SNARGs for P from LWE

Arka Rai Choudhuri
University of California, Berkeley

Abhishek Jain
Johns Hopkins University

Zhengzhong Jin
Johns Hopkins University

Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

$\mathcal{M}$, x

$\mathcal{M}$, x

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

within T steps

Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

$\mathcal{M}$, x

$\mathcal{M}$, x

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

within T steps

 wants to delegate computation to 

Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

$\mathcal{M}$, x

Π $\longrightarrow$

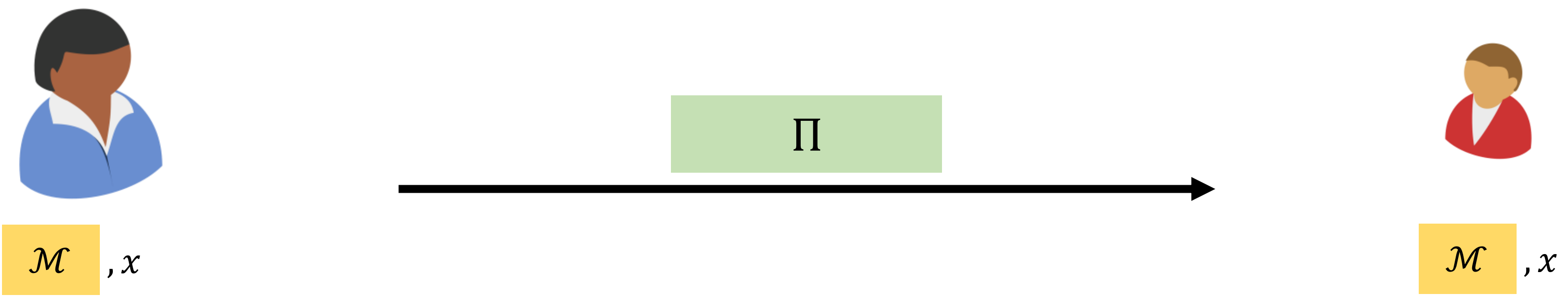
$\mathcal{M}$, x

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

within T steps

Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)



Π is publicly verifiable

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

within T steps

Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)

$\mathcal{M}$, x

$\longleftarrow$ $\text{polylog}(T)$ $\longrightarrow$

Π

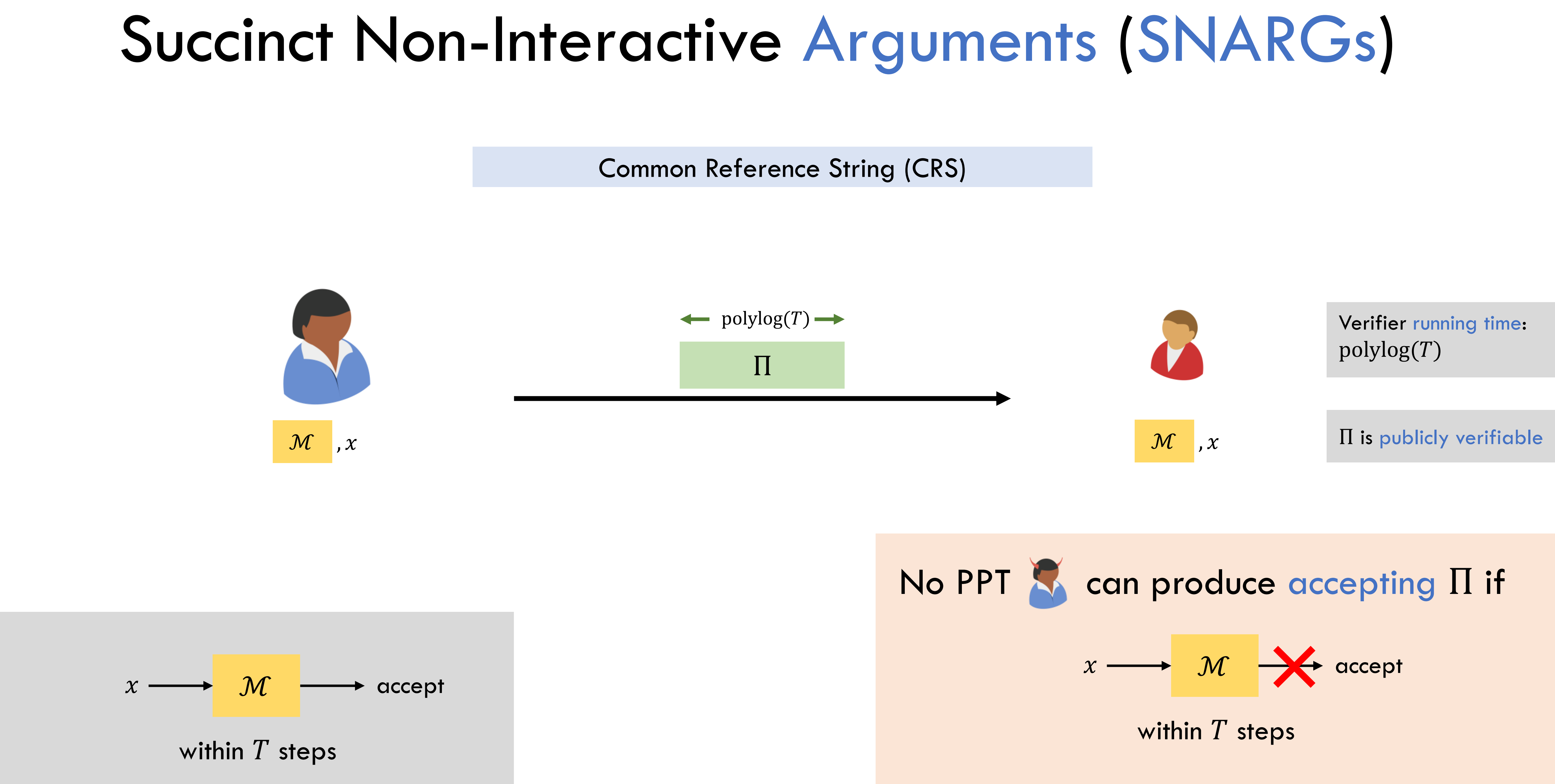
$\mathcal{M}$, x

Verifier running time: $\text{polylog}(T)$

Π is publicly verifiable

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

within T steps



Succinct Non-Interactive Arguments (SNARGs)
Common Reference String (CRS)
polylog(T)
Π
$\mathcal{M}$, x
$\mathcal{M}$, 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

$\xleftarrow{\quad}$ $\text{polylog}(T)$ $\xrightarrow{\quad}$

Π

$\mathcal{M}$, x

Verifier running time: $\text{polylog}(T)$

Π is publicly verifiable

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

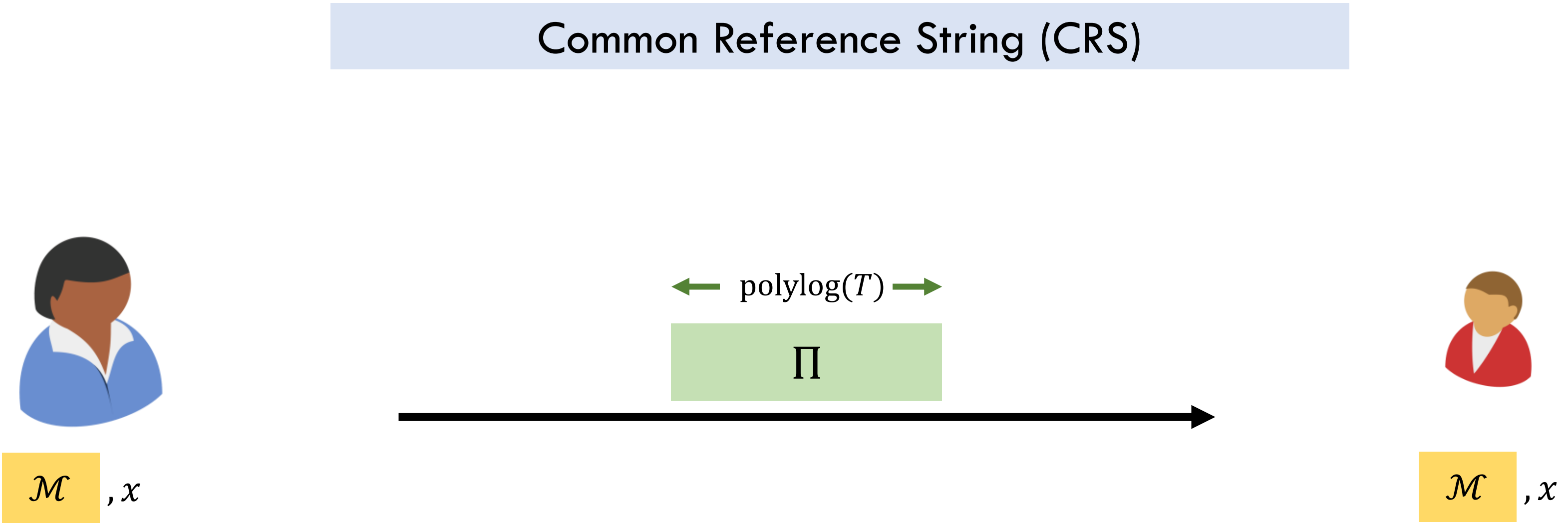
within T steps

No PPT  can produce accepting x, Π if

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

within T steps

Succinct Non-Interactive Arguments (SNARGs)



Verifier running time: polylog(T)

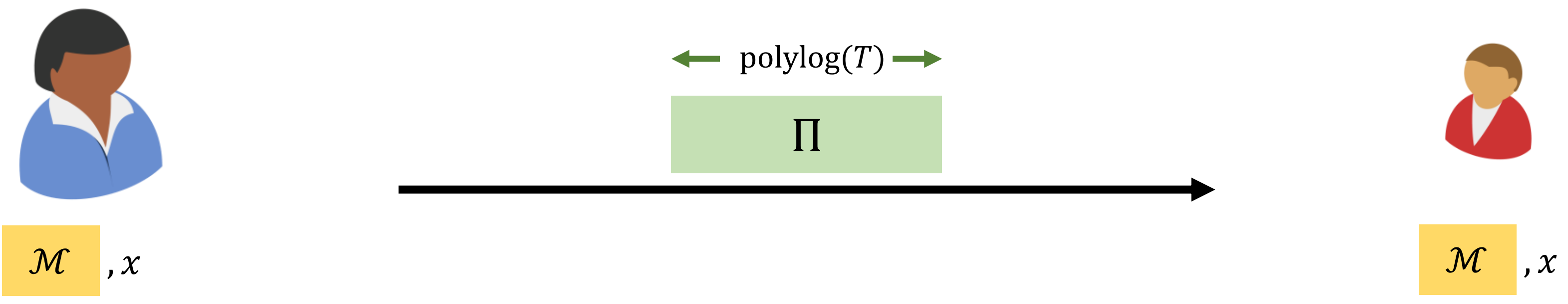
Π is publicly verifiable

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

x

Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)



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]

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

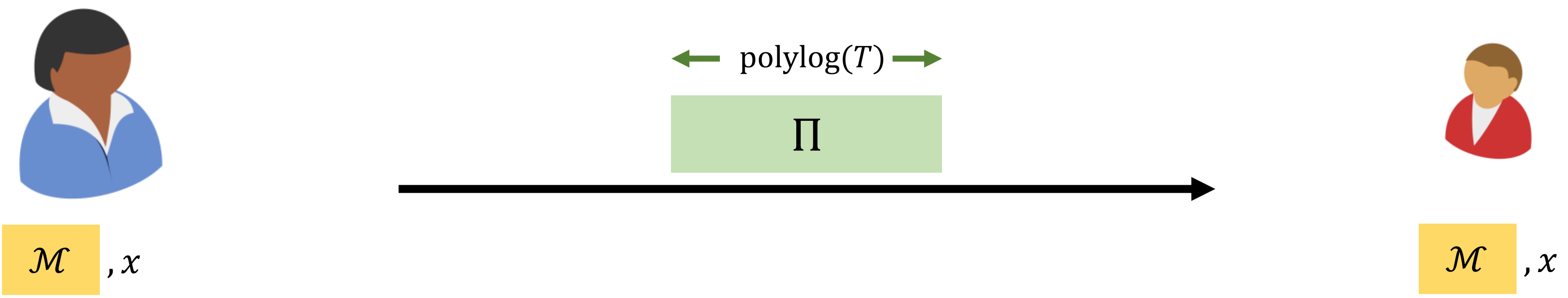
What kind of computation can we hope to delegate based on standard assumptions?

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

x

Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)



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 computation?
- Sub-classes of NP?

x

SNARGs for Batch NP

CRS

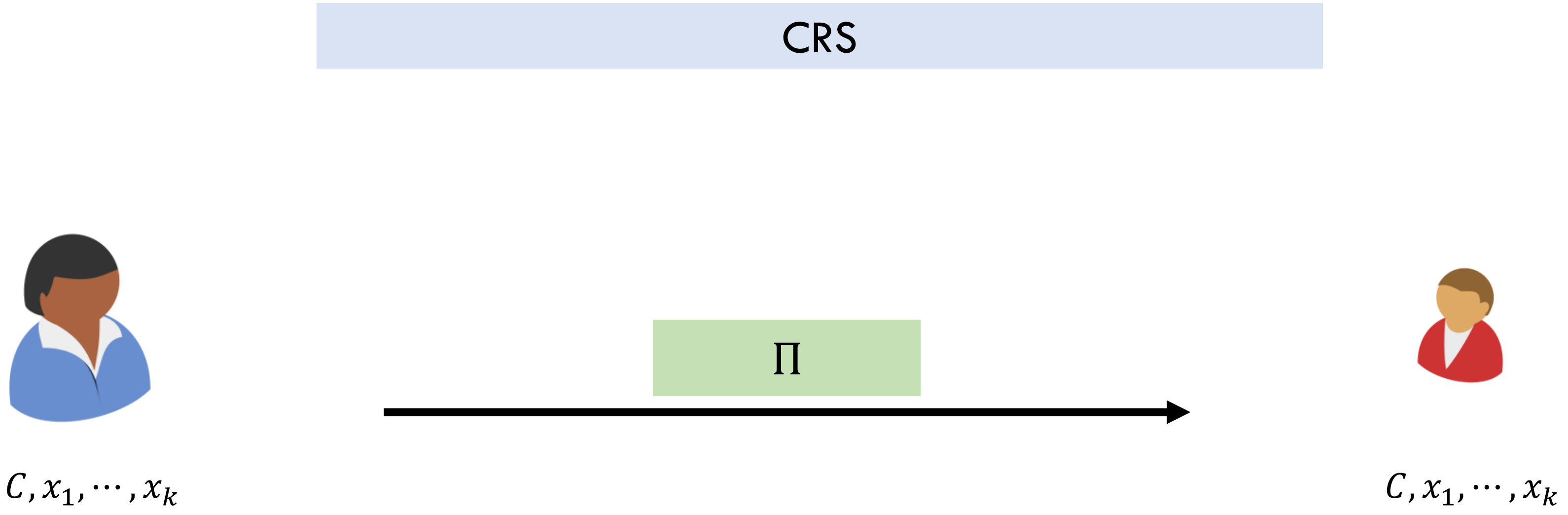
$C, x_1, \cdots, x_k$

$C, x_1, \cdots, x_k$

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

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

SNARGs for Batch NP



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

$C, x_1, \cdots, x_k$

Π

$C, x_1, \cdots, x_k$

Π is publicly verifiable

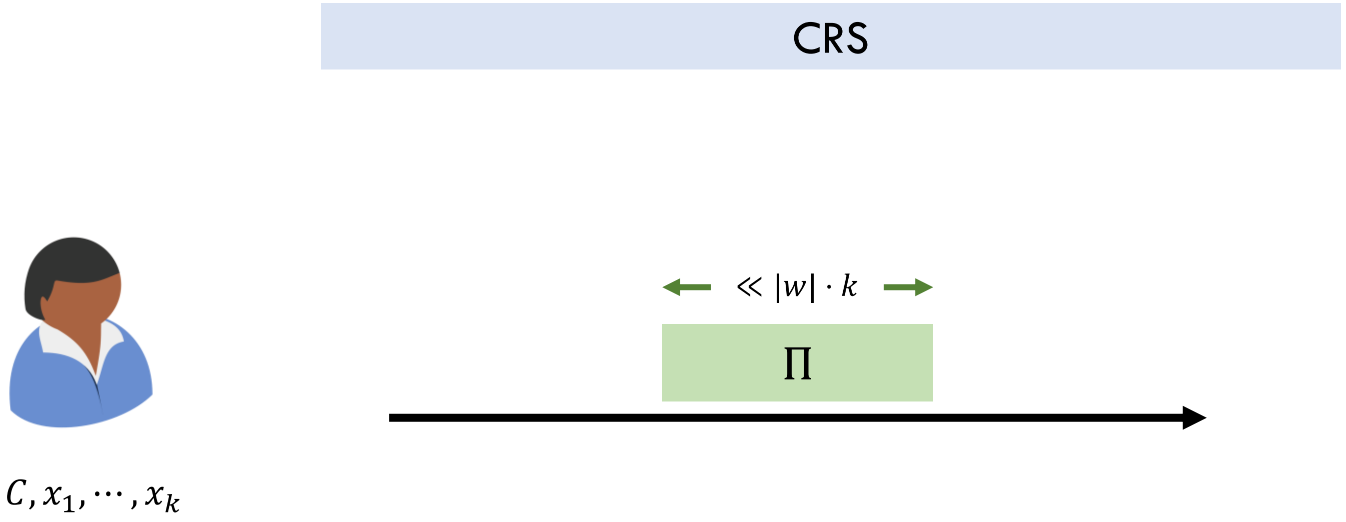
$\text{SAT} = \{(C, x) \mid \exists w \ s.t. \ C(x, w) = 1\}$

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

No PPT  can produce accepting Π if

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

SNARGs for Batch NP



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

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

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

SNARGs for Batch NP

CRS

$C, x_1, \cdots, x_k$

$\longleftarrow \ll |w| \cdot k \longrightarrow$

Π

$C, x_1, \cdots, x_k$

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

Π is publicly verifiable

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

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

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

Prior Works

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

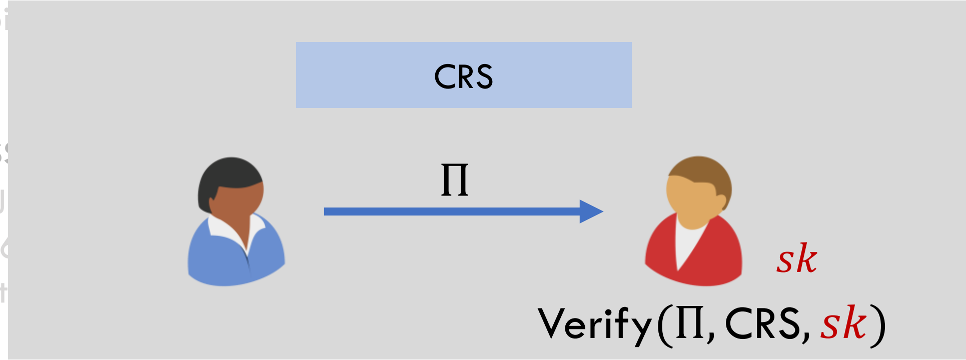
[Canetti-Holmgren-Jain-Vaikuntanathan'15, Koppula-Lewko-Waters'15, Bitansky-Garg-Lin-Pass-Telang'15, Canetti-Holmgren'16, Ananth-Chen-Chung-Lin-Lin'16, Chen-Chow-Chung-Lai-Lin-Zhou'16, Paneth-Rothblum'17, Canetti-Chen-Holmgren-Lombardi-Rothblum-Rothblum-Wichs'19, Kalai-Paneth-Yang'19]

Delegation for P and some for batch 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-Rubi



"Less standard" as

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

Some works can delegate NP

Delegation for P and some for batch NP

Designated Verifier (standard assumptions)

Prior Works

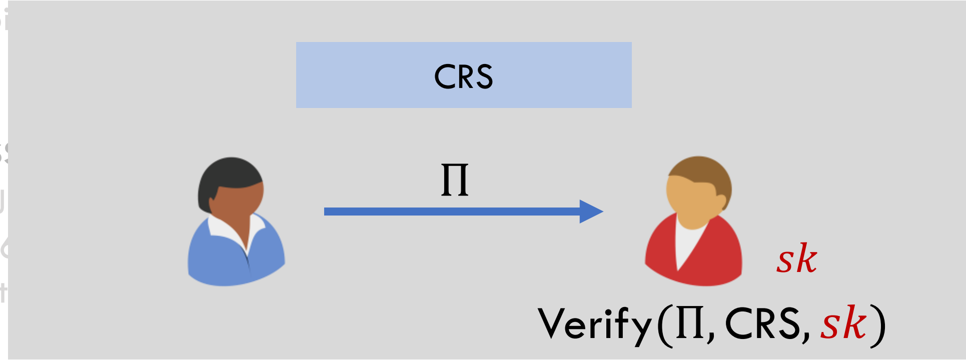
Non-falsifiable assumptions/ Random oracle model

[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

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 and some for batch NP

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]

Delegation for P and some for batch NP

Do there exists SNARGs for P and batch NP based on standard assumptions?

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.

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

Previously best known: [C-Jain-Jin'21a] assuming QR + (LWE/sub-exp DDH)

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

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

Theorem

LWE → SNARGs for Batch NP

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

LWE – Learning with Error

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

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

Results

SNARGs for Batch NP $\xrightarrow[\text{LWE}]{\text{Assuming}}$ SNARGs for polynomial time computation.

LWE – Learning with Error

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

within T steps

Results

Theorem

LWE → SNARGs for Batch NP → SNARGs for polynomial time computation.

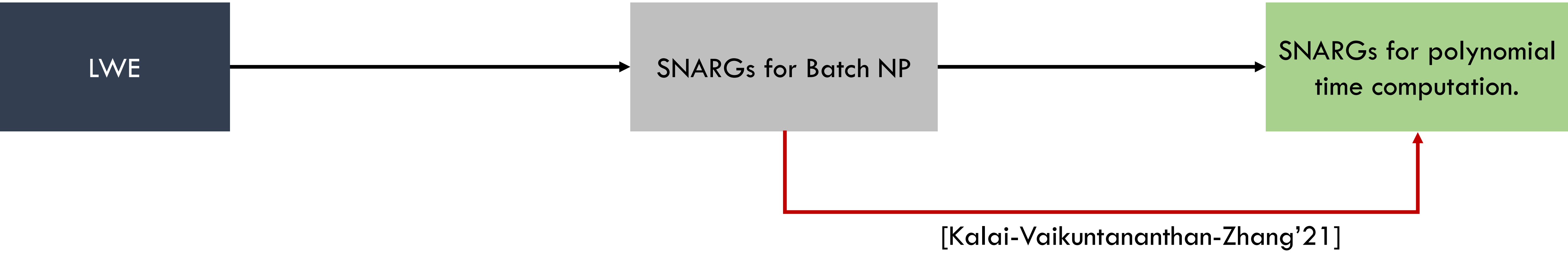
$|\mathrm{CRS}|, |\Pi|, |$$| = \mathrm{polylog}(T)$

LWE – Learning with Error

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

within T steps

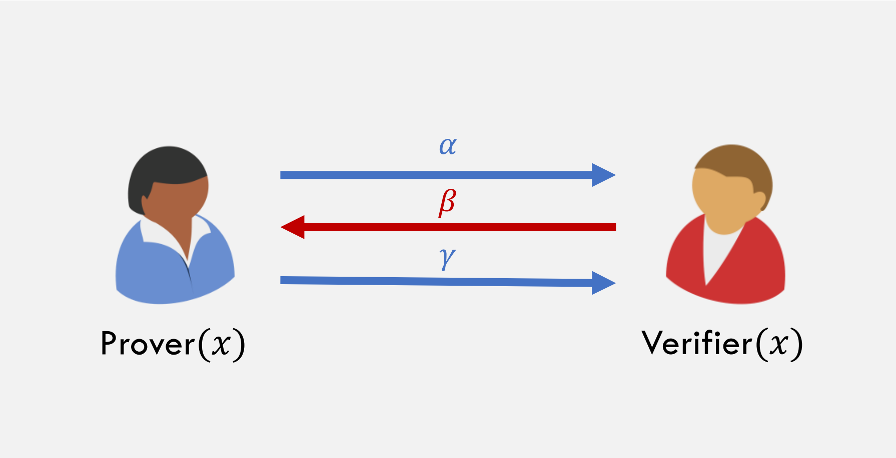
Results



LWE
SNARGs for Batch NP
SNARGs for polynomial time computation.
[Kalai-Vaikuntananthan-Zhang'21]

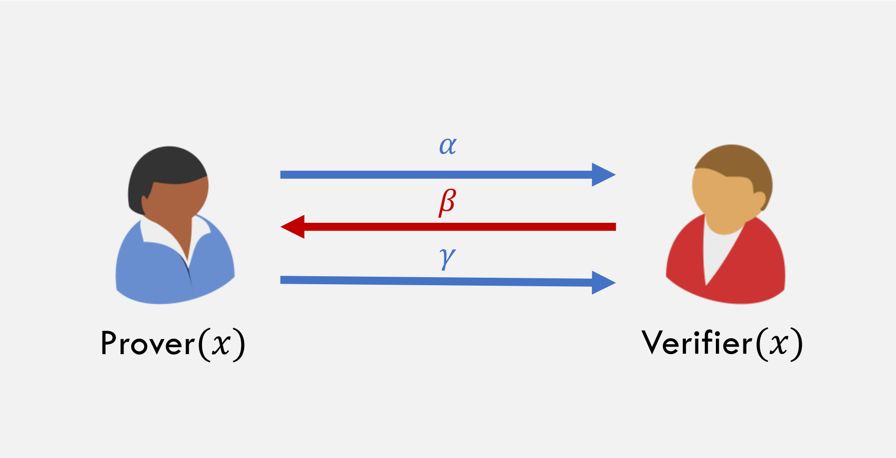
Tool: Fiat-Shamir (FS) Methodology

Fiat-Shamir (FS) Methodology



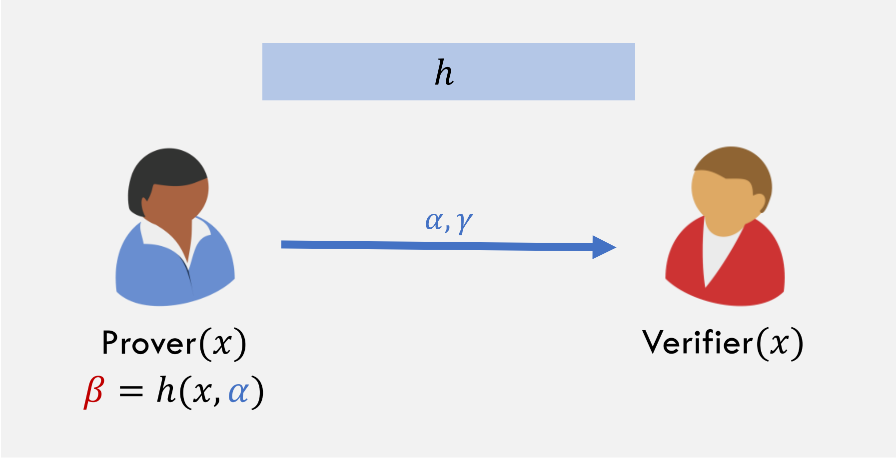
β is a random string

Fiat-Shamir (FS) Methodology

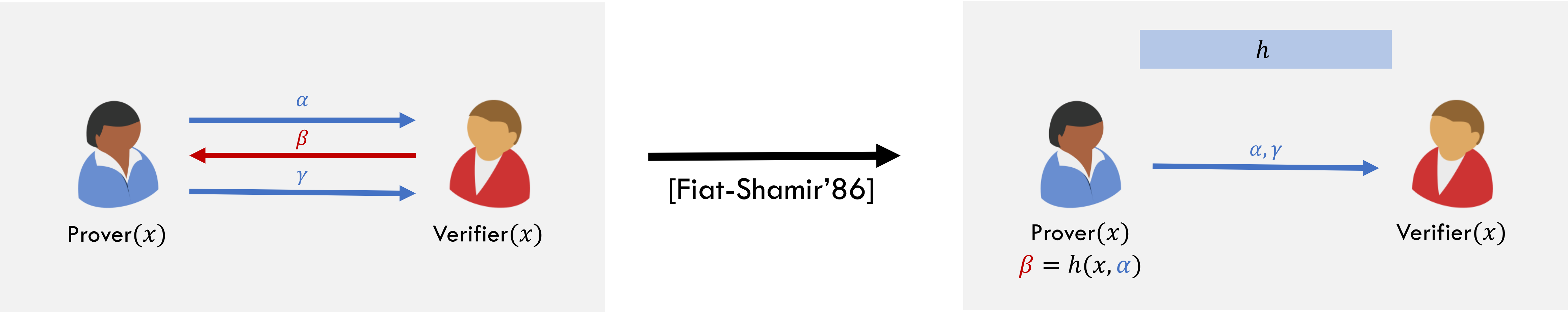


β is a random string

[Fiat-Shamir'86]



Fiat-Shamir (FS) Methodology

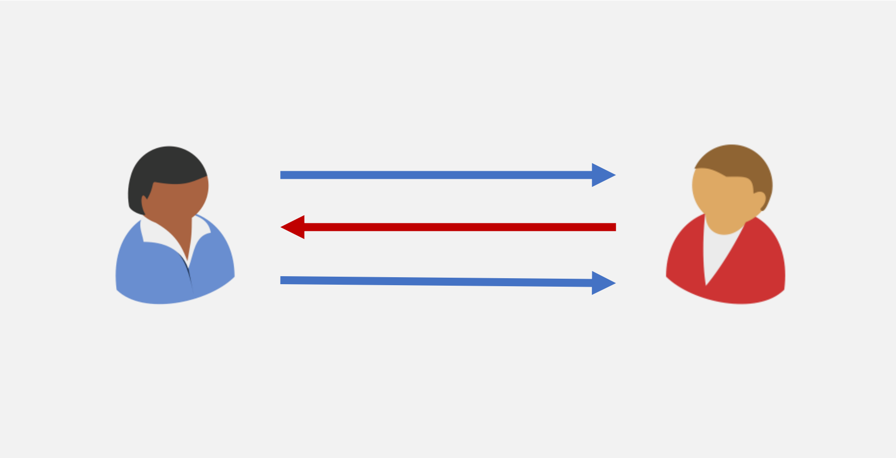


β is a random string

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

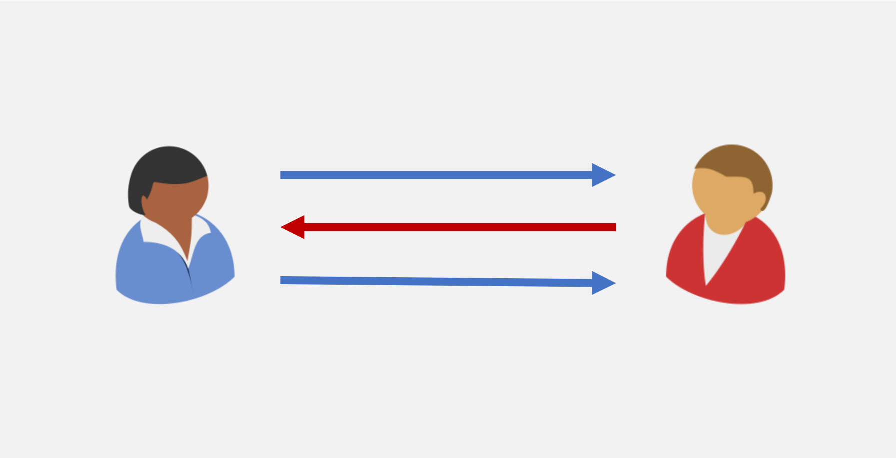
[Kalai-Rothblum-Rothblum'17, Canetti-Chen-Reyzin-Rothblum'18, Holmgren-Lombardi'18, Canetti-Chen-Holmgren-Lombardi-Rothblum-Rothblum-Wichs'19, Peikert-Sheihian'19, Brakerski-Koppula-Mour'20, Couteau-Katsumata-Ursu'20, Jain-Jin'21, Jawale-Kalai-Khurana-Zhang'21, Holmgren-Lombardi-Rothblum'21]

Fiat-Shamir (FS) Methodology



[Kilian'92]

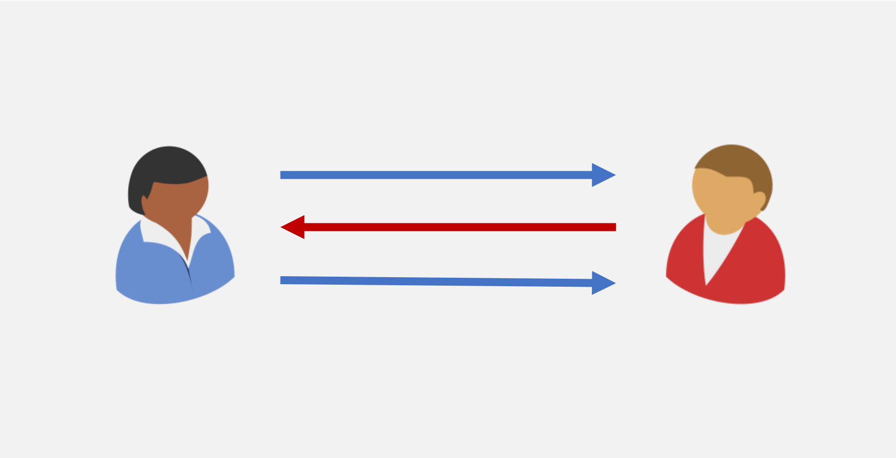
Fiat-Shamir (FS) Methodology



[Kilian'92]

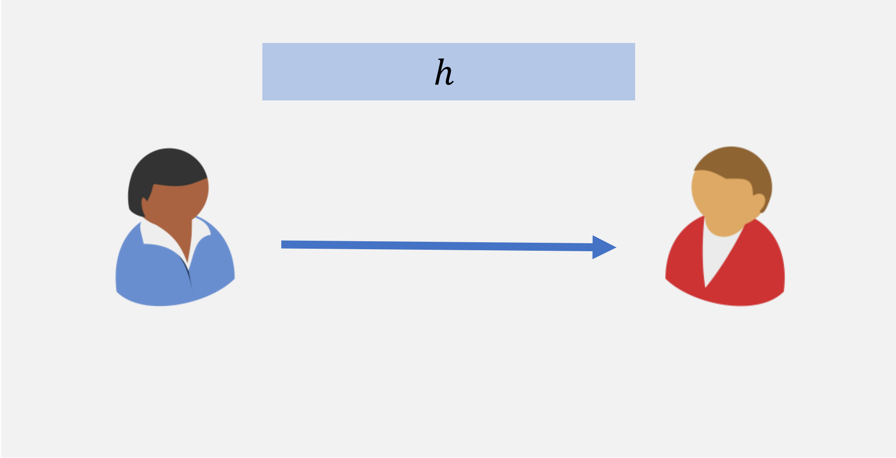
Succinct interactive arguments for NP.

Fiat-Shamir (FS) Methodology



[Kilian'92]

Instantiate with CIH

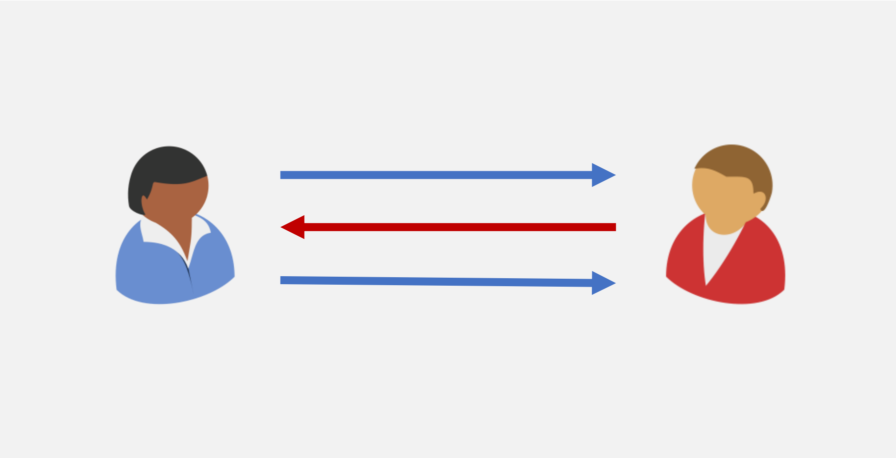


Succinct interactive arguments for NP.

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

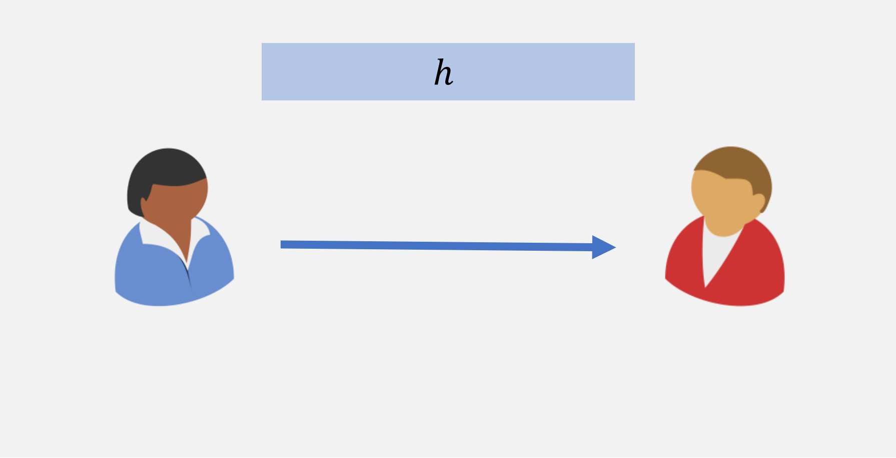
Instantiating hash function for Fiat-Shamir transformation of Kilian's protocol is hard.

Fiat-Shamir (FS) Methodology



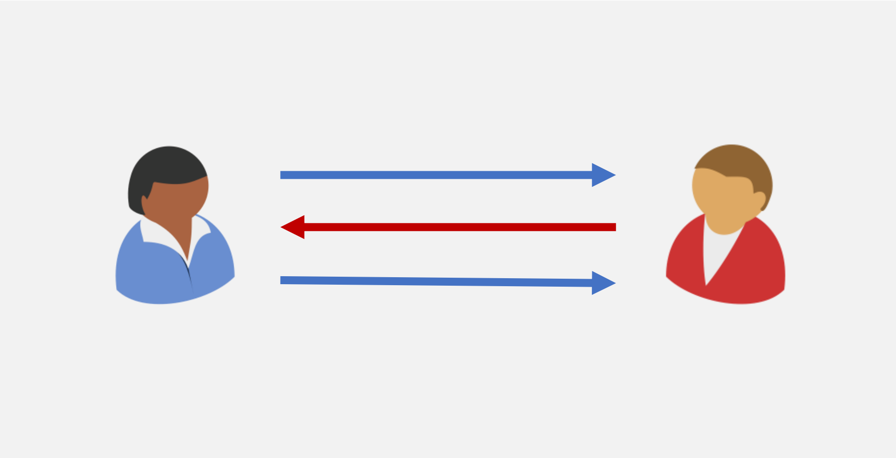
[Kilian'92]

Instantiate with CIH

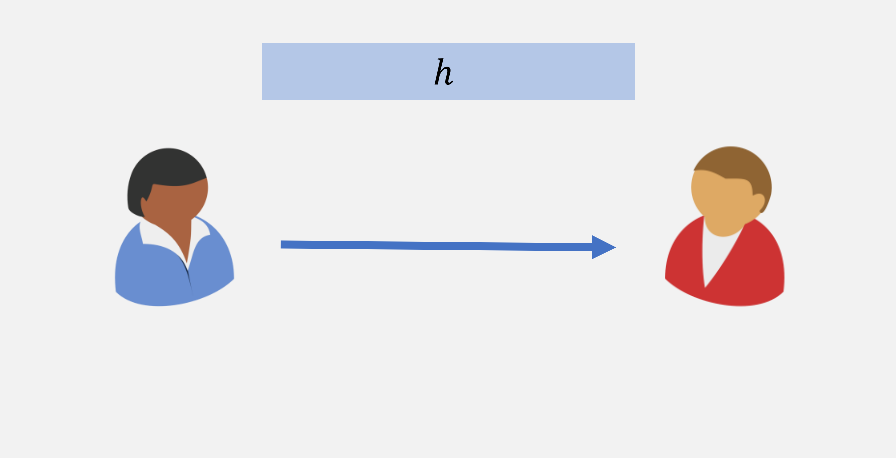


Succinct interactive arguments for NP.

Fiat-Shamir (FS) Methodology



Instantiate with CIH

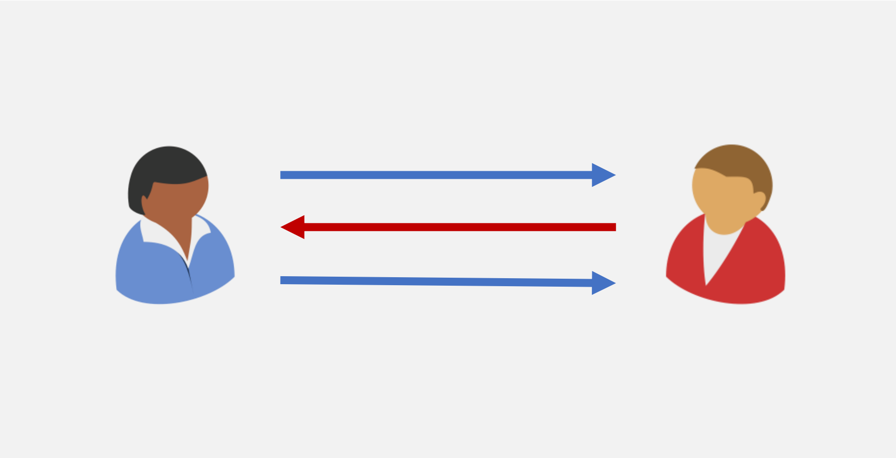


[Kilian'92]

Succinct interactive arguments for NP.

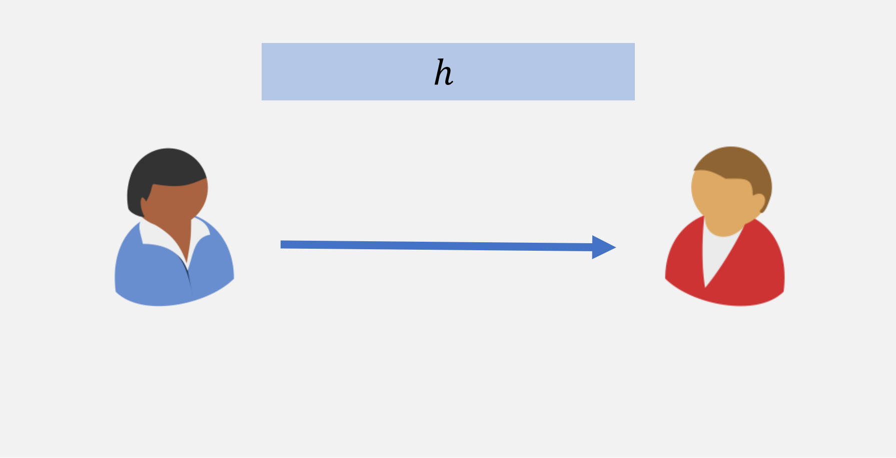
Known instantiations of CI Hash for Fiat-Shamir transform are for proofs.

Fiat-Shamir (FS) Methodology



[Goldwasser-Kalai-Rothblum'08]

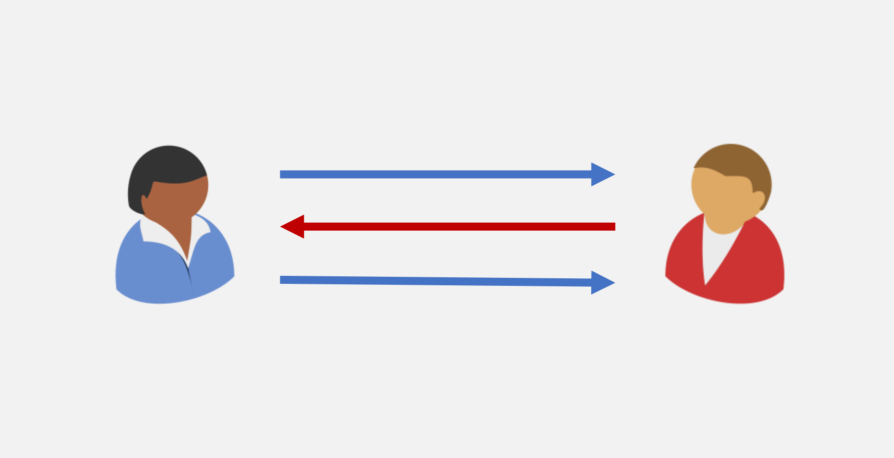
Instantiate with CIH



Succinct interactive proof for depth bounded computation.

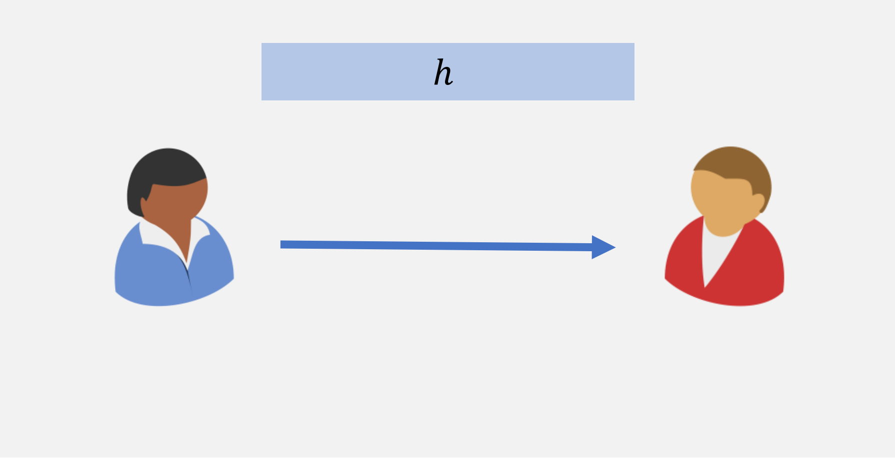
[Canetti-Chen-Holmgren-Lombardi-Rothblum-Rothblum-Wichs'19, Jawale-Kalai-Khurana-Zhang'21]

Fiat-Shamir (FS) Methodology



[Goldwasser-Kalai-Rothblum’08]

Instantiate with CIH



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

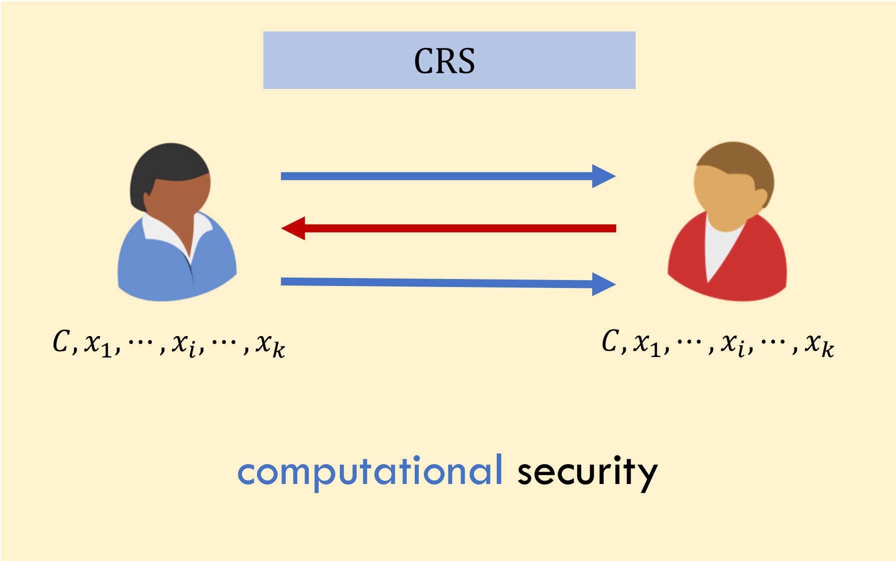
Succinct interactive proof for depth bounded computation.

- Interactive proofs for all polynomial time computation unlikely to exist.
- No known interactive proof for batch NP.

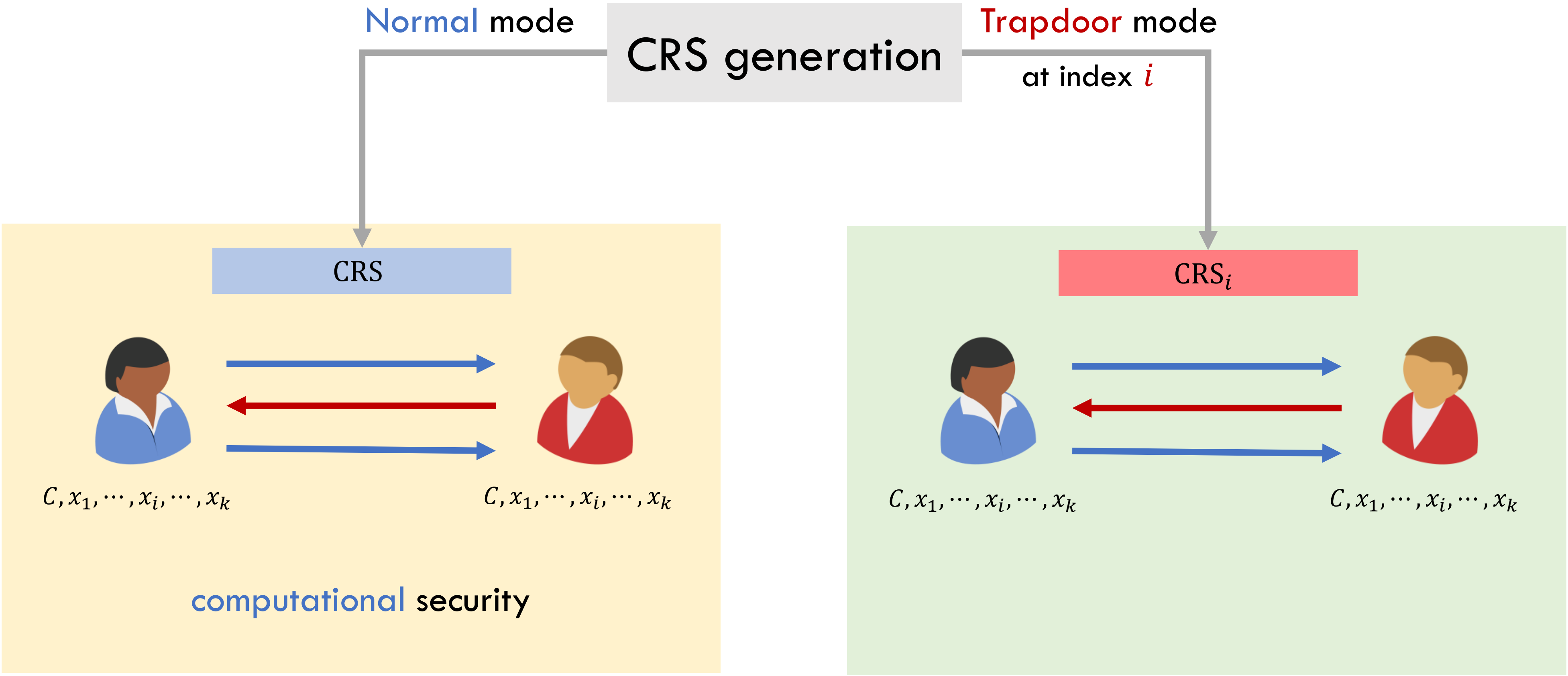
A Different Starting Point

Dual-Mode Interactive Batch Arguments [C-Jain-Jin'21a]

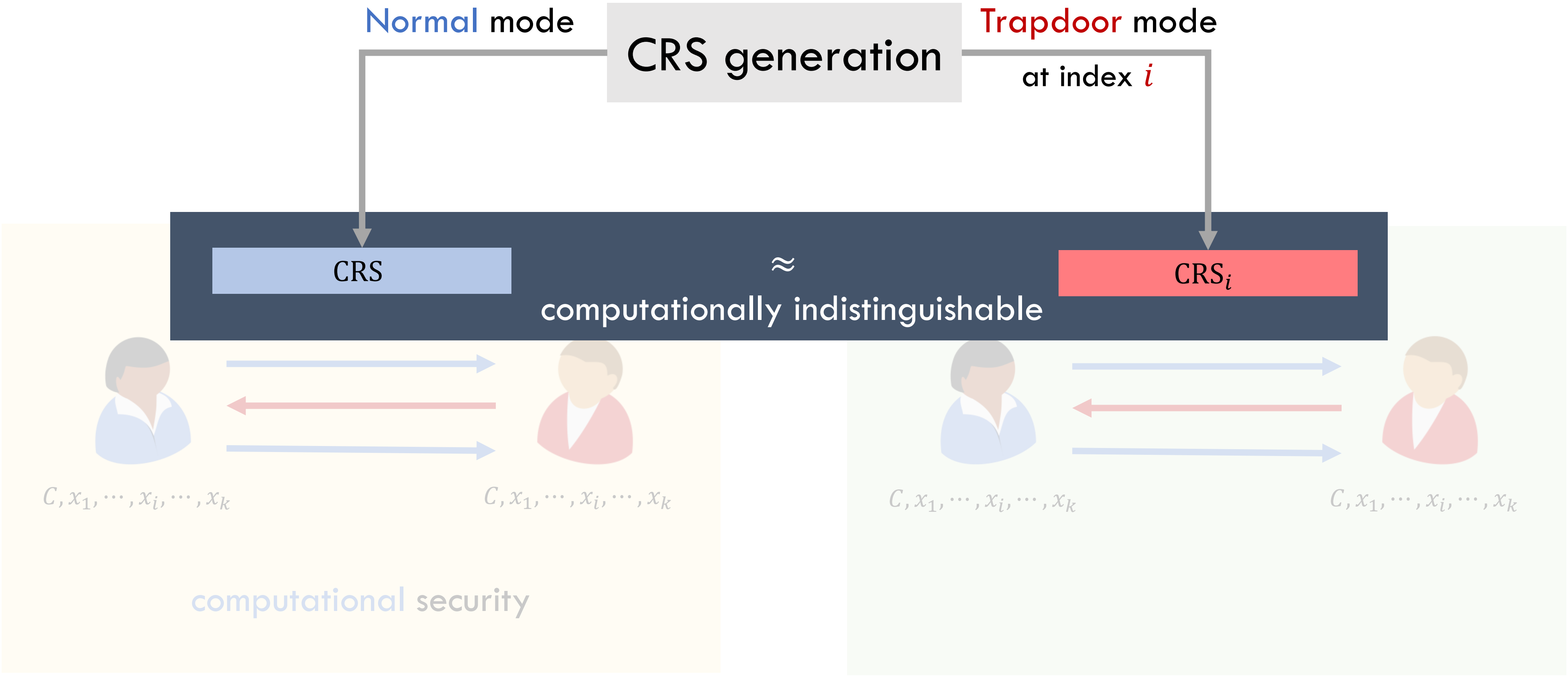
Dual-Mode Interactive Batch Arguments



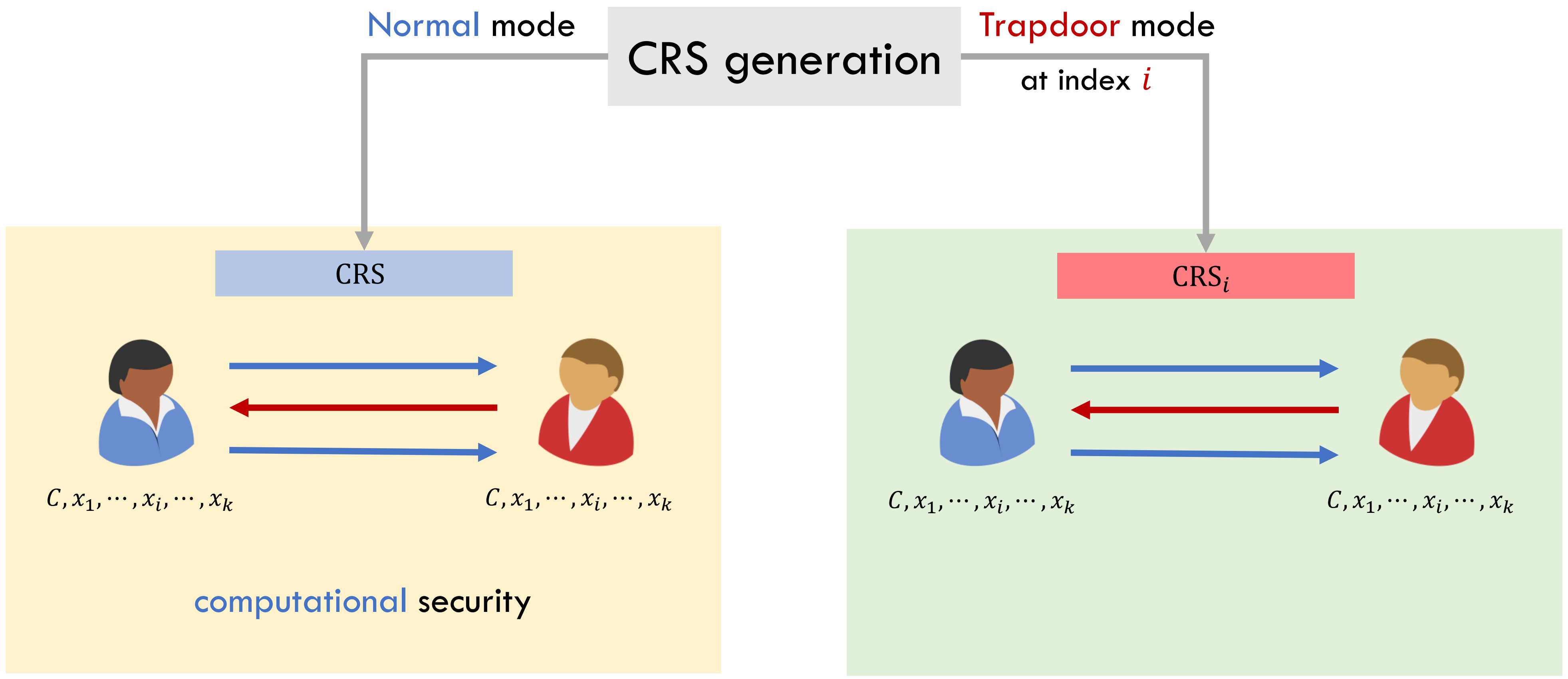
Dual-Mode Interactive Batch Arguments



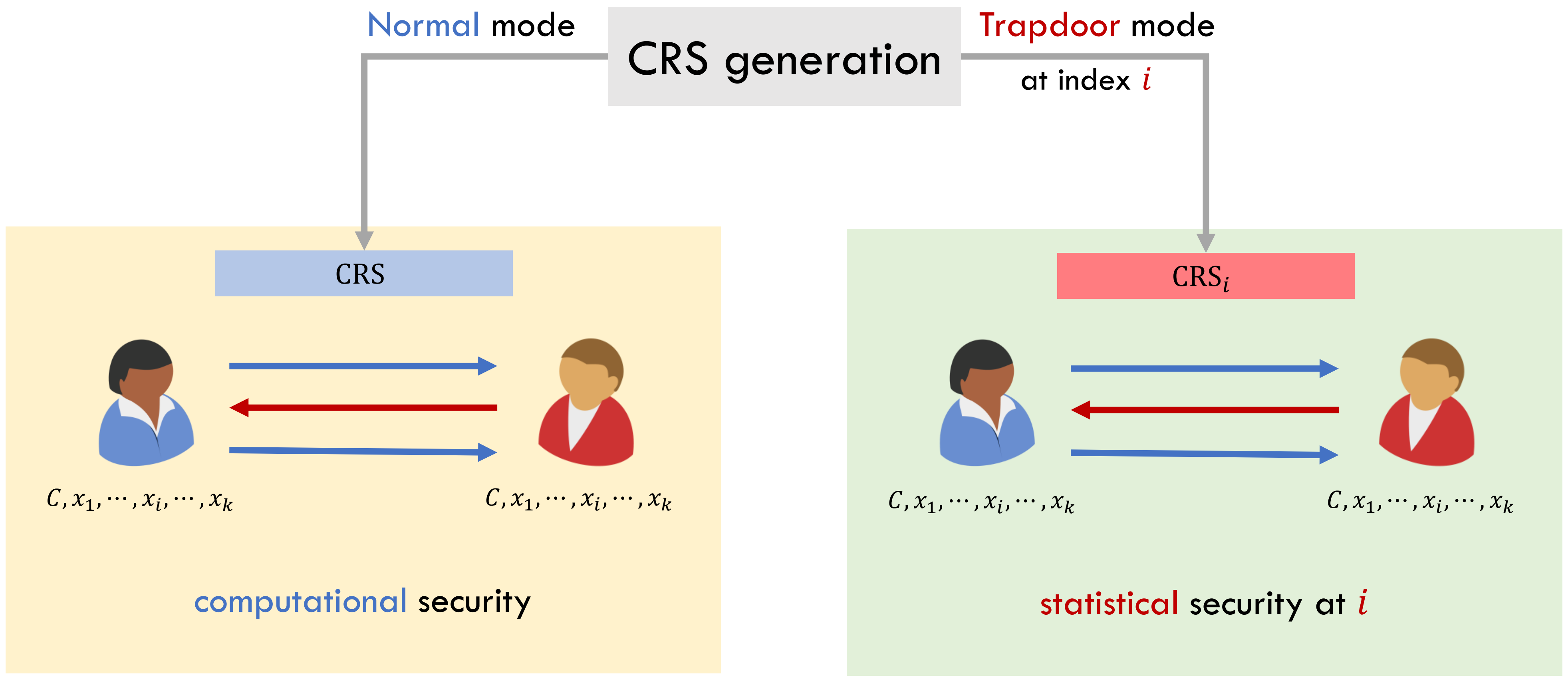
Dual-Mode Interactive Batch Arguments



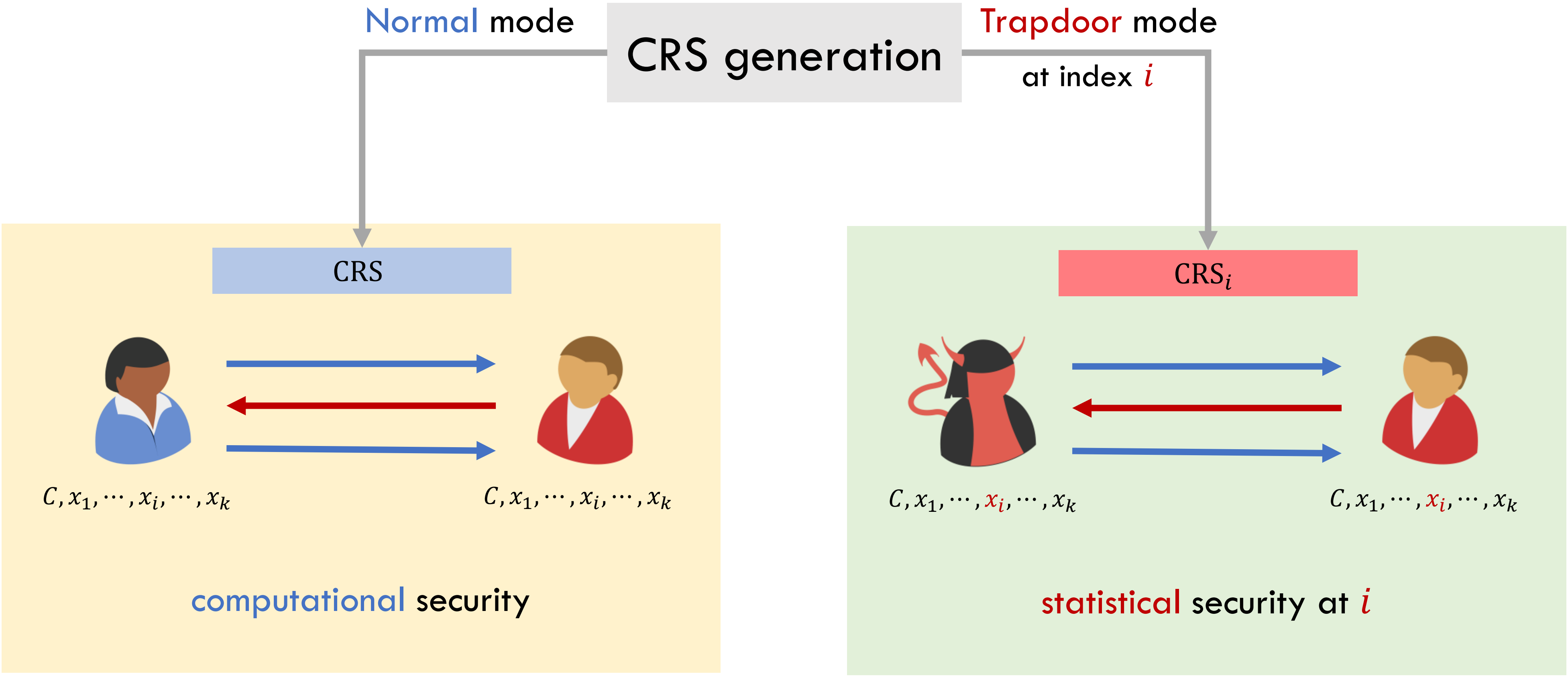
Dual-Mode Interactive Batch Arguments



Dual-Mode Interactive Batch Arguments

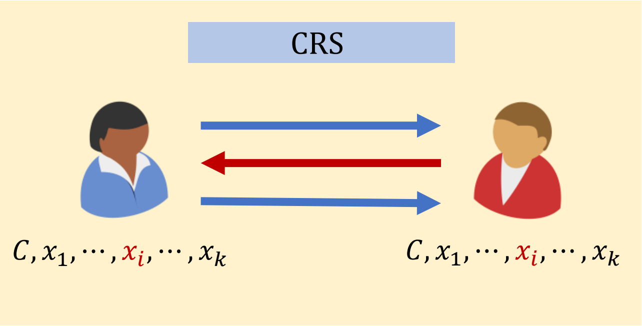


Dual-Mode Interactive Batch Arguments

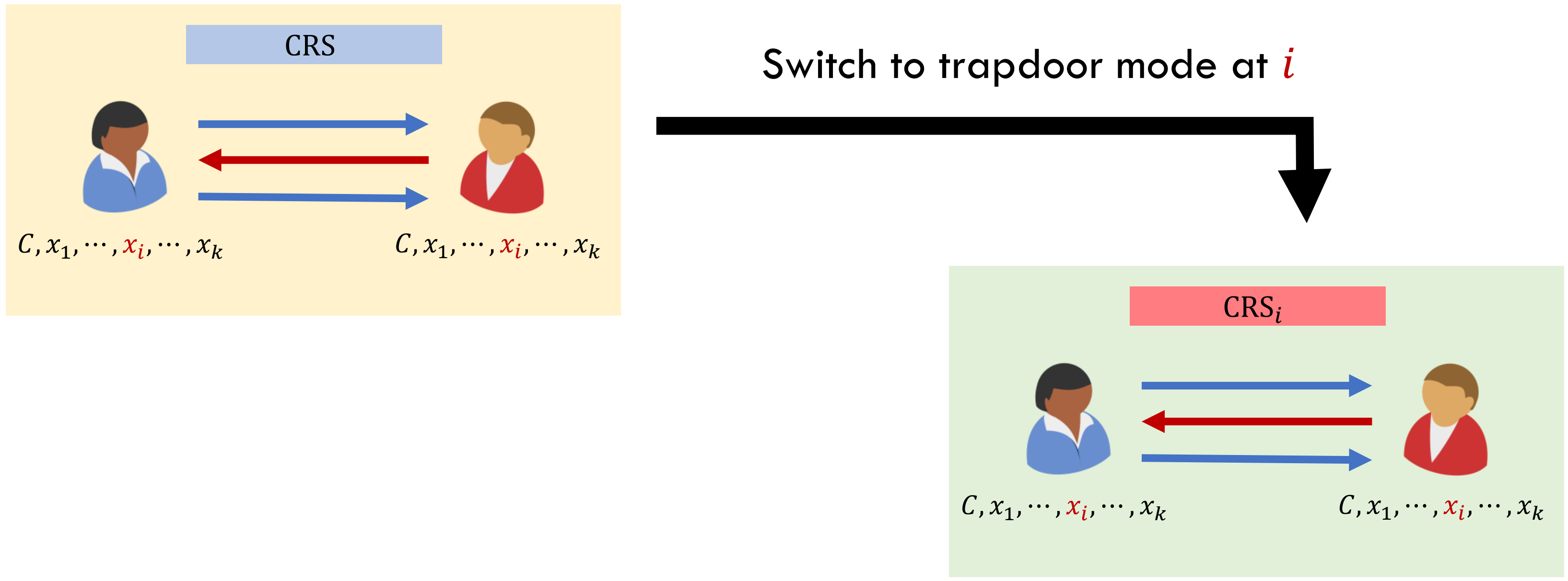


Even unbounded cannot make accept if $(C, x_i) \notin \mathrm{SAT}$

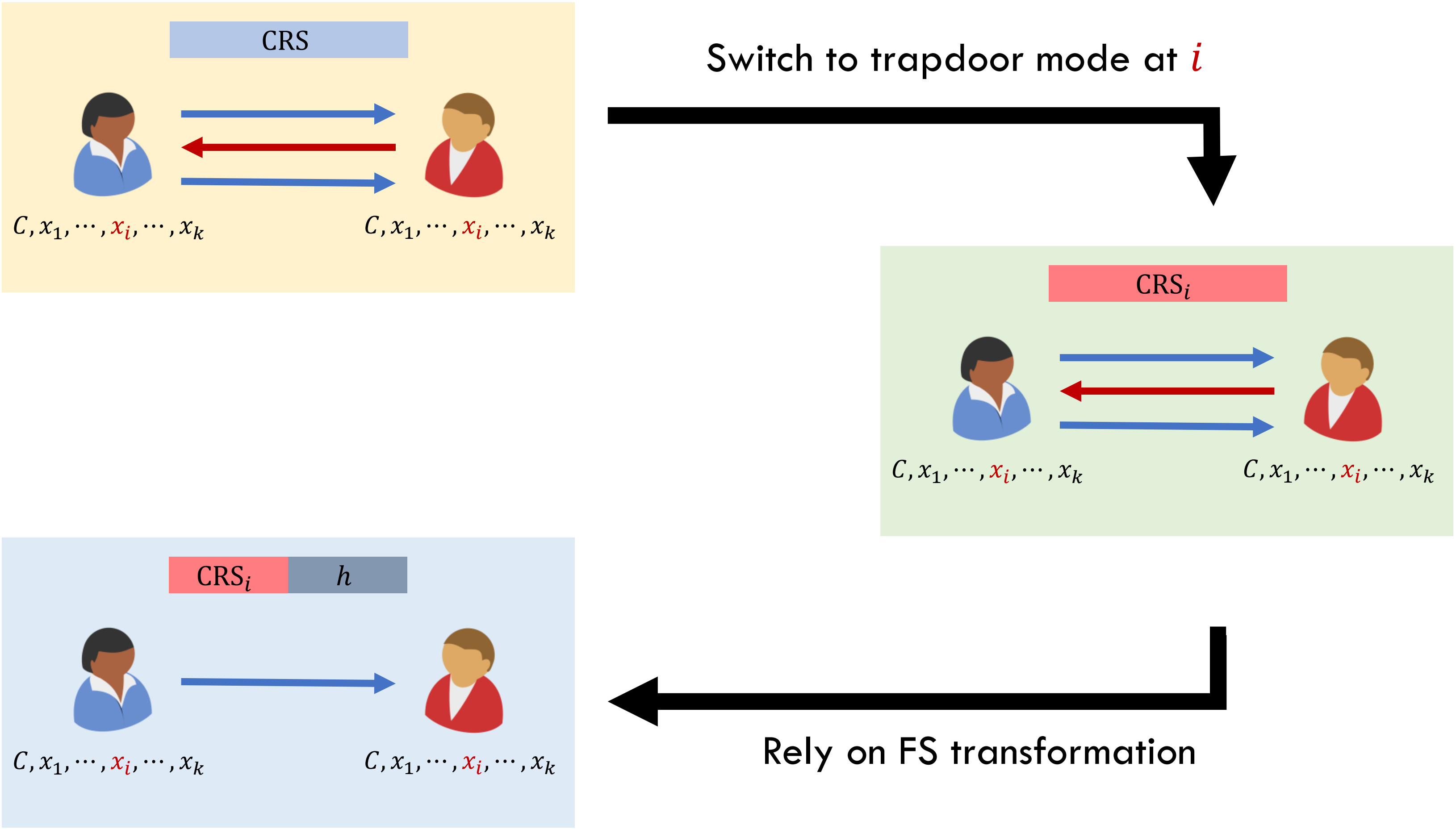
Security Intuition



Security Intuition

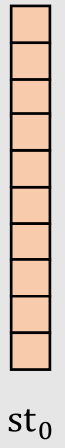


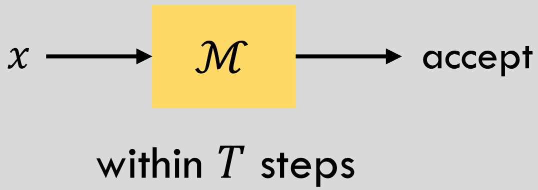
Security Intuition



SNARGs for Batch NP $\rightarrow$ SNARGs for P

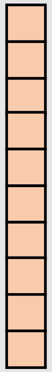
SNARGs for Polynomial-time Computation





This talk: Bounded space computation

SNARGs for Polynomial-time Computation

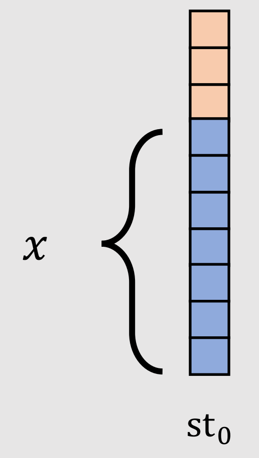


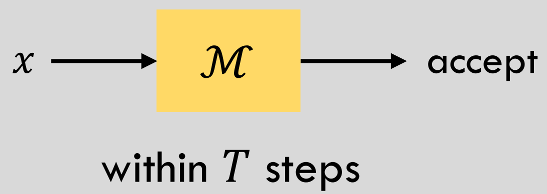
st_0

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

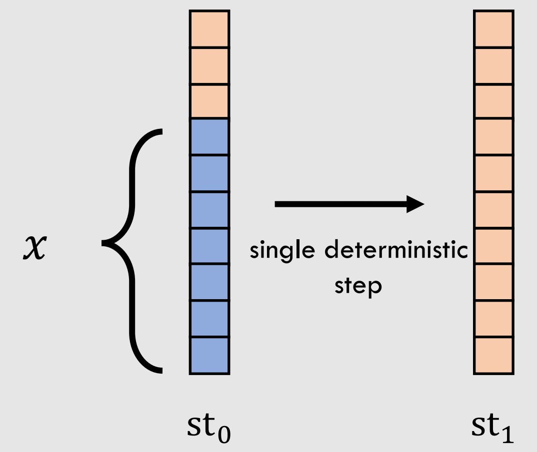
within T steps

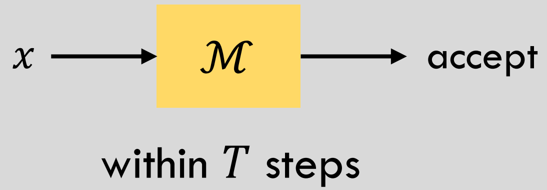
SNARGs for Polynomial-time Computation



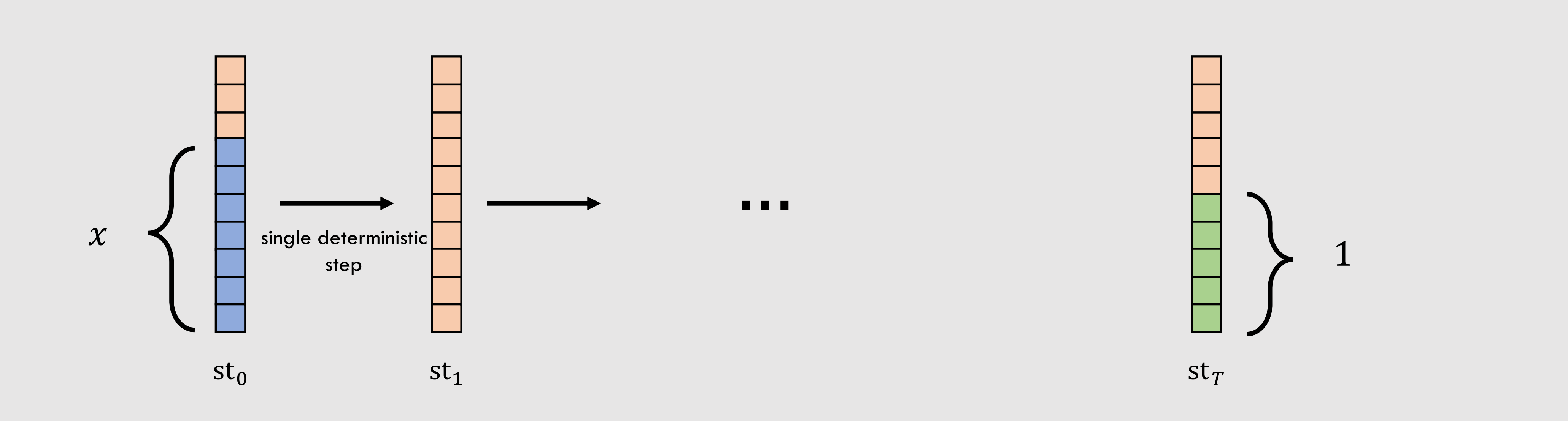


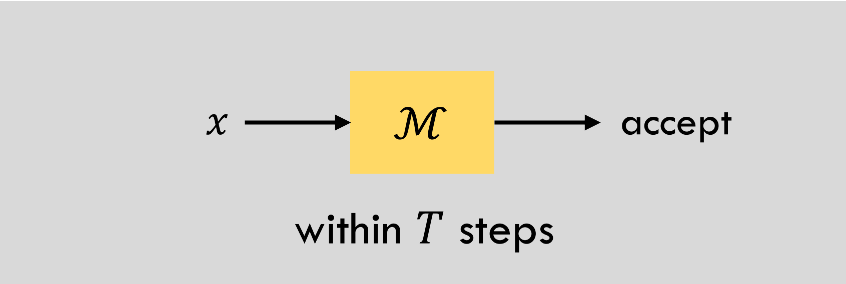
SNARGs for Polynomial-time Computation



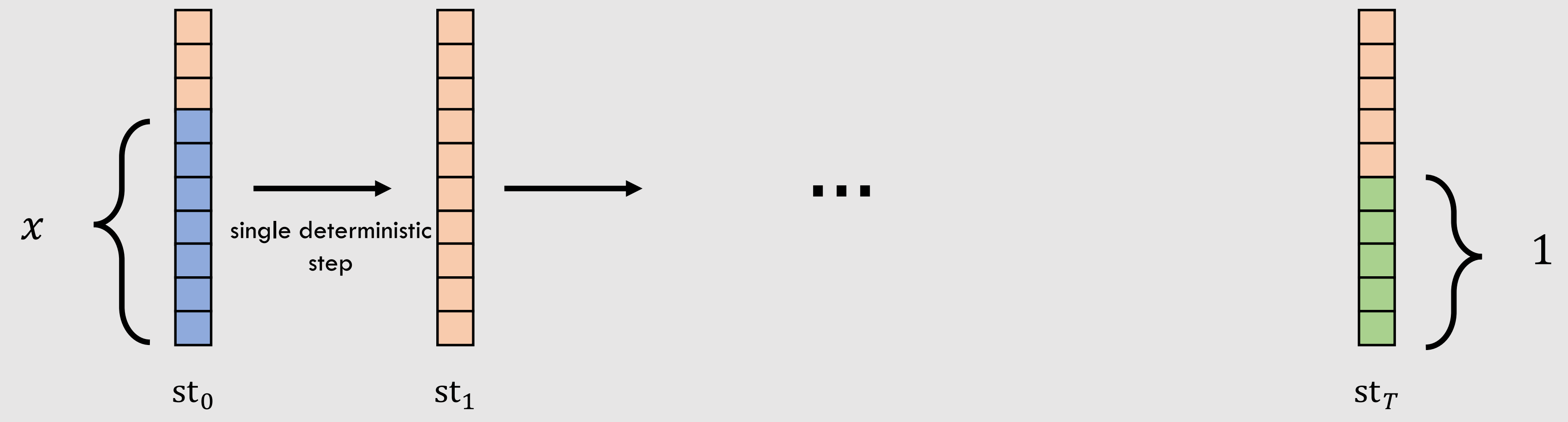


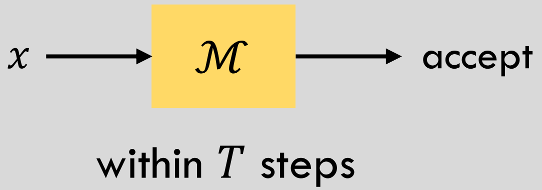
SNARGs for Polynomial-time Computation





SNARGs for Polynomial-time Computation



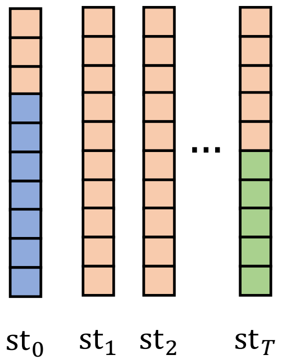


Prove for every $i \in [0, \dots, T-1]$

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

is the correct transition.

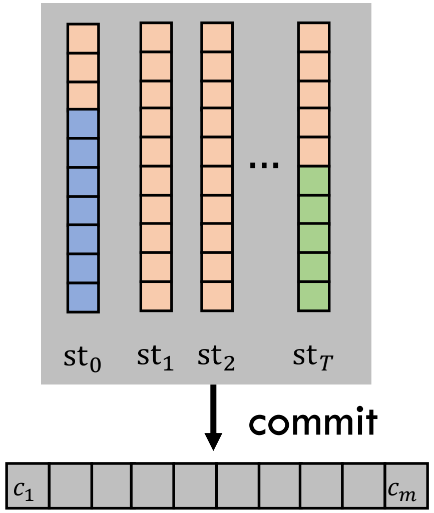
SNARGs for Polynomial-time Computation

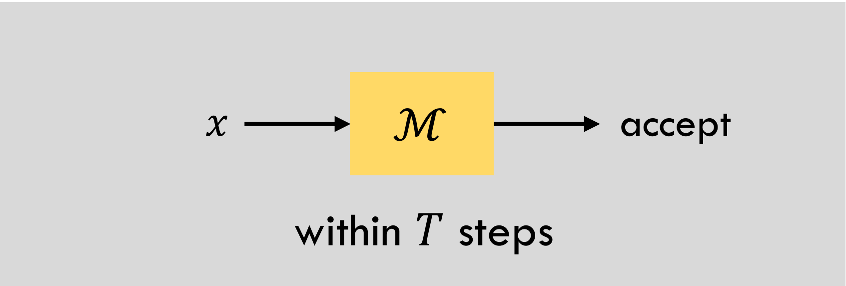


$x \longrightarrow \boxed{\mathcal{M}} \longrightarrow$ accept

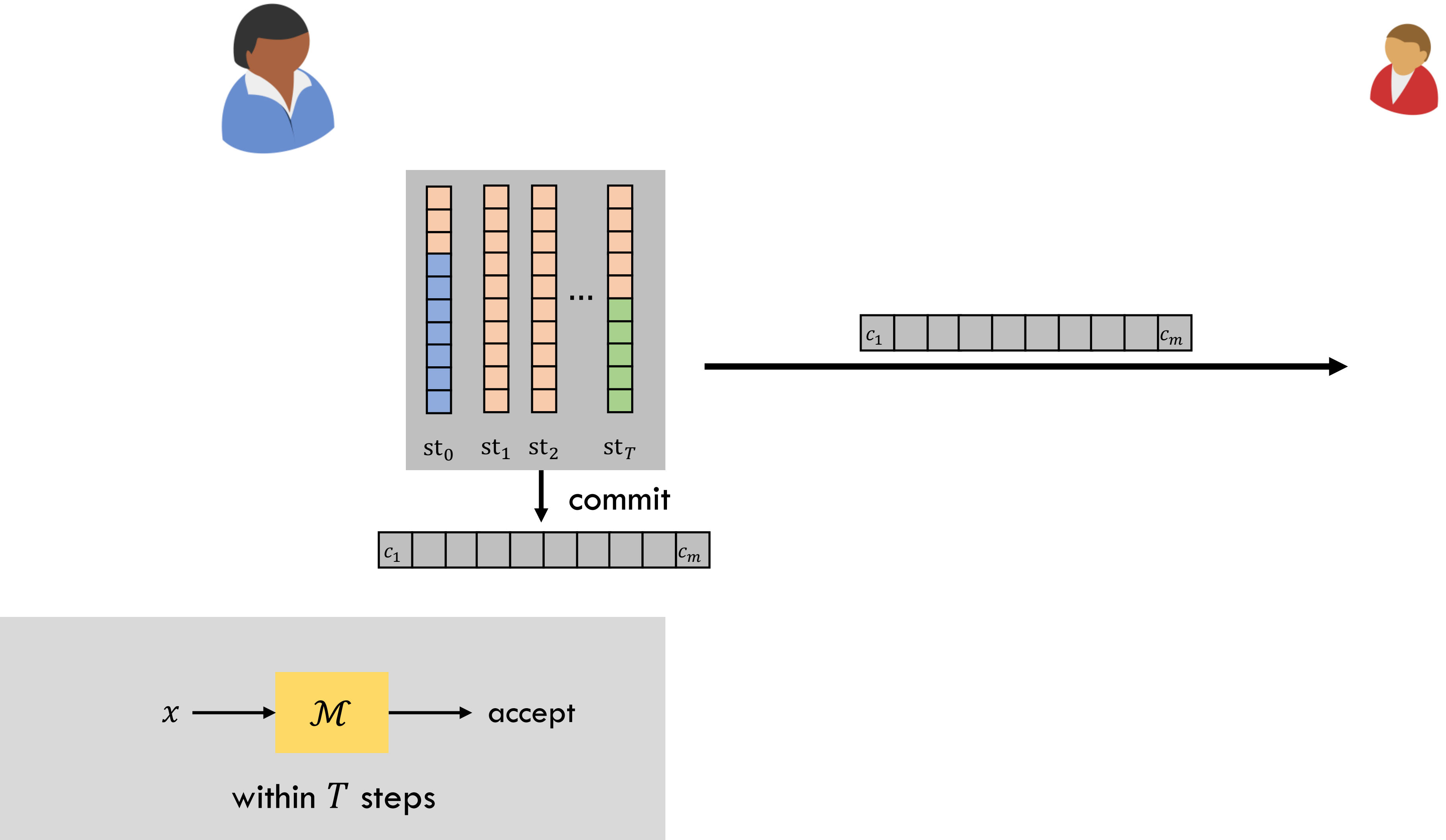
within T steps

SNARGs for Polynomial-time Computation

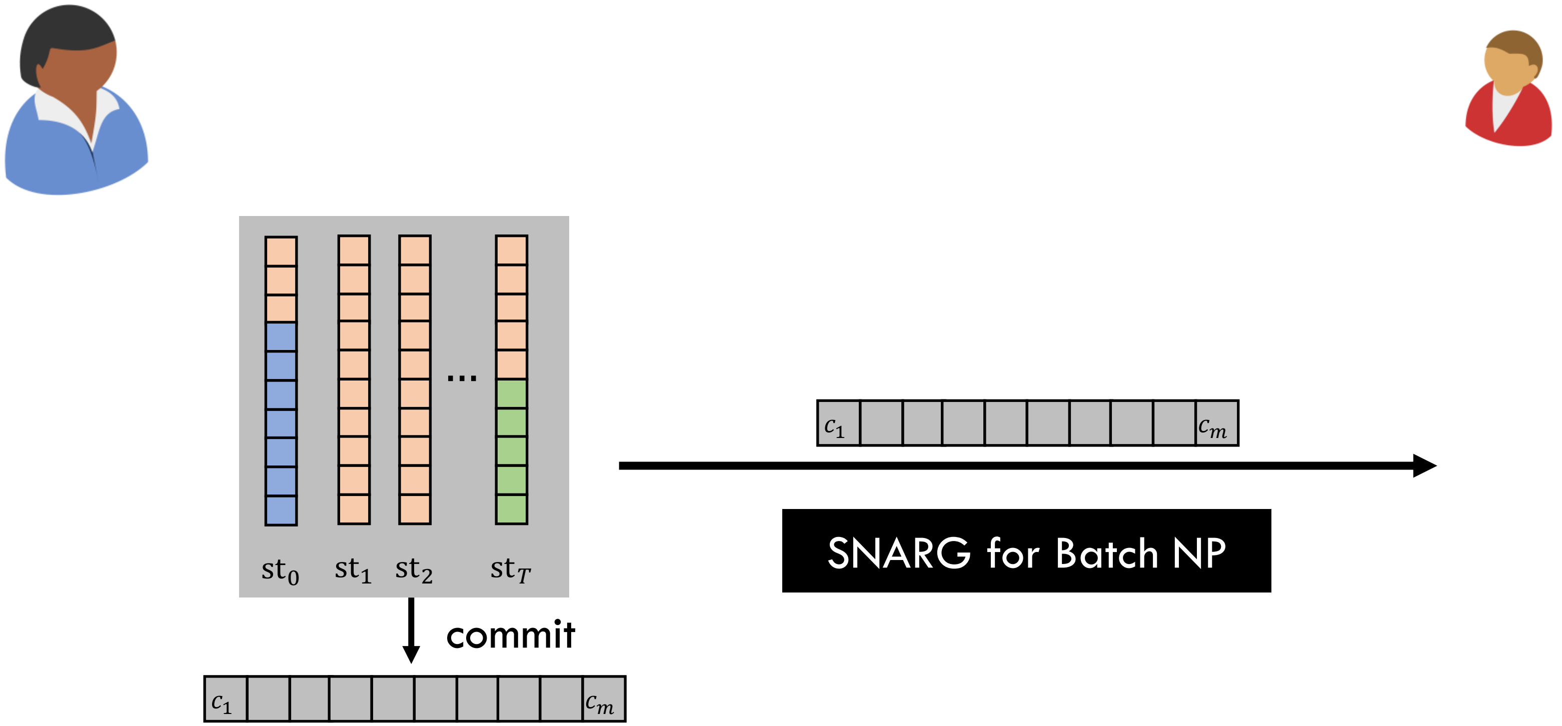


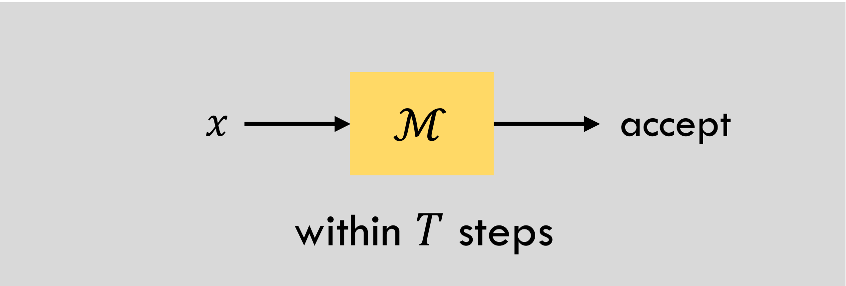


SNARGs for Polynomial-time Computation



SNARGs for Polynomial-time Computation



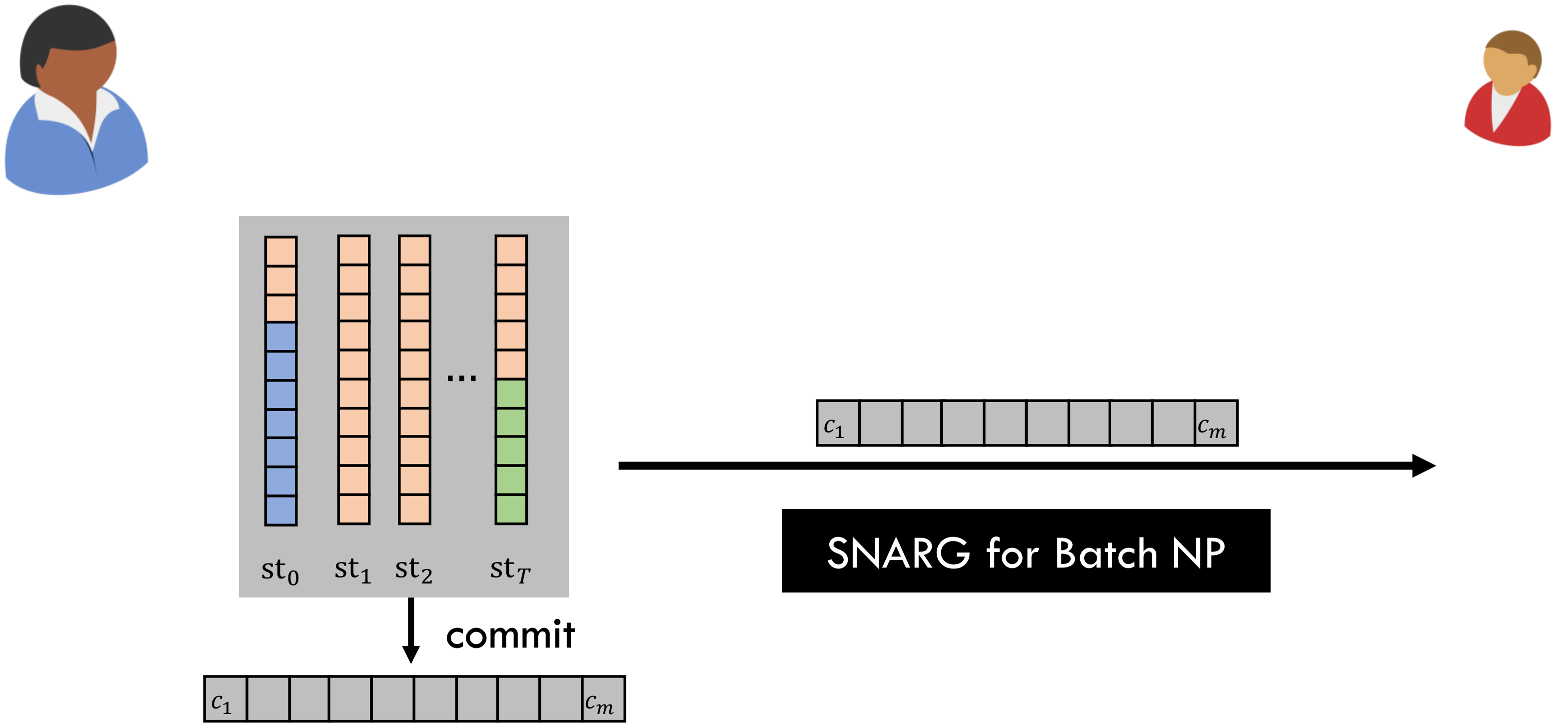


SNARG for Batch NP

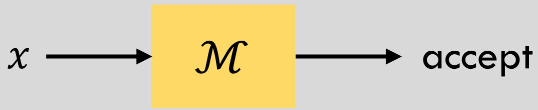
For every $i \in [0, \dots, T-1]$

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

SNARGs for Polynomial-time Computation



Verifier needs to read all the instances $\Omega(T)$



within T steps

SNARG for Batch NP

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

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

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

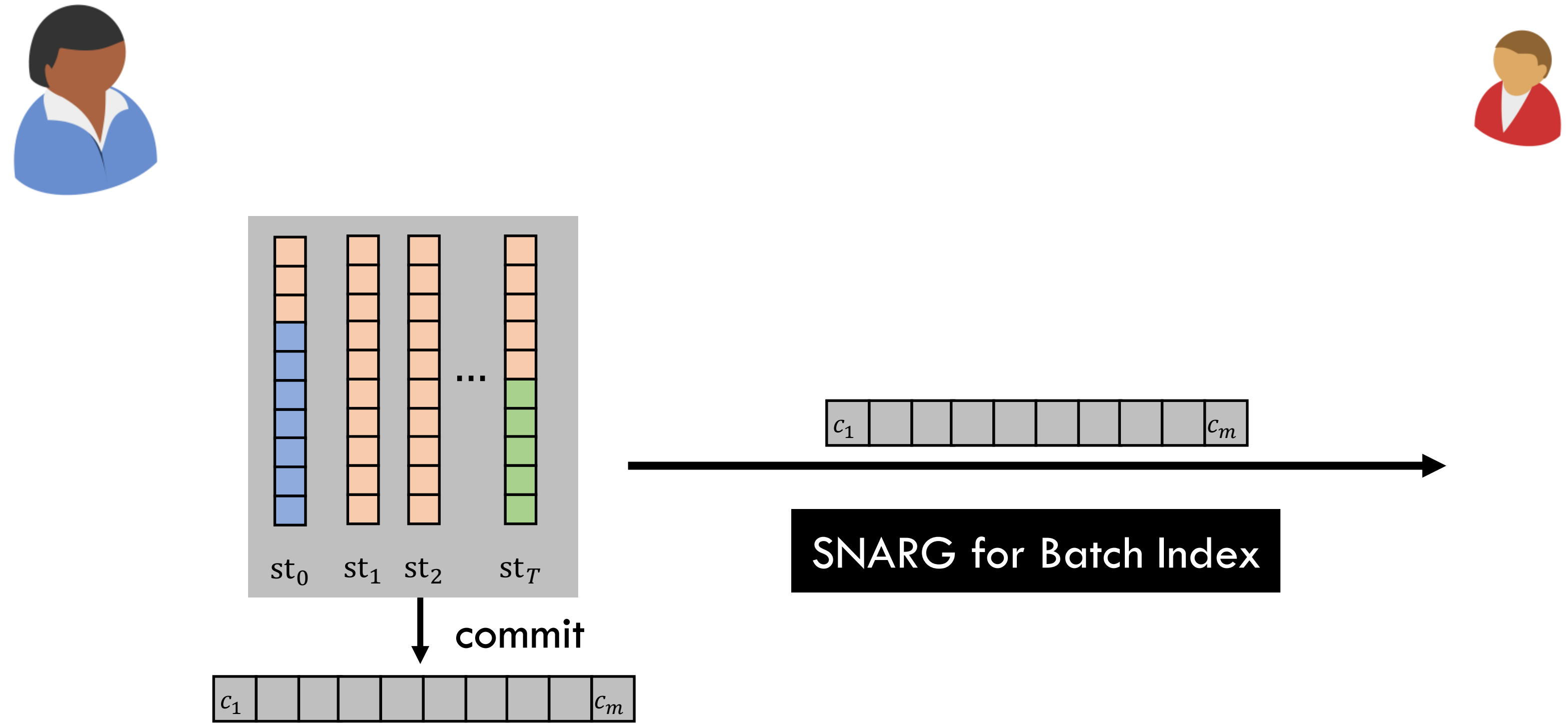
Π

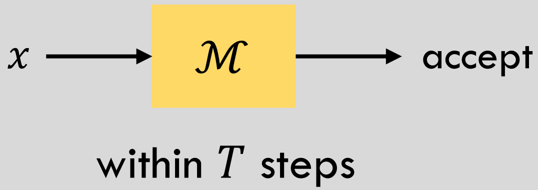
C, k

Verifier running time:

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

SNARGs for Polynomial-time Computation



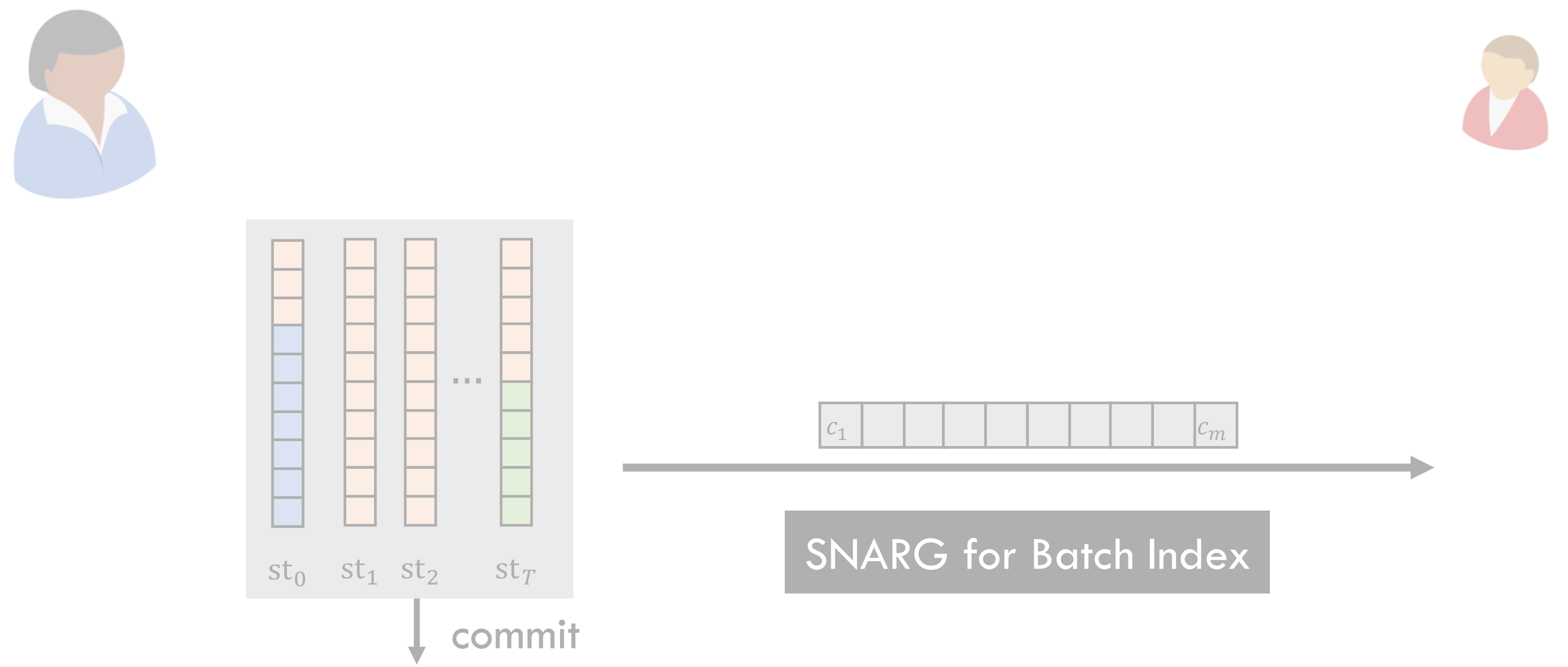


SNARG for Batch Index

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

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

SNARGs for Polynomial-time Computation

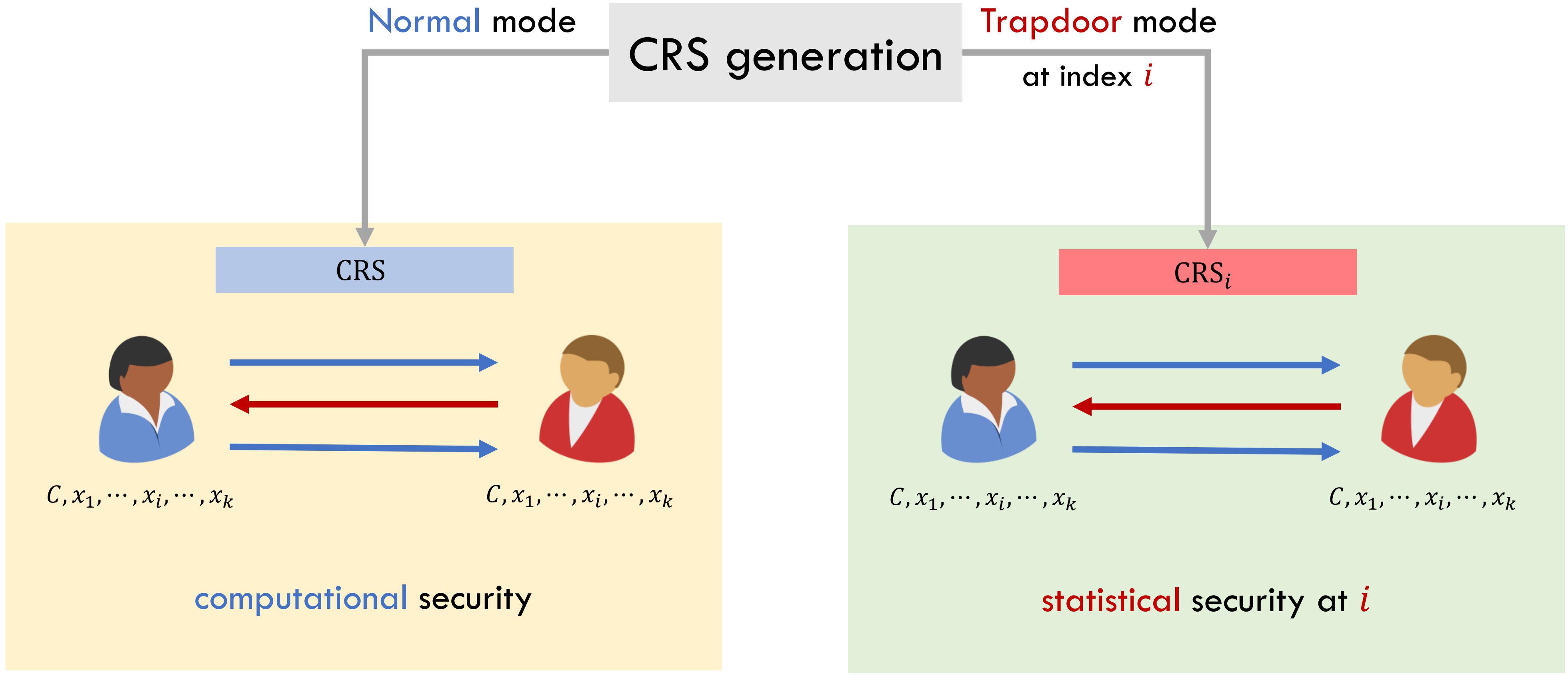


Further Challenges: Ensuring Global consistency across all states

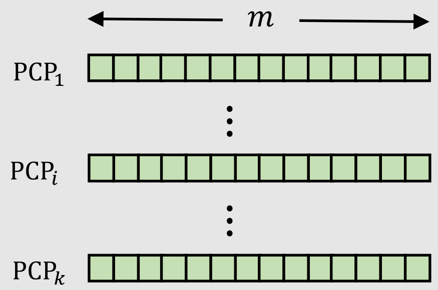
Use Somewhere Statistical Binding (SSB) Commitments [Hubáček -Wichs'15] and No-Signaling SSB Commitments [González-Zacharakis'21]

SNARGs for Batch Index

Dual-Mode Interactive Batch Arguments



Dual Mode Argument for Batch Index



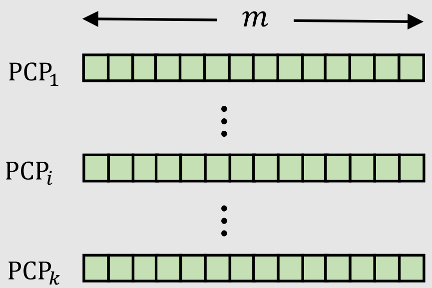
$L_C = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

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

Dual Mode Argument for Batch Index

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

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

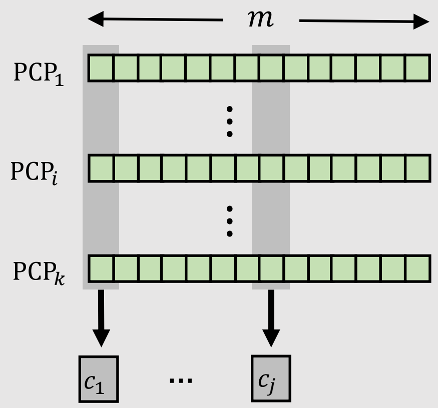


commitment key K

Dual Mode Argument for Batch Index

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

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

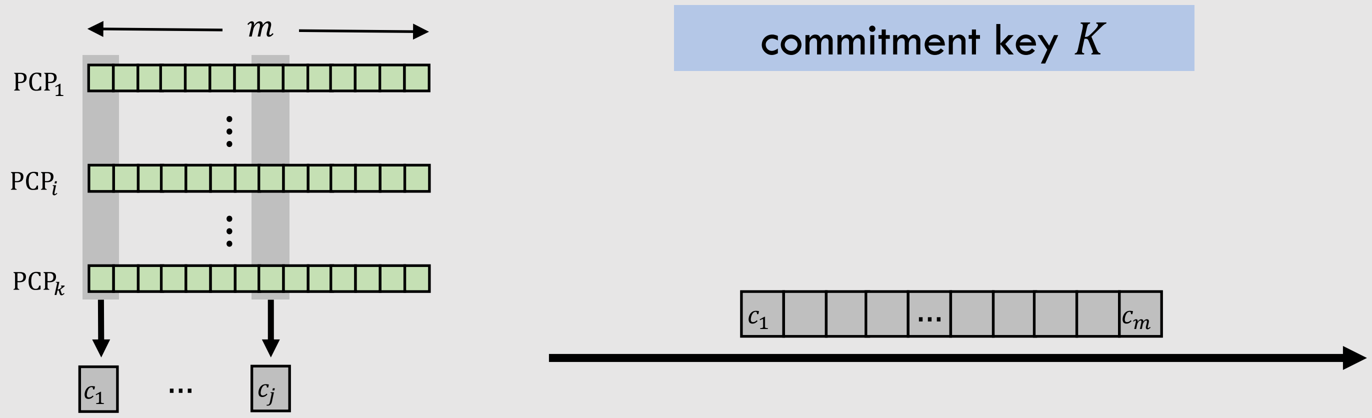


commitment key K

Dual Mode Argument for Batch Index

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

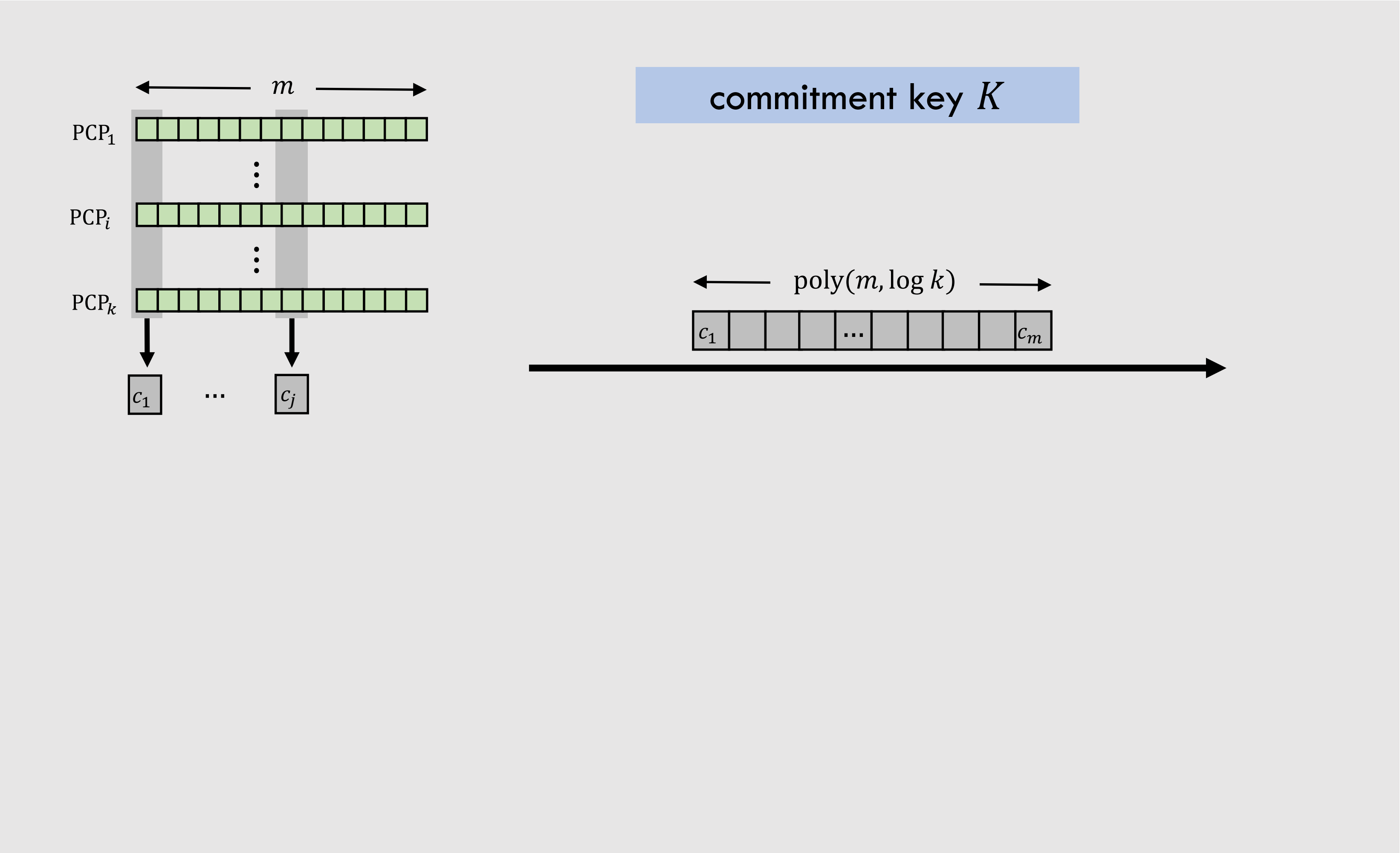
$\forall i \in [k], i \in L_C$



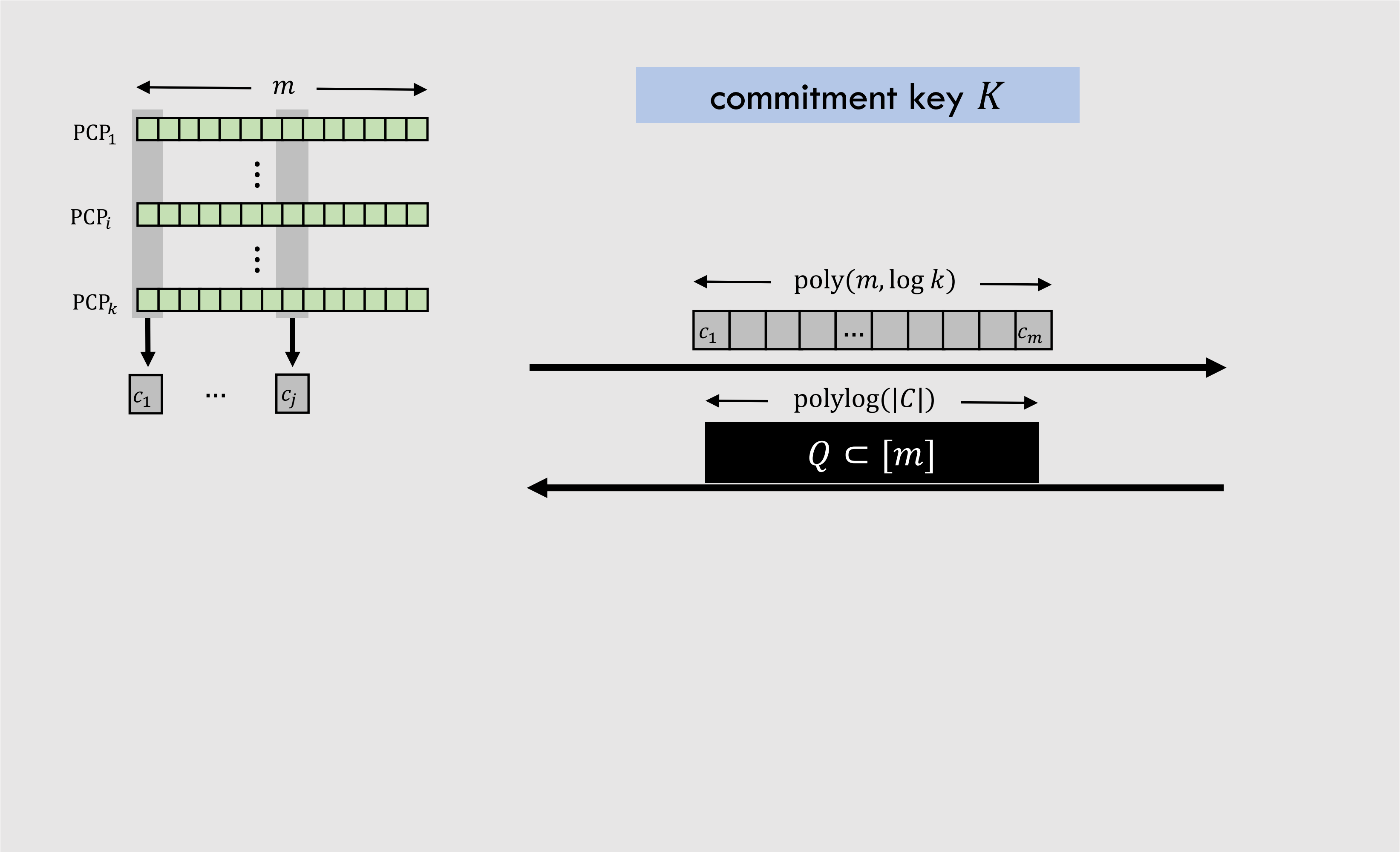
Dual Mode Argument for Batch Index

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

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



Dual Mode Argument for Batch Index



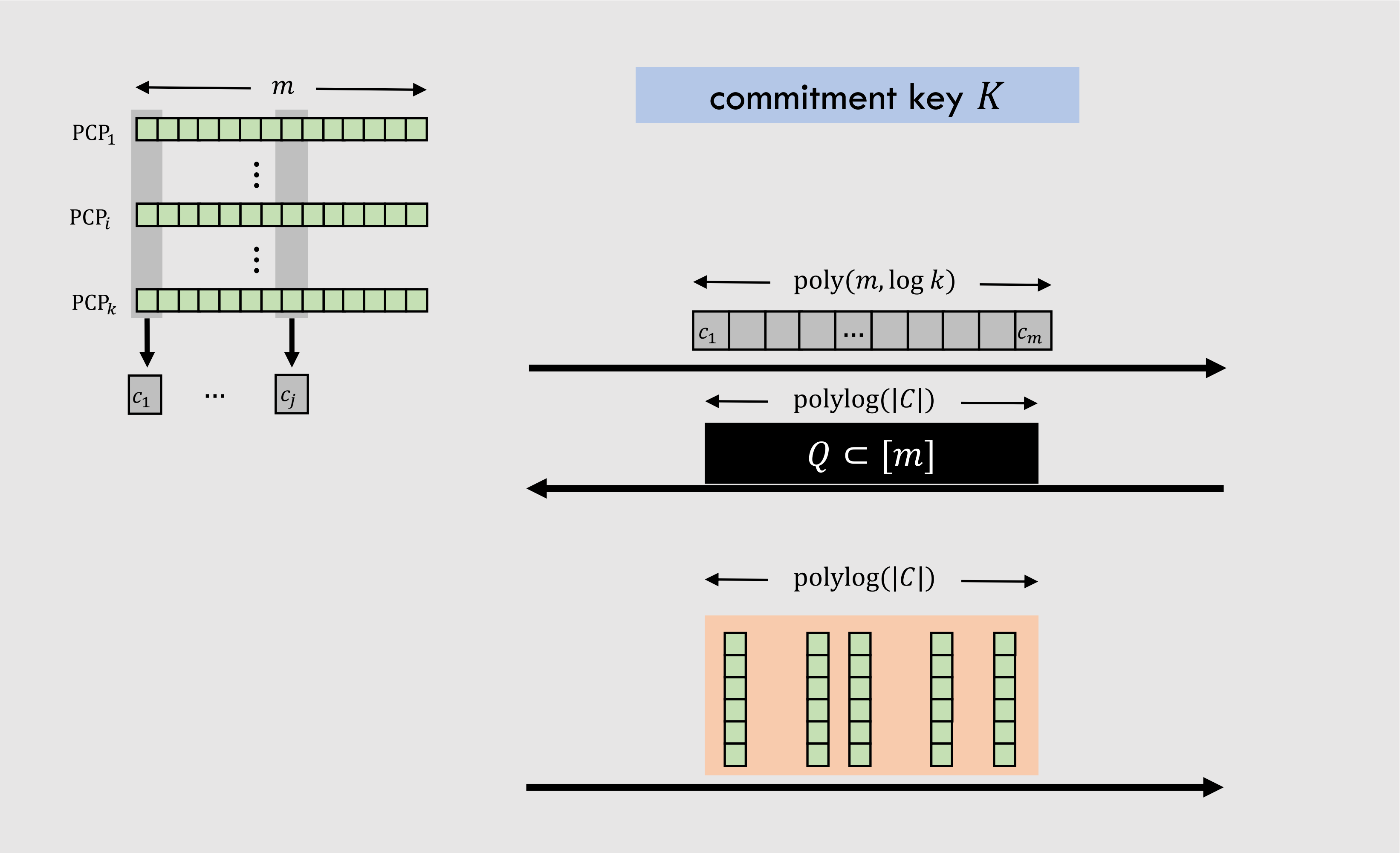
$L_C = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

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

Dual Mode Argument for Batch Index

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

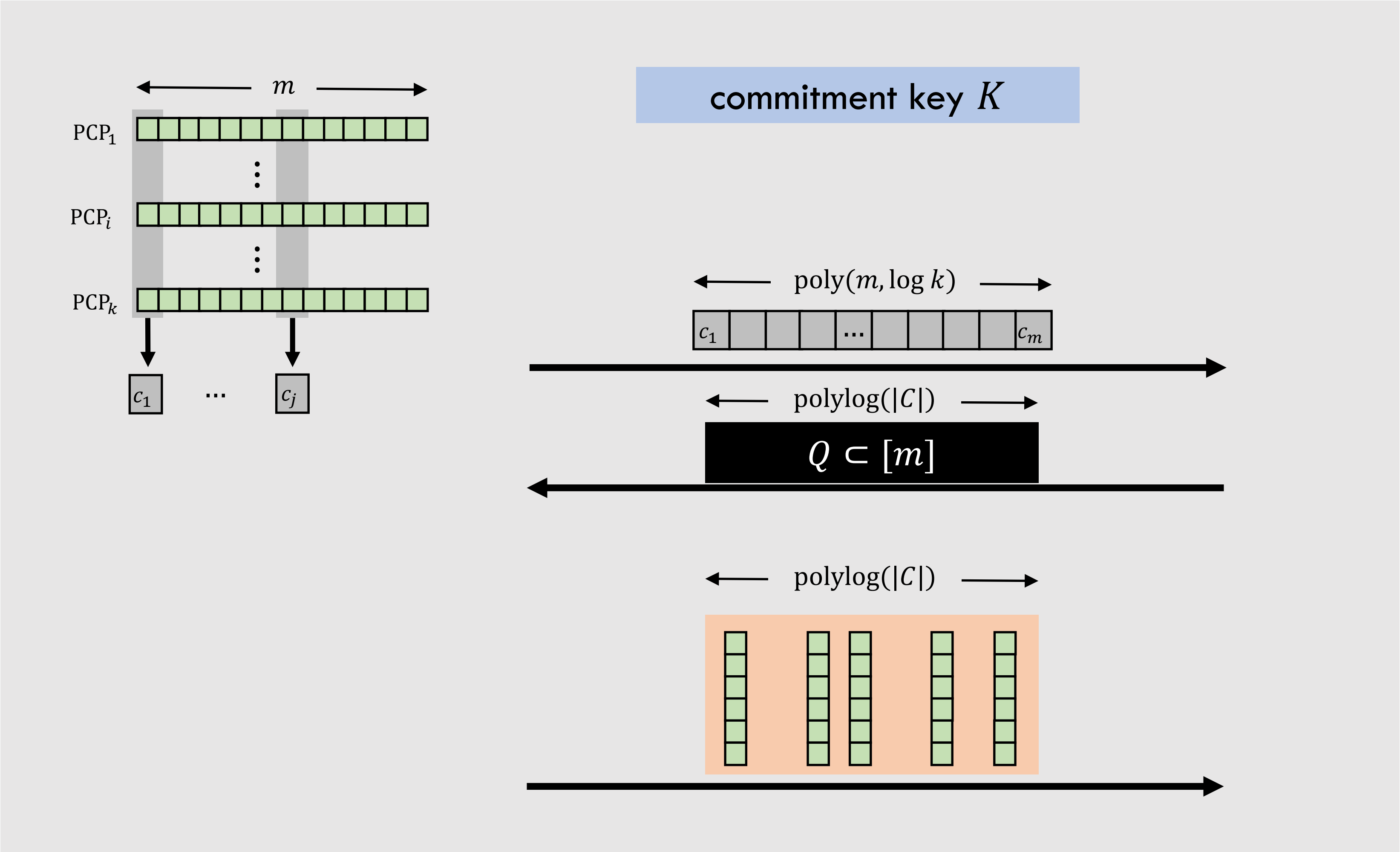
$\forall i \in [k], i \in L_C$



Dual Mode Argument for Batch Index

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

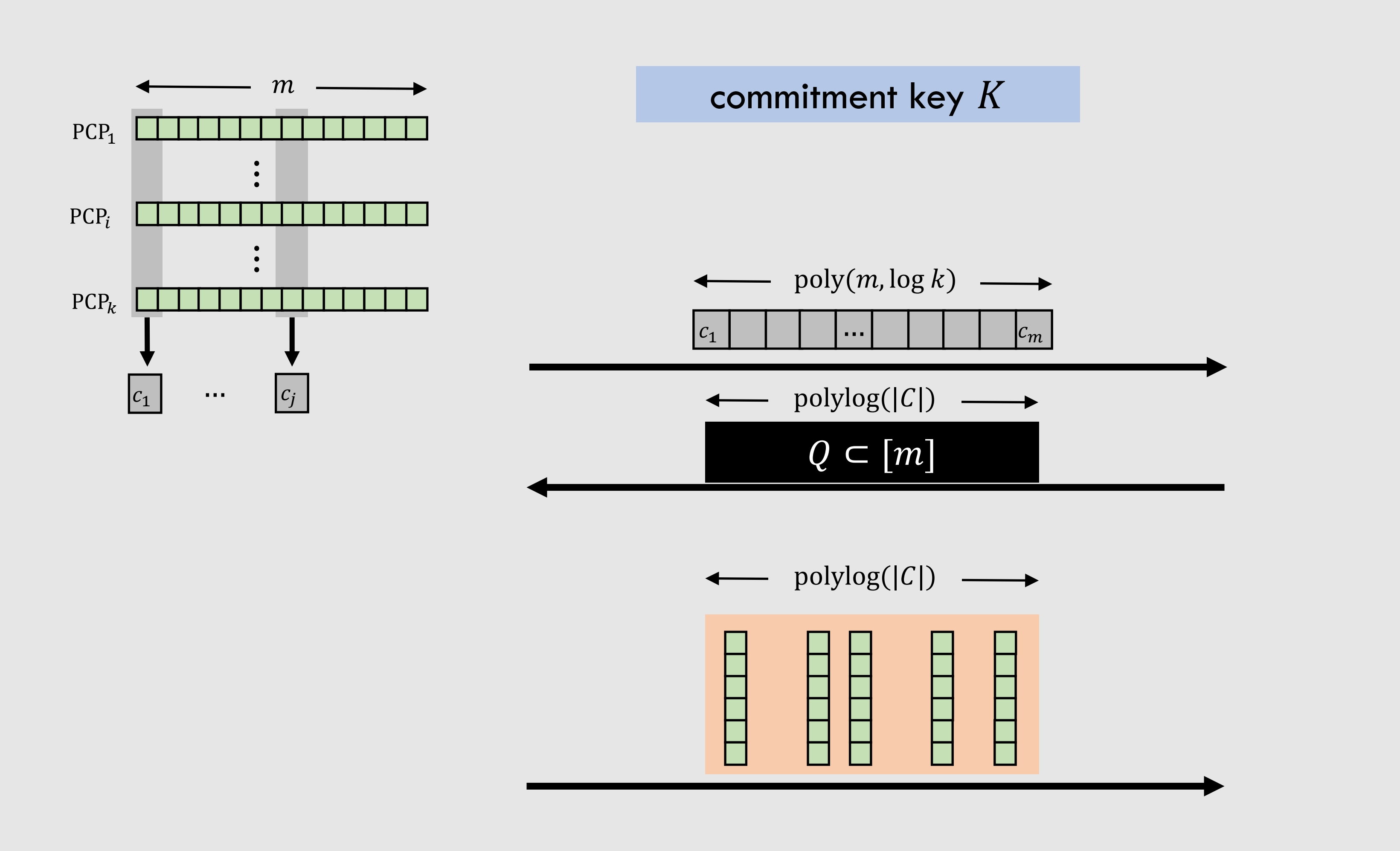
$\forall i \in [k], i \in L_C$



Verify:

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

Dual Mode Argument for Batch Index



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

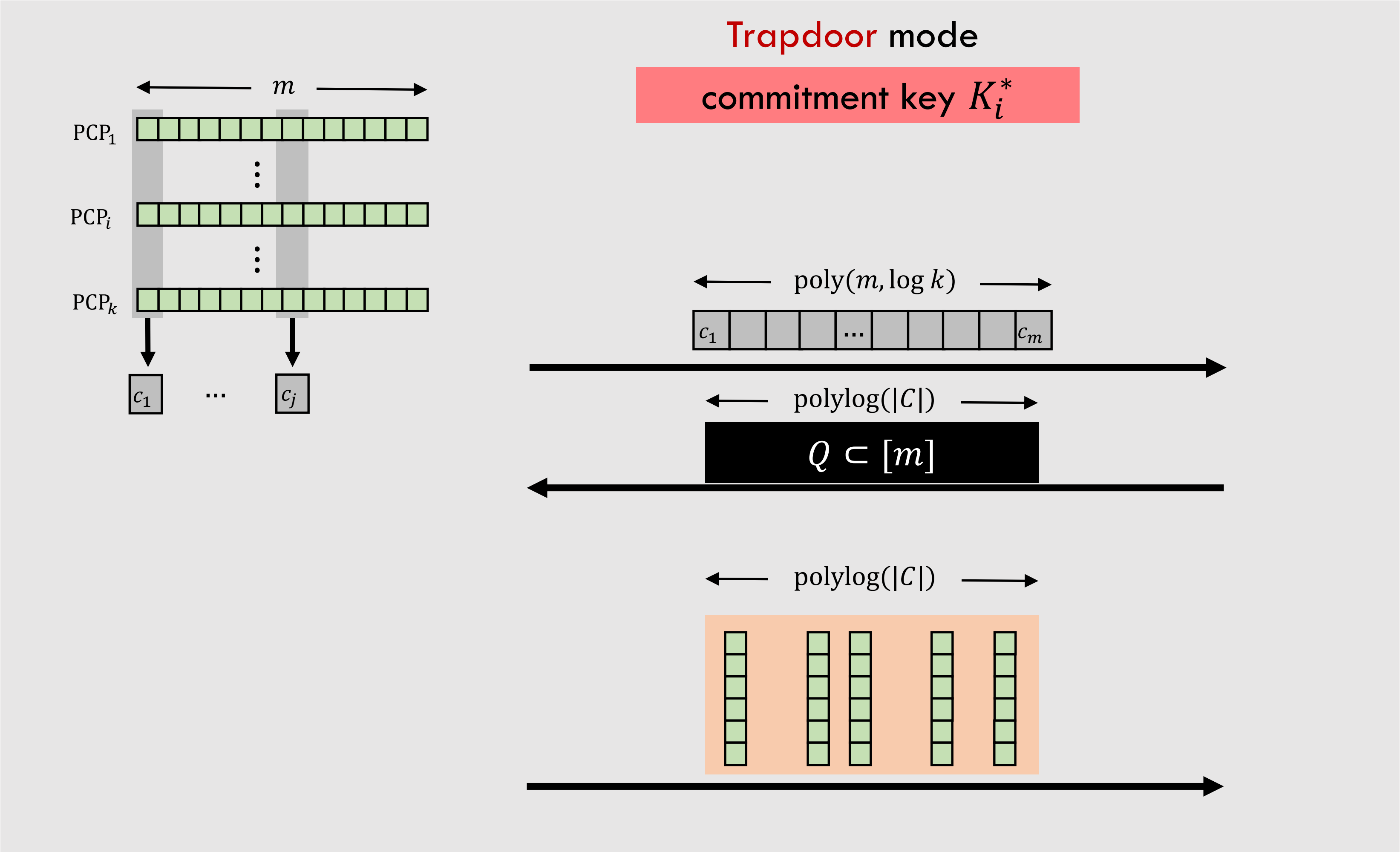
$\forall i \in [k], i \in L_C$

Somewhere Statistically Binding (SSB) Commitment Scheme

Verify:

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

Dual Mode Argument for Batch Index



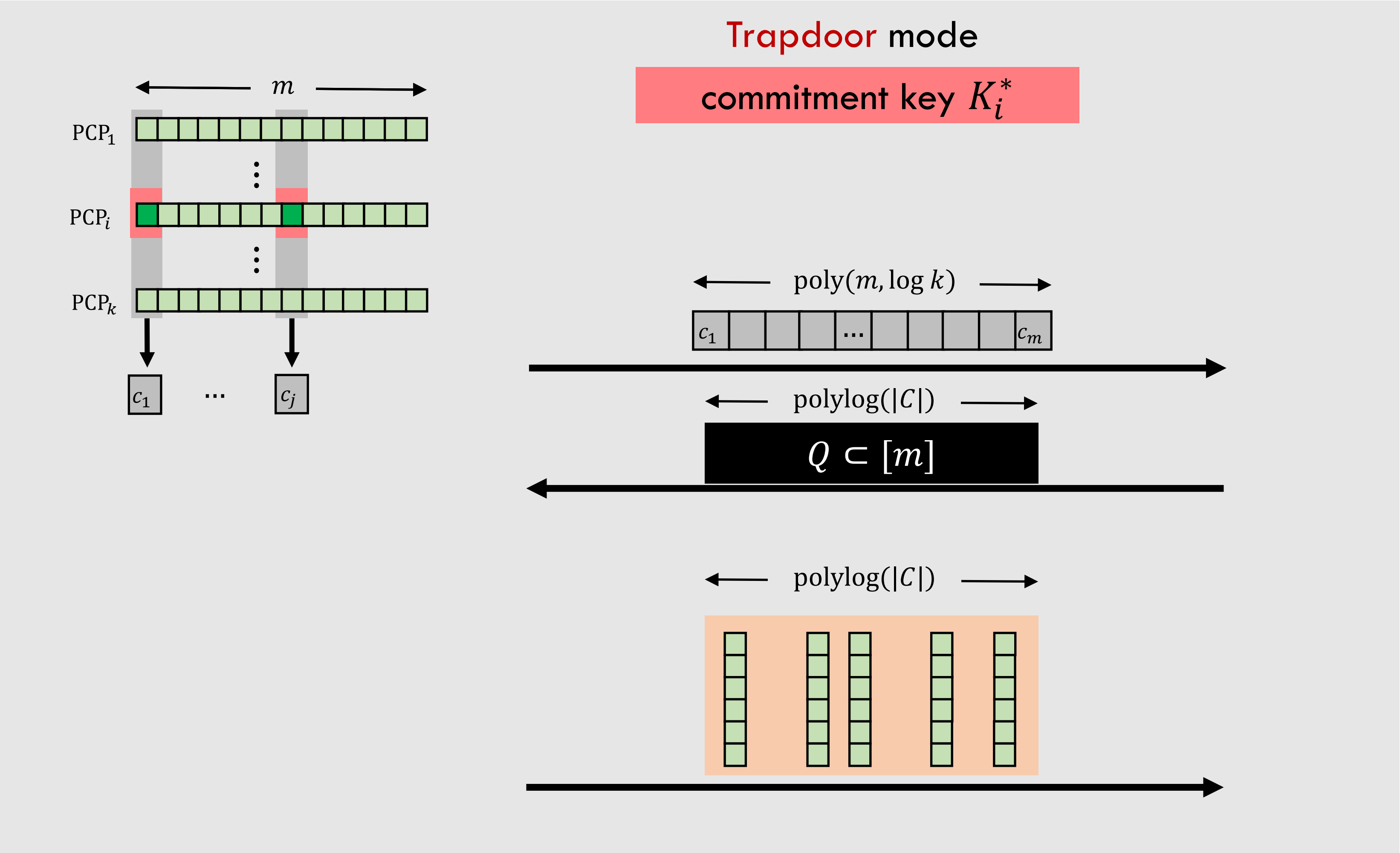
$L_C = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

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

Somewhere Statistically Binding (SSB) Commitment Scheme

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

Dual Mode Argument for Batch Index



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

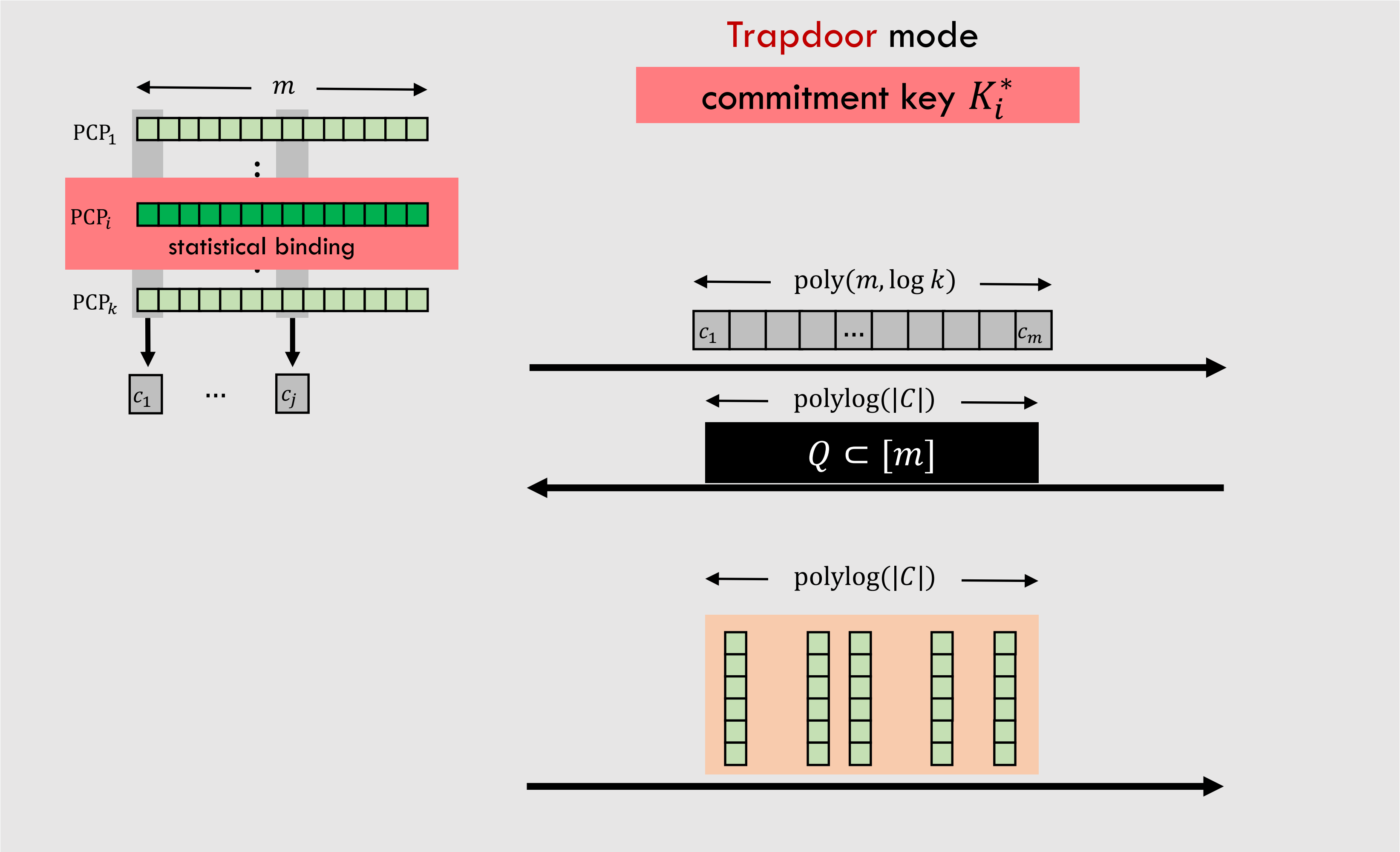
$\forall i \in [k], i \in L_C$

Somewhere Statistically Binding (SSB) Commitment Scheme

Verify:

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

Dual Mode Argument for Batch Index



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

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

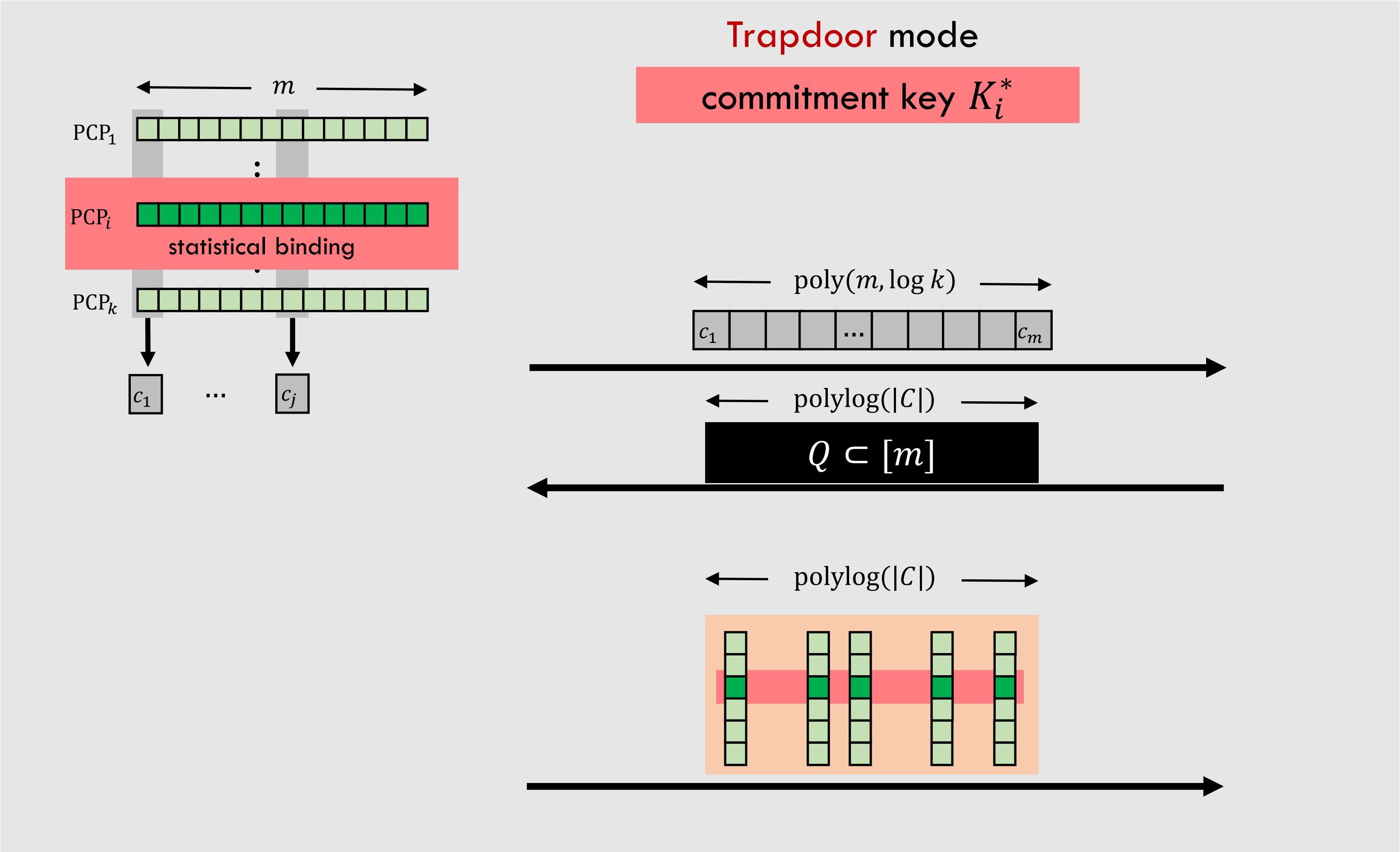
Verify:

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

Dual Mode Argument for Batch Index

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

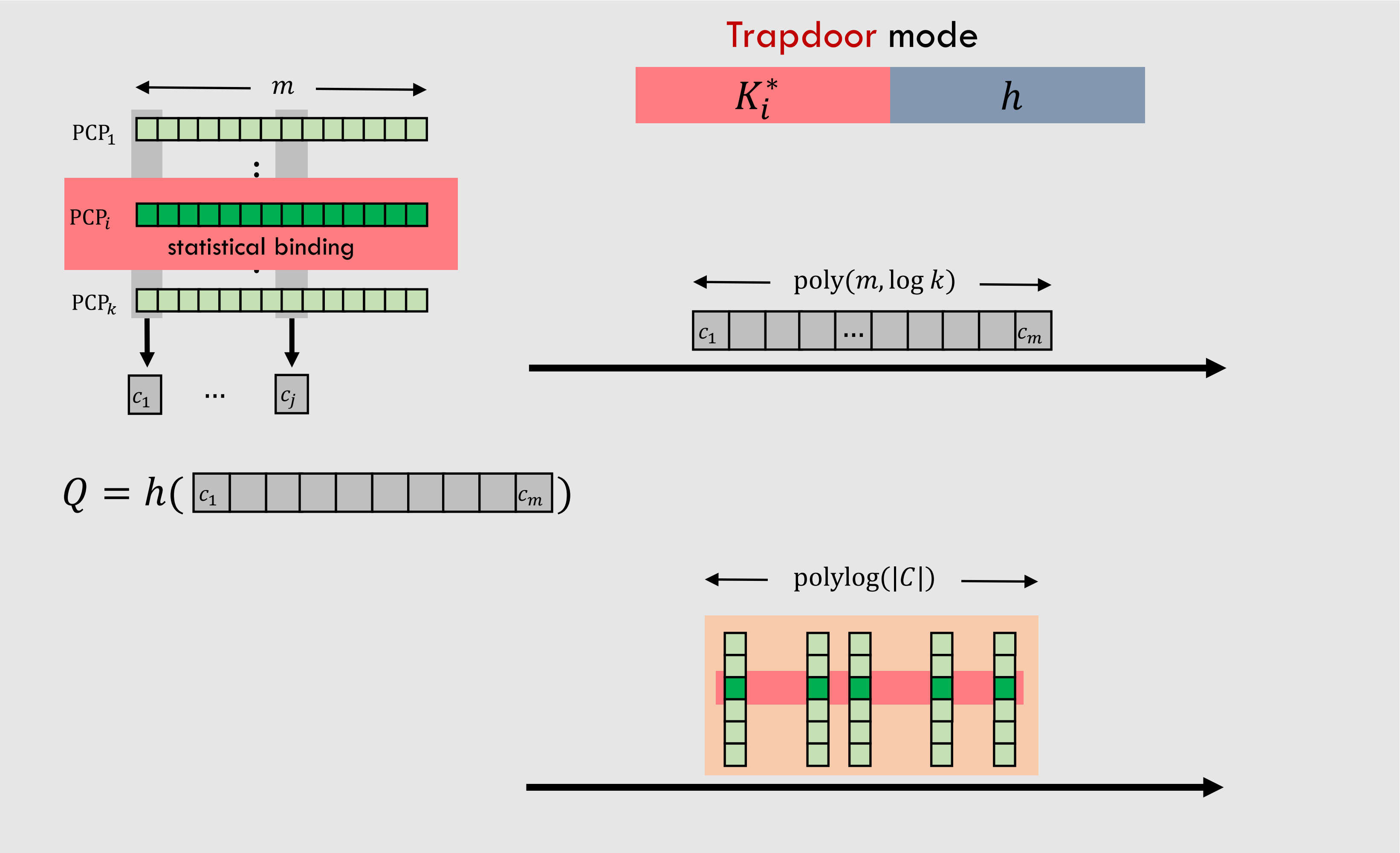
$\forall i \in [k], i \in L_C$



Verify:

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

Dual Mode Argument for Batch Index



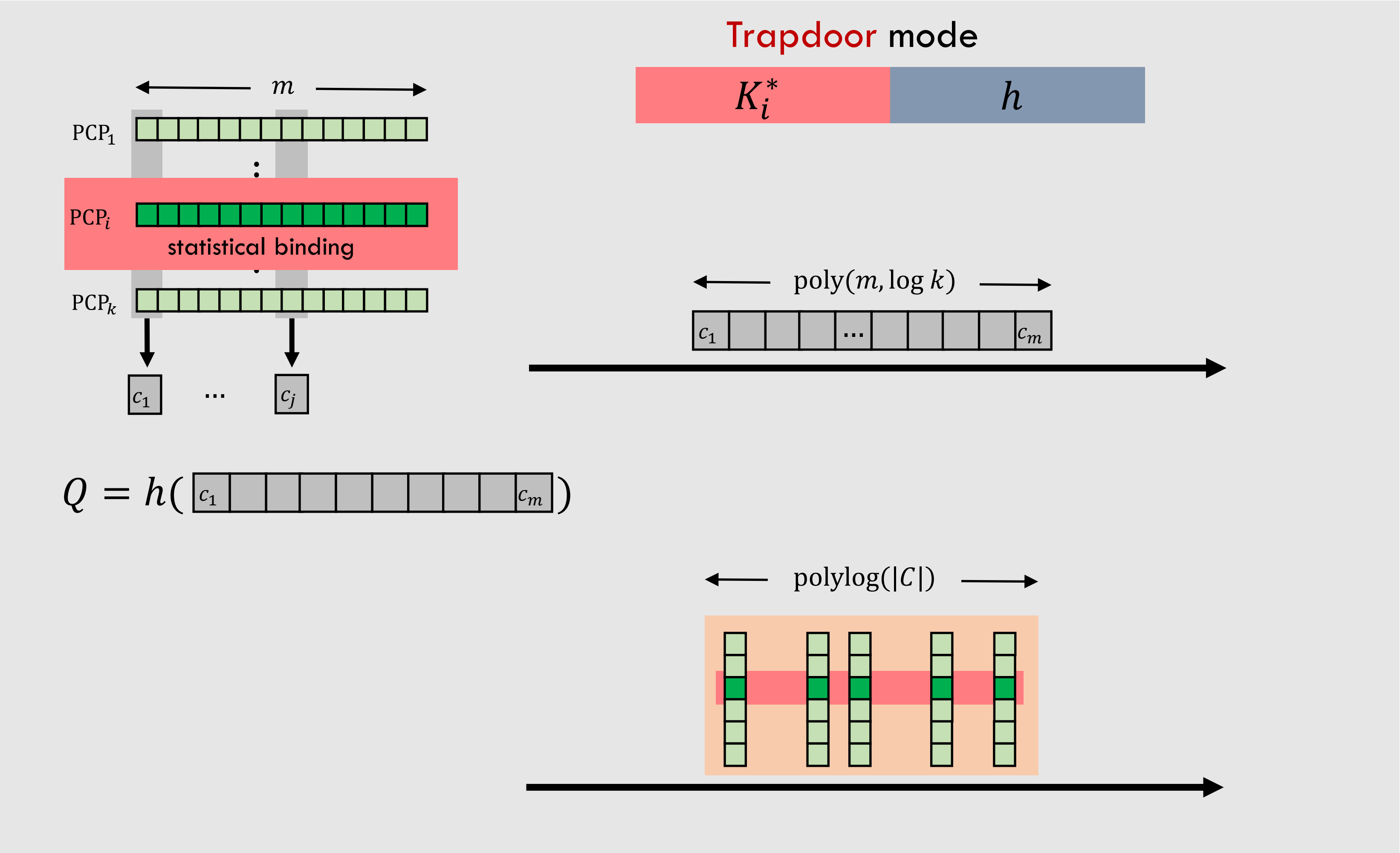
$L_C = \{i \mid \exists w \text{ s.t. } C(i,w) = 1\}$

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

Verify:

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

Dual Mode Argument for Batch Index



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

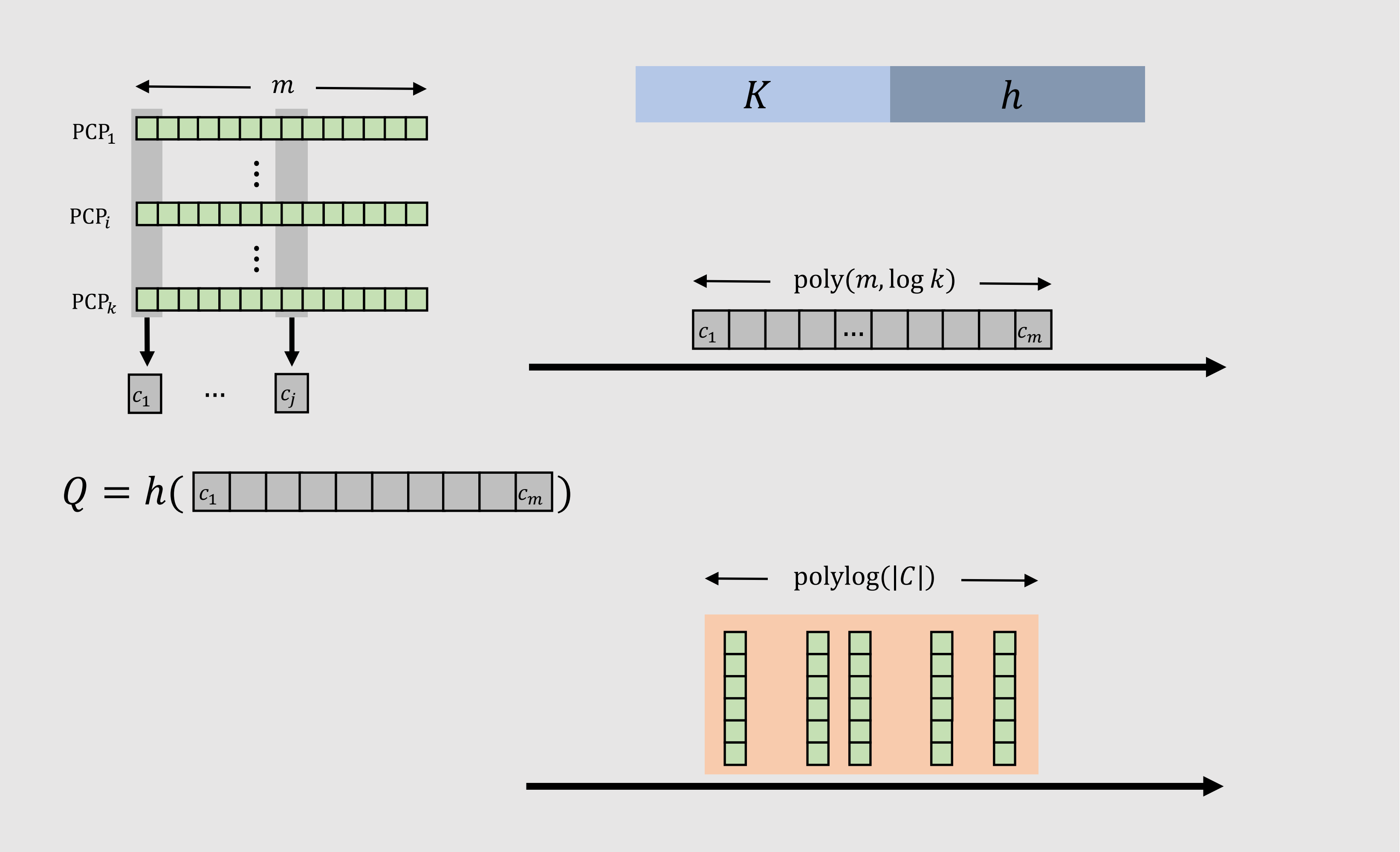
$\forall i \in [k], i \in L_C$

[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



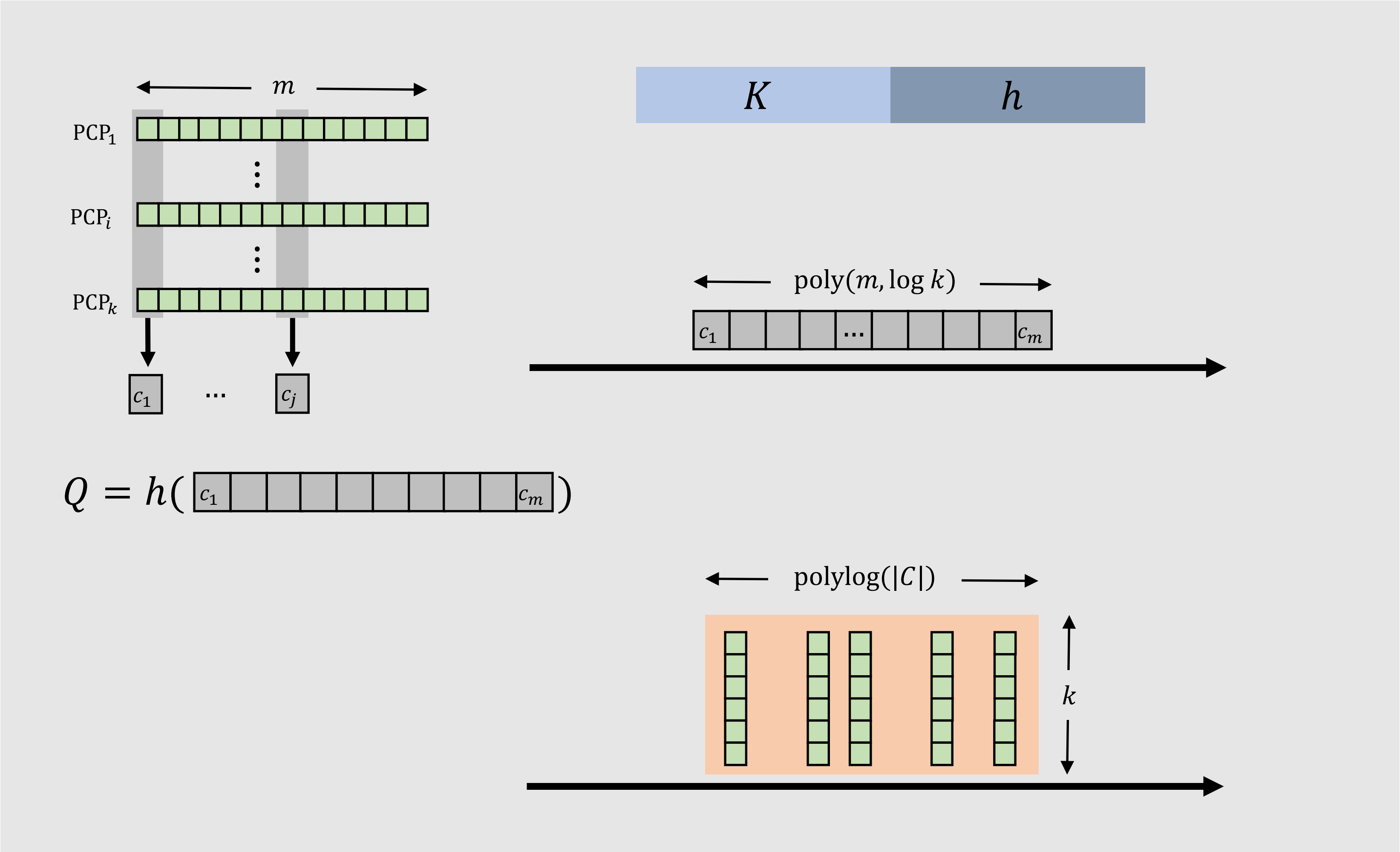
$L_C \; = \{i \mid \exists w \; s.t. \; C(i,w) = 1\}$

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

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

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

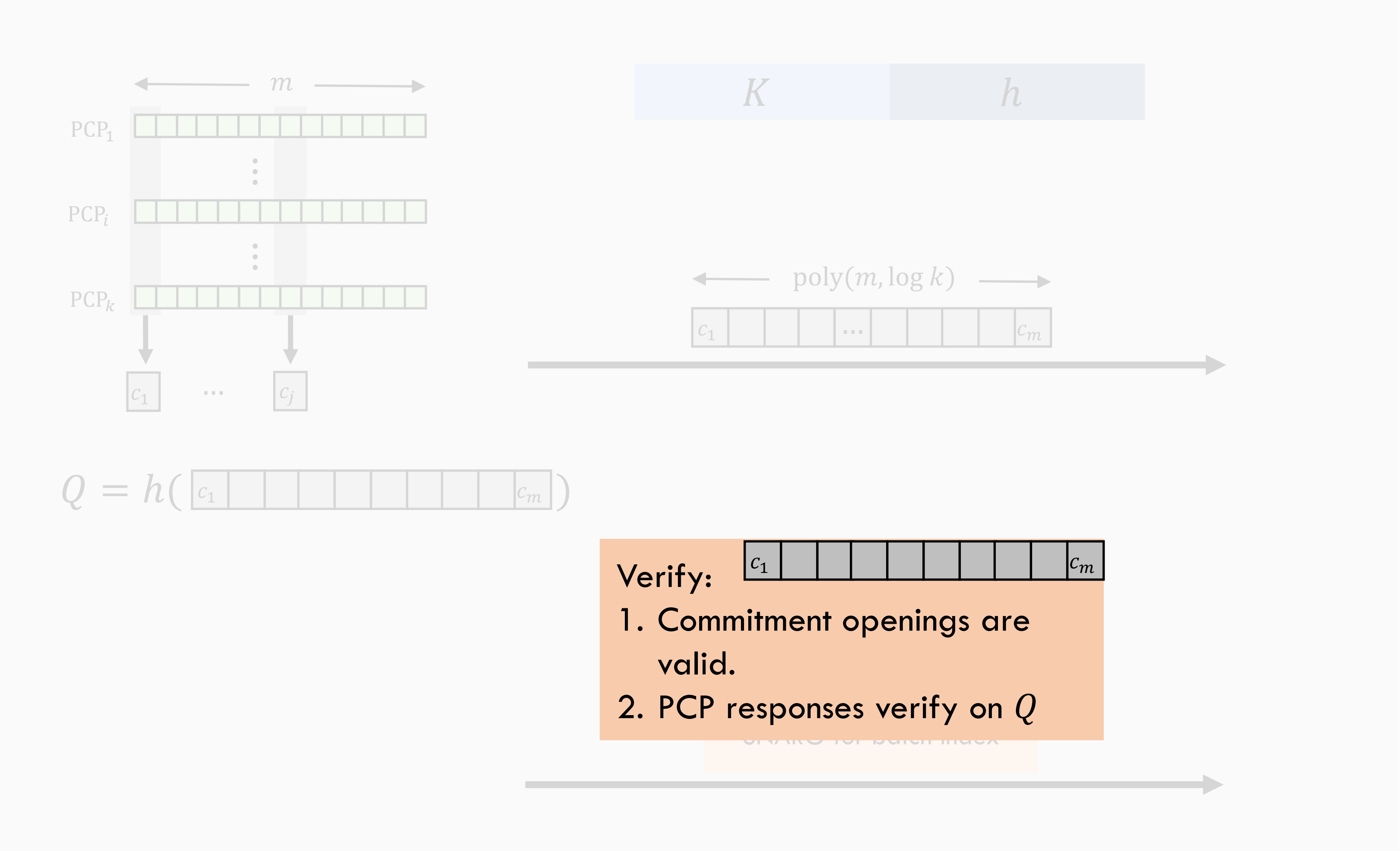
$\forall i \in [k], i \in L_C$

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

Verify:

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

SNARG for Batch Index



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

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

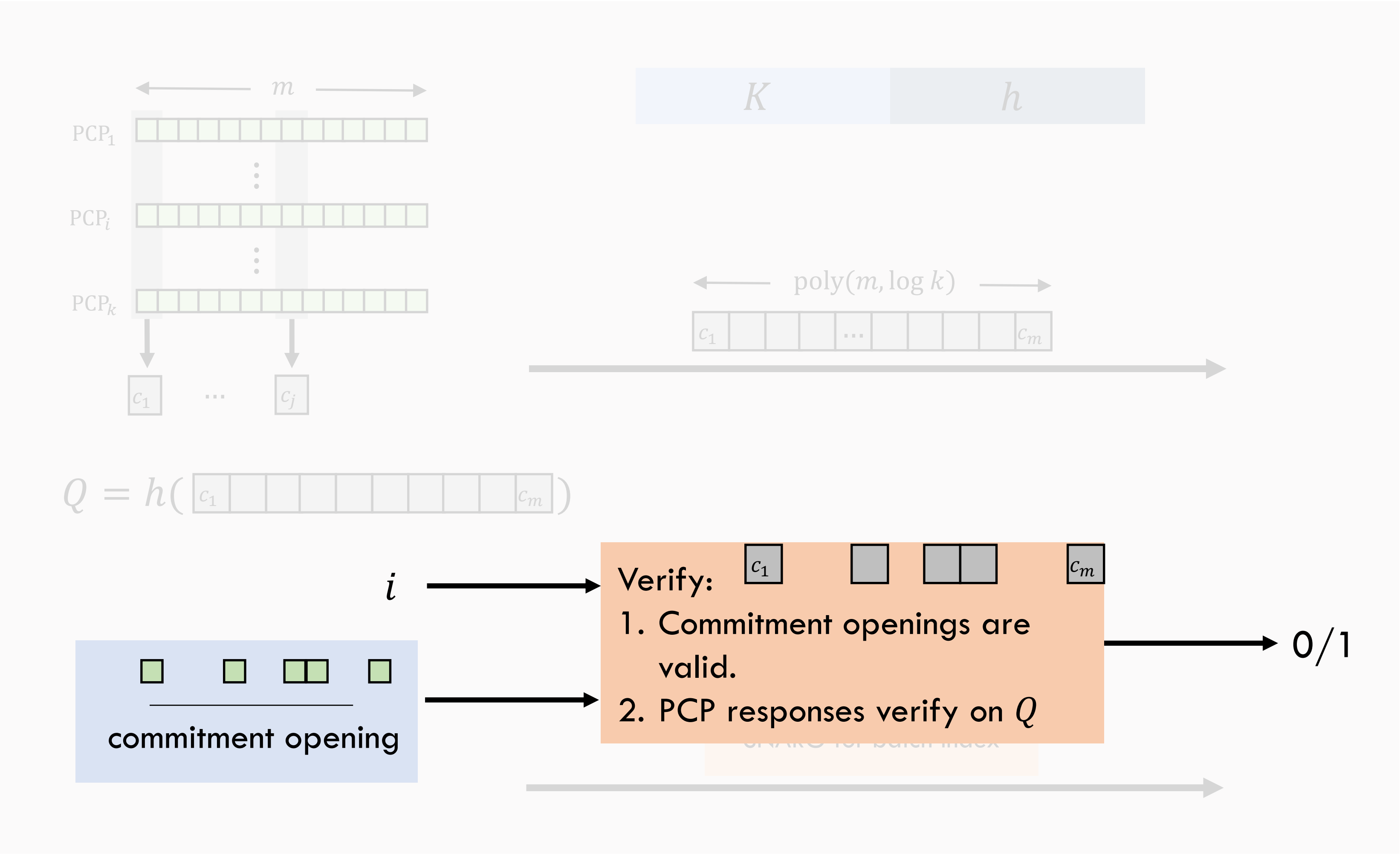
$\forall i \in [k], i \in L_C$

|  | $= \Omega(k)$

Verify: c_1 c_m

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

SNARG for Batch Index

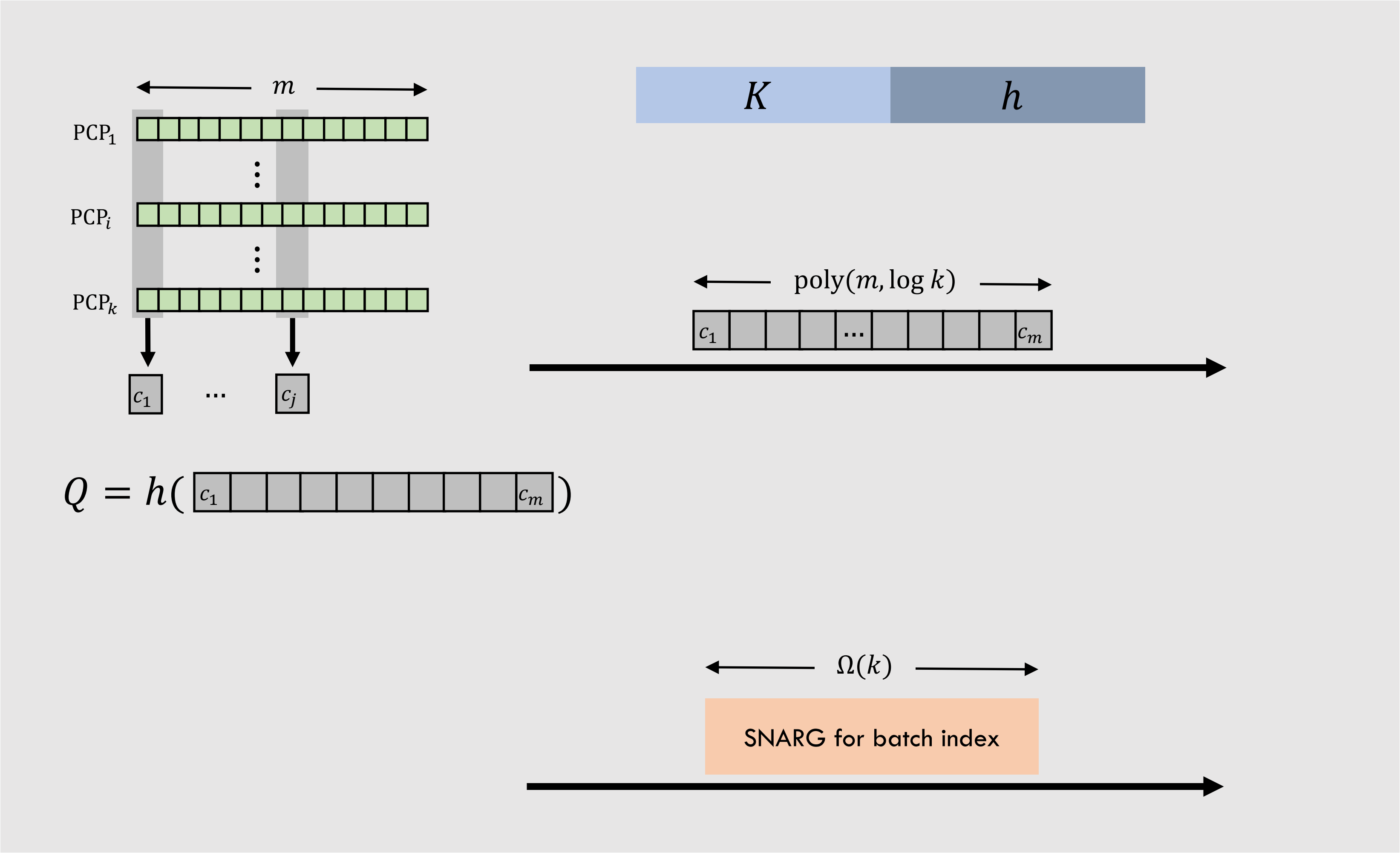


$L_C = \{i \mid \exists w \text{ s.t. } C(i,w) = 1\}$

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

|  | $= \Omega(k)$

SNARG for Batch Index



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

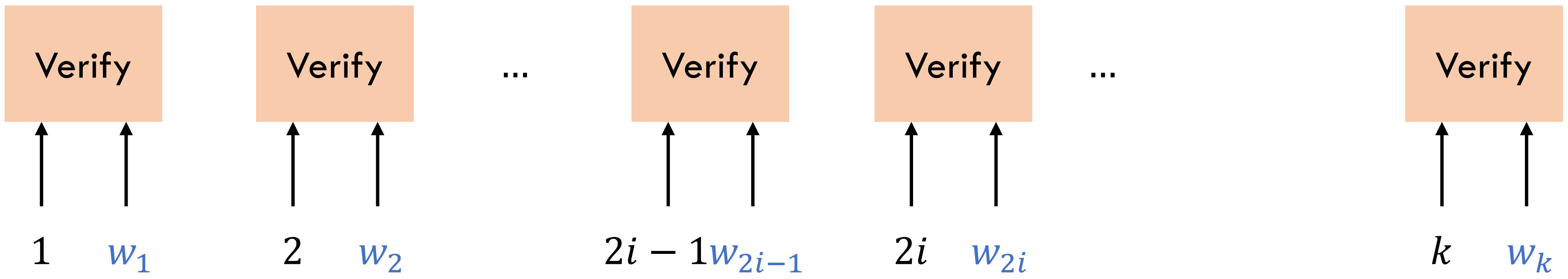
$\forall i \in [k], i \in L_C$

|  | $= \Omega(k)$

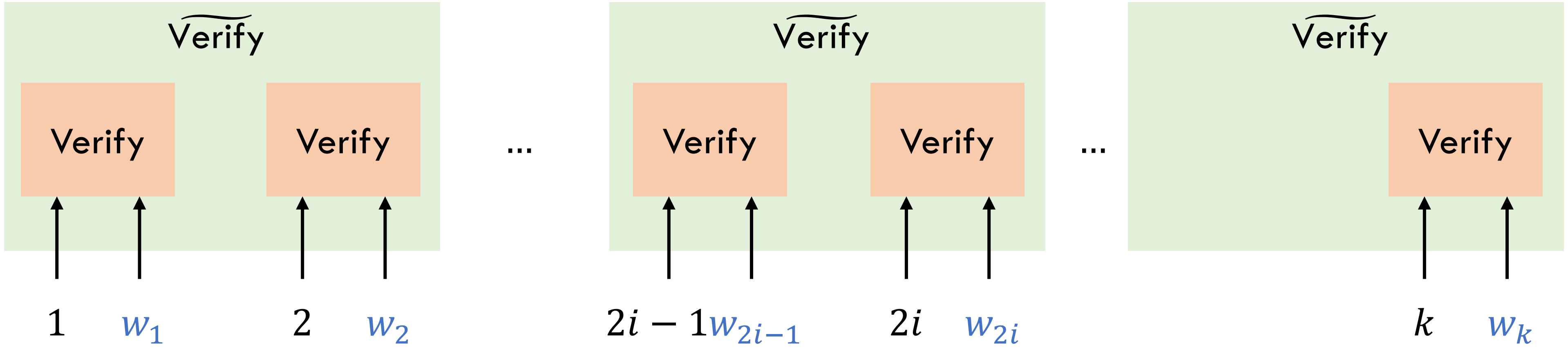
Verify: c_1 c_m

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

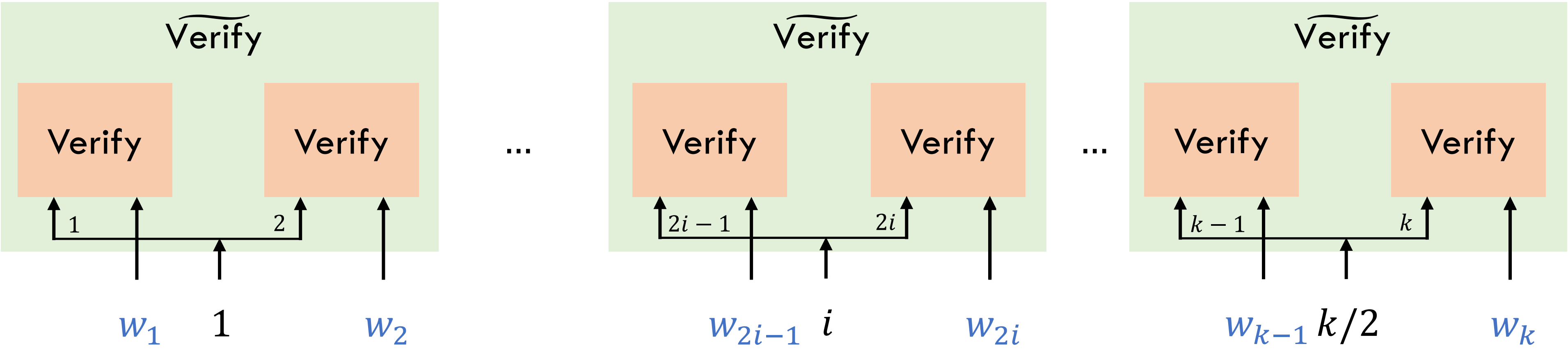
Grouping Instances for Recursion



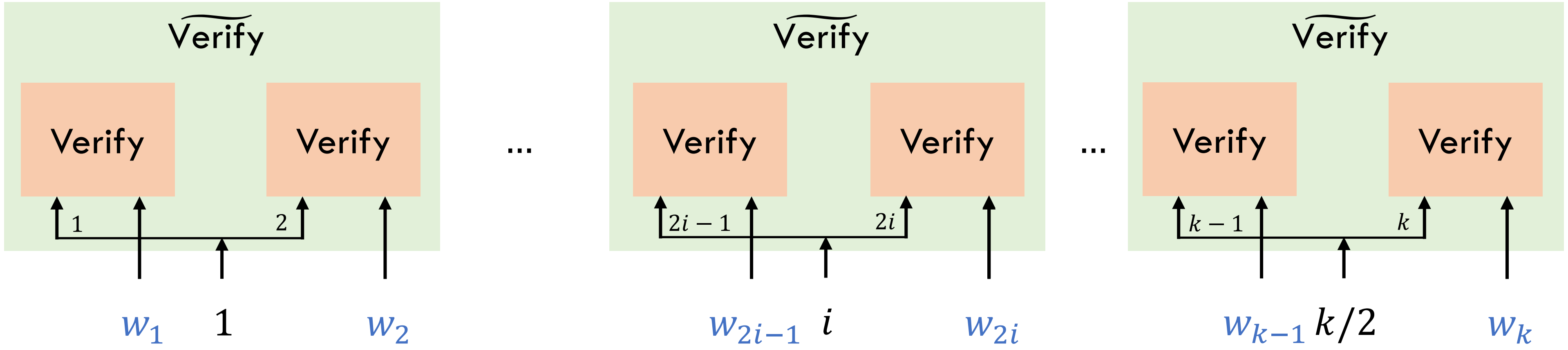
Grouping Instances for Recursion



Grouping Instances for Recursion



Grouping Instances for Recursion



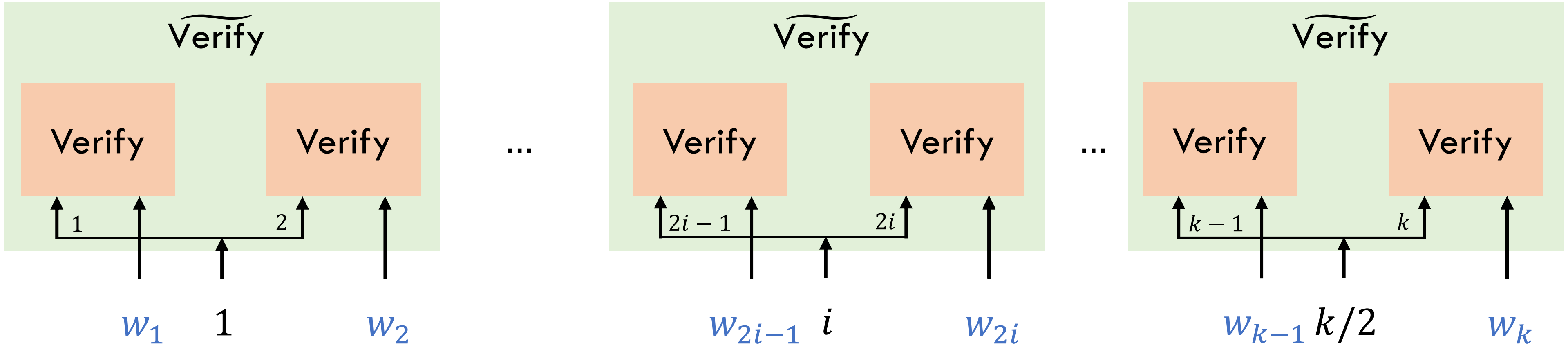
$k/2$ instances for circuit $\widetilde{\text{Verify}}$

Grouping Instances for Recursion

Recurse $\log(k)$ times

Further Challenges: Prevent exponential growth in $\widetilde{\text{Verify}}$

- Tool: PCP with Fast Online verification



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

Recap

LWE → SNARGs for Batch NP → SNARGs for polynomial time computation.

Dual Mode Interactive Batch Arguments + Fiat-Shamir instantiation via CIH

No Signaling SSB Commitments

Thank you. Questions?

Arka Rai Choudhuri

arkarc@berkeley.edu

ia.cr/2021/808