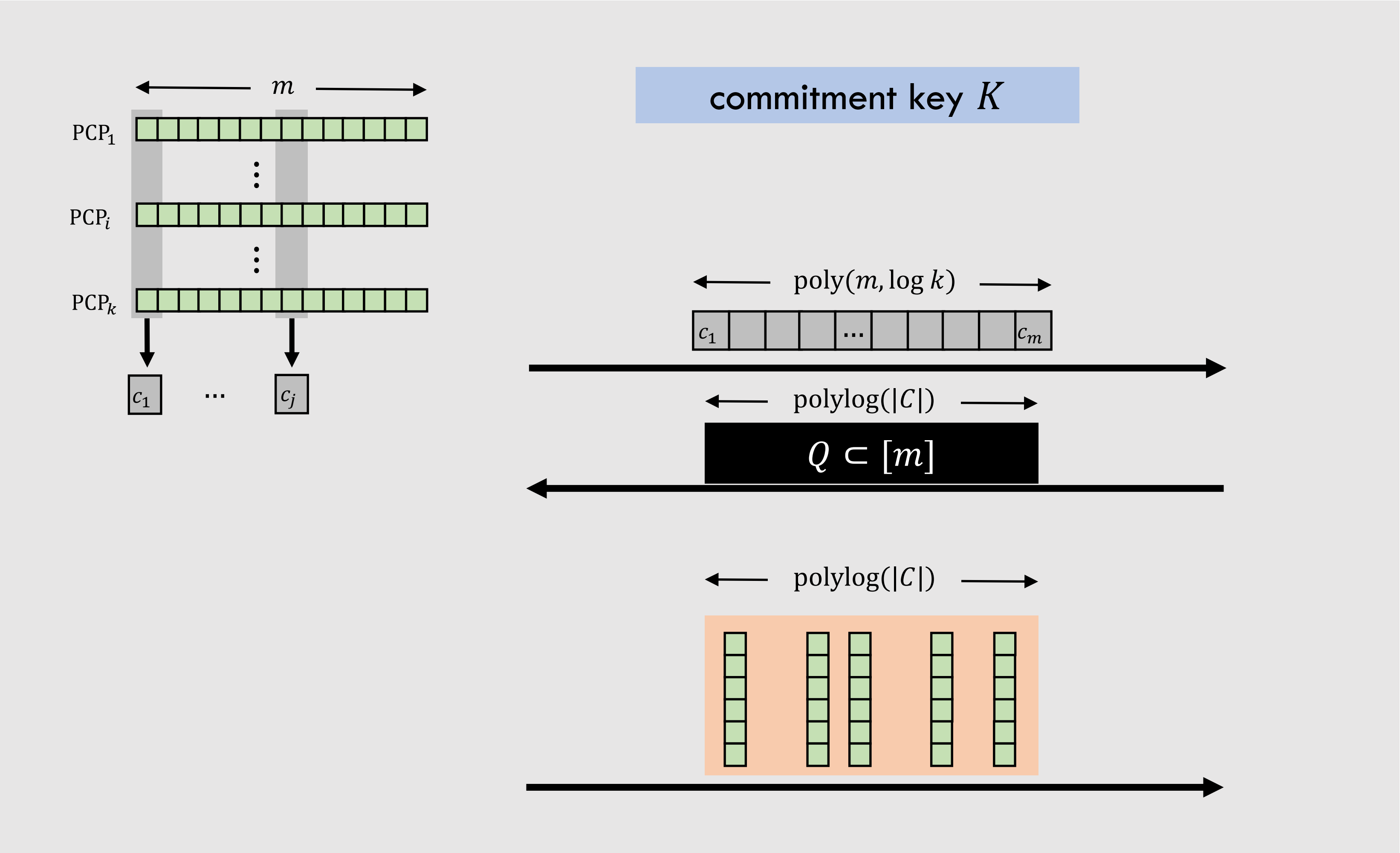
$\forall i \in [k], i \in L_C$



Dual Mode Argument for Batch Index

$L_C = \{i \mid \exists w \text{ s.t. } C(i,w) = 1\}$

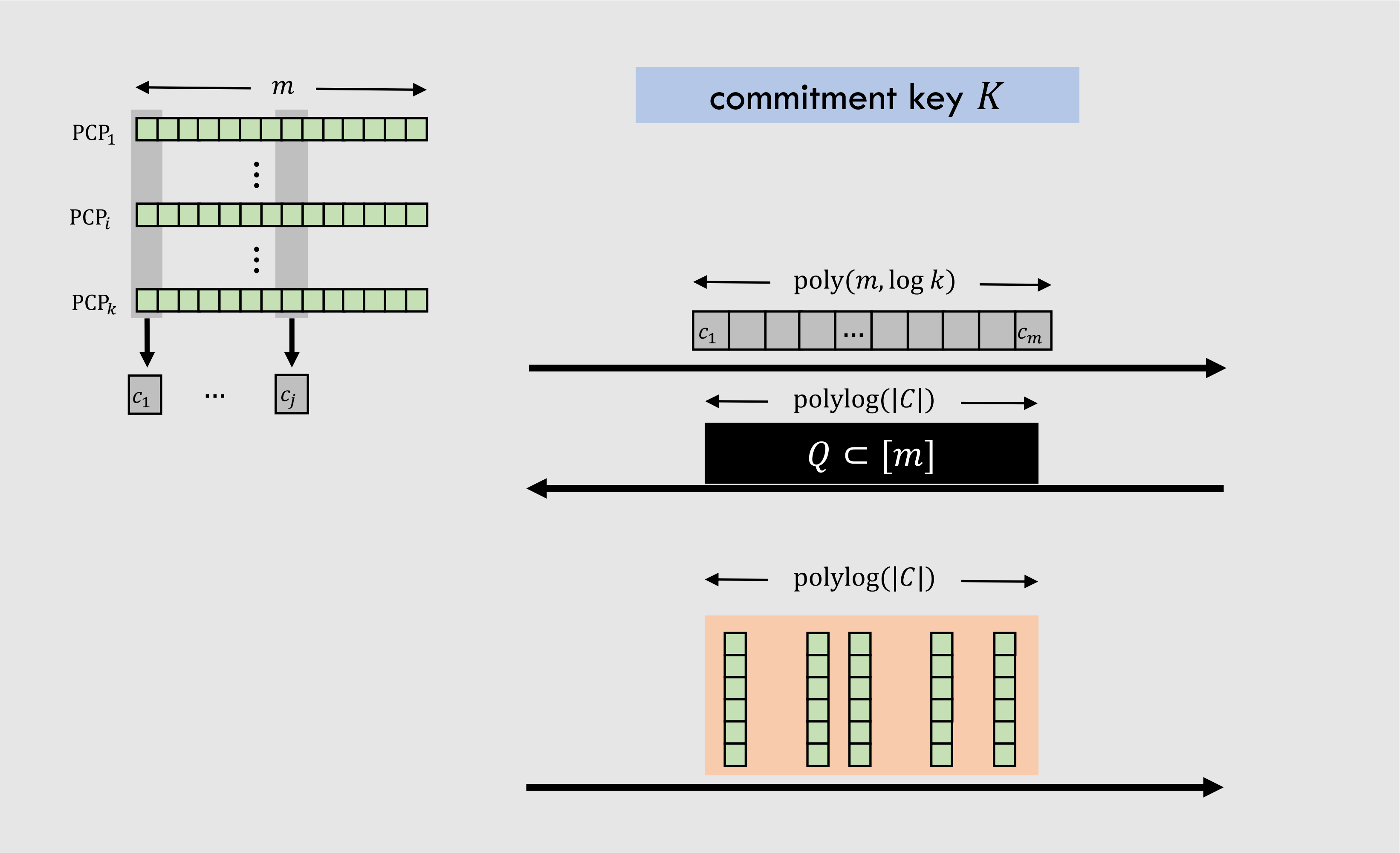
$\forall i \in [k], i \in L_C$



Dual Mode Argument for Batch Index

$L_C = \{i \mid \exists w \ s.t. \ C(i,w) = 1\}$

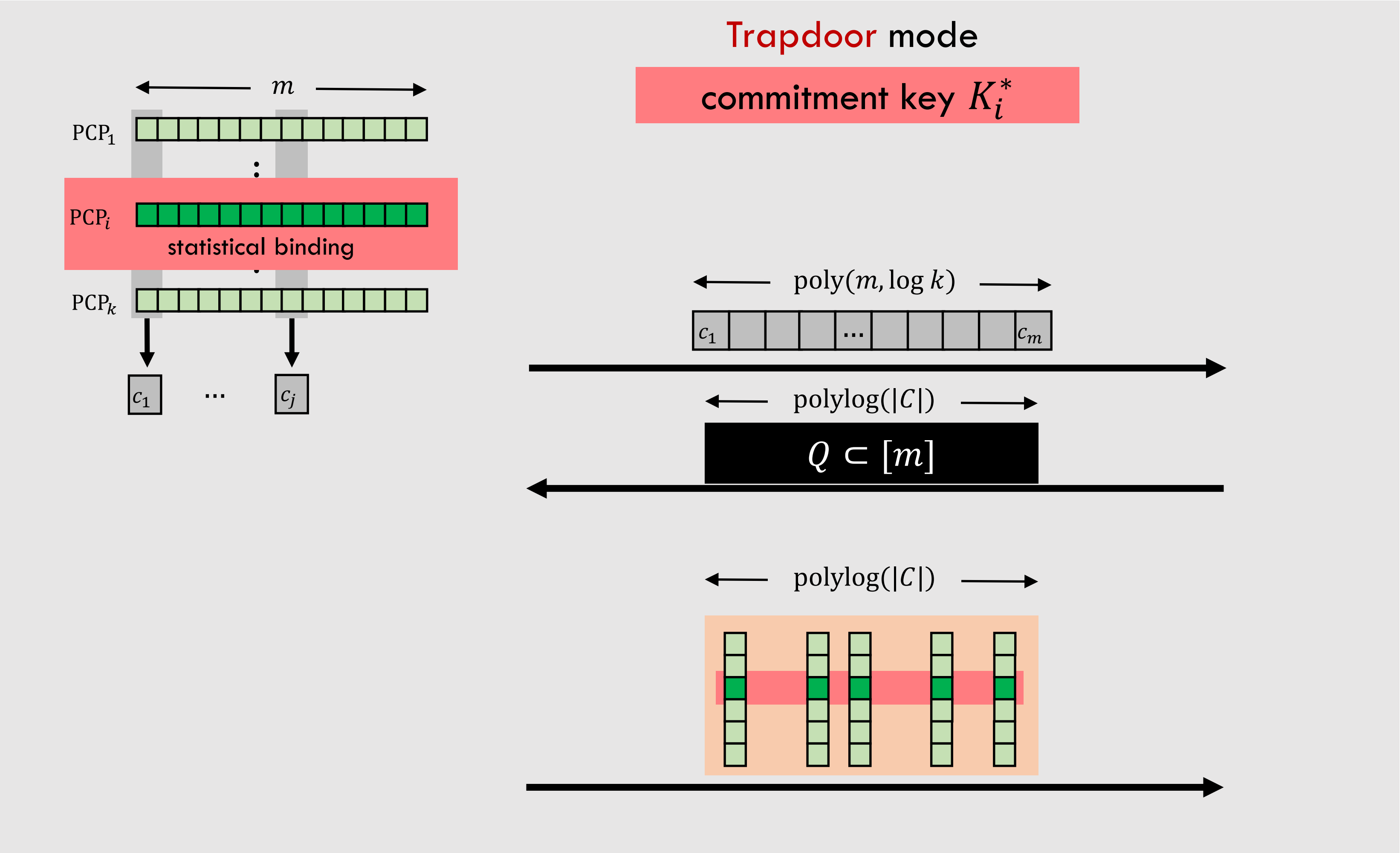
$\forall i \in [k], i \in L_C$



Verify:

1. Commitment openings are valid.
2. PCP responses verify on Q

Dual Mode Argument for Batch Index

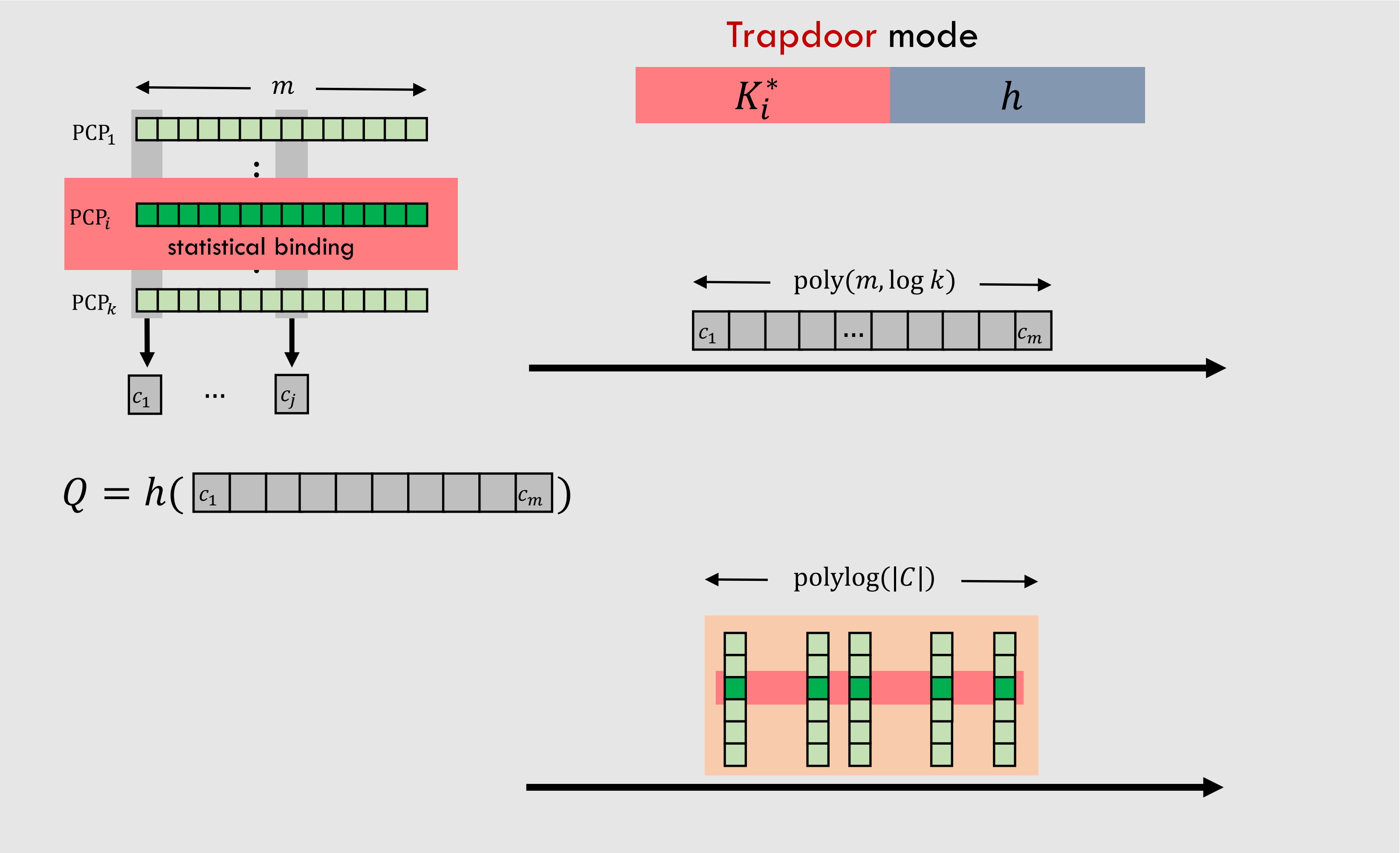


$L_C = \{i \mid \exists w \text{ s.t. } C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

Verify:
1. Commitment openings are valid.
2. PCP responses verify on Q

Dual Mode Argument for Batch Index

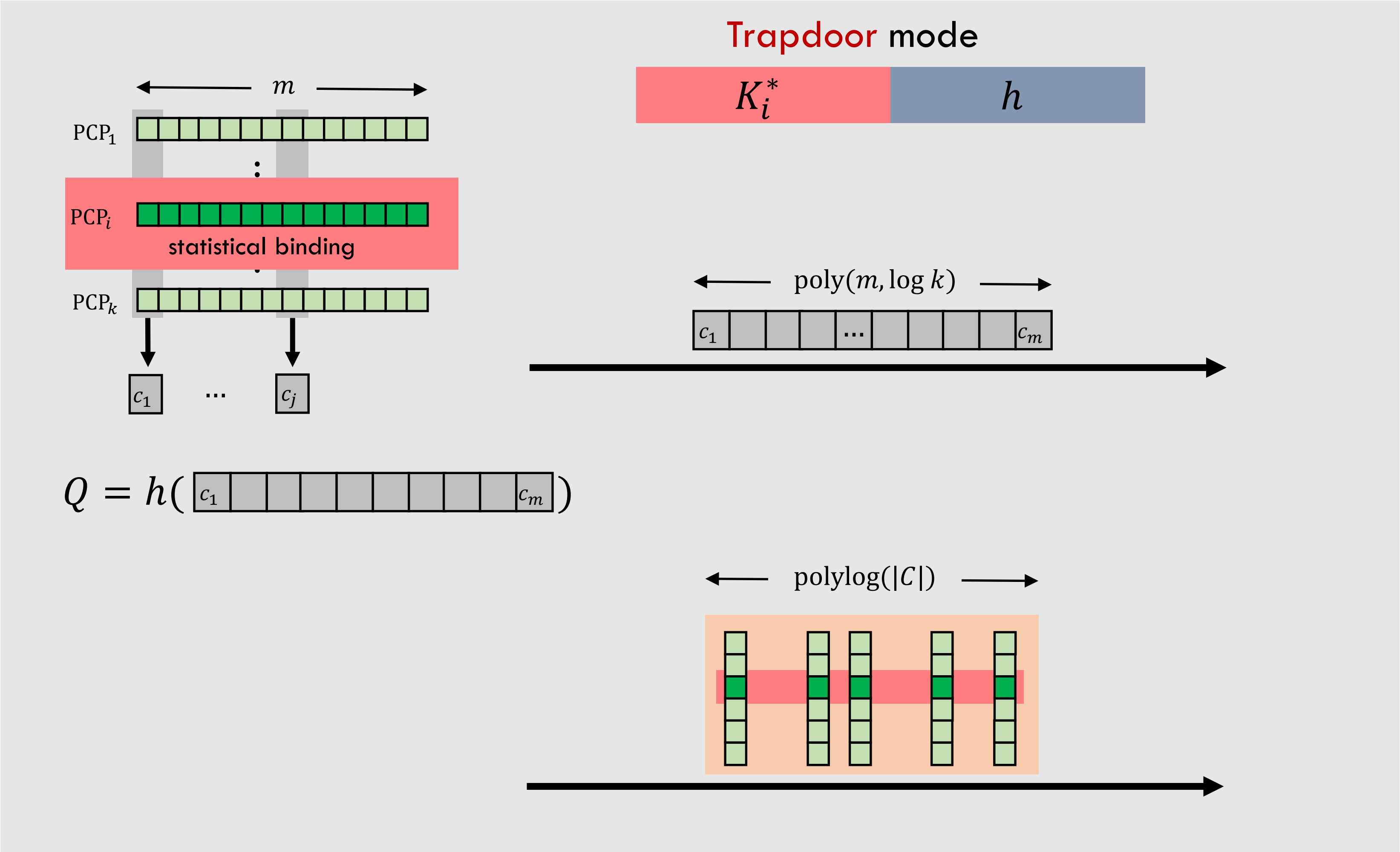


$L_C = \{i \mid \exists w \text{ s.t. } C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

Verify:
1. Commitment openings are valid.
2. PCP responses verify on Q

Dual Mode Argument for Batch Index



$L_C = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

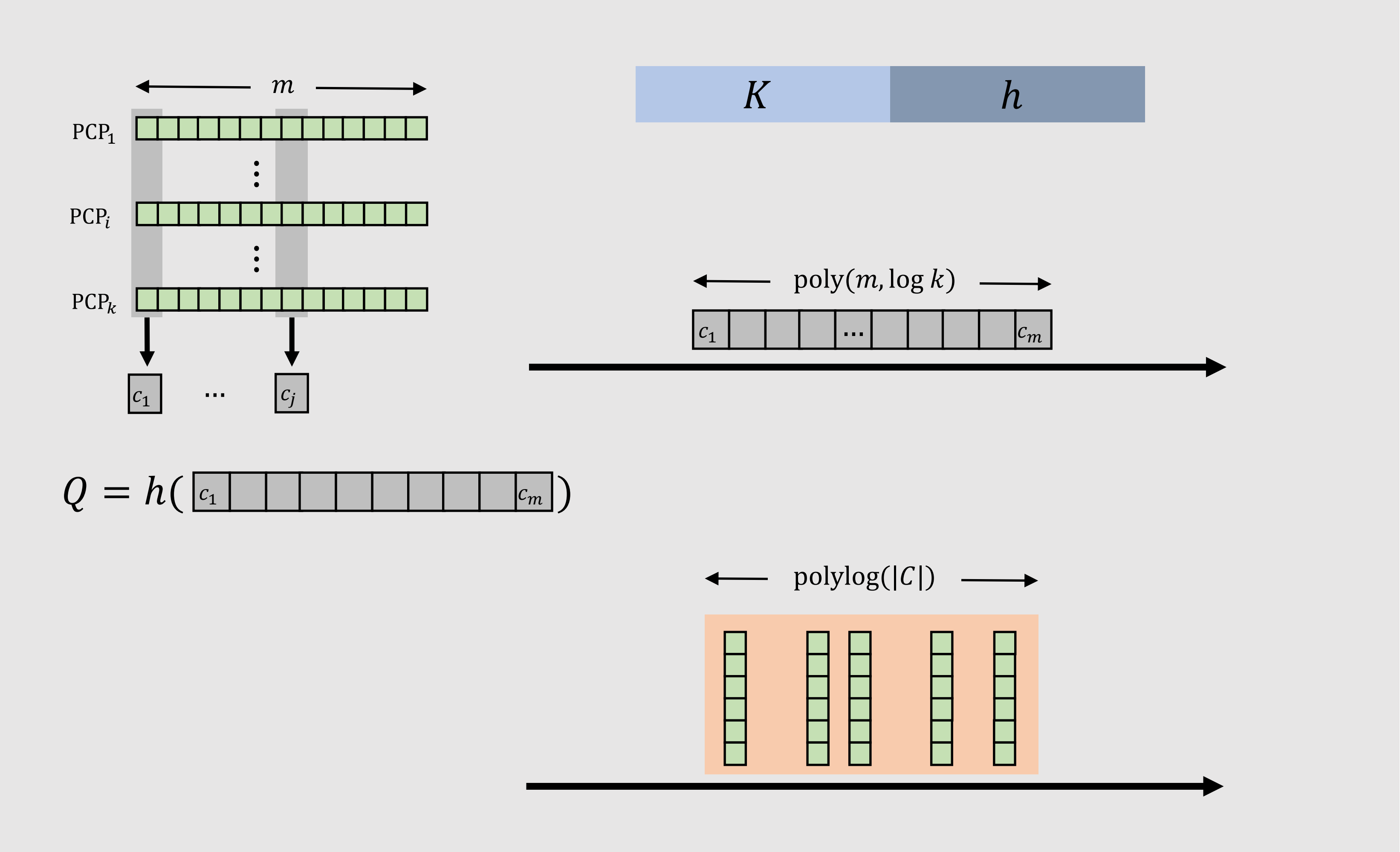
$\forall i \in [k], i \in L_C$

[Holmgren-Lombardi-Rothblum'21]
Assuming LWE, the transformation is sound.

Verify:

1. Commitment openings are valid.
2. PCP responses verify on Q

SNARG for Batch Index



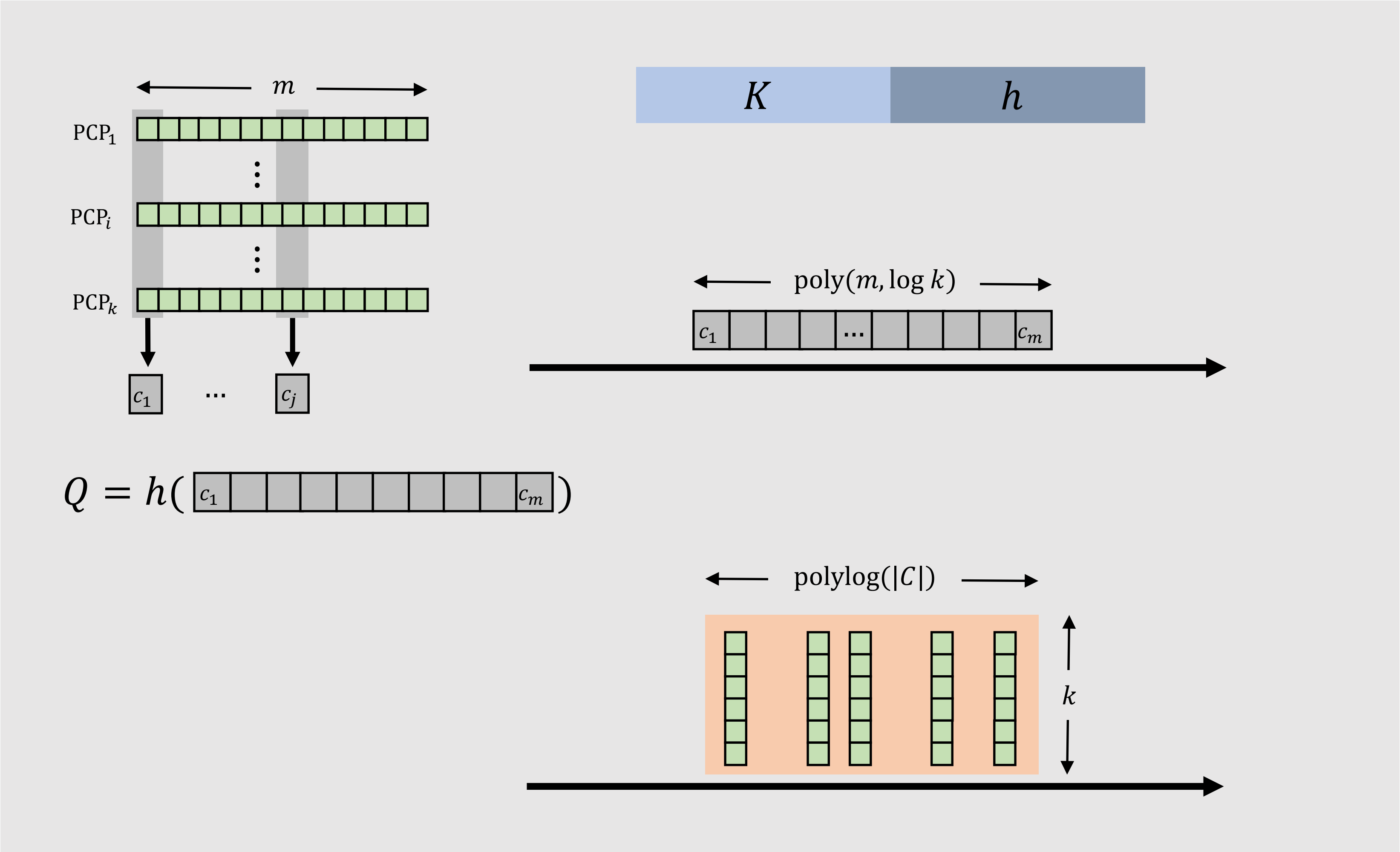
$L_C = \{i \mid \exists w \ s.t.\ C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

Verify:

1. Commitment openings are valid.
2. PCP responses verify on Q

SNARG for Batch Index



$L_C = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

|  | $= \Omega(k)$

Verify:

1. Commitment openings are valid.
2. PCP responses verify on Q

SNARG for Batch Index

Send proof of verification

$L_C = \{i \mid \exists w \text{ s.t. } C(i,w) = 1\}$

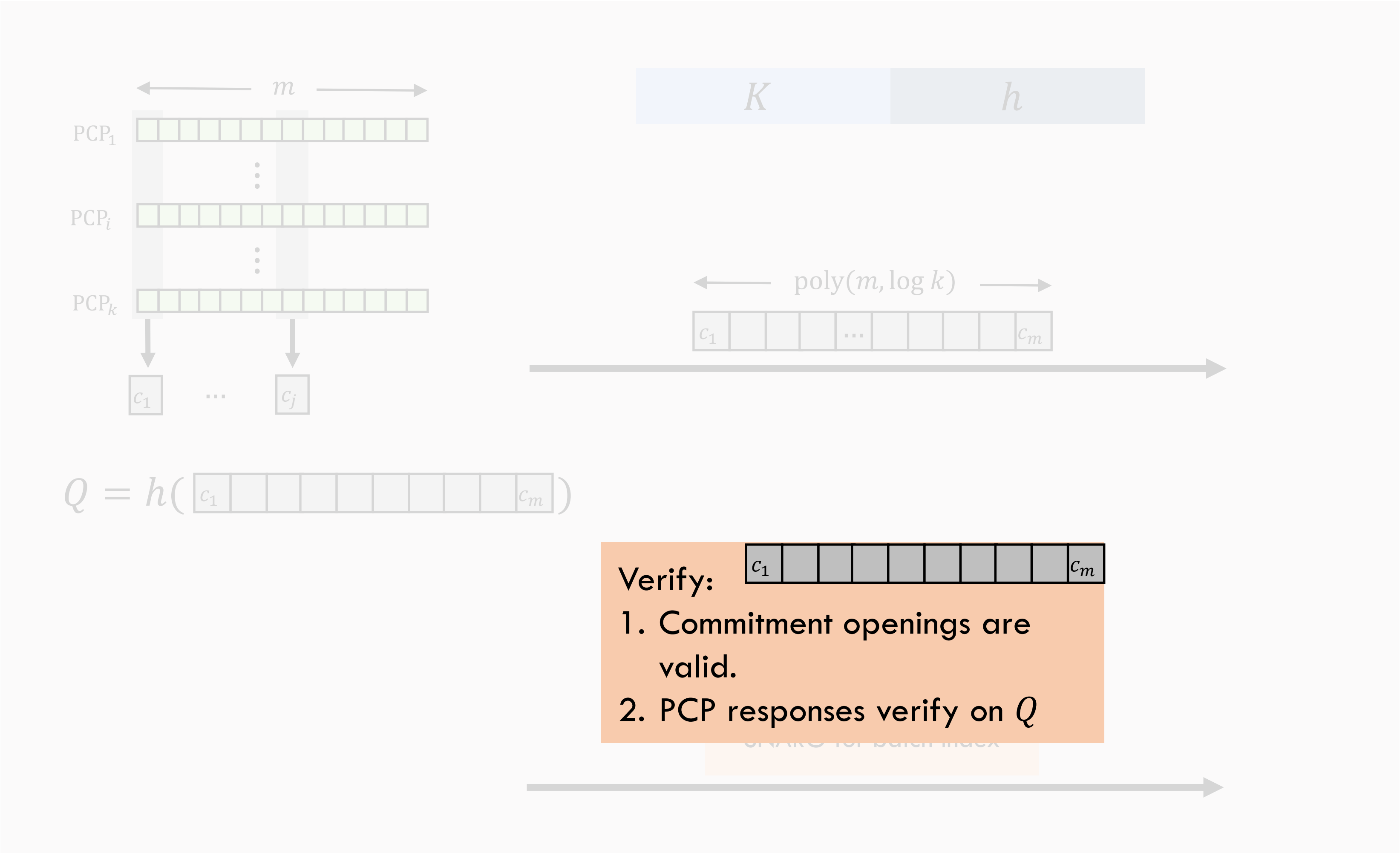
$\forall i \in [k], i \in L_C$

|  | $= \Omega(k)$

Verify:

1. Commitment openings are valid.
2. PCP responses verify on Q

SNARG for Batch Index



Verify: $c_1 \ \ldots \ c_m$

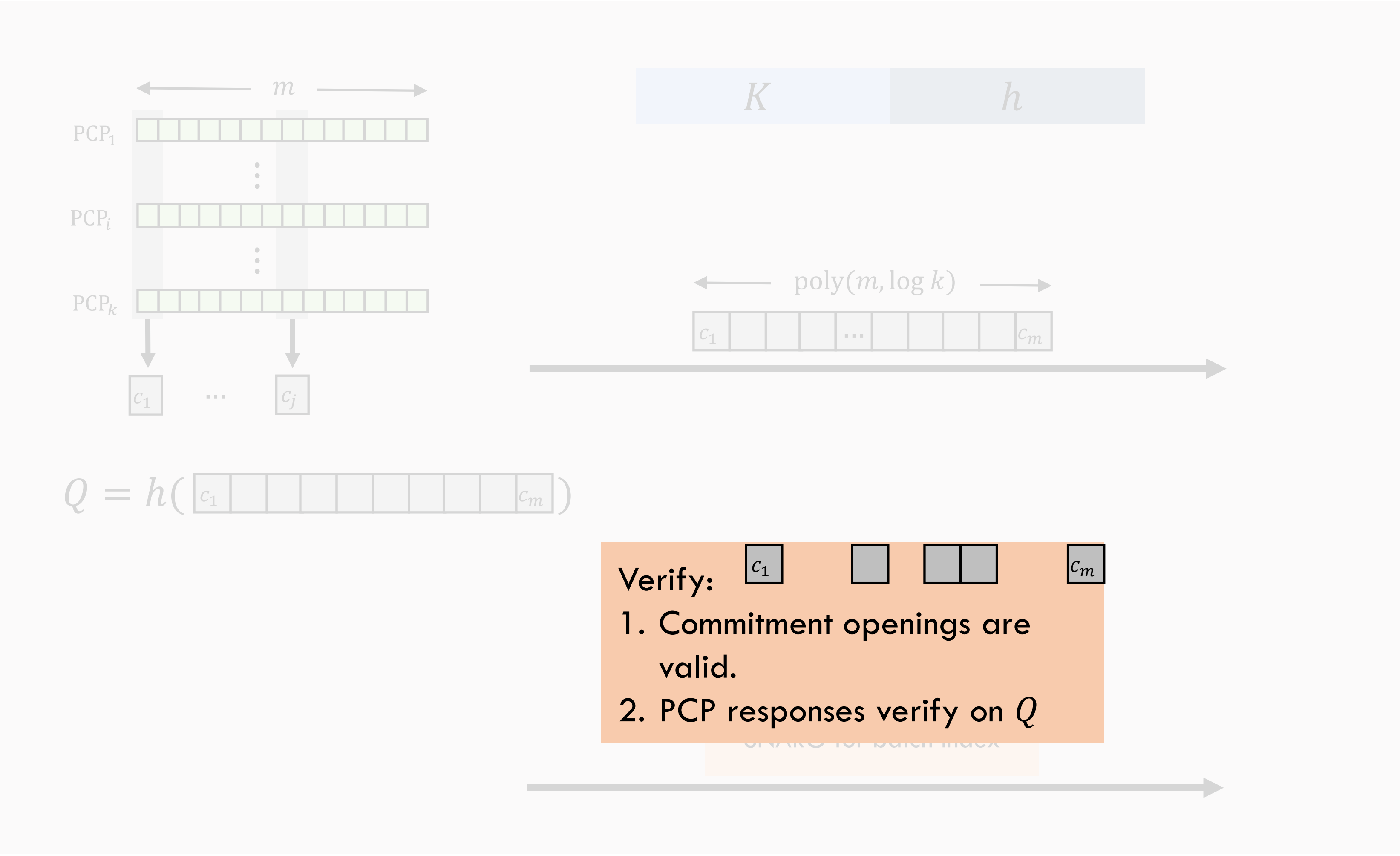
1. Commitment openings are valid.
2. PCP responses verify on Q

$L_C = \{i \mid \exists w \ s.t. \ C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

|  | $= \Omega(k)$

SNARG for Batch Index



Verify: c_1 … c_m

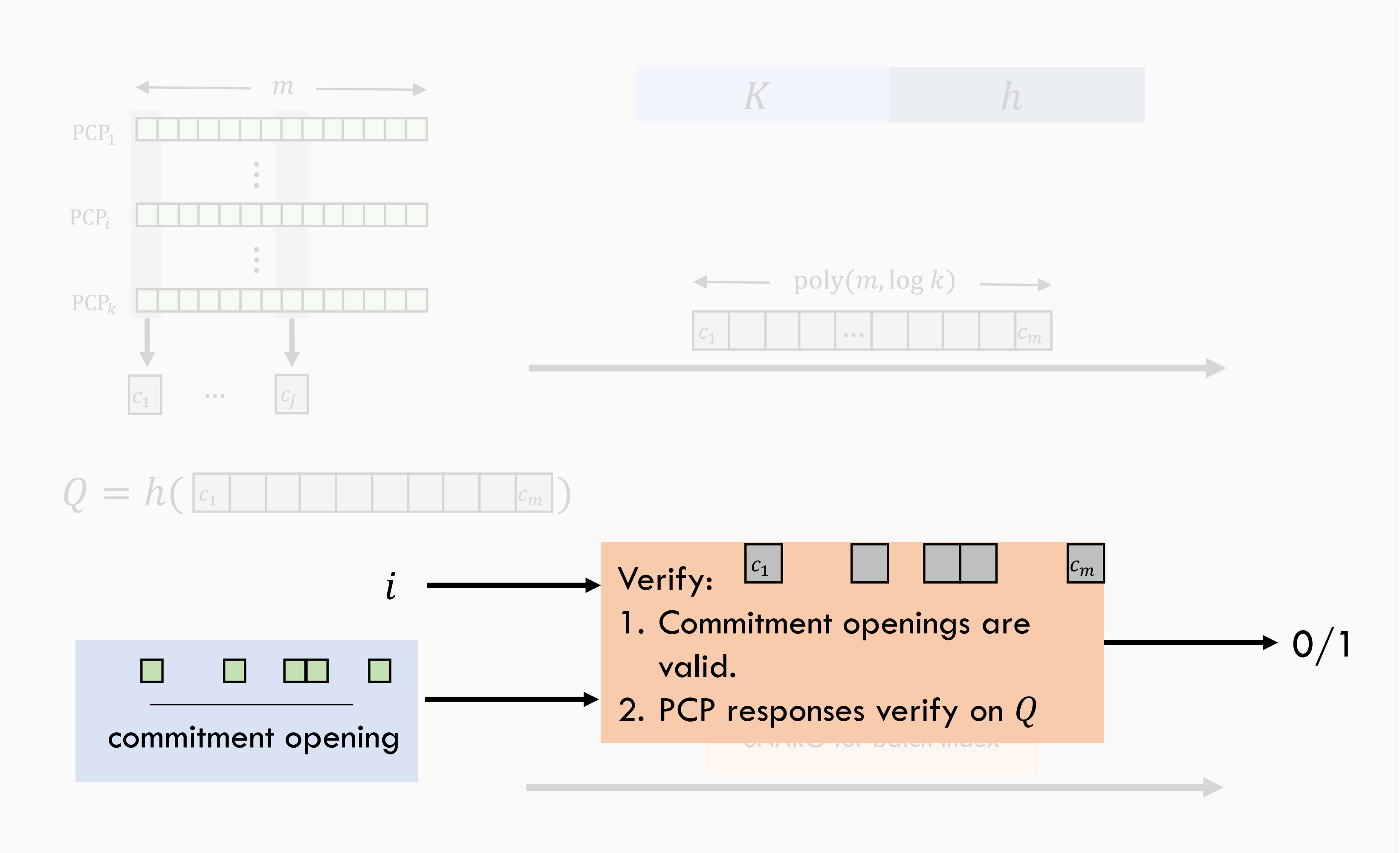
1. Commitment openings are valid.
2. PCP responses verify on Q

$L_C = \{i \mid \exists w\ s.t.\ C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

|  | $= \Omega(k)$

SNARG for Batch Index

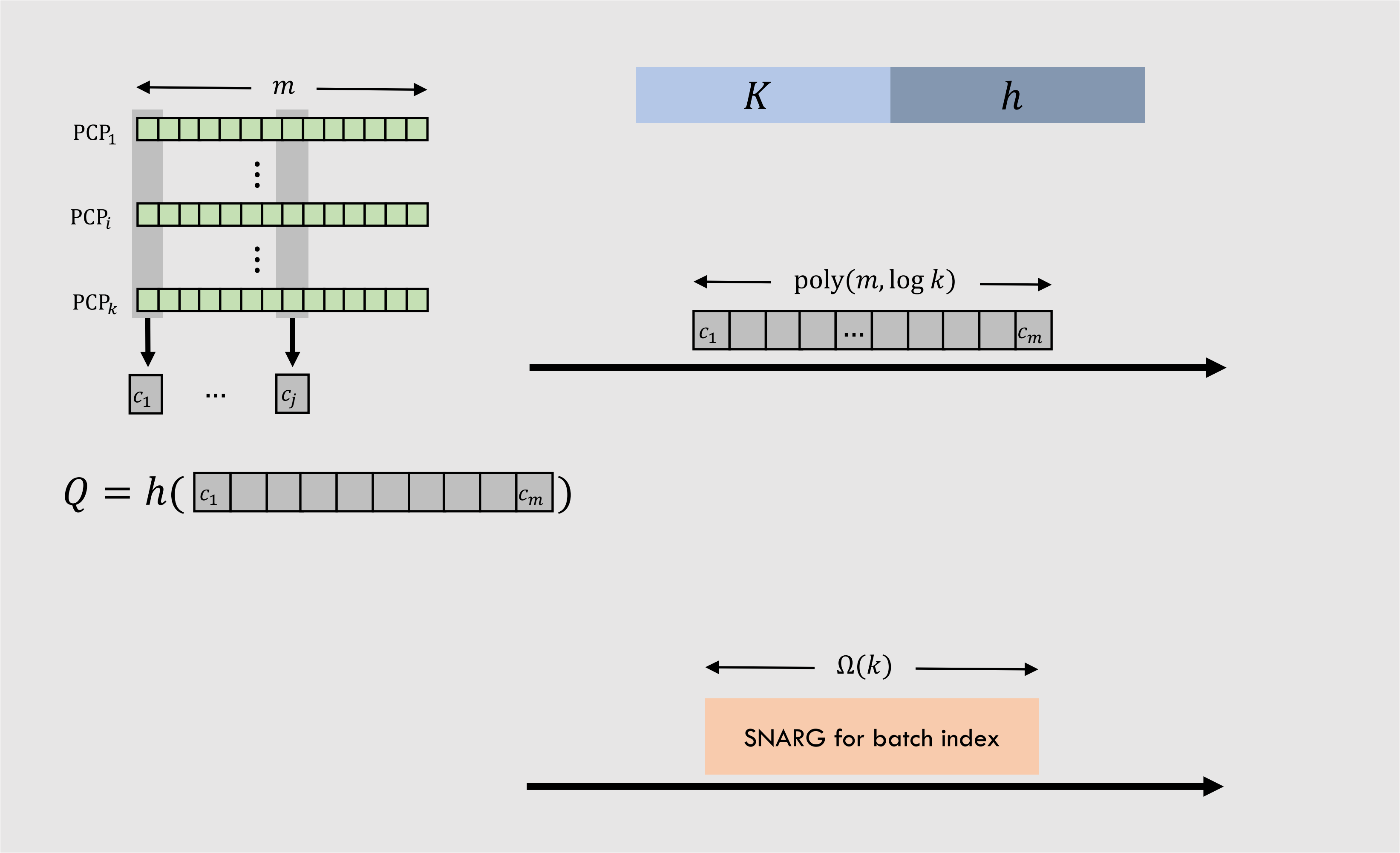


$L_C = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

$| \; \cdot \; | = \Omega(k)$

SNARG for Batch Index



$L_C = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

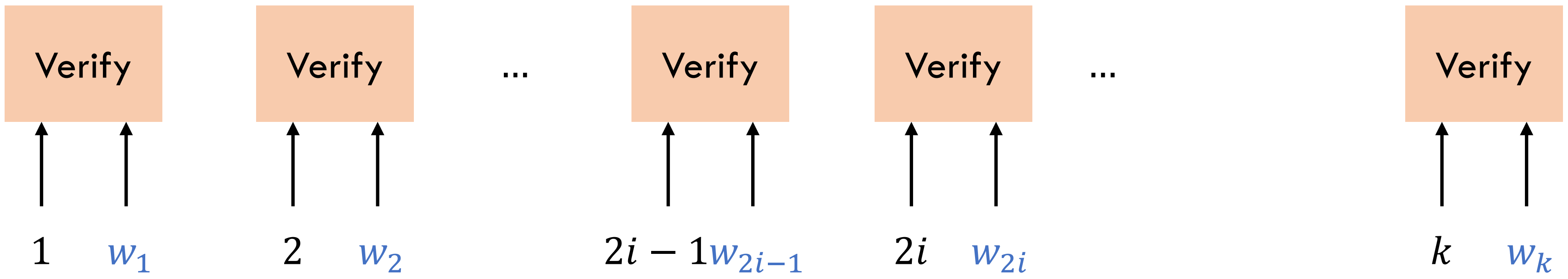
$\forall i \in [k], i \in L_C$

|  | $= \Omega(k)$

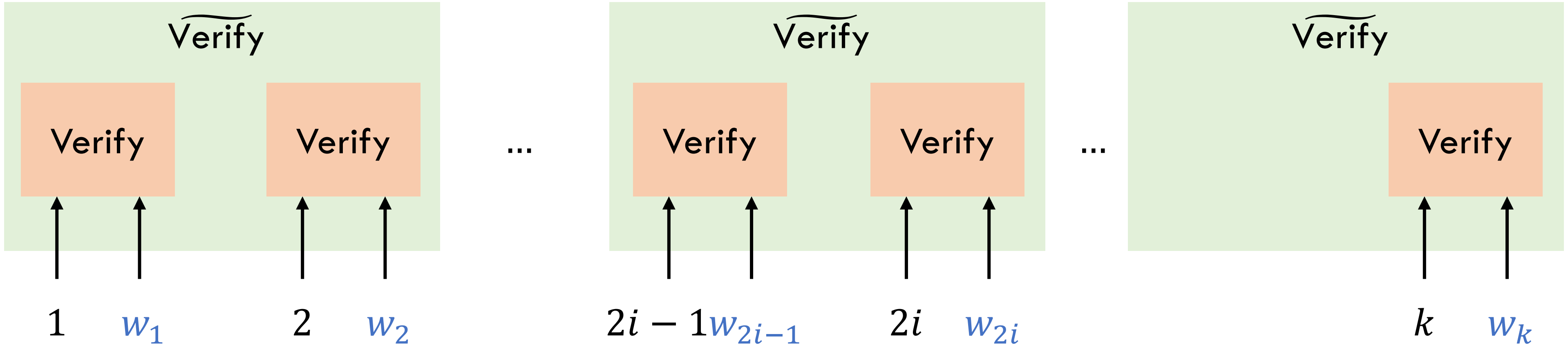
Verify: c_1 c_m

1. Commitment openings are valid.
2. PCP responses verify on Q

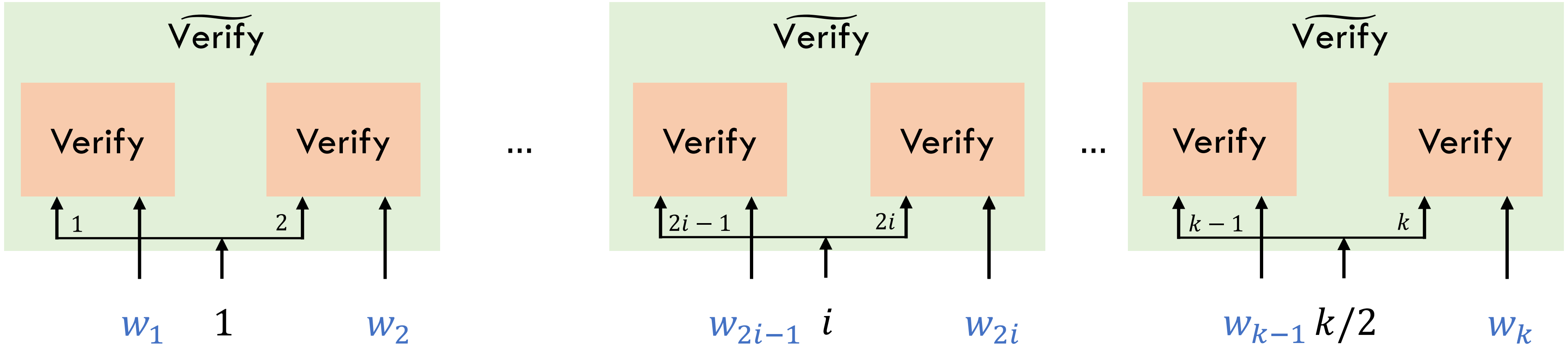
Grouping Instances for Recursion



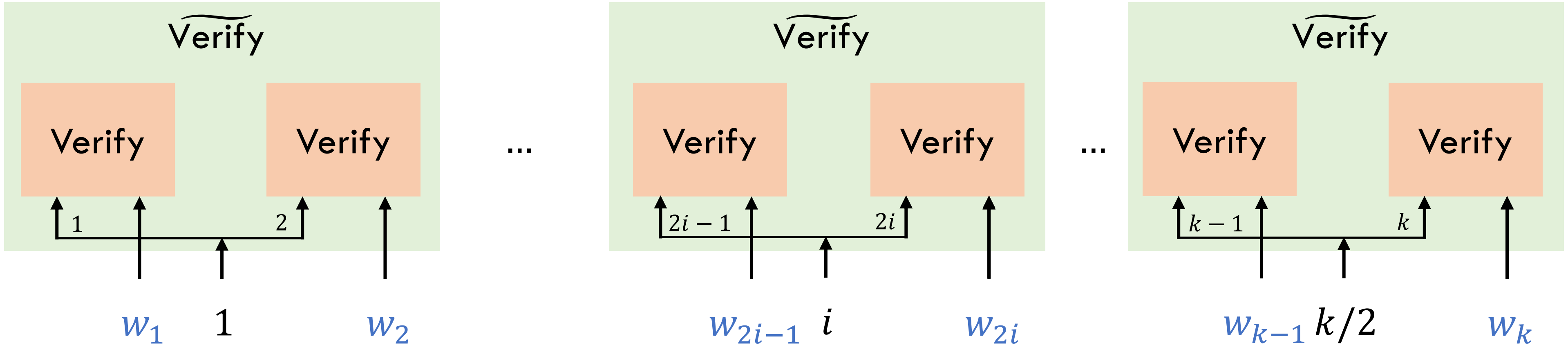
Grouping Instances for Recursion



Grouping Instances for Recursion



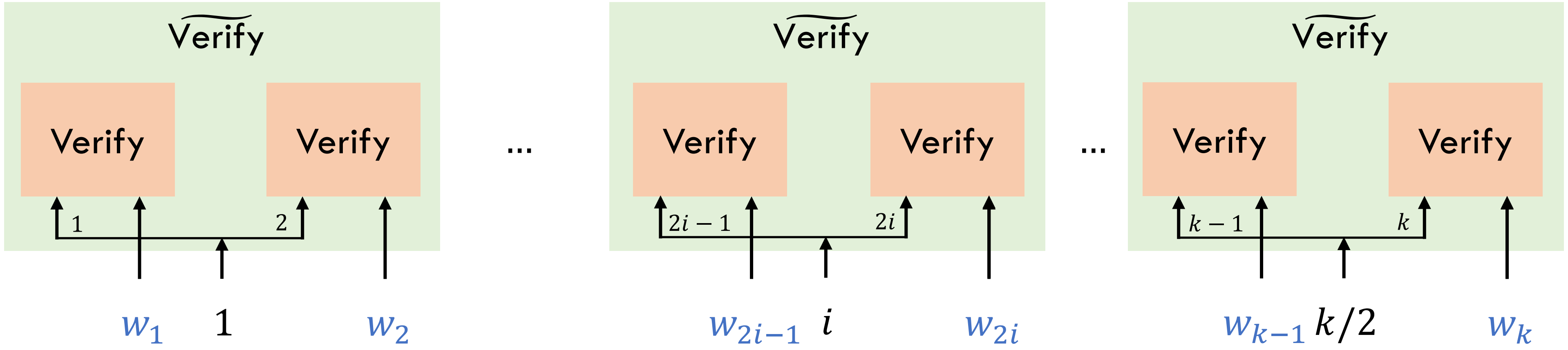
Grouping Instances for Recursion



$k/2$ instances for circuit $\widetilde{\text{Verify}}$

Grouping Instances for Recursion

$|\widetilde{\text{Verify}}| \geq 2|\text{Verify}|$



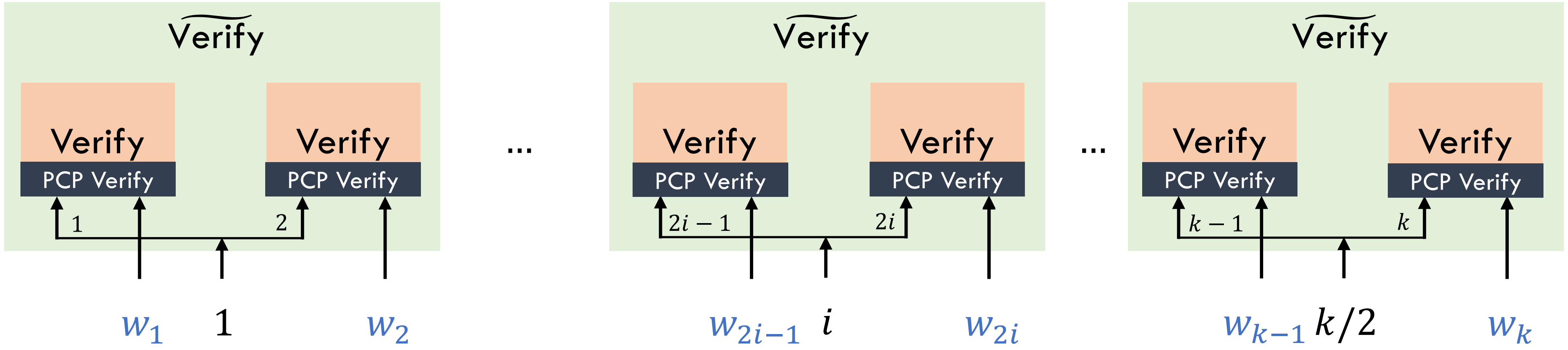
$k/2$ instances for circuit $\widetilde{\text{Verify}}$

Grouping Instances for Recursion

$L_C \; = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

$|\widetilde{\text{Verify}}| \geq 2|\text{Verify}|$



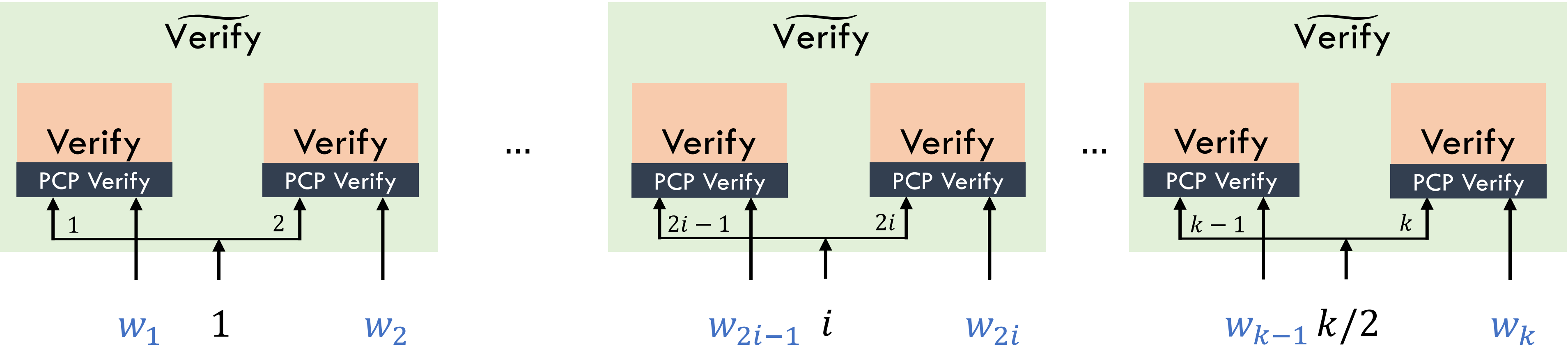
$k/2$ instances for circuit $\widetilde{\text{Verify}}$

Grouping Instances for Recursion

$L_C = \{i \mid \exists w \ s.t.\ C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

$\left|\widetilde{\text{Verify}}\right| \geq 2|\text{Verify}| \geq 2|C|$



$k/2$ instances for circuit $\widetilde{\text{Verify}}$

Grouping Instances for Recursion

$L_C = \{i \mid \exists w \text{ s.t. } C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

$|\widetilde{\text{Verify}}| \geq 2|\text{Verify}| \geq 2|C|$

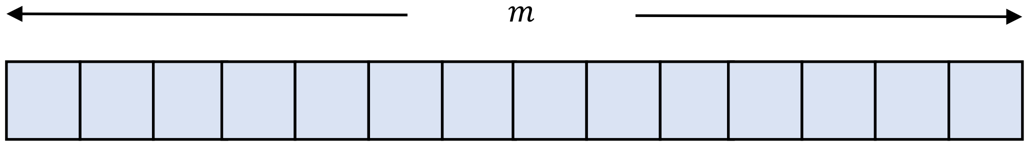
k instances for circuits of size $|C|$ → $k/2$ instances for circuits of size $2|C|$ → … → $O(1)$ instances for circuits of size $k|C|$

$\log k$ steps of recursion

Tool: PCP with Fast Online Verification

C, x, w

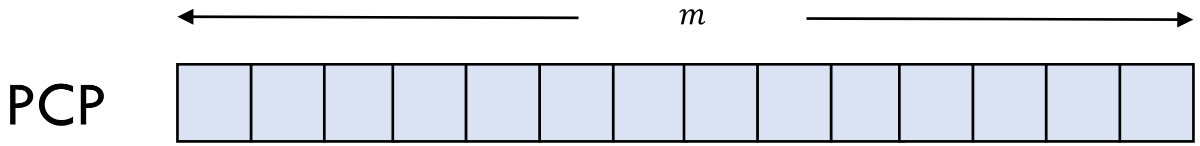
PCP



C, x

Tool: PCP with Fast Online Verification

C, x, w



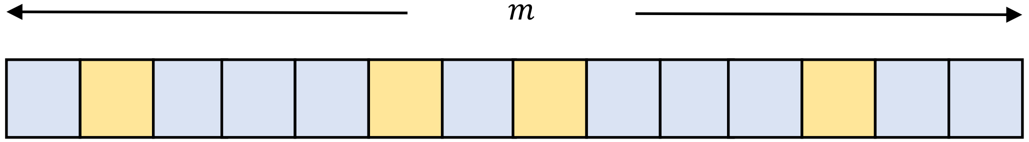
C, x

$Q \leftarrow [m]$

Tool: PCP with Fast Online Verification

C, x, w

PCP

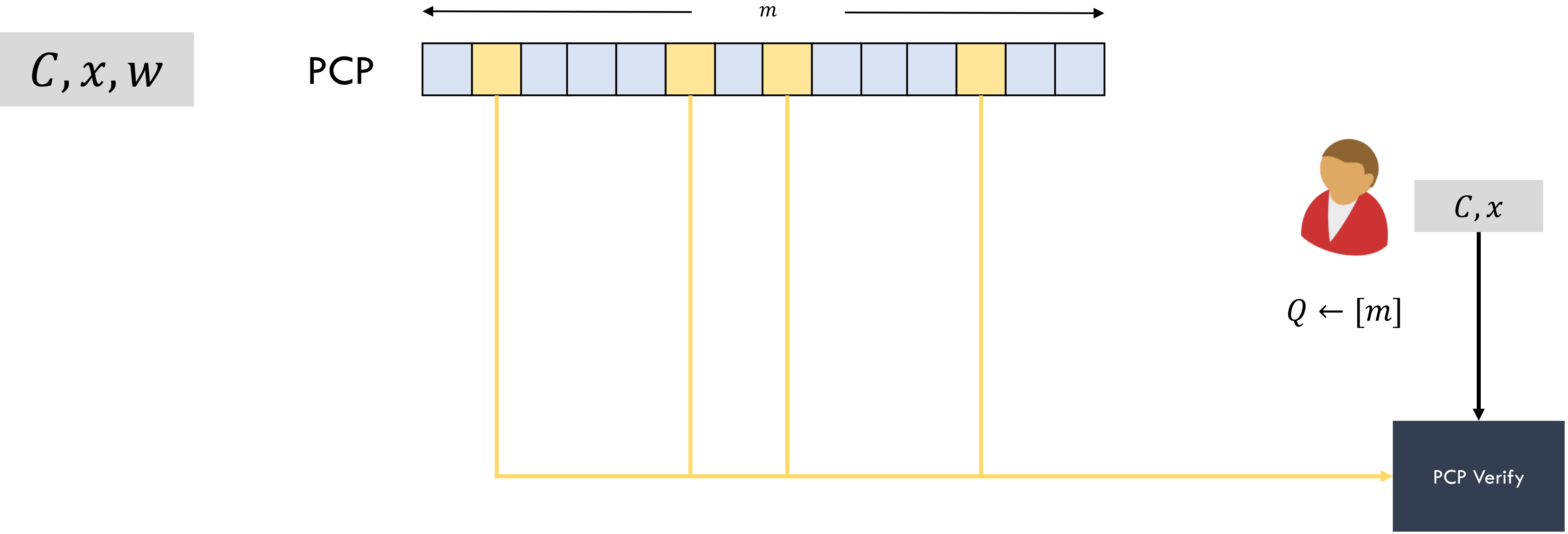


C, x

$Q \leftarrow [m]$

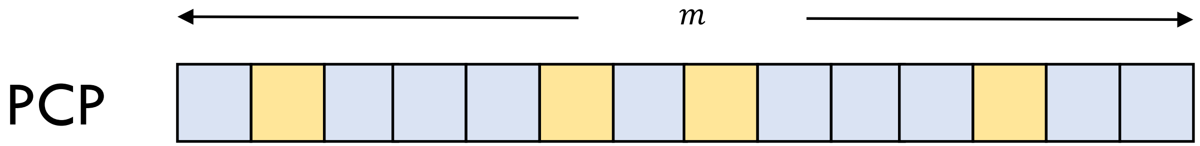
PCP Verify

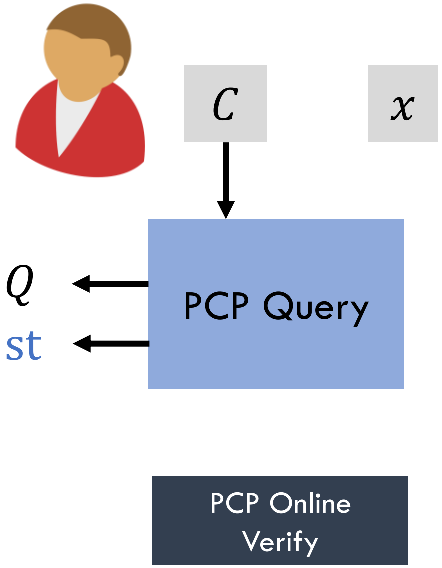
Tool: PCP with Fast Online Verification



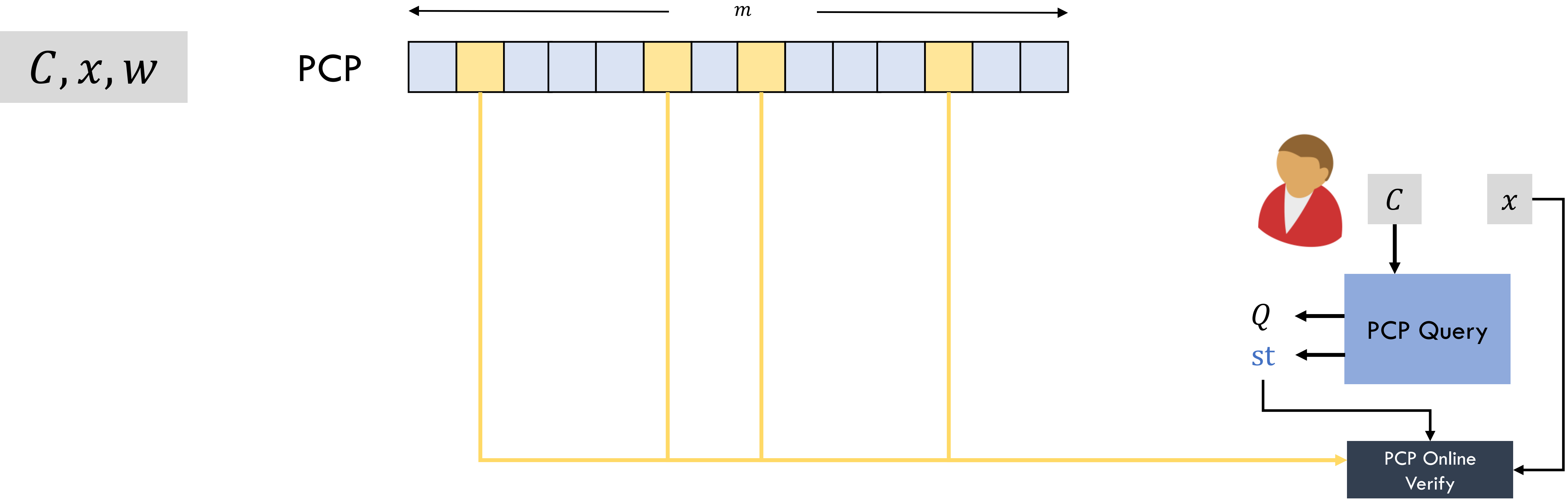
Tool: PCP with Fast Online Verification

C, x, w



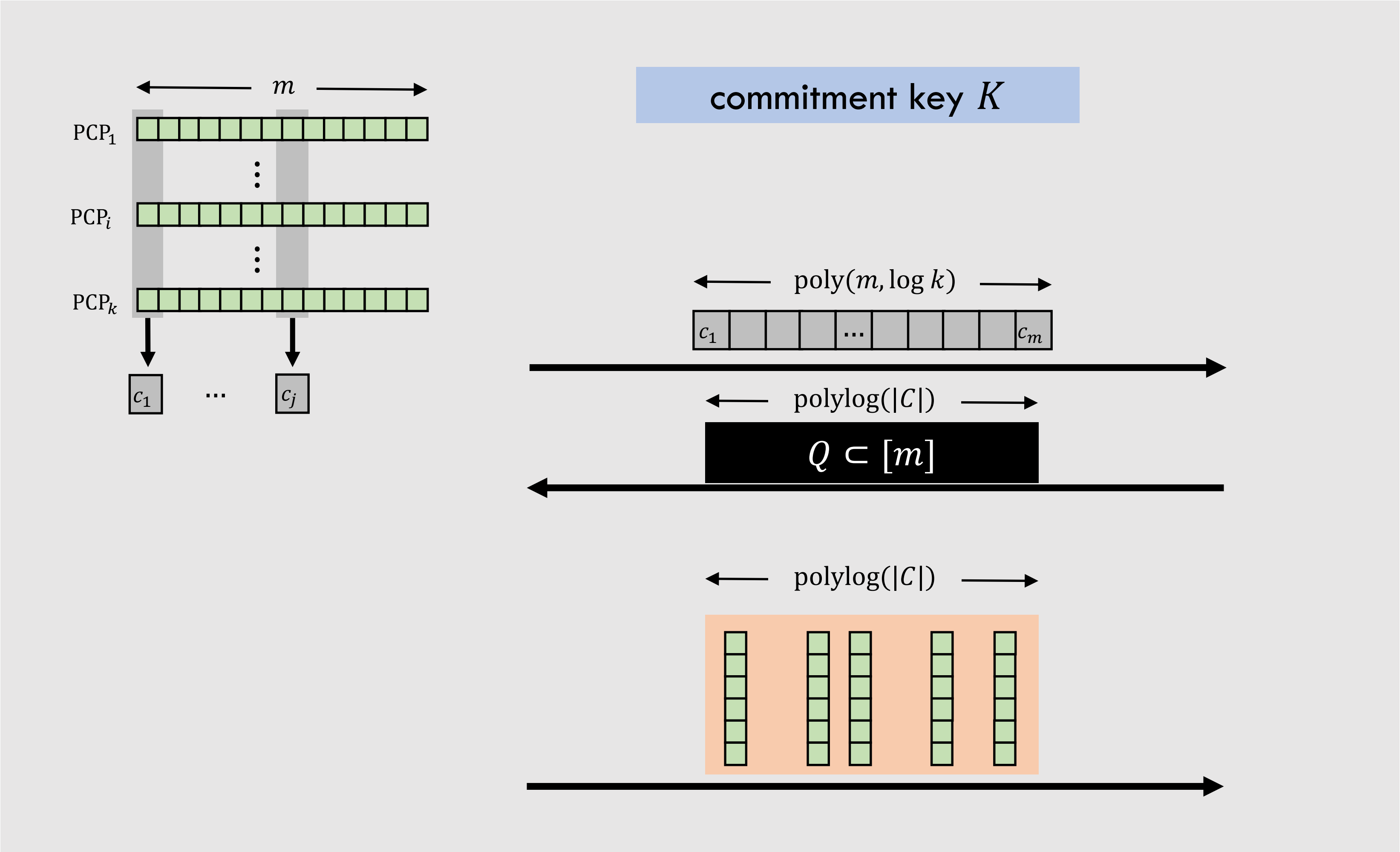


Tool: PCP with Fast Online Verification



$|\text{PCP Online Verify}| \approx O(|x|)$

Dual Mode Argument for Batch Index



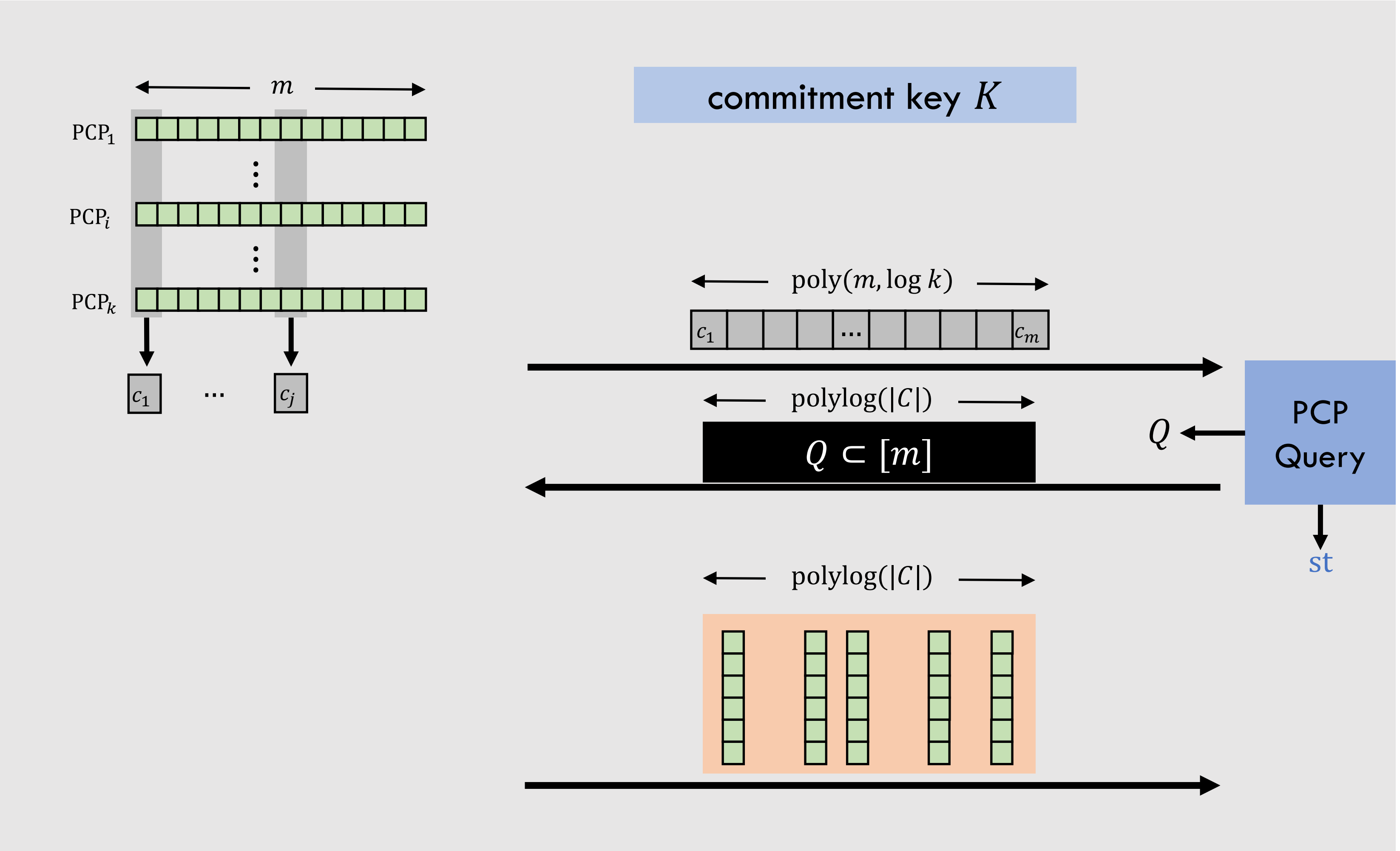
$L_C = \{i \mid \exists w \text{ s.t. } C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

Verify:

1. Commitment openings are valid.
2. PCP responses verify on Q

Dual Mode Argument for Batch Index



$L_C = \{i \mid \exists w \text{ s.t. } C(i,w) = 1\}$

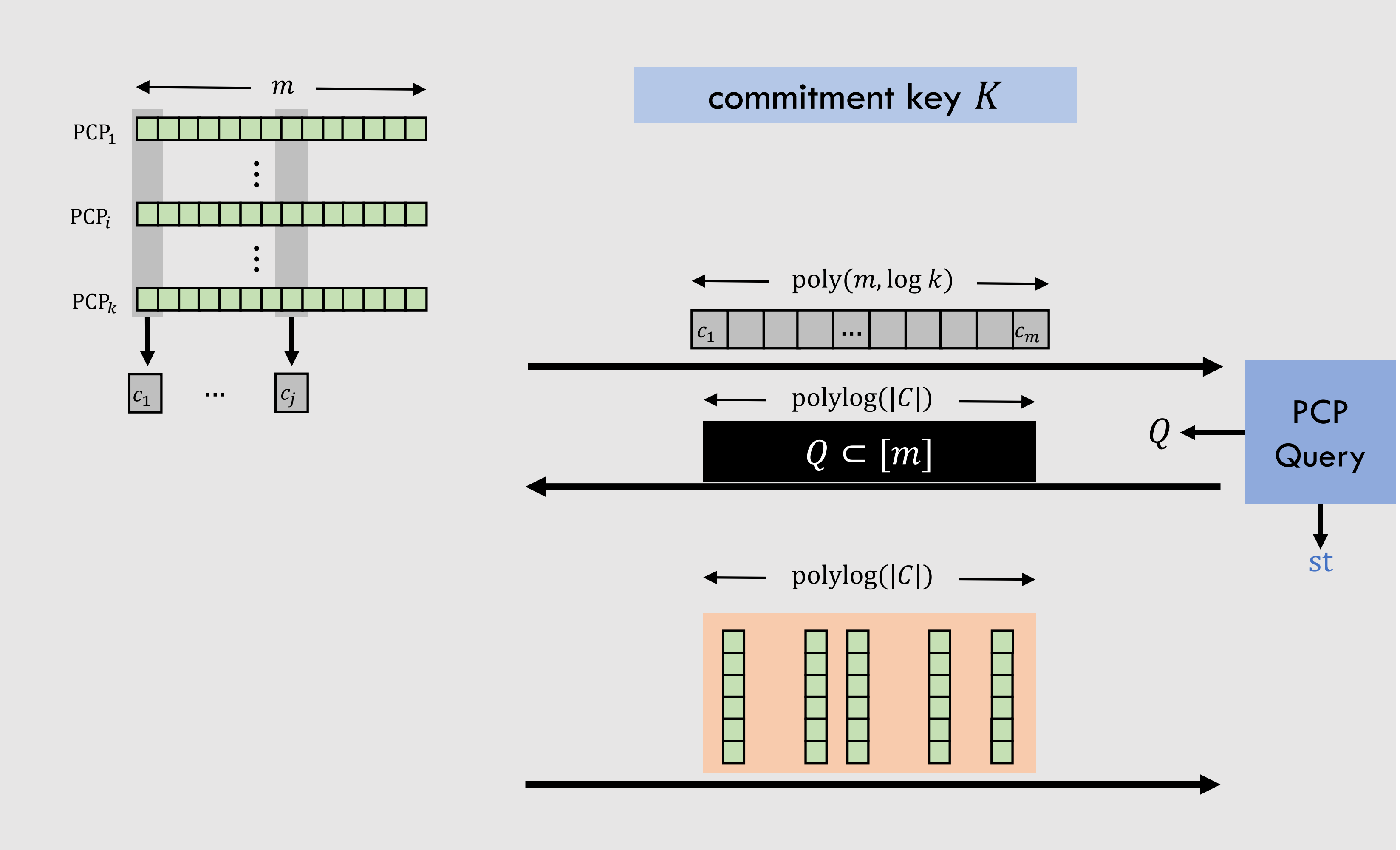
$\forall i \in [k], i \in L_C$

PCP Verify

Verify:

1. Commitment openings are valid.
2. PCP responses verify on Q, st

Dual Mode Argument for Batch Index



$L_C = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

PCP ○Verify

Verify:

1. Commitment openings are valid.
2. PCP responses verify on Q, st

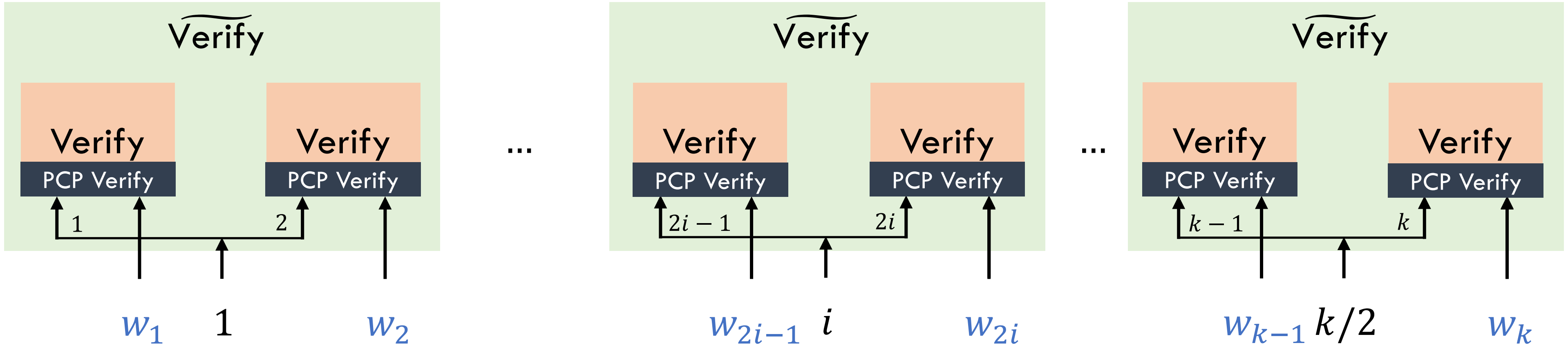
$|\mathrm{Verify}| \approx \mathrm{polylog}(k, |C|)$

Grouping Instances for Recursion

$L_C = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

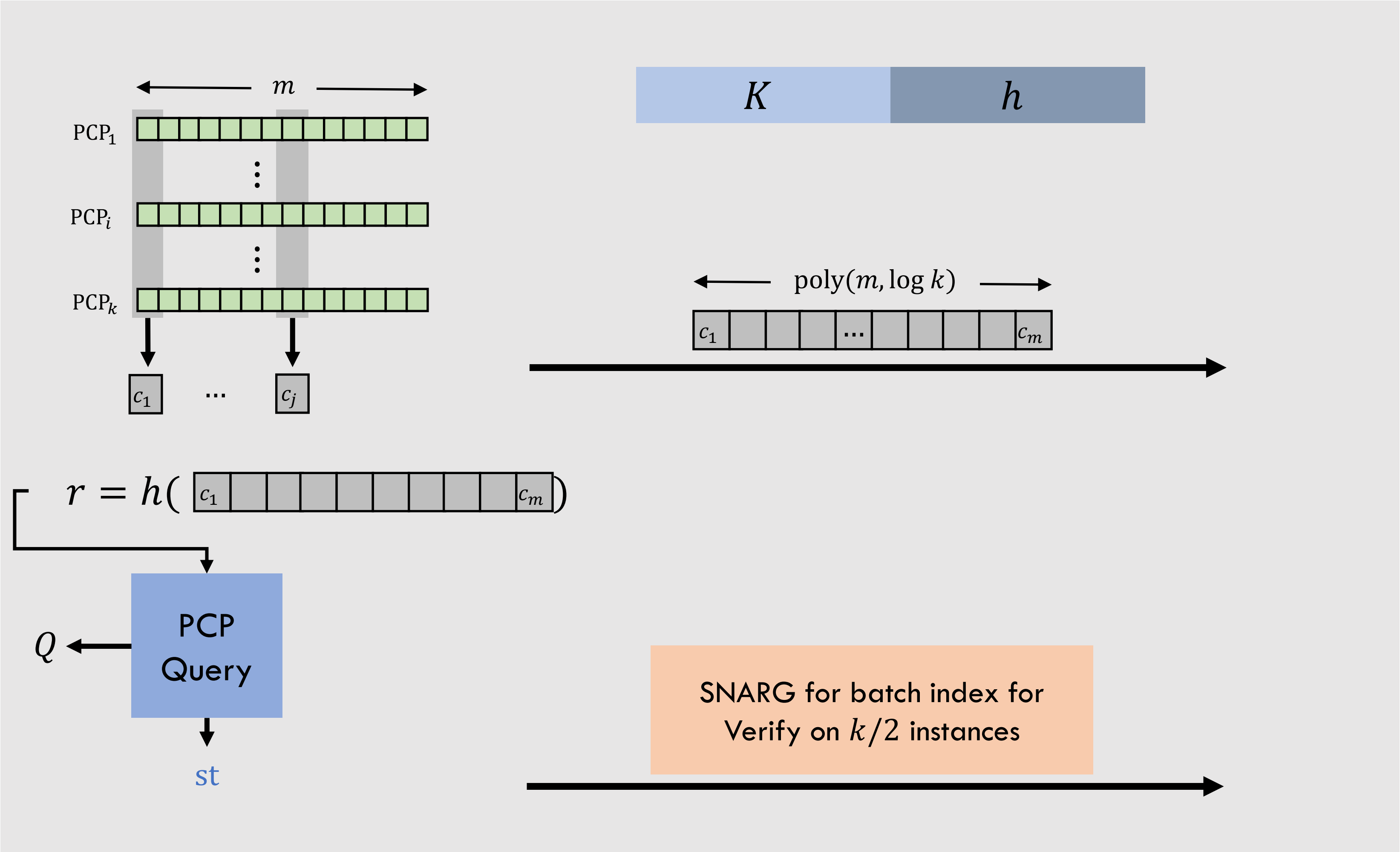
$\forall i \in [k], i \in L_C$

$|\widetilde{\text{Verify}}| \geq 2|\text{Verify}| \geq 2 \, \text{polylog}(k, |C|)$



$k/2$ instances for circuit $\widetilde{\text{Verify}}$

SNARG for Batch Index



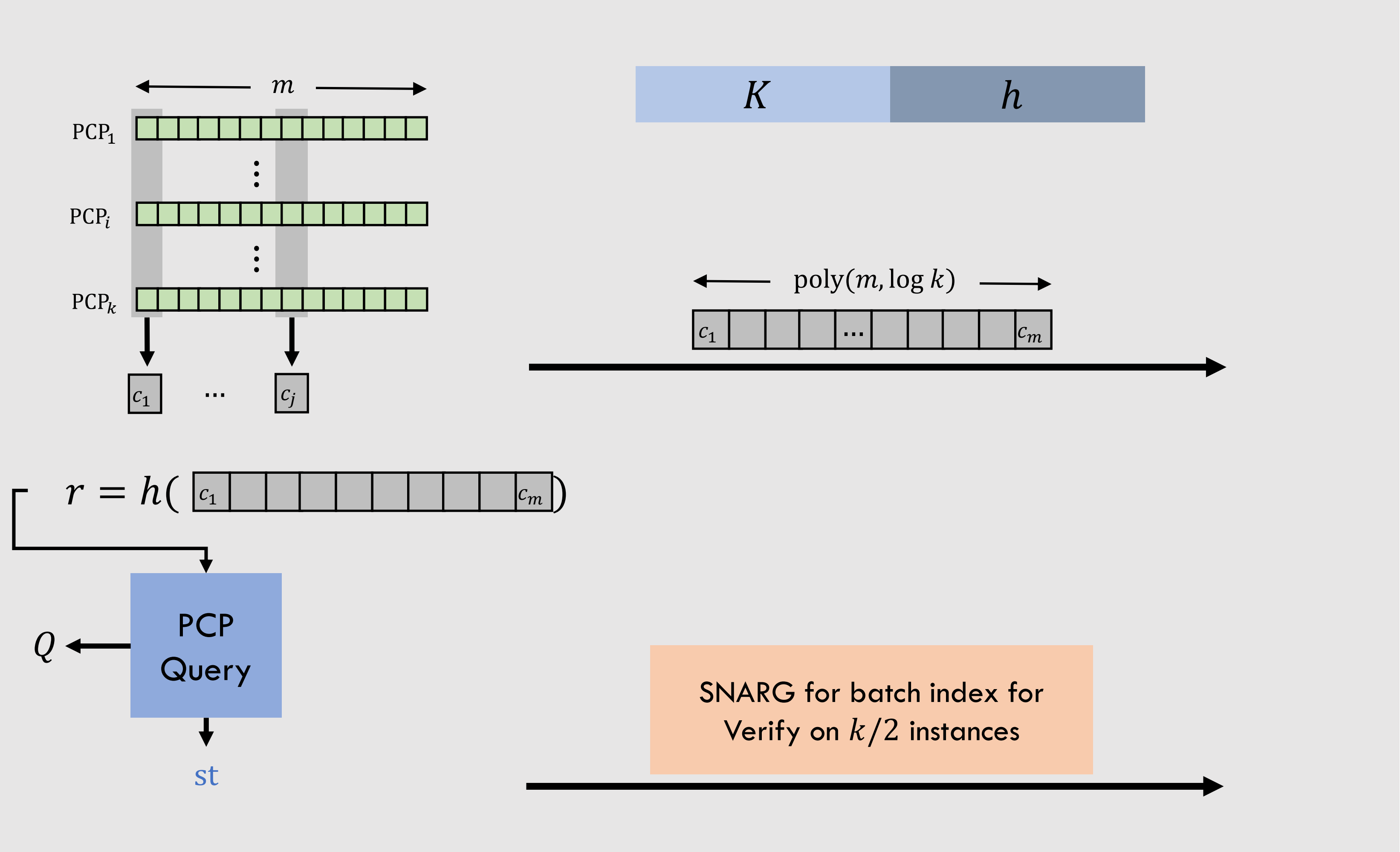
$L_C = \{i \mid \exists w \ s.t. \ C(i,w) = 1\}$

$\forall i \in [k], i \in L_C$

Verify: c_1 … c_m

1. Commitment openings are valid.
2. PCP responses verify on Q, st

SNARG for Batch Index



$L_C = \{i \mid \exists w \ s.t.\ C(i,w) = 1\}$

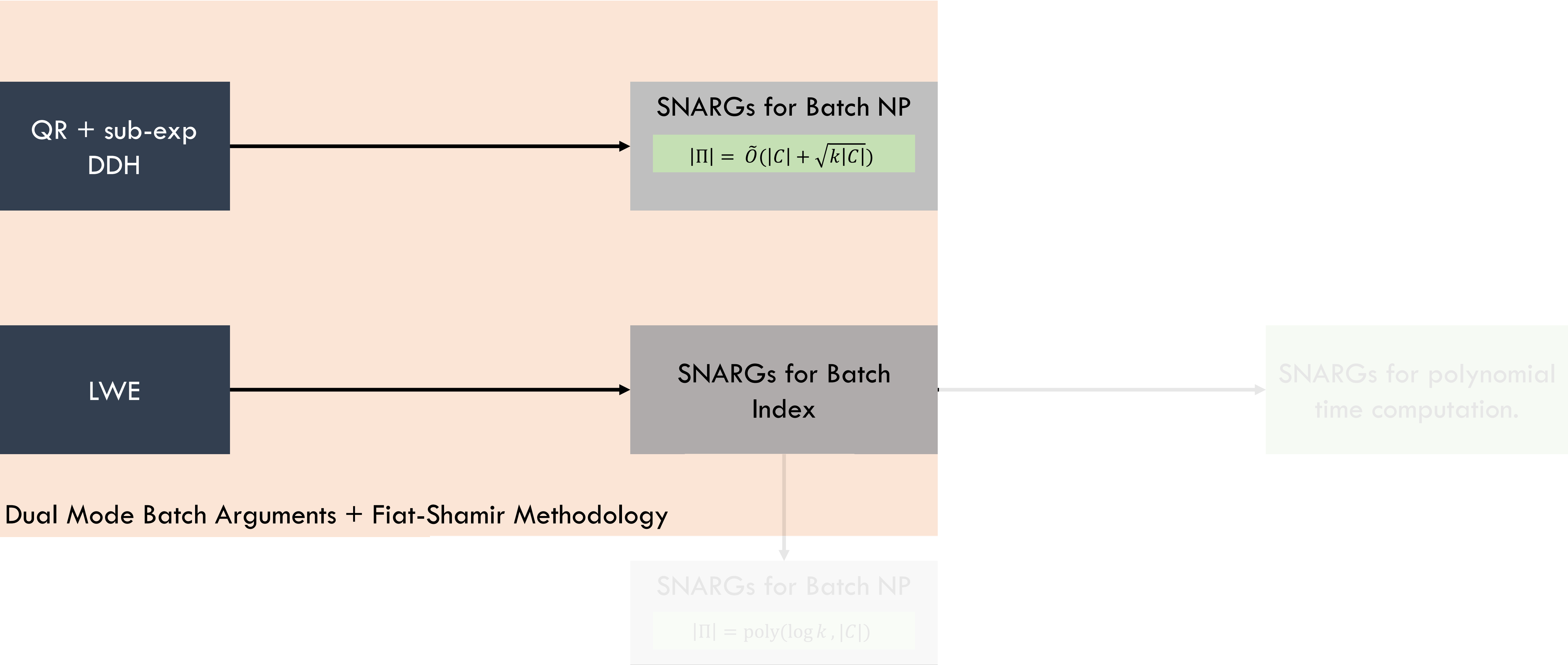
$\forall i \in [k], i \in L_C$

Recurse $\log k$ times

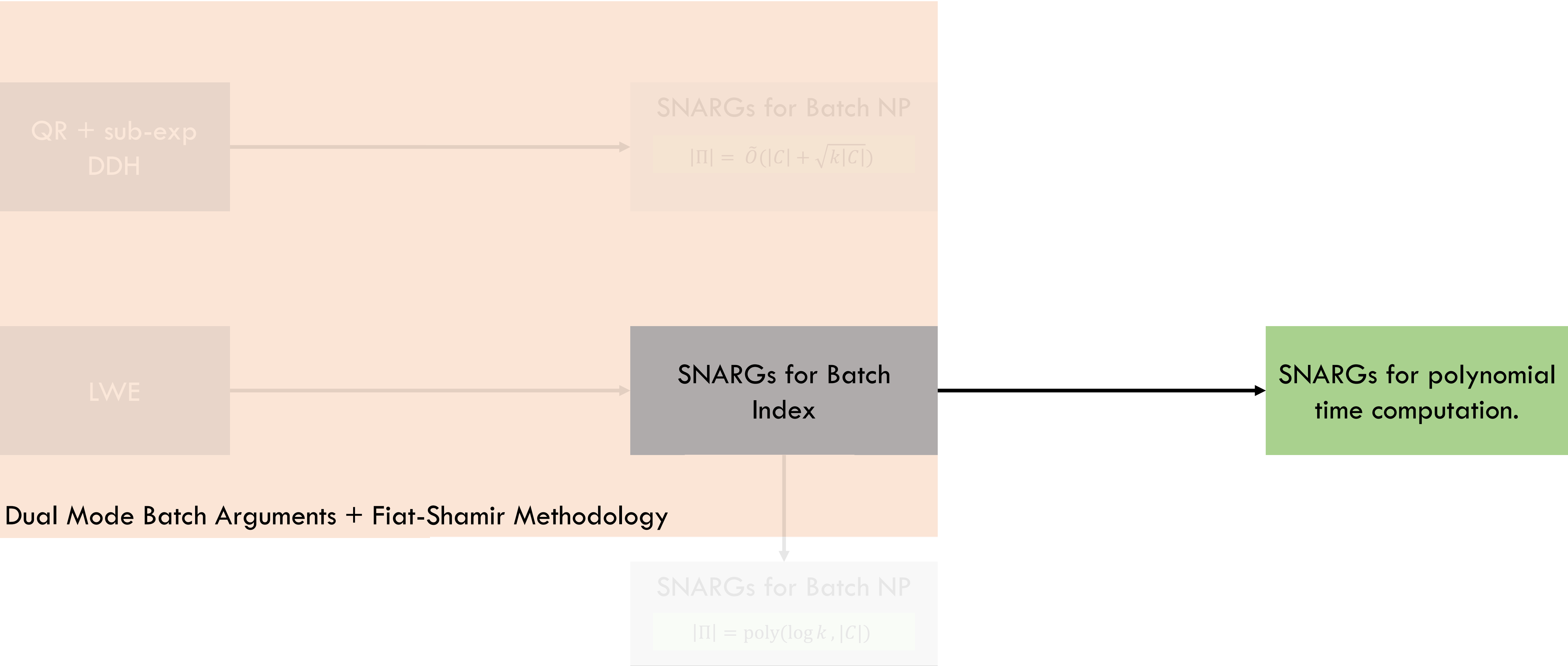
Verify: c_1 c_m

1. Commitment openings are valid.
2. PCP responses verify on Q, st

Results Overview

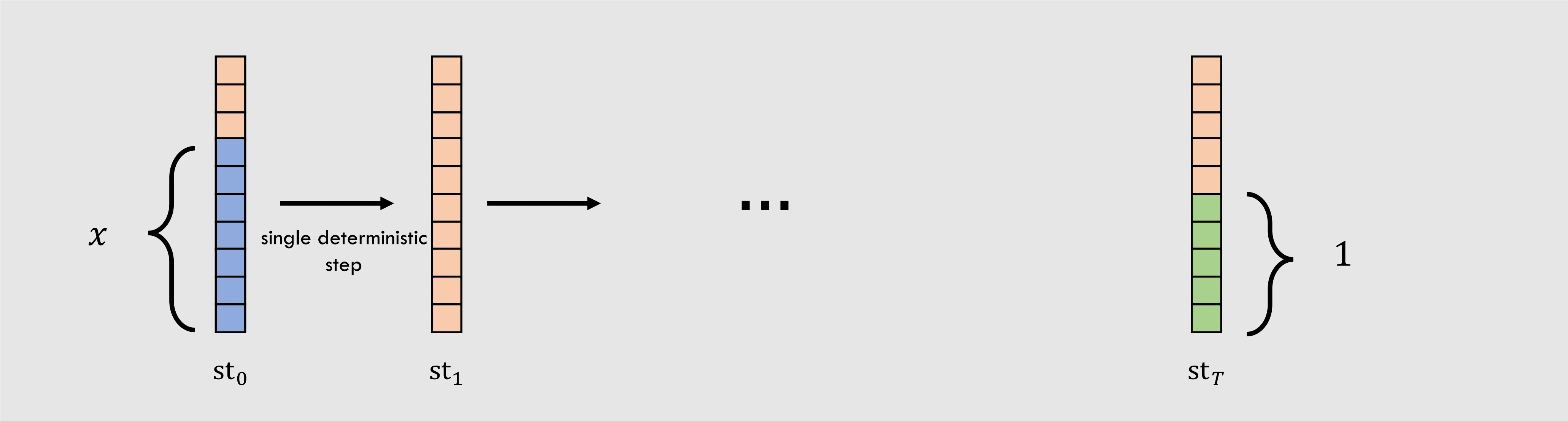


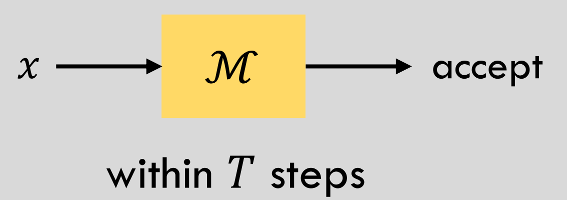
Results Overview



SNARGs for Batch Index → SNARGs for P

Delegation via Batching [Reingold-Rothblum-Rothblum'16]



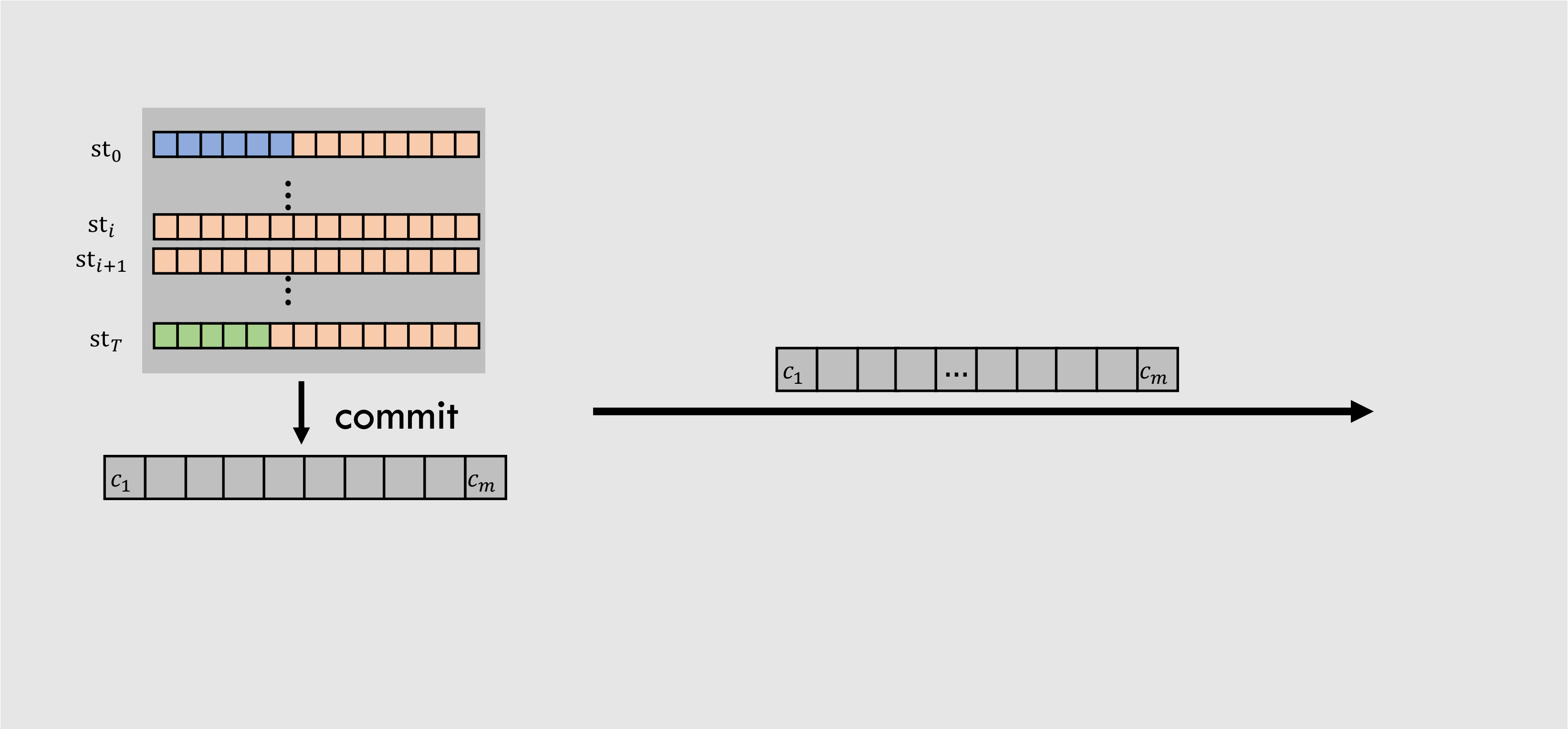


Prove for every $i \in [0, \dots, T-1]$

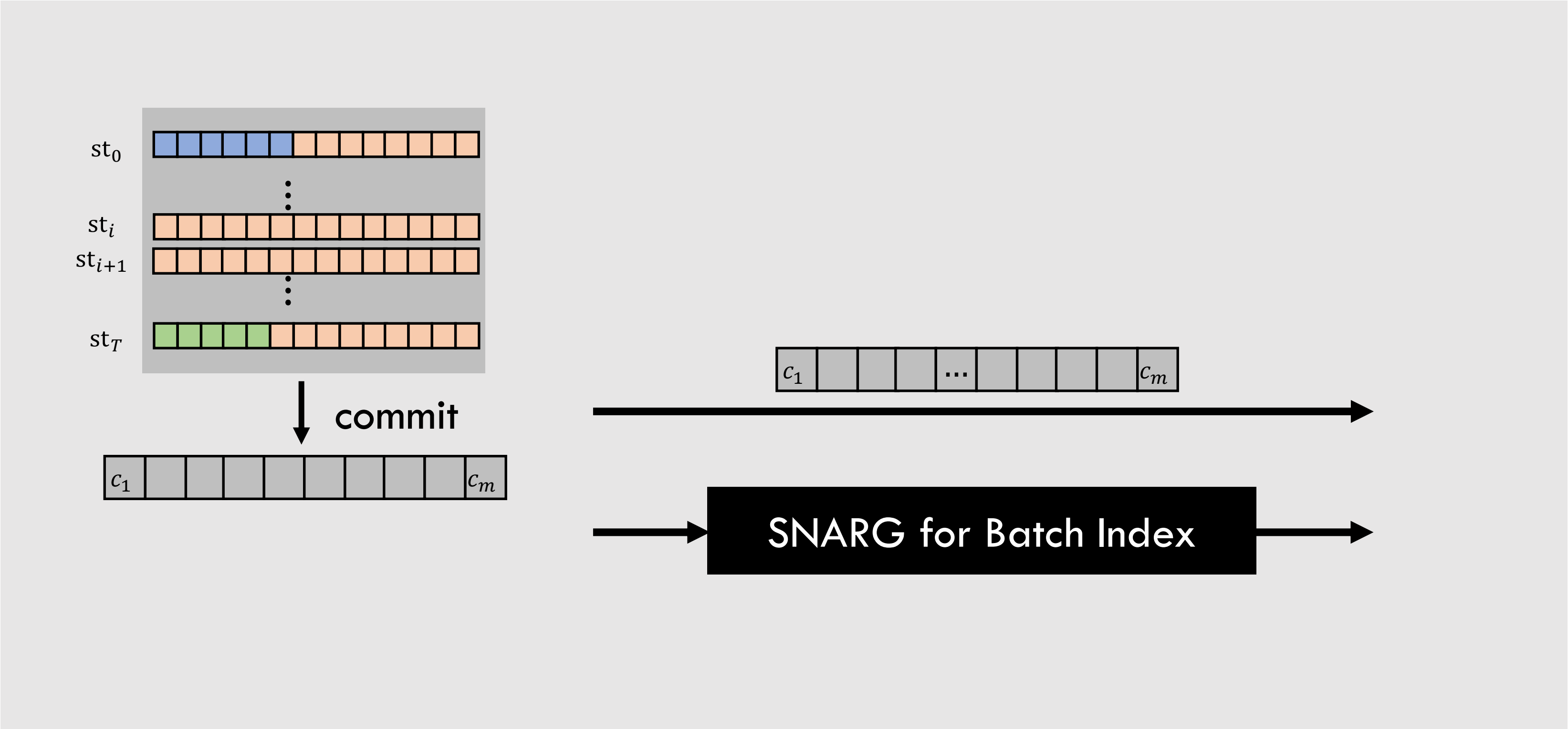
$$\mathsf{st}_i \to \mathsf{st}_{i+1}$$

is the correct transition.

SNARGs for Polynomial-time Computation



SNARGs for Polynomial-time Computation

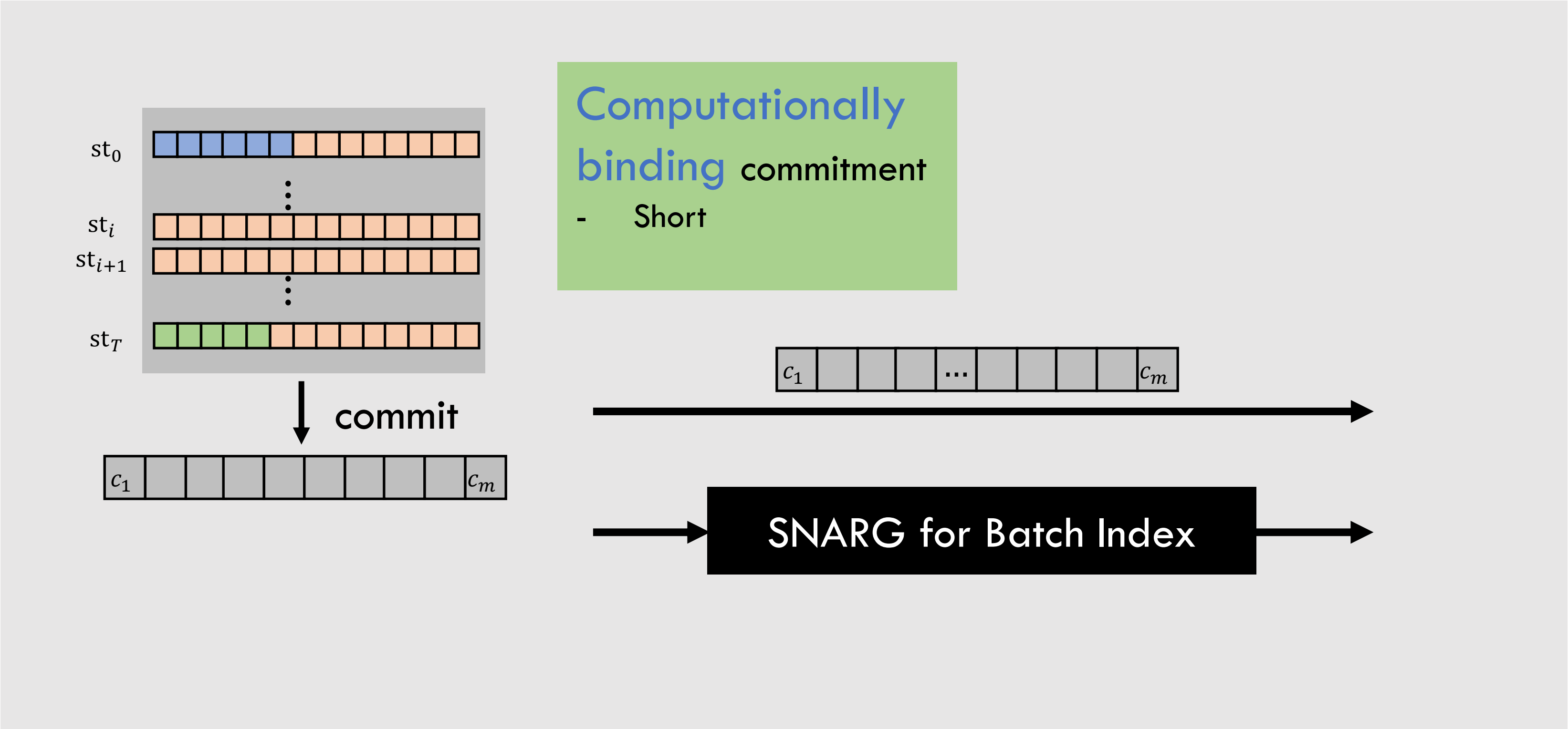


SNARG for Batch Index

For every $i \in [0, \ldots, T-1]$

1. Commitment contains st_i and st_{i+1}
2. Valid transition $\mathrm{st}_i \to \mathrm{st}_{i+1}$

SNARGs for Polynomial-time Computation

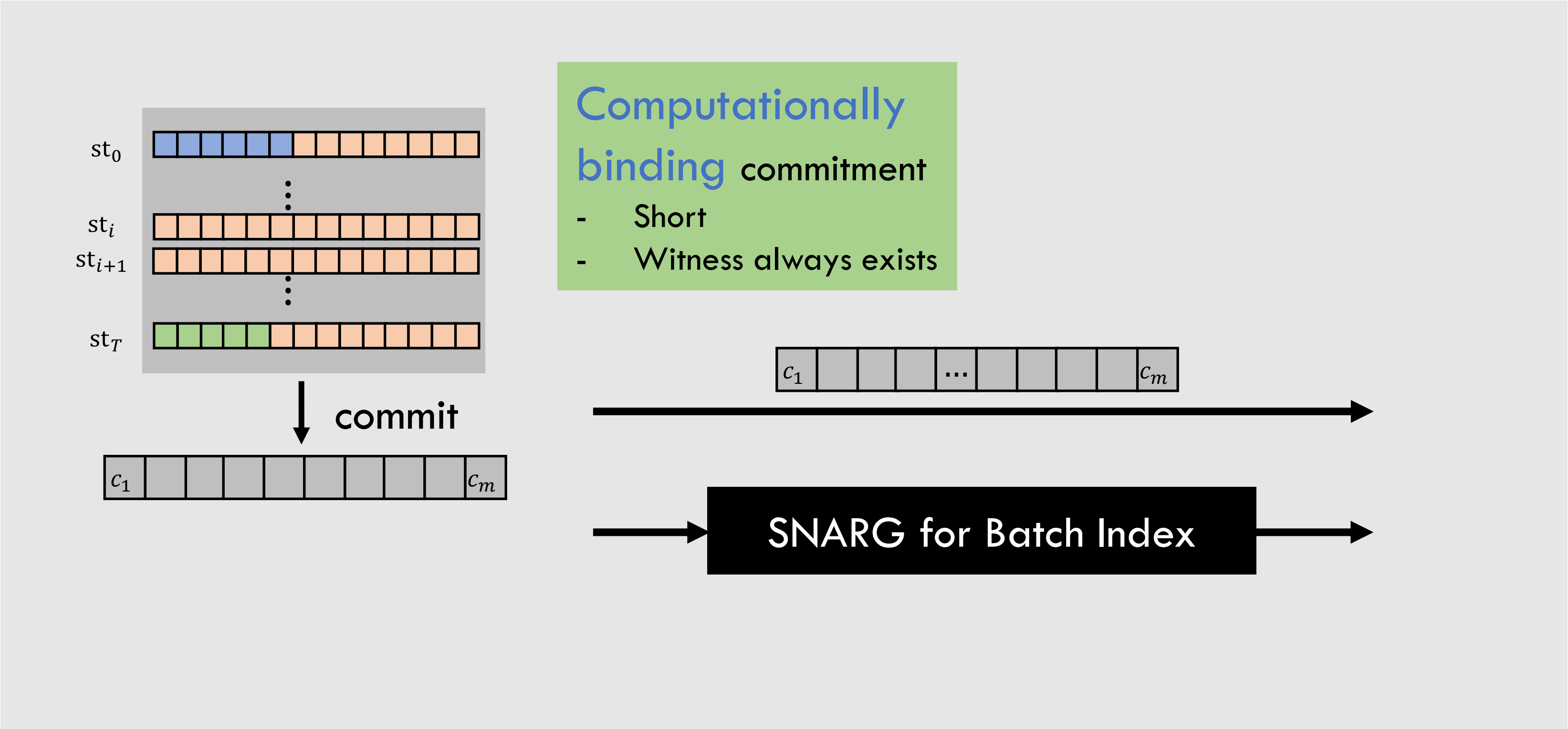


SNARG for Batch Index

For every $i \in [0, \dots, T-1]$

1. Commitment contains st_i and st_{i+1}
2. Valid transition $\mathsf{st}_i \rightarrow \mathsf{st}_{i+1}$

SNARGs for Polynomial-time Computation

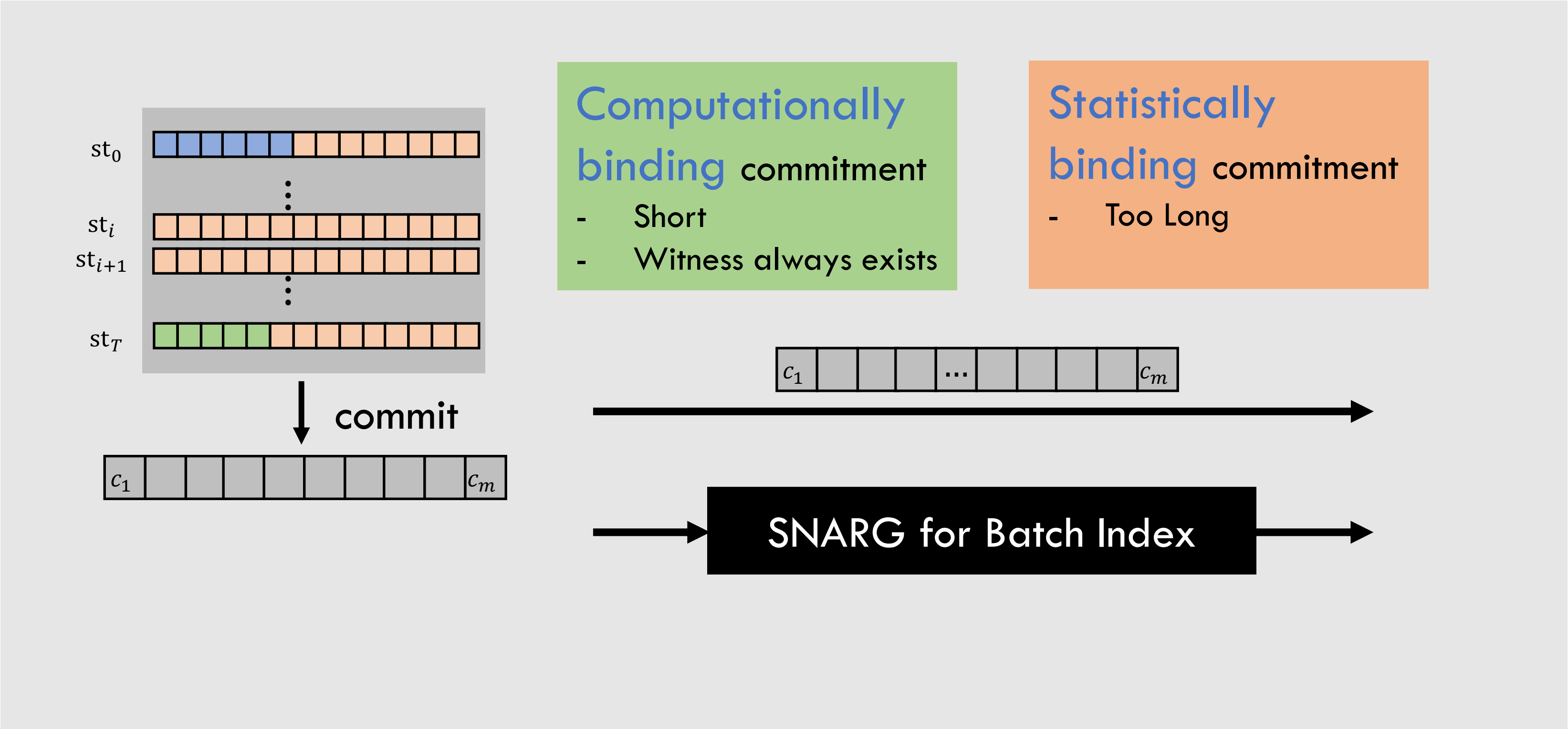


SNARG for Batch Index

For every $i \in [0, \dots, T-1]$

1. Commitment opening to st_i and st_{i+1}
2. Valid transition $\mathsf{st}_i \to \mathsf{st}_{i+1}$

SNARGs for Polynomial-time Computation

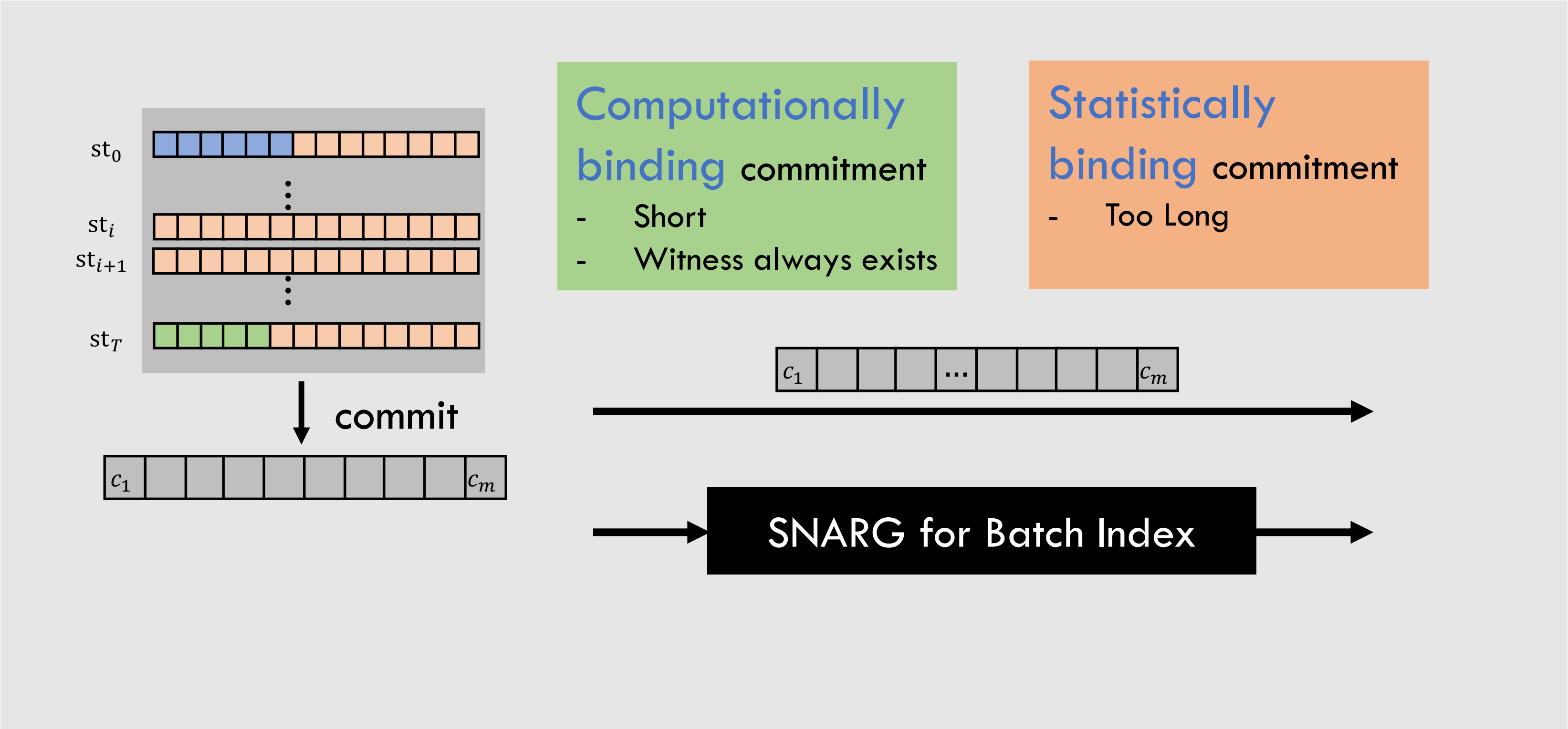


SNARG for Batch Index

For every $i \in [0, \ldots, T-1]$

1. Commitment opening to st_i and st_{i+1}
2. Valid transition $\mathsf{st}_i \to \mathsf{st}_{i+1}$

SNARGs for Polynomial-time Computation



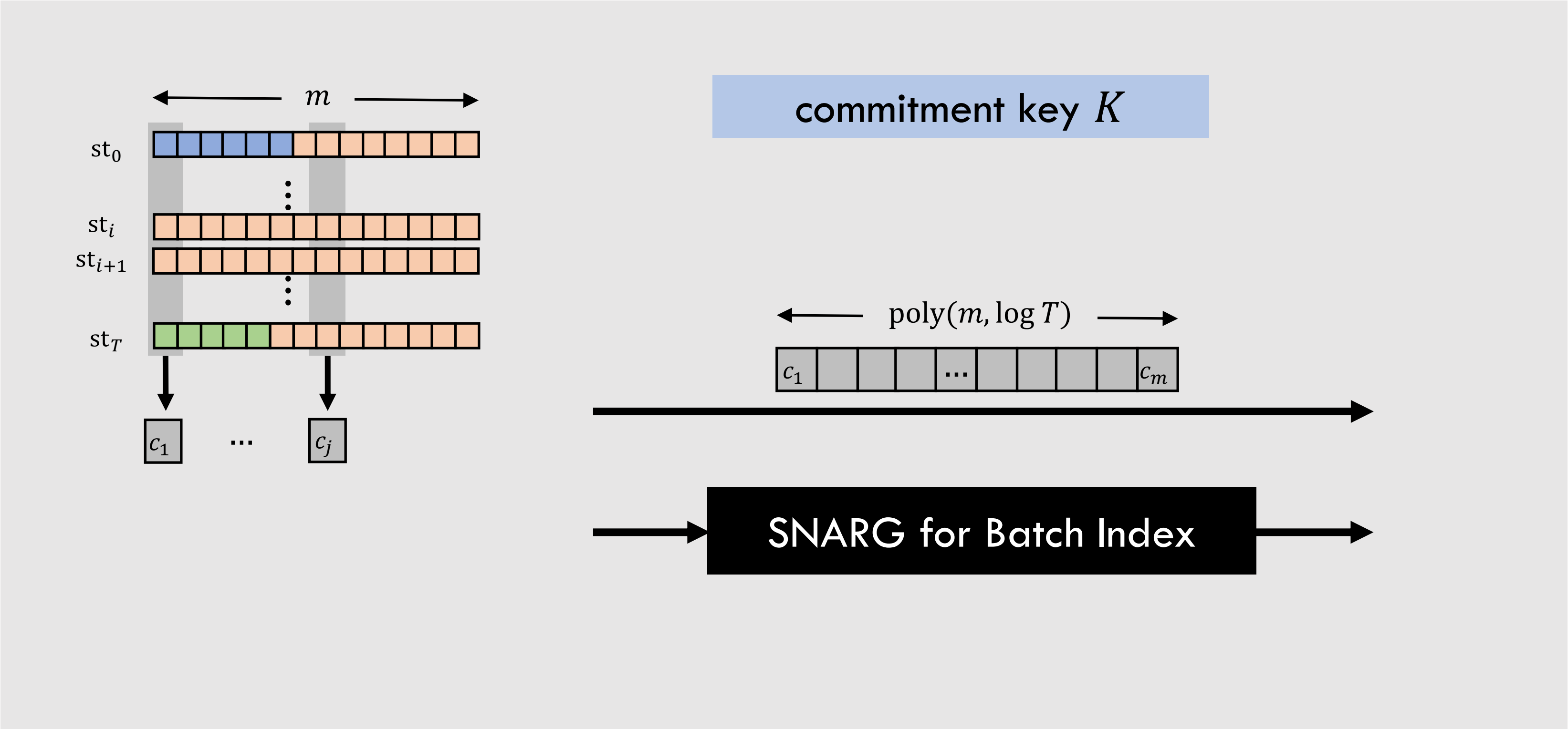
Use SSB Commitments

SNARG for Batch Index

For every $i \in [0, \dots, T-1]$

1. Commitment opening to st_i and st_{i+1}
2. Valid transition $\mathsf{st}_i \rightarrow \mathsf{st}_{i+1}$

SNARGs for Polynomial-time Computation

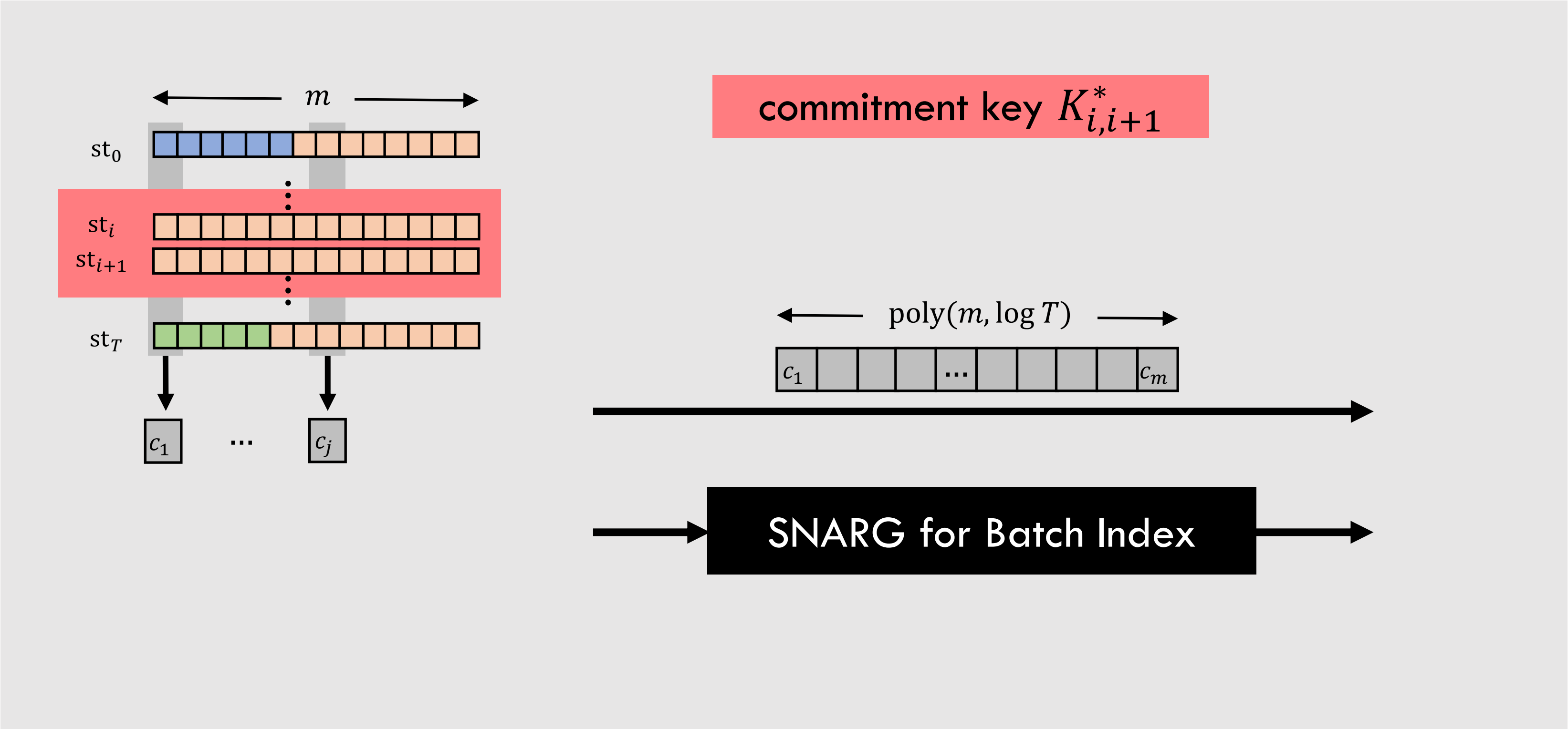


SNARG for Batch Index

For every $i \in [0, \dots, T-1]$

1. Commitment opening to st_i and st_{i+1}
2. Valid transition $\mathsf{st}_i \to \mathsf{st}_{i+1}$

SNARGs for Polynomial-time Computation

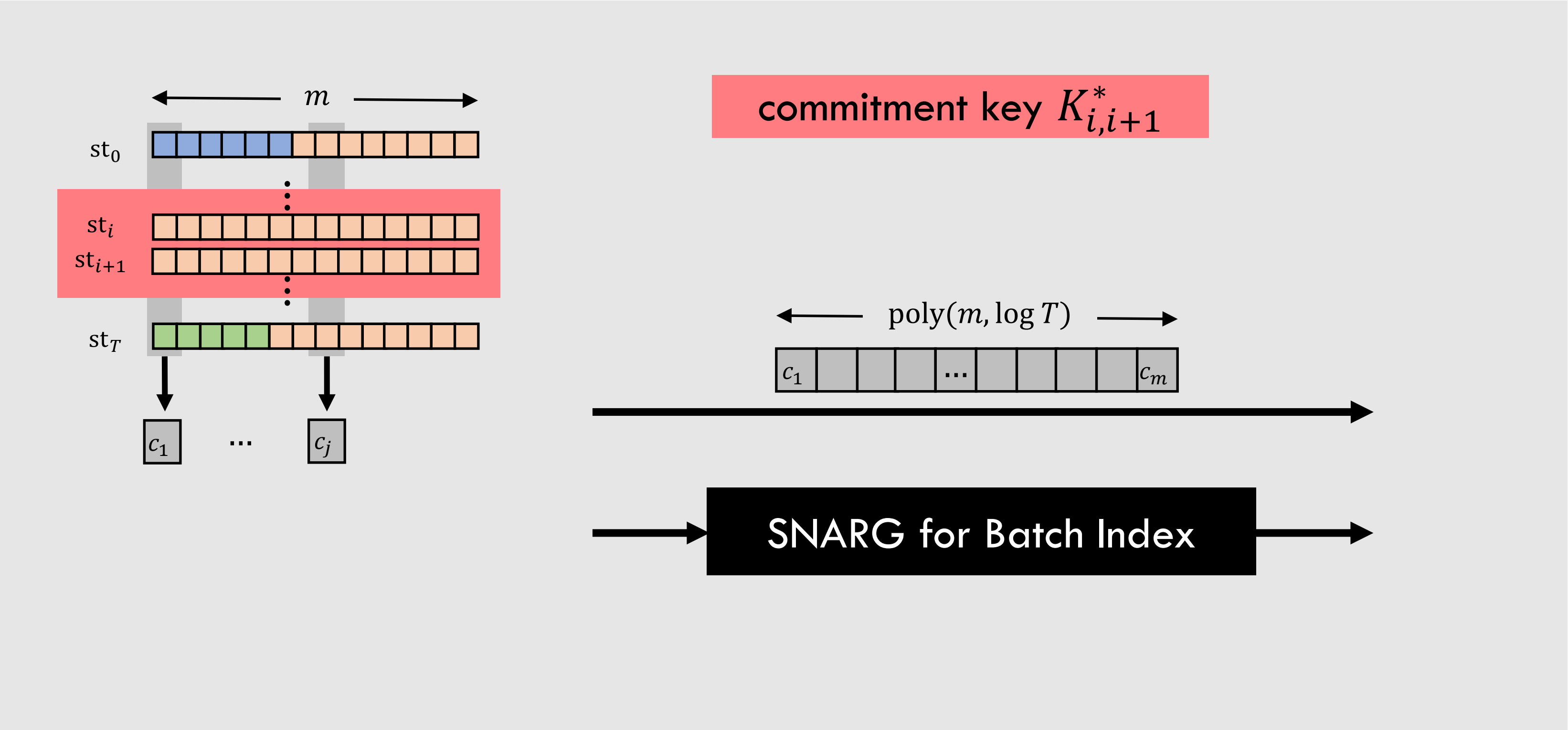


SNARG for Batch Index

For every $i \in [0, \dots, T-1]$

1. Commitment opening to st_i and st_{i+1}
2. Valid transition $\mathsf{st}_i \to \mathsf{st}_{i+1}$

SNARGs for Polynomial-time Computation



Local Soundness

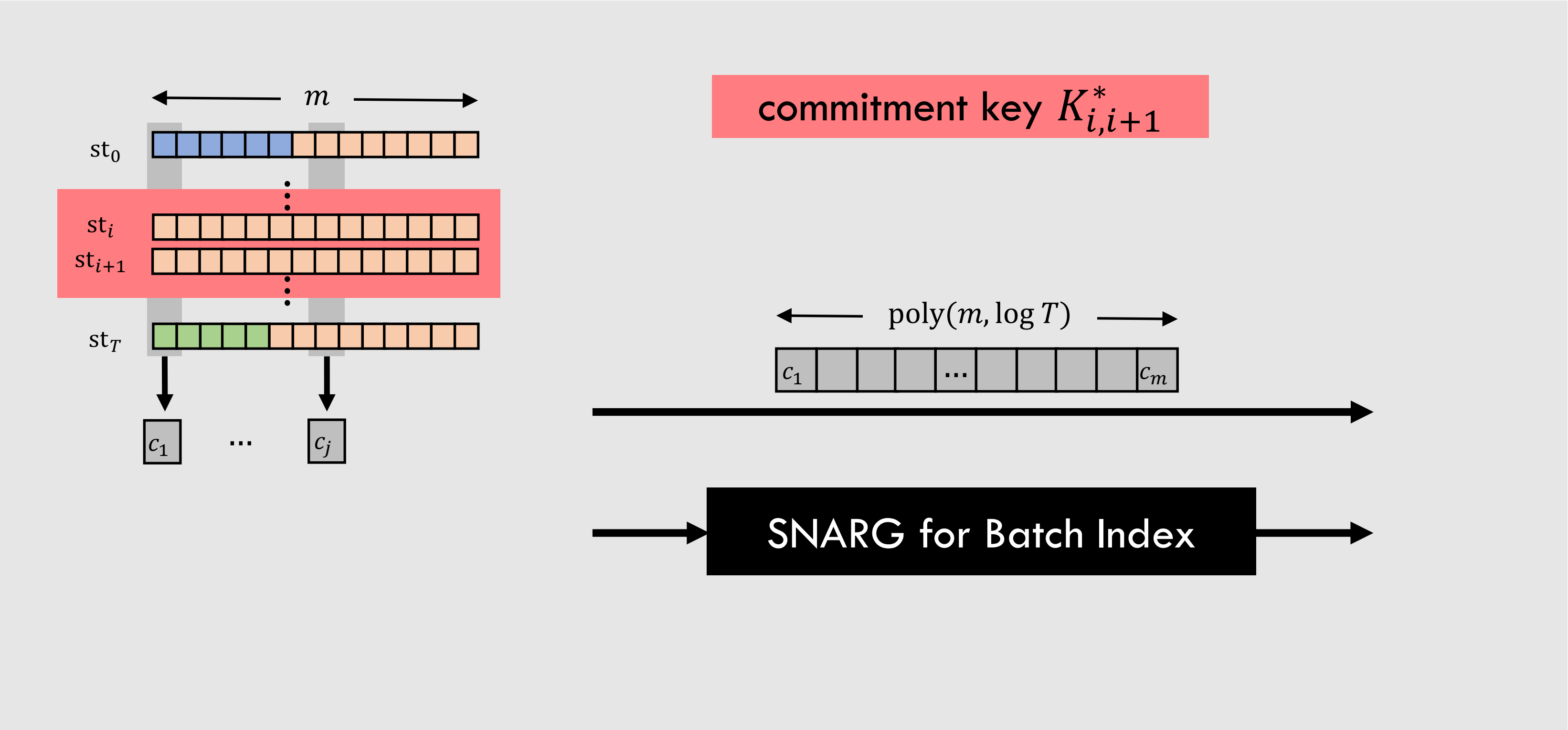
i-th state transition correct

SNARG for Batch Index

For every $i \in [0, \dots, T-1]$

1. Commitment opening to st_i and st_{i+1}
2. Valid transition $\mathrm{st}_i \to \mathrm{st}_{i+1}$

SNARGs for Polynomial-time Computation



Local Soundness

i-th state transition correct

→

Global Soundness

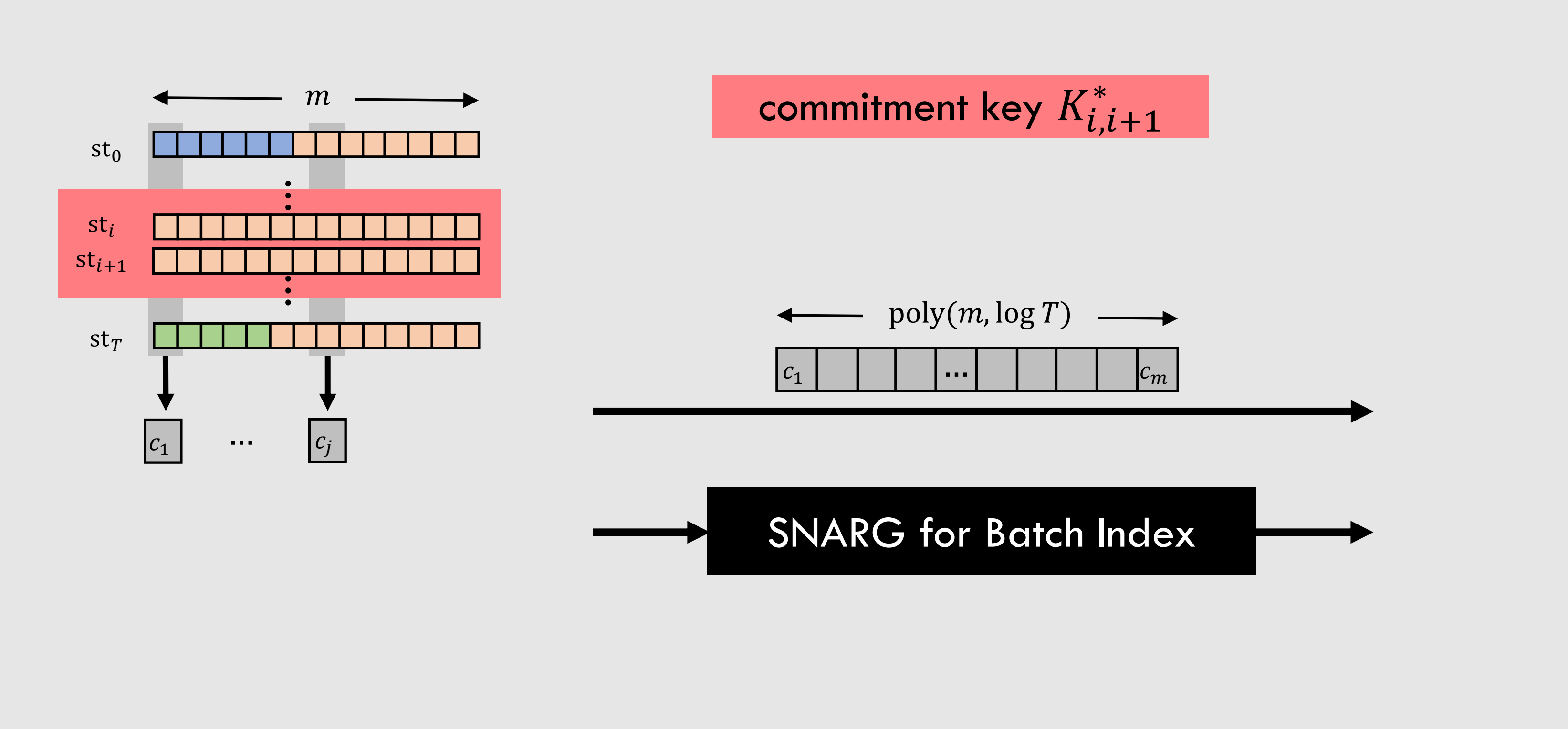
Local soundness at all i

SNARG for Batch Index

For every $i \in [0, \dots, T-1]$

1. Commitment opening to st_i and st_{i+1}
2. Valid transition $\mathsf{st}_i \to \mathsf{st}_{i+1}$

SNARGs for Polynomial-time Computation



No-Signaling SSB Commitment Scheme [González-Zacharakis'21]

Local Soundness

i-th state transition correct

Global Soundness

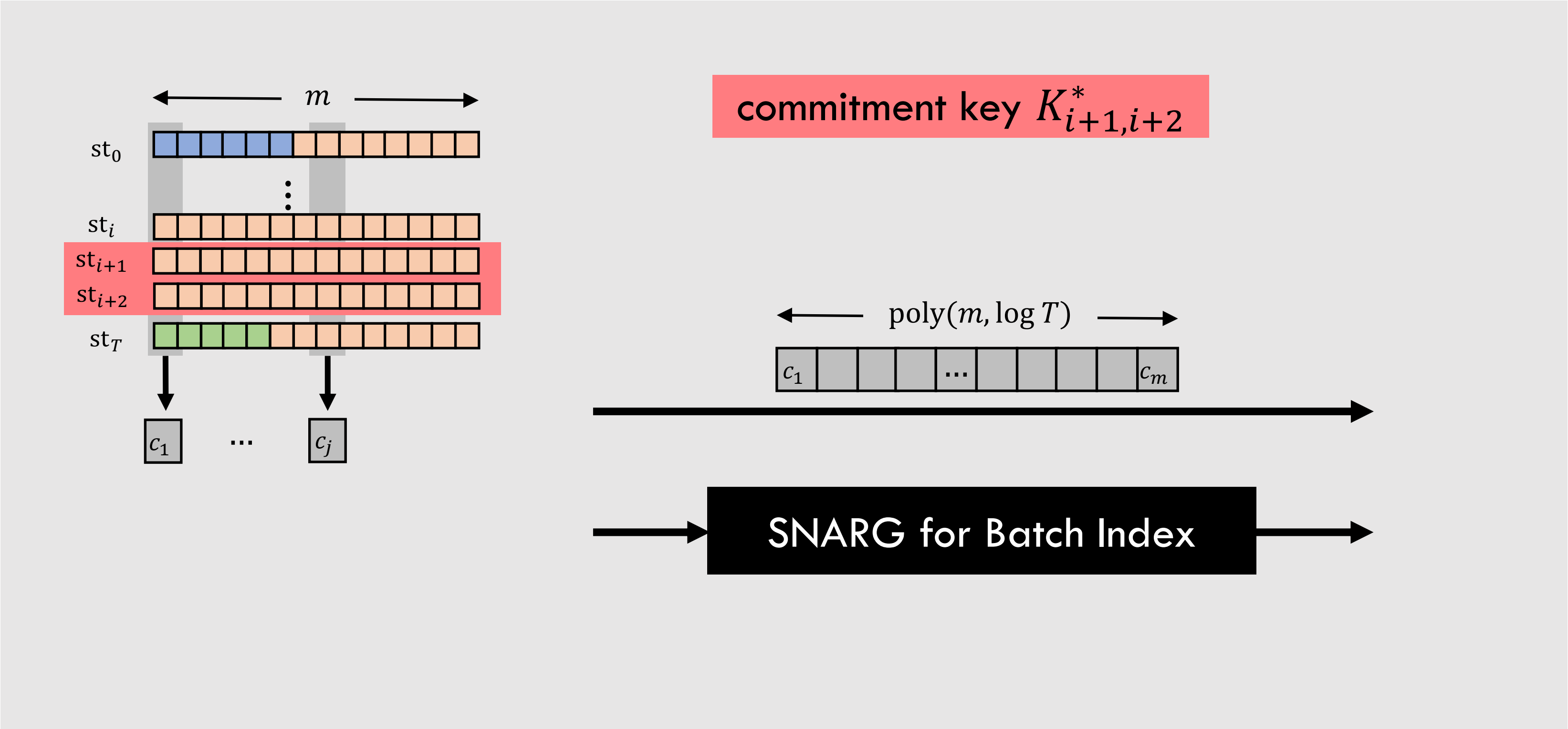
Local soundness at all i

SNARG for Batch Index

For every $i \in [0, \dots, T-1]$

1. Commitment opening to st_i and st_{i+1}
2. Valid transition $\mathsf{st}_i \to \mathsf{st}_{i+1}$

SNARGs for Polynomial-time Computation



No-Signaling SSB Commitment Scheme [González-Zacharakis'21]

Local Soundness

i-th state transition correct

Global Soundness

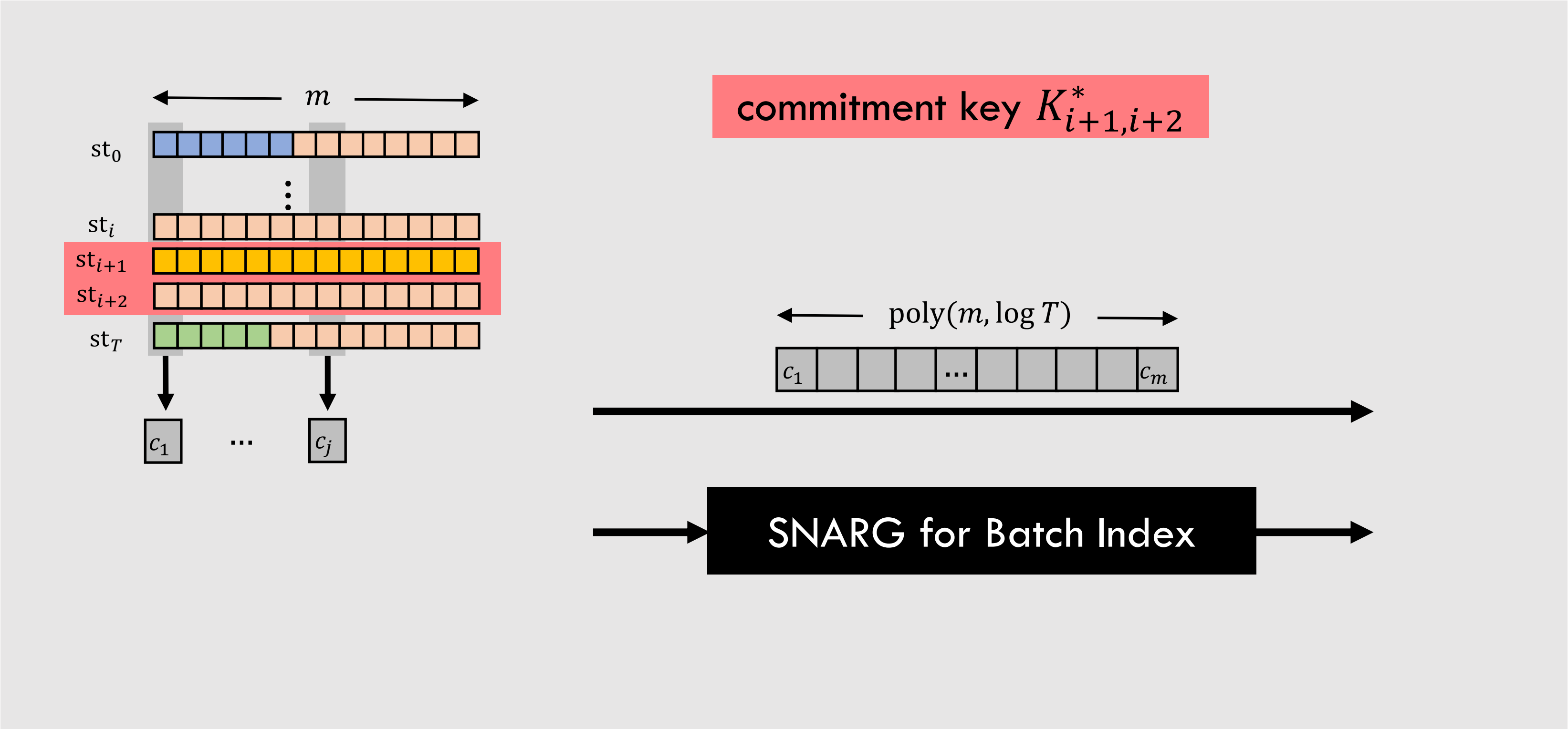
Local soundness at all i

SNARG for Batch Index

For every $i \in [0, \ldots, T-1]$

1. Commitment opening to st_i and st_{i+1}
2. Valid transition $\mathsf{st}_i \rightarrow \mathsf{st}_{i+1}$

SNARGs for Polynomial-time Computation



No-Signaling SSB Commitment Scheme [González-Zacharakis'21]

Local Soundness

i-th state transition correct

→

Global Soundness

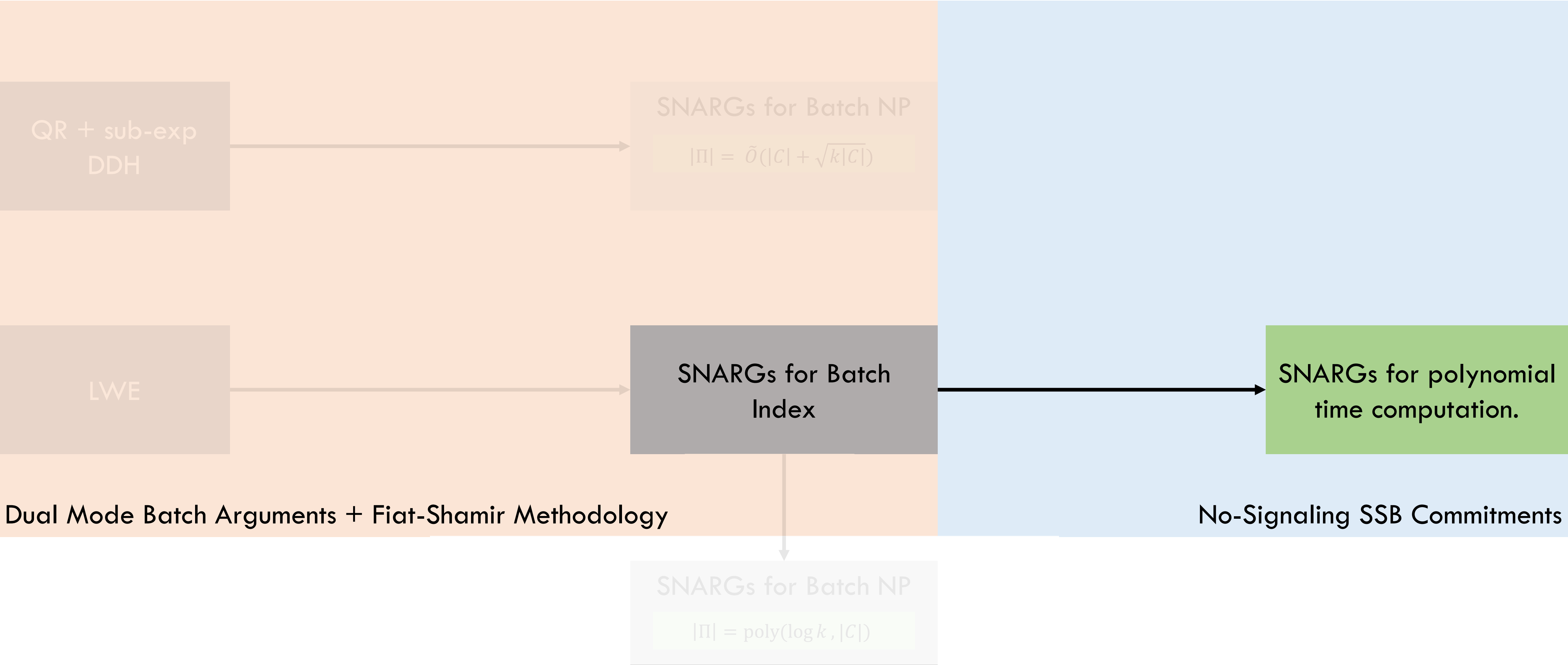
Local soundness at all i

SNARG for Batch Index

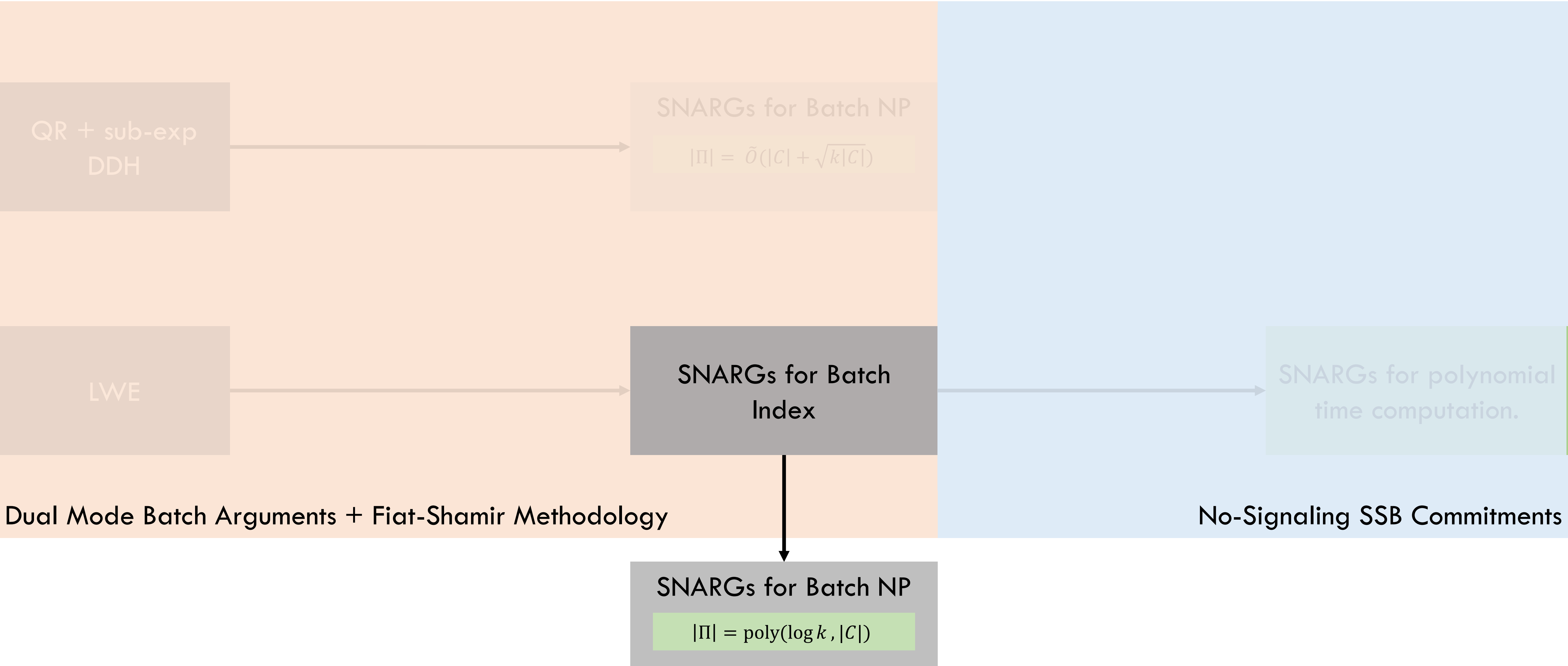
For every $i \in [0, \ldots, T-1]$

1. Commitment opening to st_i and st_{i+1}
2. Valid transition $\mathsf{st}_i \to \mathsf{st}_{i+1}$

Results Overview

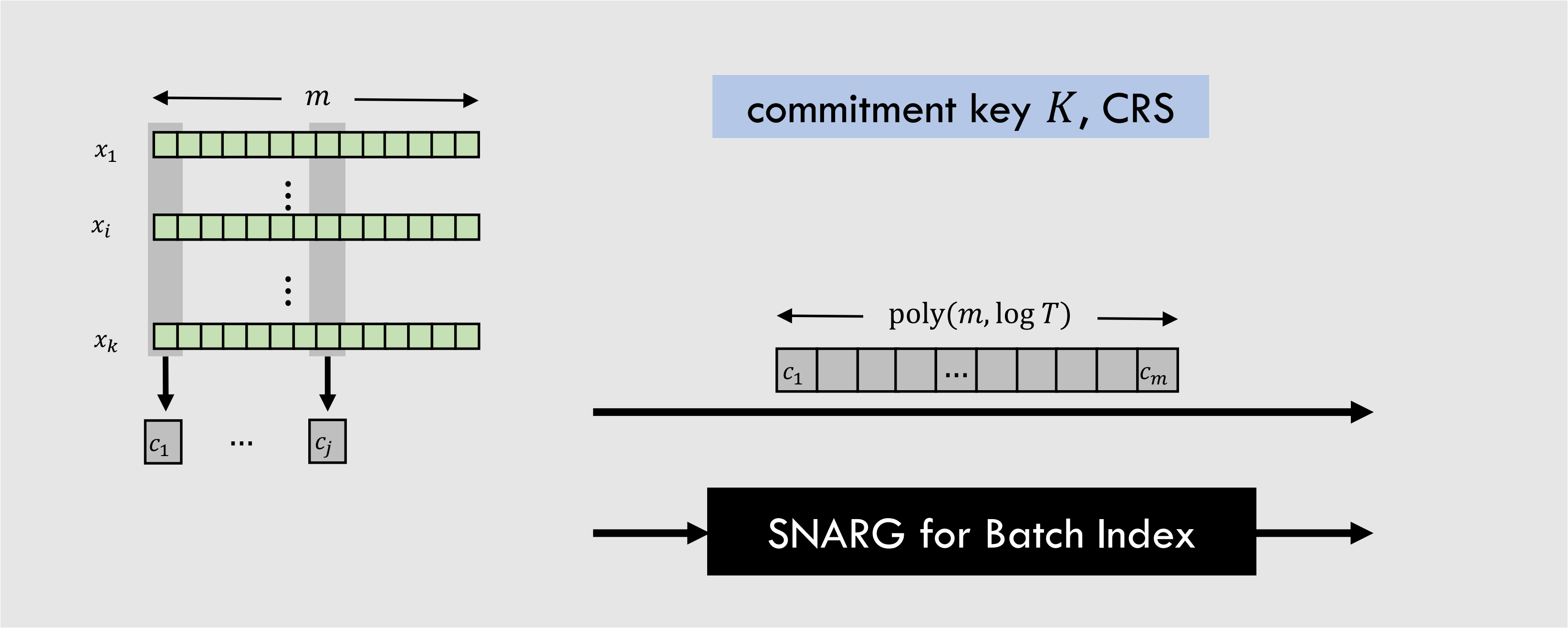


Results Overview



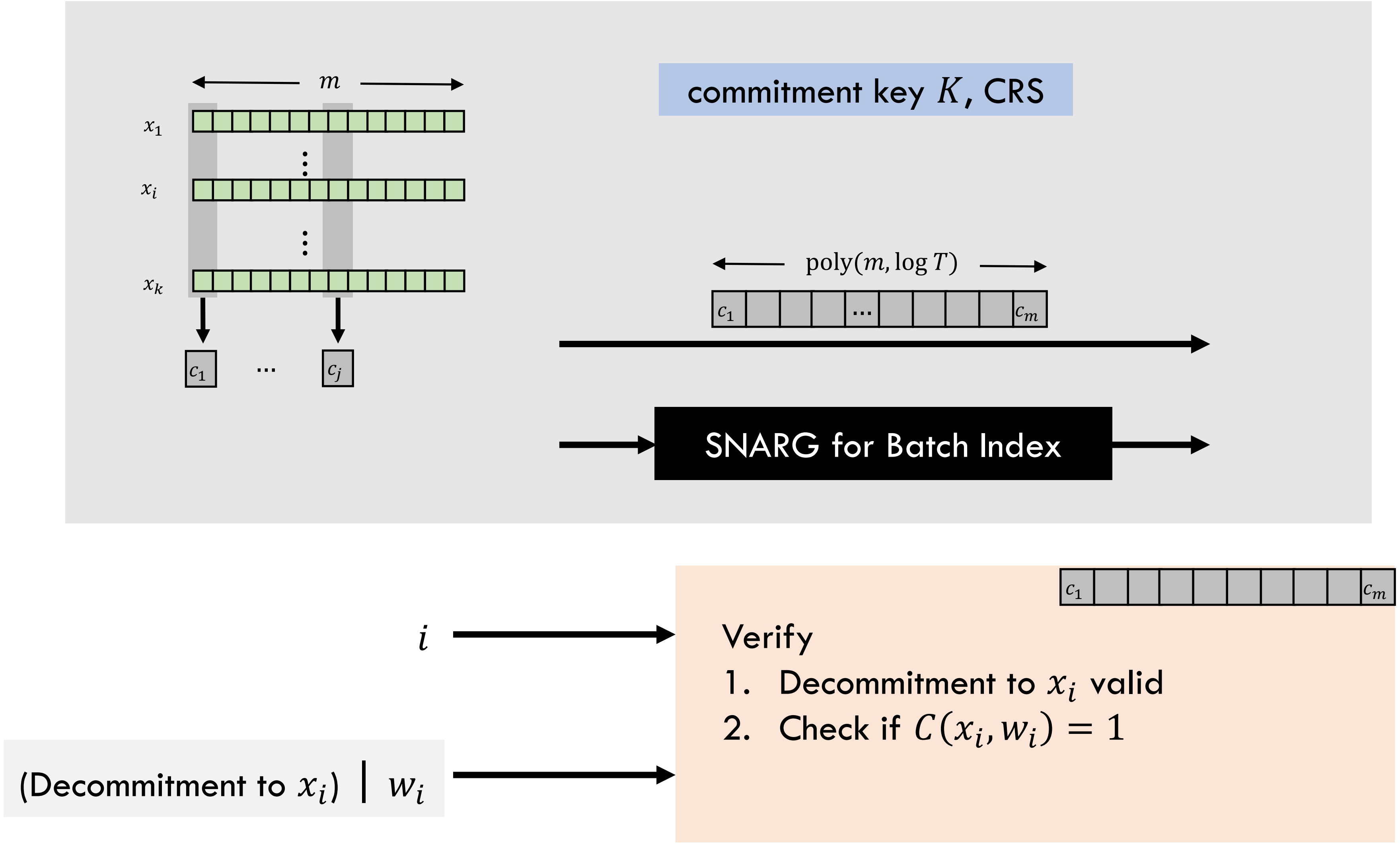
SNARGs for Batch Index $\rightarrow$ SNARGs for Batch NP

SNARGs for Batch NP



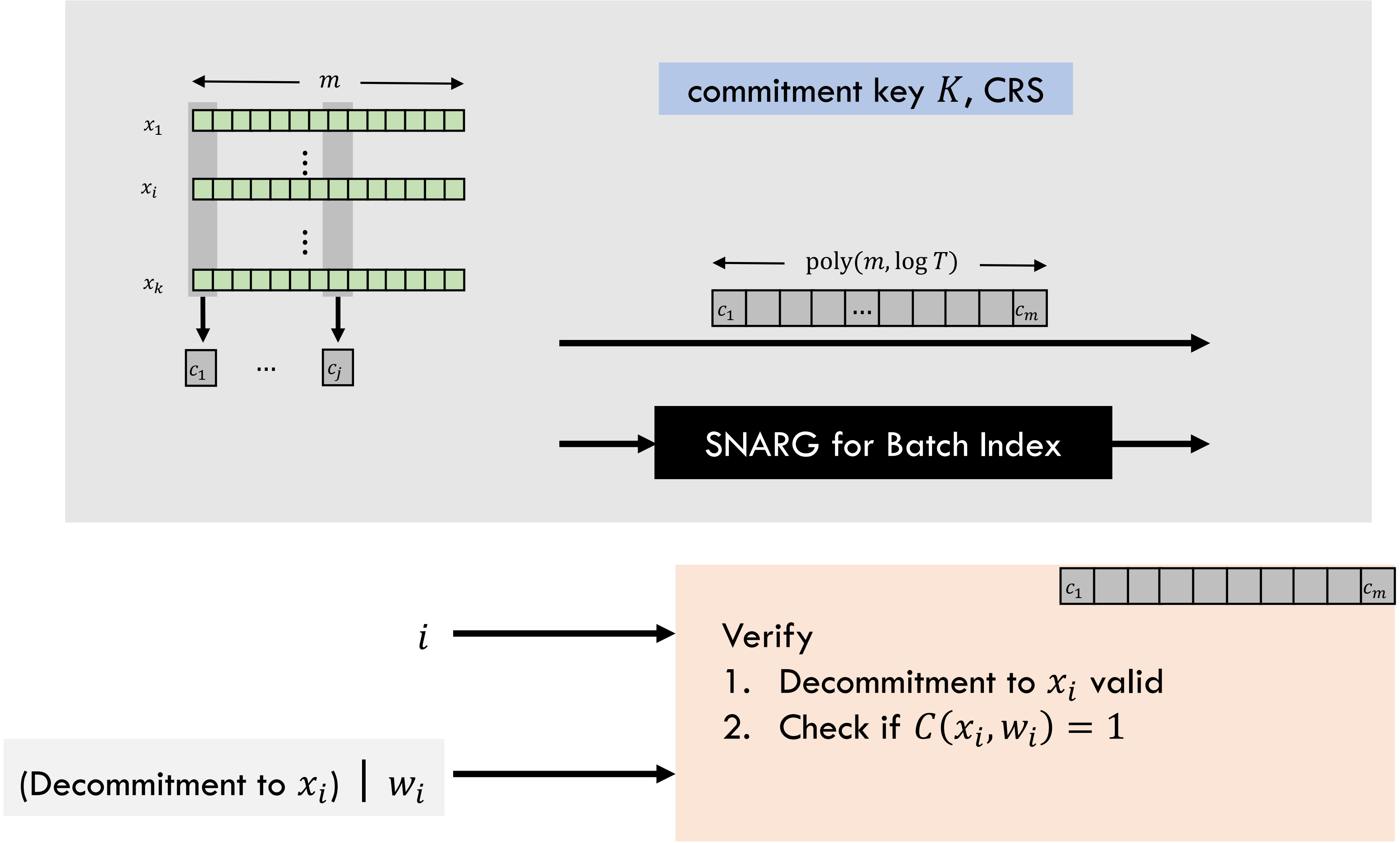
Use fixed randomness for the commitment (say 0)

SNARGs for Batch NP



Use fixed randomness for the commitment (say 0)

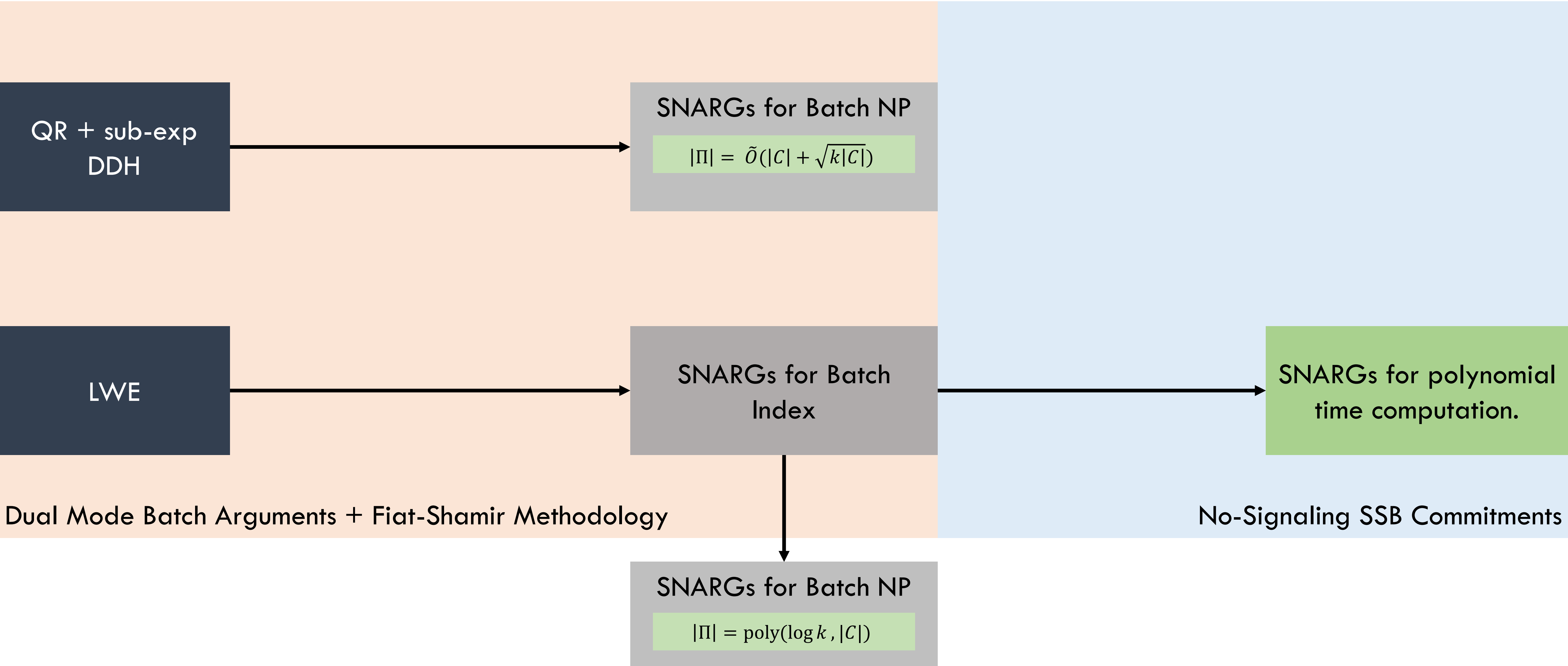
SNARGs for Batch NP



Use fixed randomness for the commitment (say 0)

Require SSB Commitments to rely on SNARG soundness.

Recap


QR + sub-exp DDH
SNARGs for Batch NP
$|\Pi| = \tilde{O}(|C| + \sqrt{k|C|})$
LWE
SNARGs for Batch Index
SNARGs for polynomial time computation.
Dual Mode Batch Arguments + Fiat-Shamir Methodology
No-Signaling SSB Commitments
SNARGs for Batch NP
$|\Pi| = \mathrm{poly}(\log k, |C|)$

Open Questions

Achieving succinct delegation from DDH?

Incrementally verifiable computation from LWE?

Establishing hardness of complexity classes such as PPAD, PLS?

Thank you. Questions?

Arka Rai Choudhuri

arkarc@berkeley.edu

ia.cr/2021/807, ia.cr/2021/808