Correlation Intractability and SNARGs from Sub-exponential DDH

Arka Rai Choudhuri
NTT Research

Sanjam Garg
UC Berkeley and NTT Research

Abhishek Jain
Johns Hopkins University and NTT Research

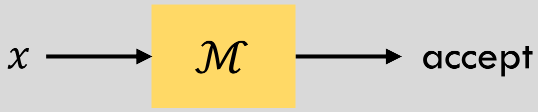
Zhengzhong Jin
MIT

Jiaheng Zhang
UC Berkeley

Succinct Non-Interactive Arguments (SNARGs)

$\mathcal{M}$, x

$\mathcal{M}$, x



Succinct Non-Interactive Arguments (SNARGs)

$\mathcal{M}$, x

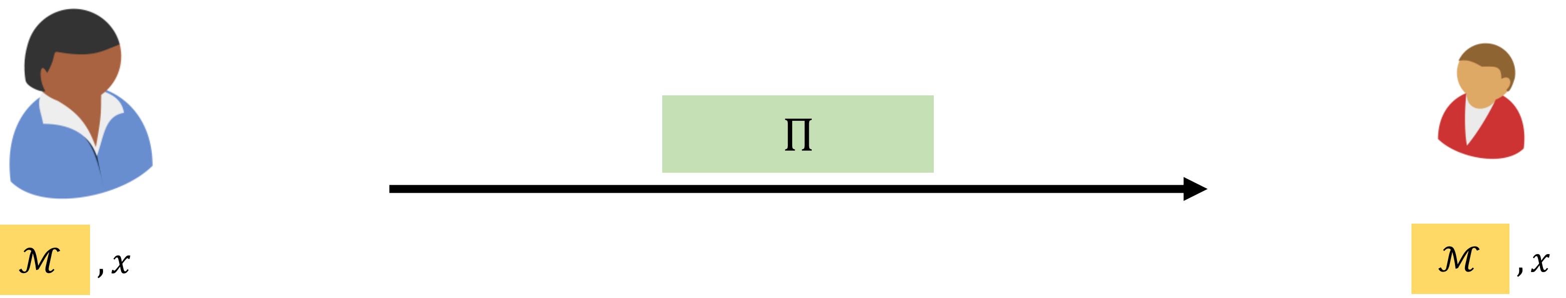
$\mathcal{M}$, x

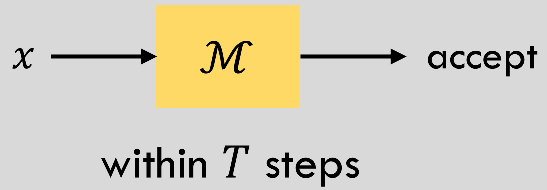
$x \longrightarrow \mathcal{M} \longrightarrow$ accept

within T steps

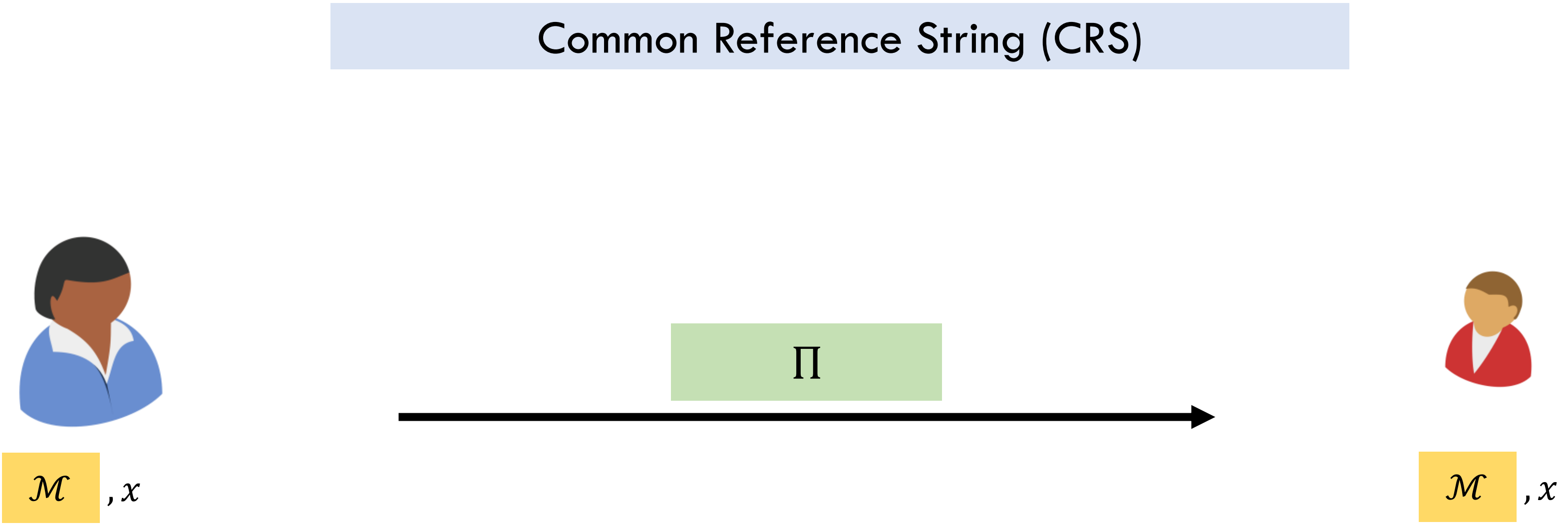
 wants to delegate computation to 

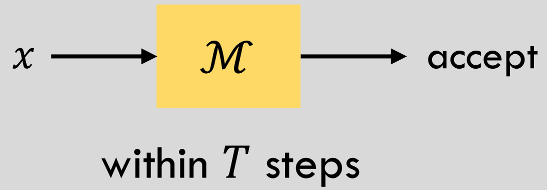
Succinct Non-Interactive Arguments (SNARGs)





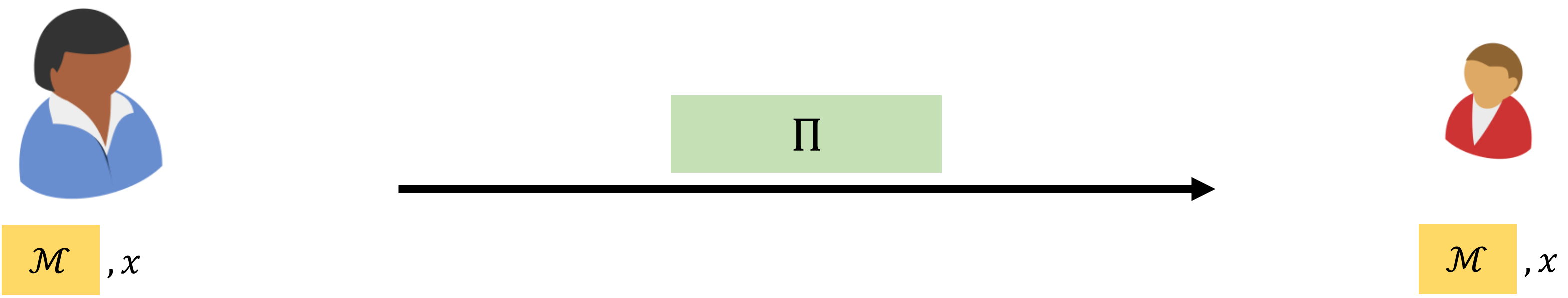
Succinct Non-Interactive Arguments (SNARGs)





Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)



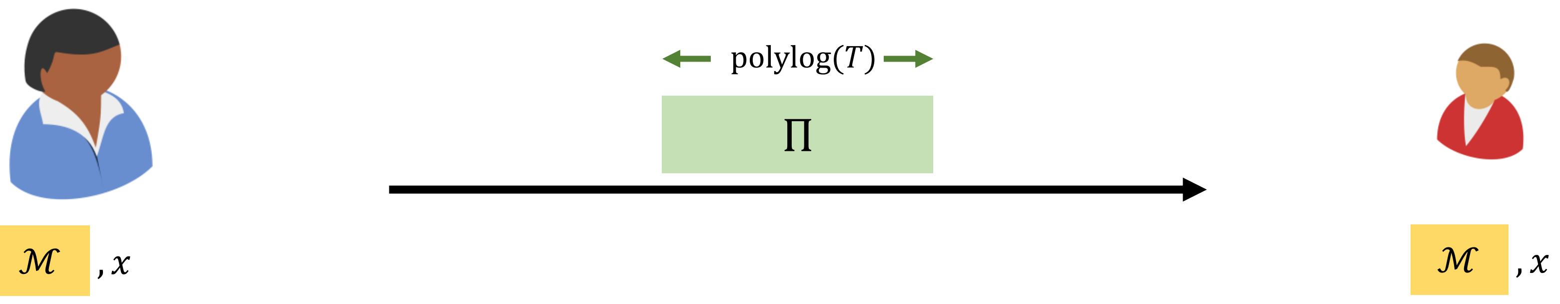
Π is publicly verifiable

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

within T steps

Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)



Verifier running time:
polylog(T)

Π is publicly verifiable

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

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

Π

$\mathcal{M}$, x

$\mathcal{M}$, x

Verifier running time: polylog(T)

Π is publicly verifiable

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

within T steps

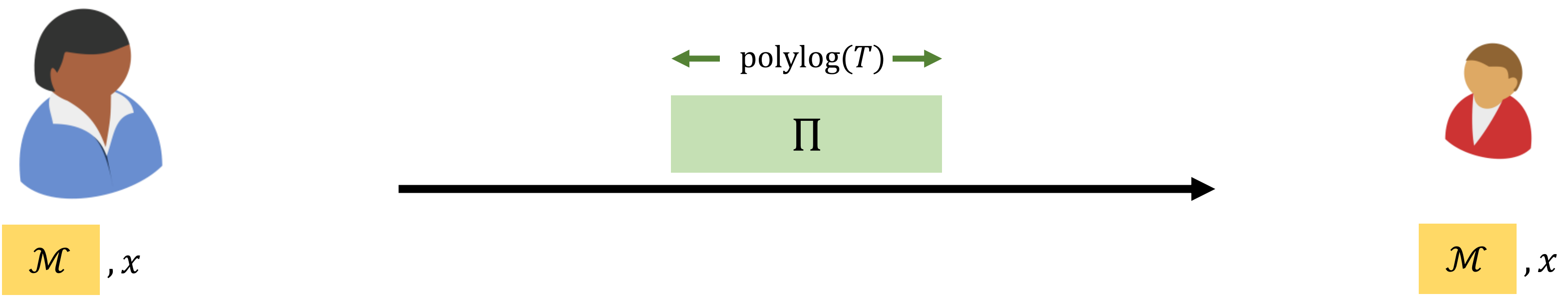
No PPT can produce accepting x, Π if

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

within T steps

Succinct Non-Interactive Arguments (SNARGs)

Common Reference String (CRS)



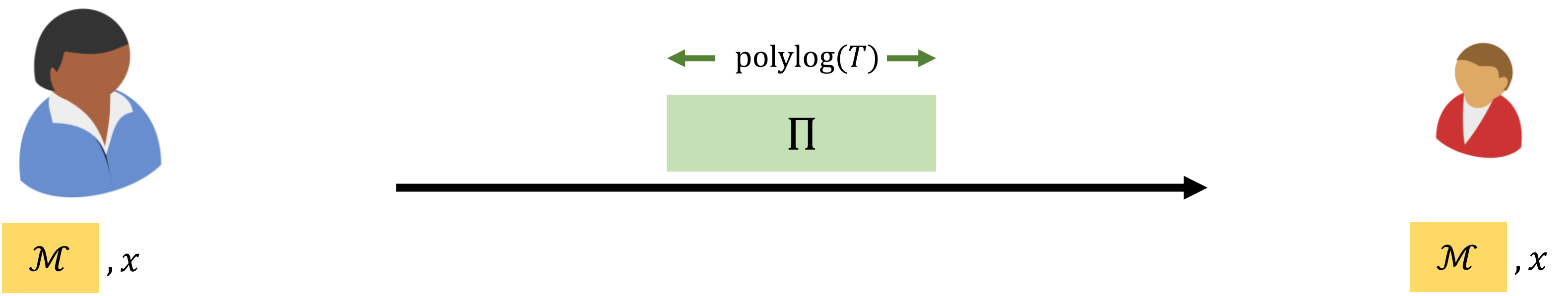
Verifier running time: polylog(T)

Π is publicly verifiable

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

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]

SNARGs for Batch NP (or BARGs)

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 (or BARGs)

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 (or BARGs)

CRS

$C, x_1, \cdots, x_k$

Π

$C, x_1, \cdots, x_k$

Π is publicly verifiable

No PPT  can produce accepting Π if

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

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

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

SNARGs for Batch NP (or BARGs)

CRS

$C, x_1, \cdots, x_k$

Π

$C, x_1, \cdots, x_k$

Π is publicly verifiable

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

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

SNARGs for Batch NP (or BARGs)

CRS

$C, x_1, \cdots, x_k$

$w_1, \dots, w_k$ →

$C, x_1, \cdots, x_k$

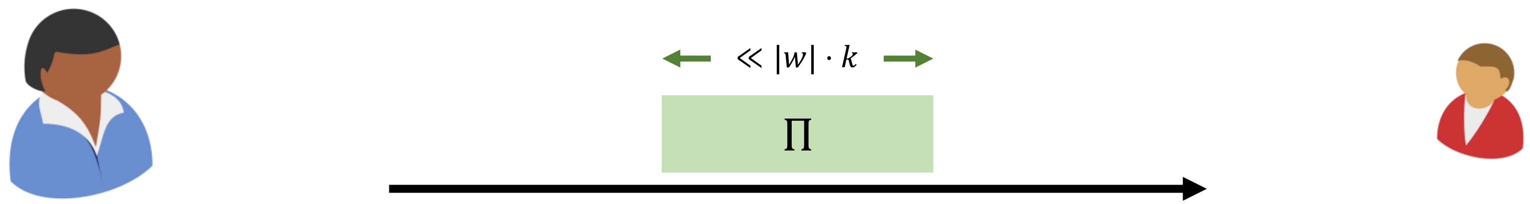
Π is publicly verifiable

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

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

SNARGs for Batch NP (or BARGs)

CRS



$C, x_1, \cdots, x_k$

$C, x_1, \cdots, x_k$

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 (or BARGs)

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

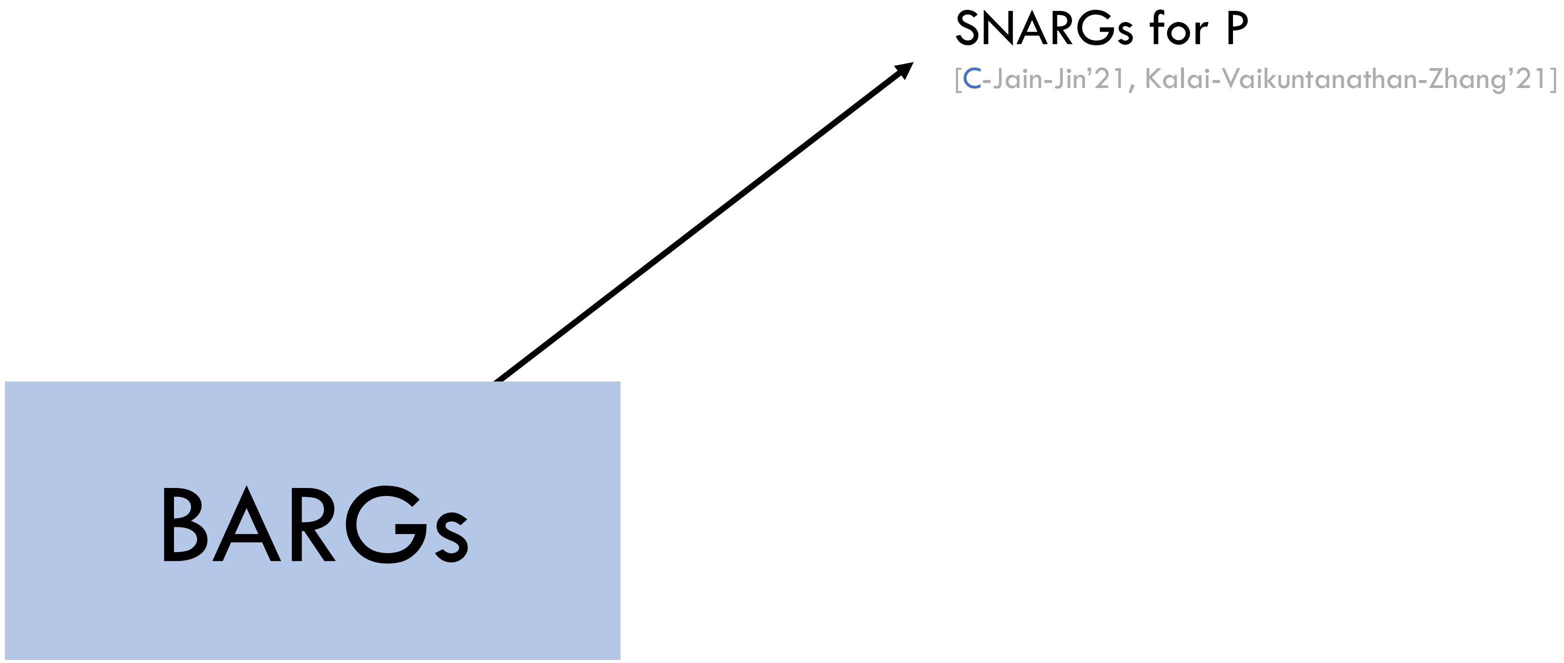
$\text{SAT} = \{(C, x) \mid \exists w \ s.t. \ C(x, w) = 1\}$

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

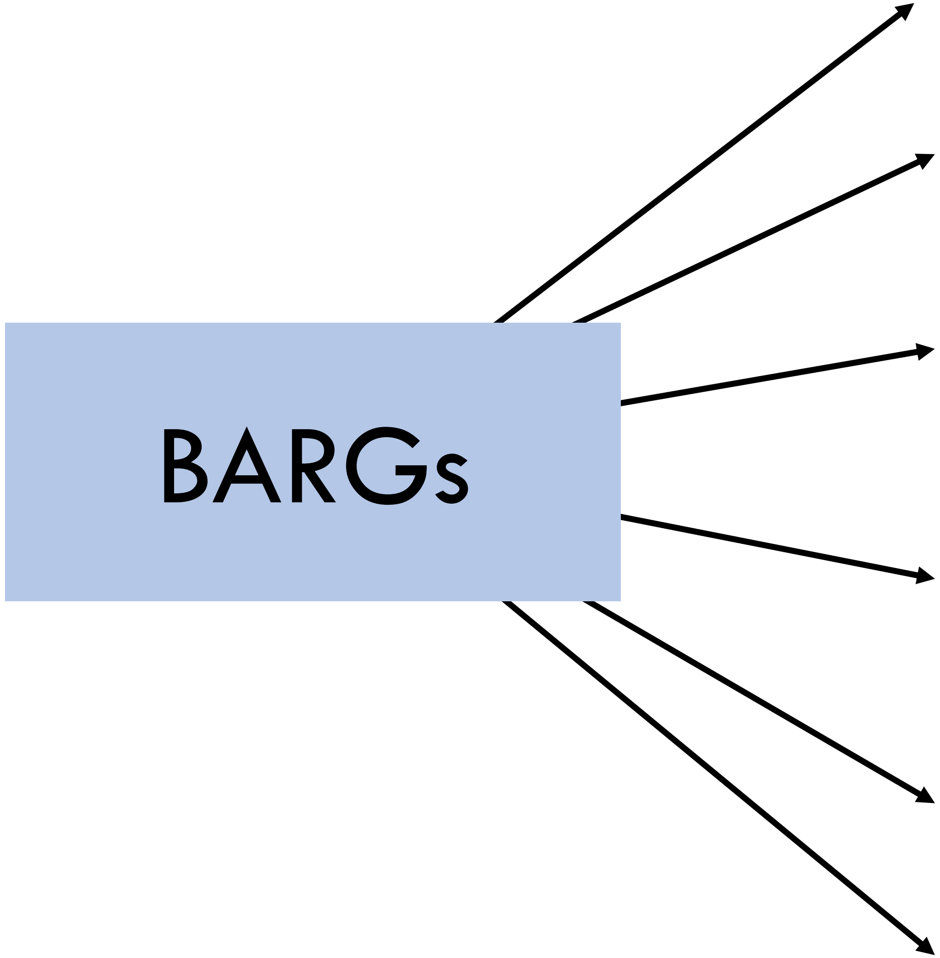
Usefulness of BARGs

BARGs

Usefulness of BARGs

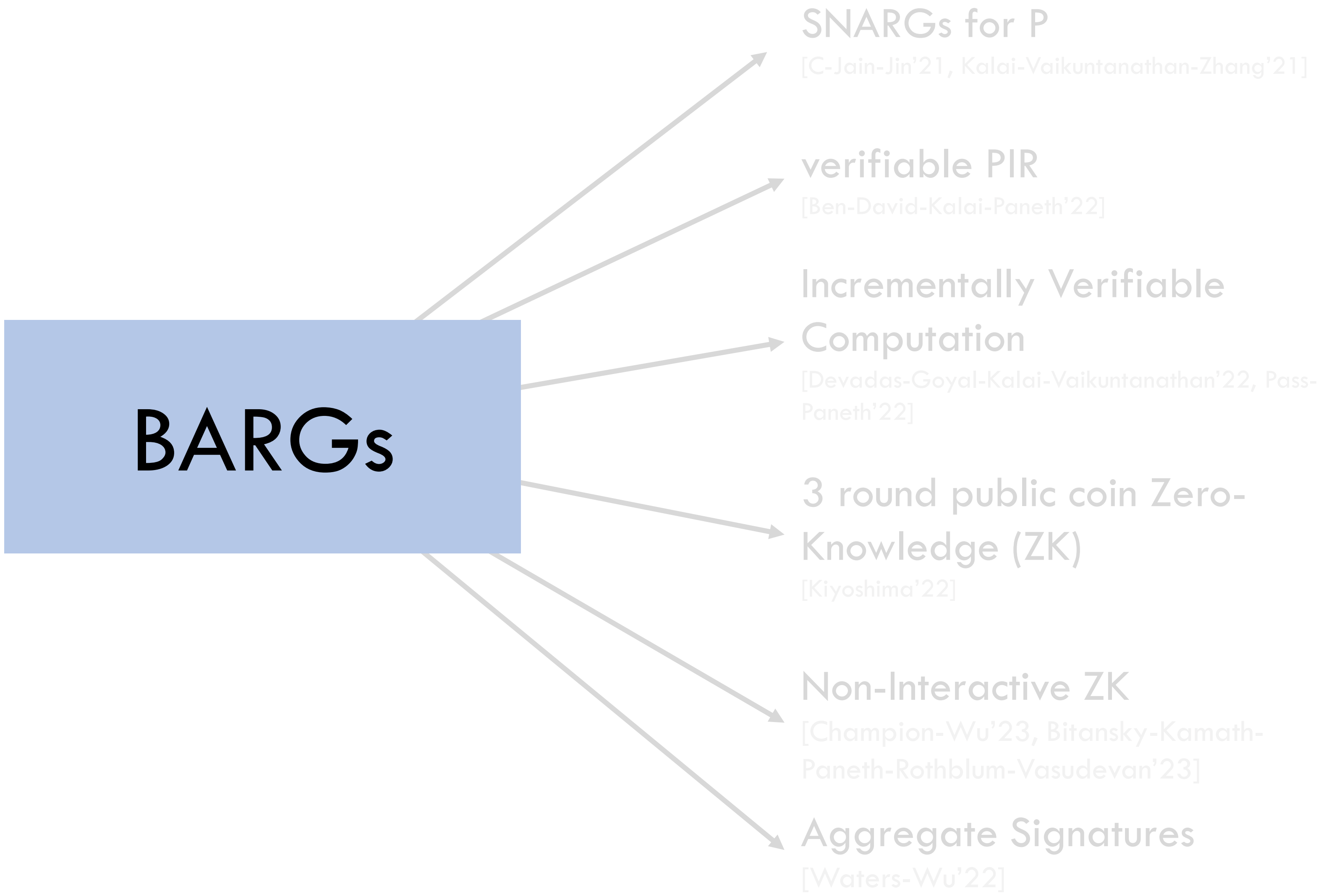


Usefulness of BARGs



- SNARGs for P
  [C-Jain-Jin'21, Kalai-Vaikuntanathan-Zhang'21]
- verifiable PIR
  [Ben-David-Kalai-Paneth'22]
- Incrementally Verifiable Computation
  [Devadas-Goyal-Kalai-Vaikuntanathan'22, Pass-Paneth'22]
- 3 round public coin Zero-Knowledge (ZK)
  [Kiyoshima'22]
- Non-Interactive ZK
  [Champion-Wu'23, Bitansky-Kamath-Paneth-Rothblum-Vasudevan'23]
- Aggregate Signatures
  [Waters-Wu'22]

Construction of BARGs



Construction of BARGs

LWE
[C-Jain-Jin'21]

DLIN
[Waters-Wu'22]

Sub-exp DDH + QR
[C-Jain-Jin'21a, Hulett-Jawale-Khurana-Srinivasan'22]

BARGs

SNARGs for P
[C-Jain-Jin'21, Kalai-Vaikuntanathan-Zhang'21]

verifiable PIR
[Ben-David-Kalai-Paneth'22]

Incrementally Verifiable Computation
[Devadas-Goyal-Kalai-Vaikuntanathan'22, Pass-Paneth'22]

3 round public coin Zero-Knowledge (ZK)
[Kiyoshima'22]

Non-Interactive ZK
[Champion-Wu'23, Bitansky-Kamath-Paneth-Rothblum-Vasudevan'23]

Aggregate Signatures
[Waters-Wu'22]

QR – Quadratic residuosity, LWE – Learning with Error, DDH – Decisional Diffie-Hellman, DLIN – Decisional Linear Assumption over Bilinear Groups.

Construction of BARGs

LWE
[C-Jain-Jin'21]

DLIN
[Waters-Wu'22]

Sub-exp DDH
+ QR
[C-Jain-Jin'21a, Hulett-Jawale-Khurana-Srinivasan'22]

Sub-exp DDH
This work

BARGs

SNARGs for P
[C-Jain-Jin'21, Kalai-Vaikuntanathan-Zhang'21]

verifiable PIR
[Ben-David-Kalai-Paneth'22]

Incrementally Verifiable Computation
[Devadas-Goyal-Kalai-Vaikuntanathan'22, Pass-Paneth'22]

3 round public coin Zero-Knowledge (ZK)
[Kiyoshima'22]

Non-Interactive ZK
[Champion-Wu'23, Bitansky-Kamath-Paneth-Rothblum-Vasudevan'23]

Aggregate Signatures
[Waters-Wu'22]

QR – Quadratic residuosity, LWE – Learning with Error, DDH – Decisional Diffie-Hellman, DLIN – Decisional Linear Assumption over Bilinear Groups.

Our Results

Theorem 1

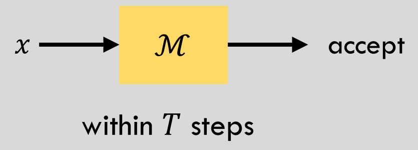
Assuming sub-exponential hardness of DDH, there exists SNARGs for batch NP where

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

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

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

Our Results



Theorem 2

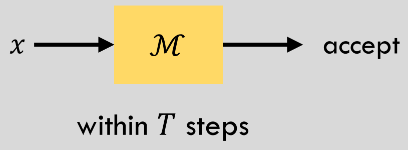
Assuming sub-exponential hardness of DDH, there exists SNARGs for P where

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

Our Results

Recent concurrent work [Kalai-Lombardi-Vaikuntanathan'23]:

SNARGs for bounded depth circuits assuming sub-exponential hardness of DDH.



Theorem 2

Assuming sub-exponential hardness of DDH, there exists SNARGs for P where

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

Our Results

Theorem 1

Assuming sub-exponential hardness of DDH, there exists SNARGs for batch NP where

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

Theorem 2

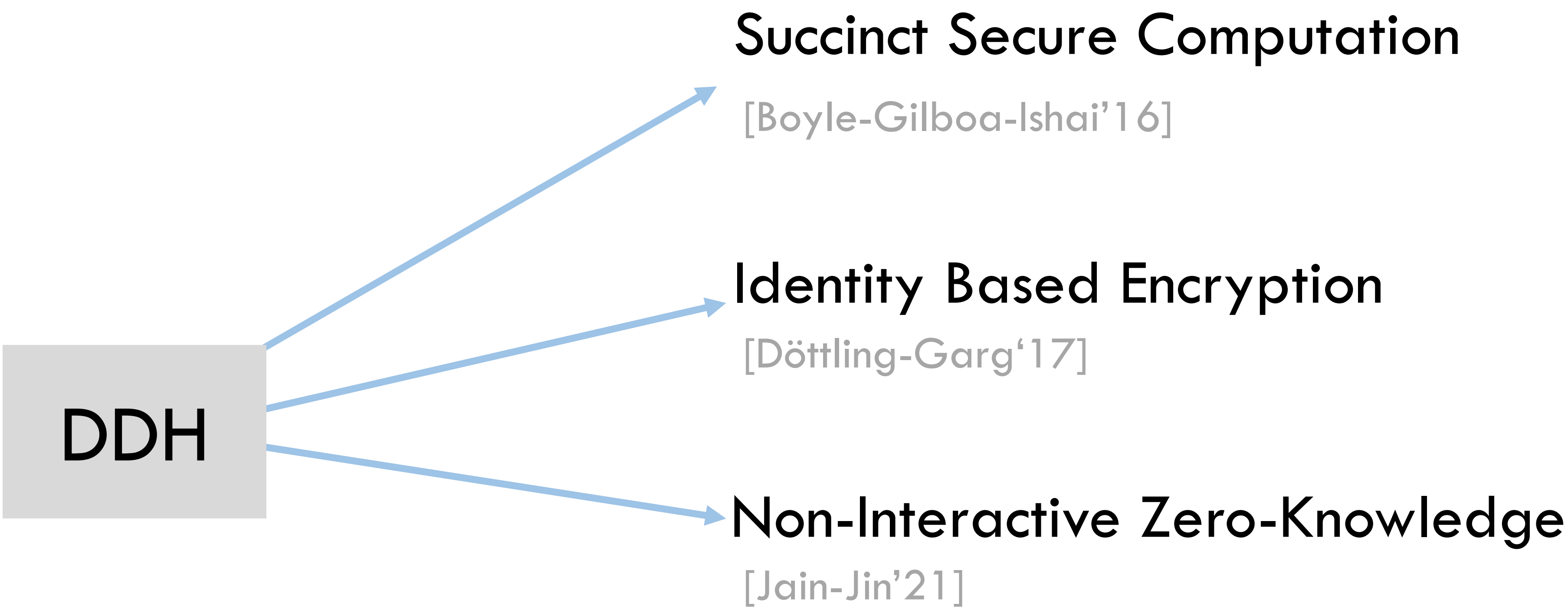
Assuming sub-exponential hardness of DDH, there exists SNARGs for P where

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

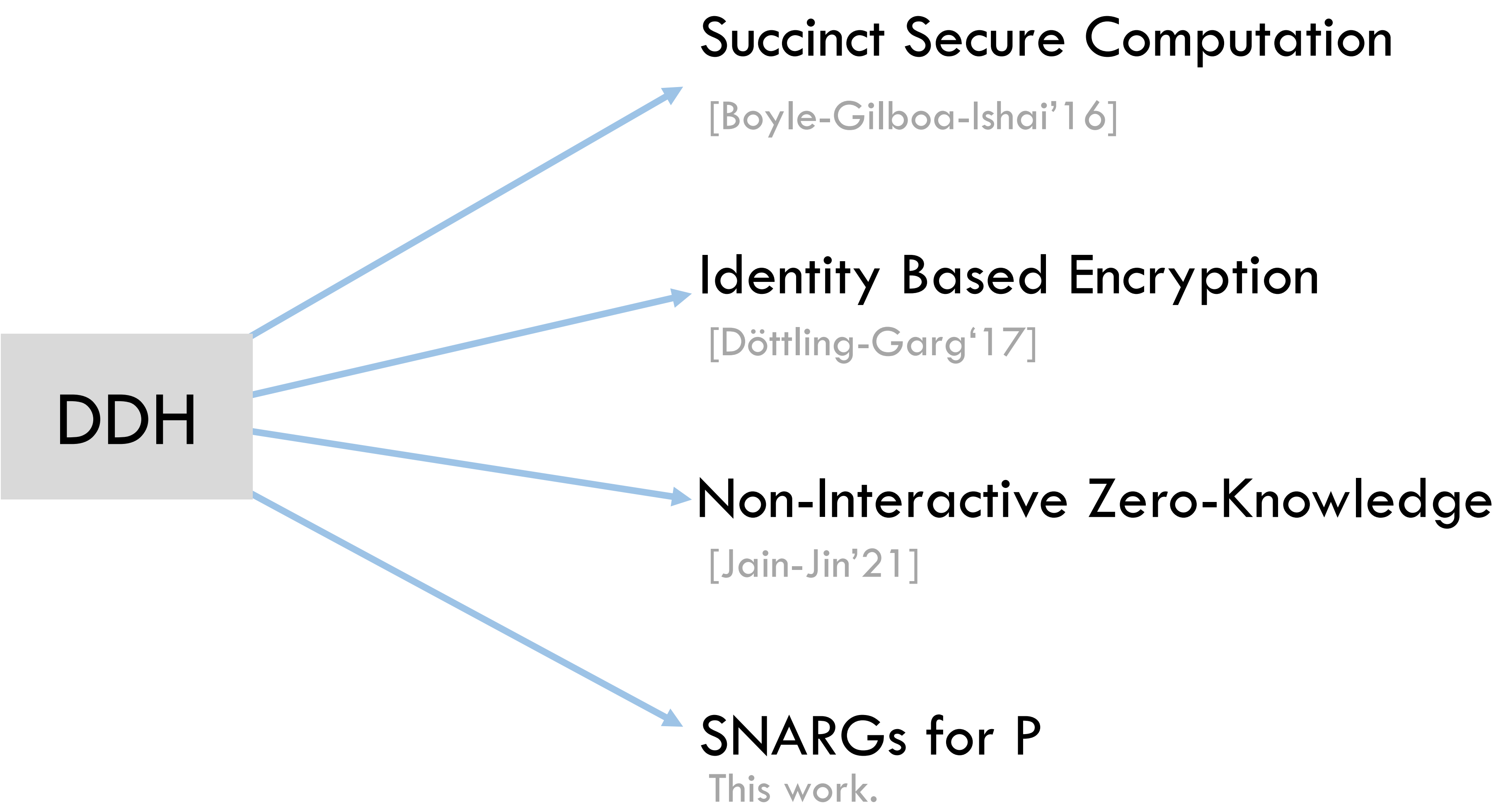
Meta View: Advanced Primitives from DDH

DDH

Meta View: Advanced Primitives from DDH

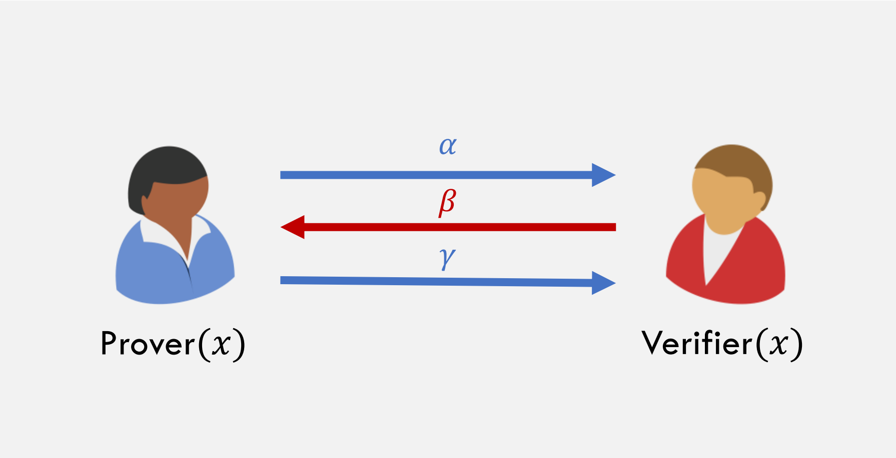


Meta View: Advanced Primitives from DDH



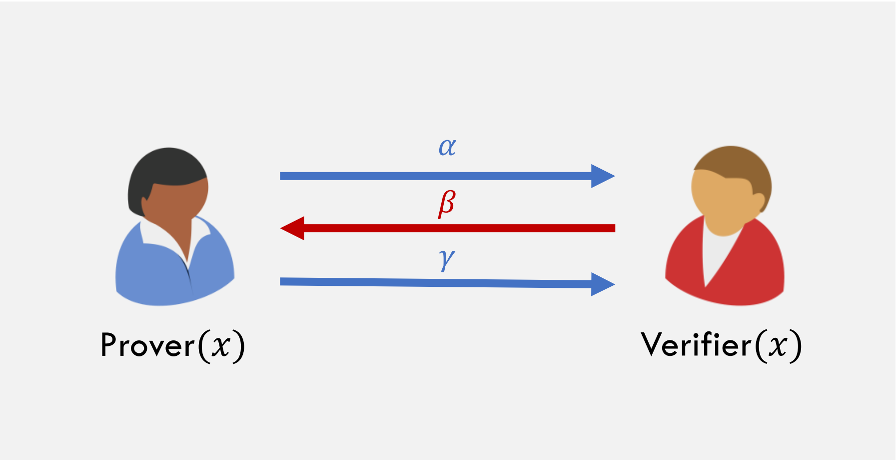
Tools and Techniques

Fiat-Shamir (FS) Methodology: Recipe for Success



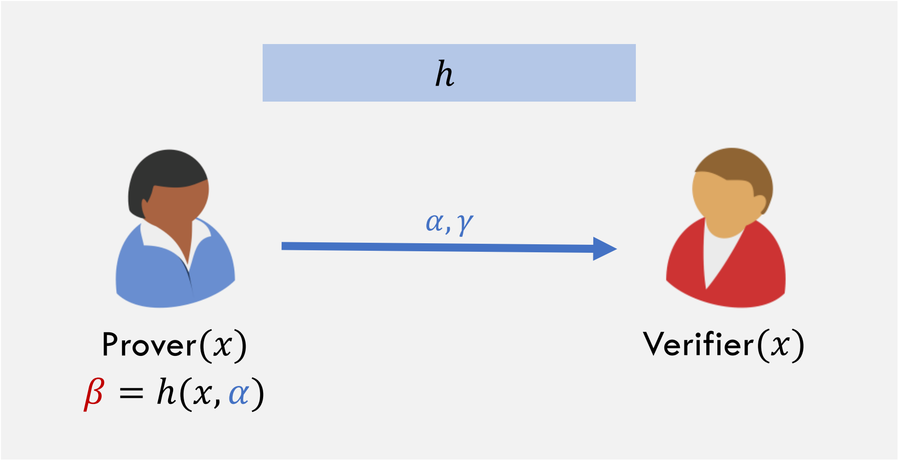
β is a random string

Fiat-Shamir (FS) Methodology

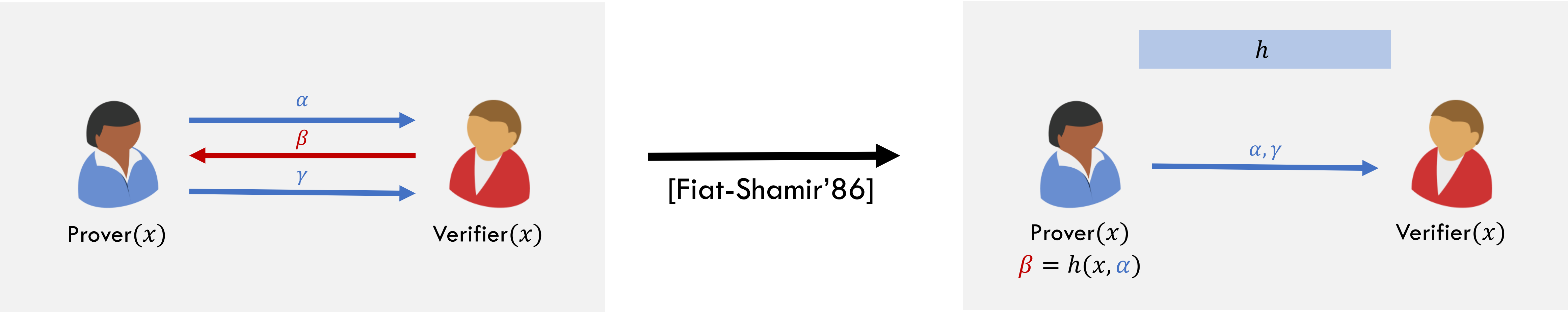


β is a random string

[Fiat-Shamir'86]



Fiat-Shamir (FS) Methodology

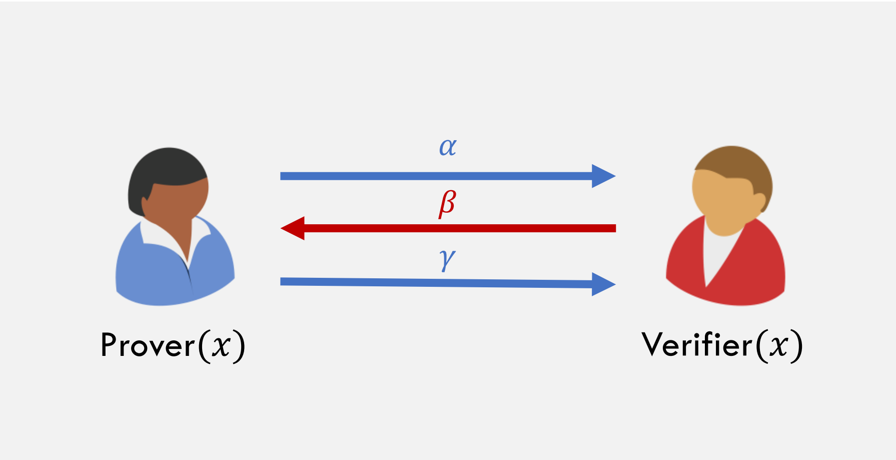


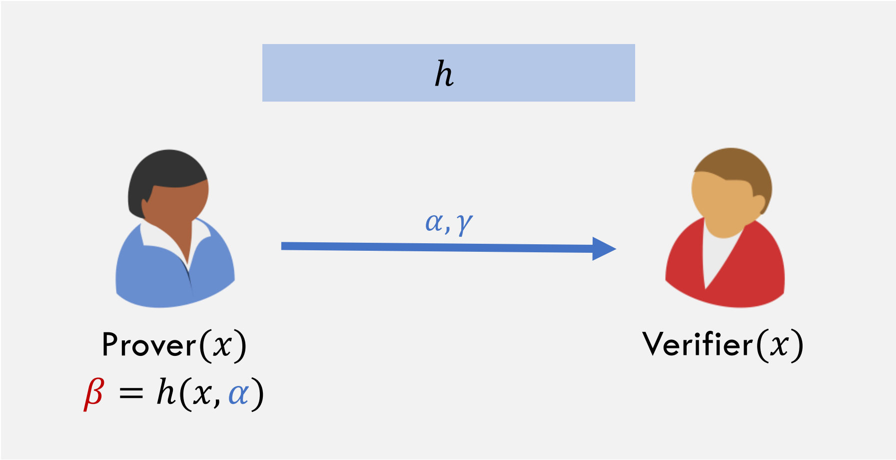
β is a random string

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

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

Fiat-Shamir (FS) Methodology





β is a random string

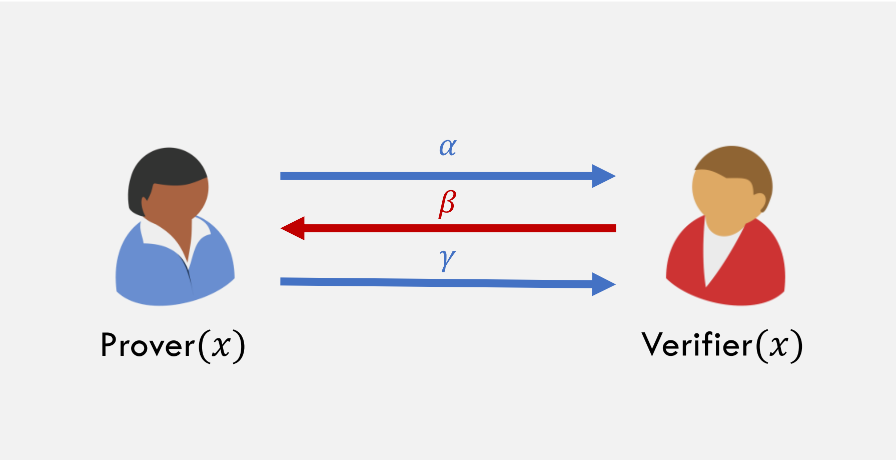
$\forall\, x \notin \mathcal{L}$

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

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

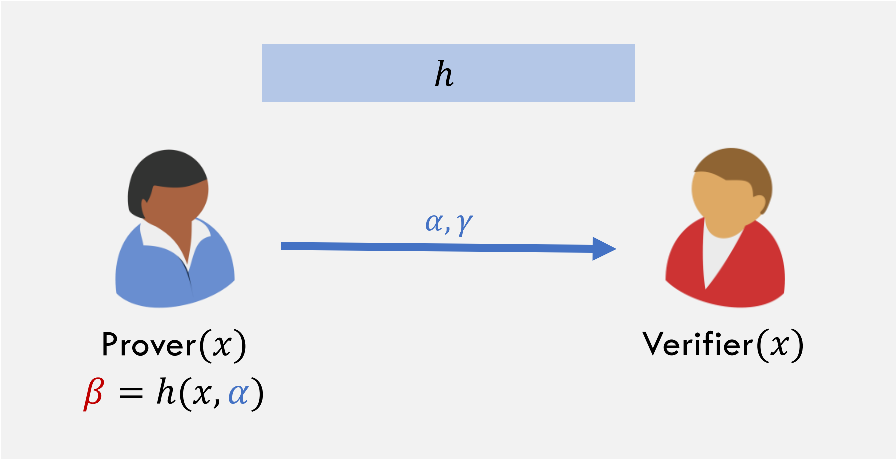
$$h(x, \alpha) \in \mathrm{BAD}_{x,\alpha}$$

Correlation Intractability [Canetti-Goldreich-Halevi'98]



β is a random string

[Fiat-Shamir'86]



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

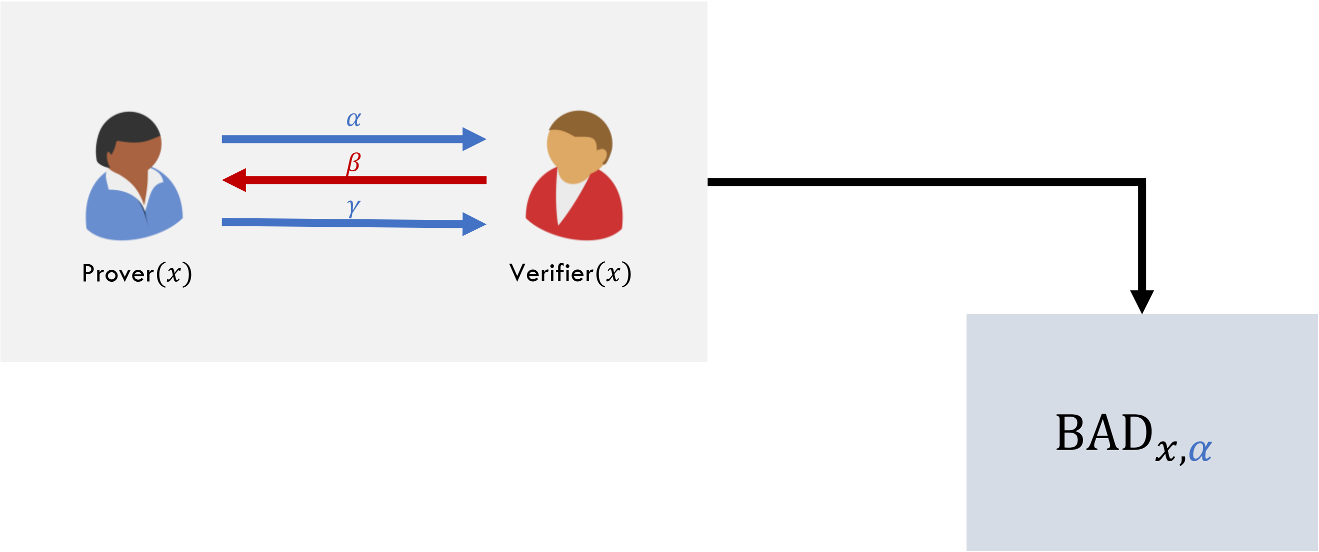
$\mathrm{BAD}_{x,\alpha} = \{\beta \mid \exists \gamma$ s.t. Verifier accepts $(\alpha, \beta, \gamma)\}$

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

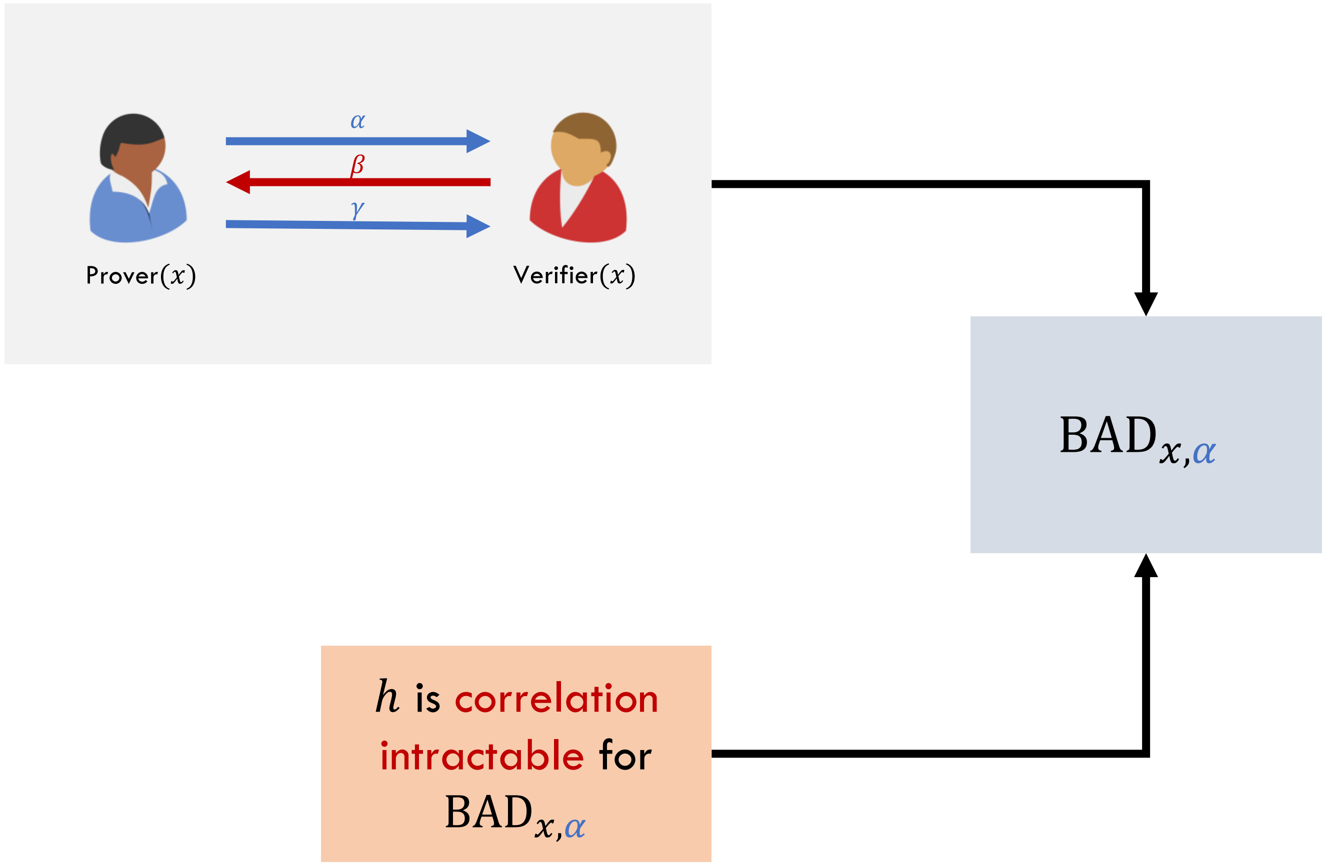
$$h(x, \alpha) \in \mathrm{BAD}_{x,\alpha}$$

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

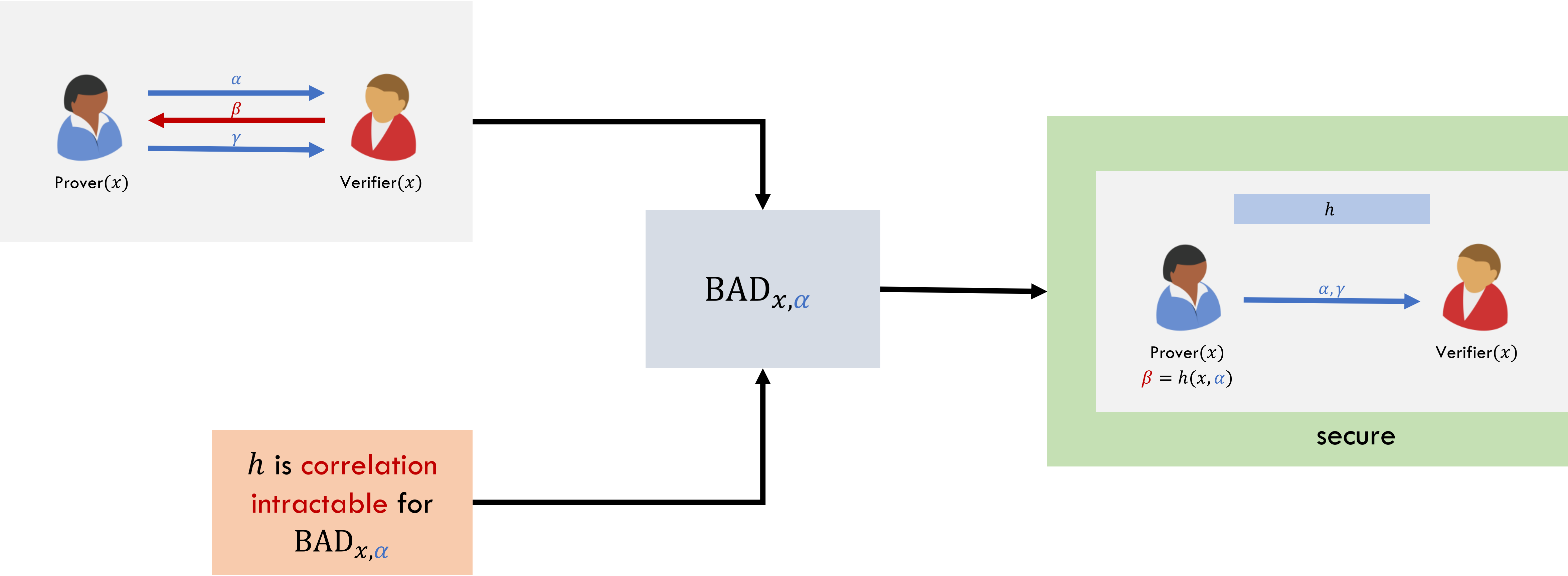
Instantiating the FS Transform



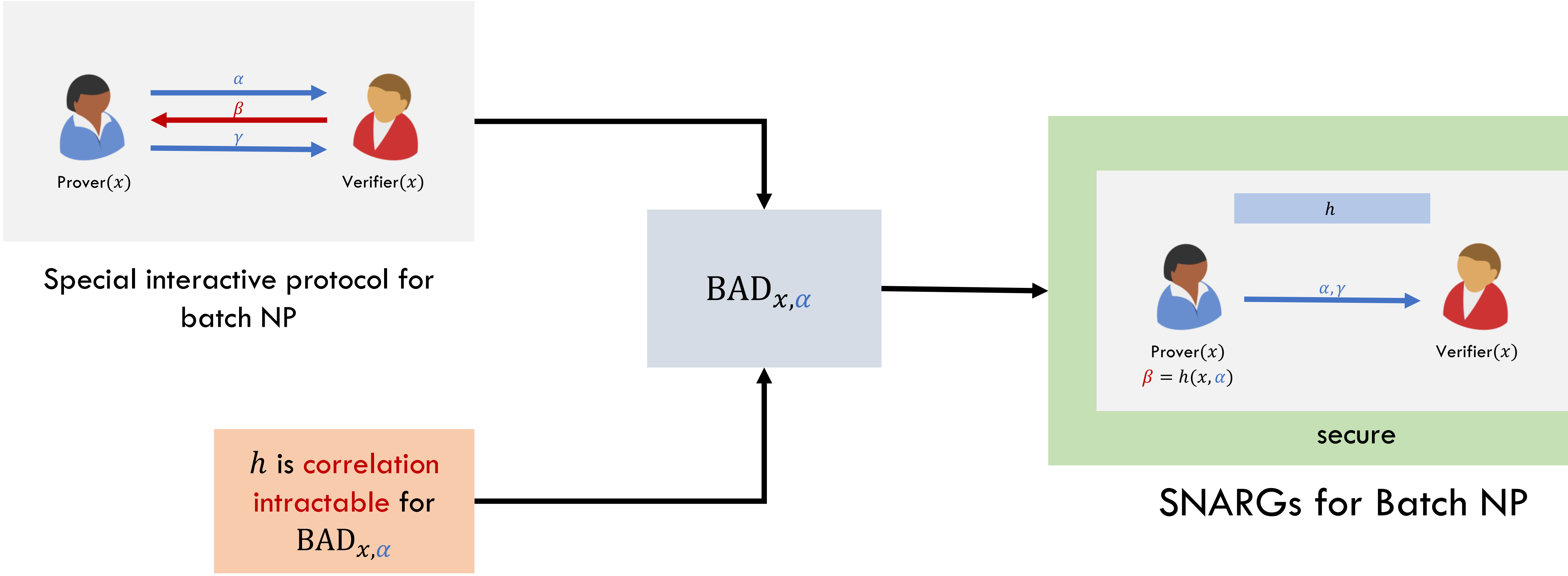
Instantiating the FS Transform



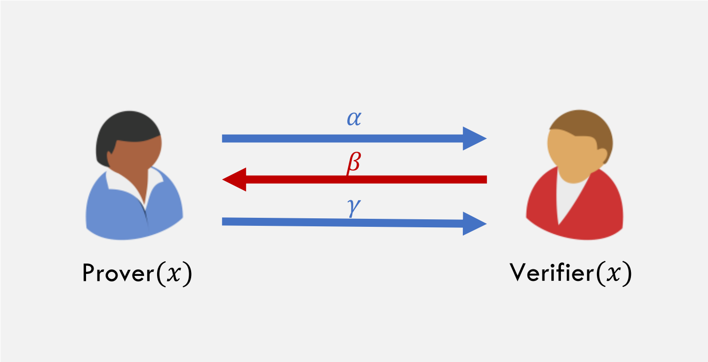
Instantiating the FS Transform



[C-Jain-Jin'21] Methodology

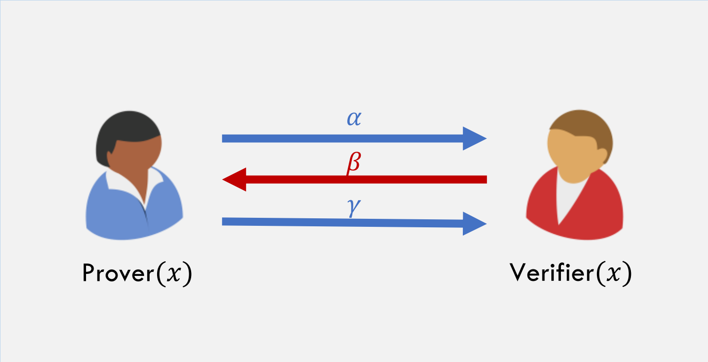


[C-Jain-Jin'21] Methodology



Special interactive protocol for batch NP

[C-Jain-Jin'21] Methodology



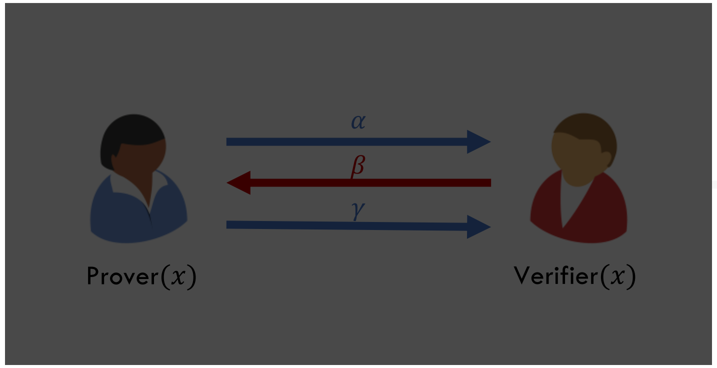
Special interactive protocol for batch NP

This work

sub-exp
DDH

see paper for details

[C-Jain-Jin'21] Methodology

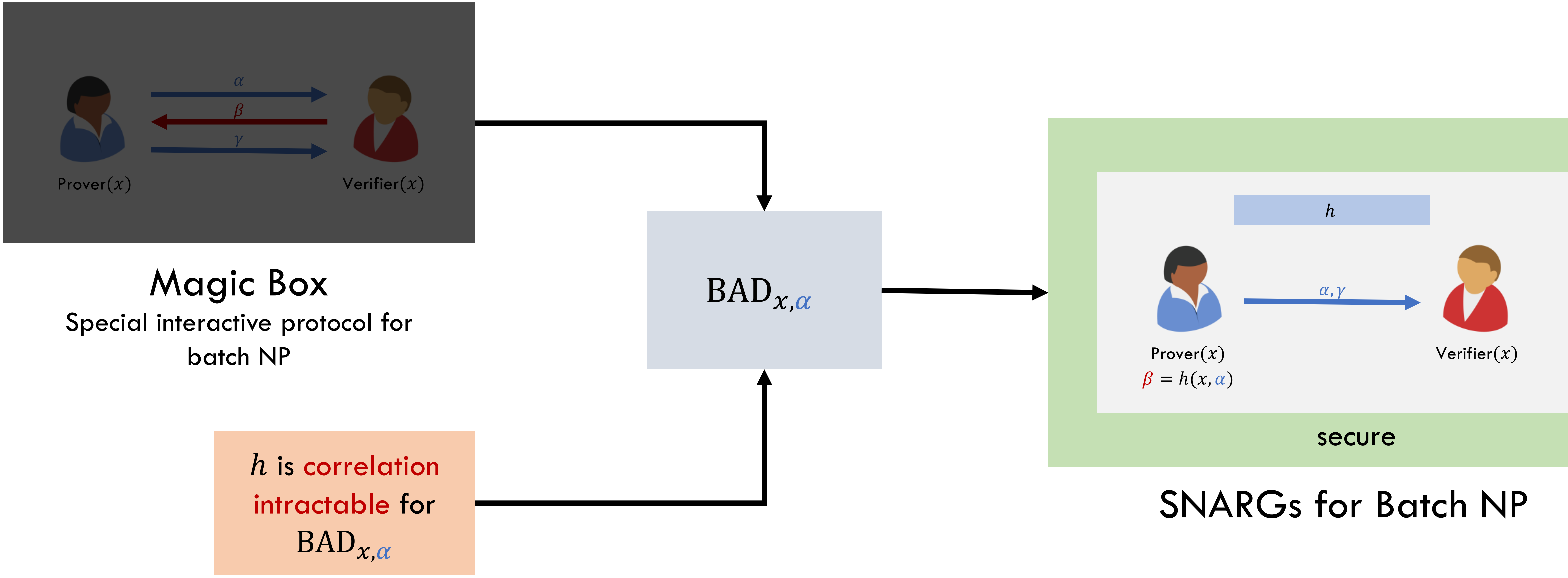


Magic Box

Special interactive protocol for batch NP

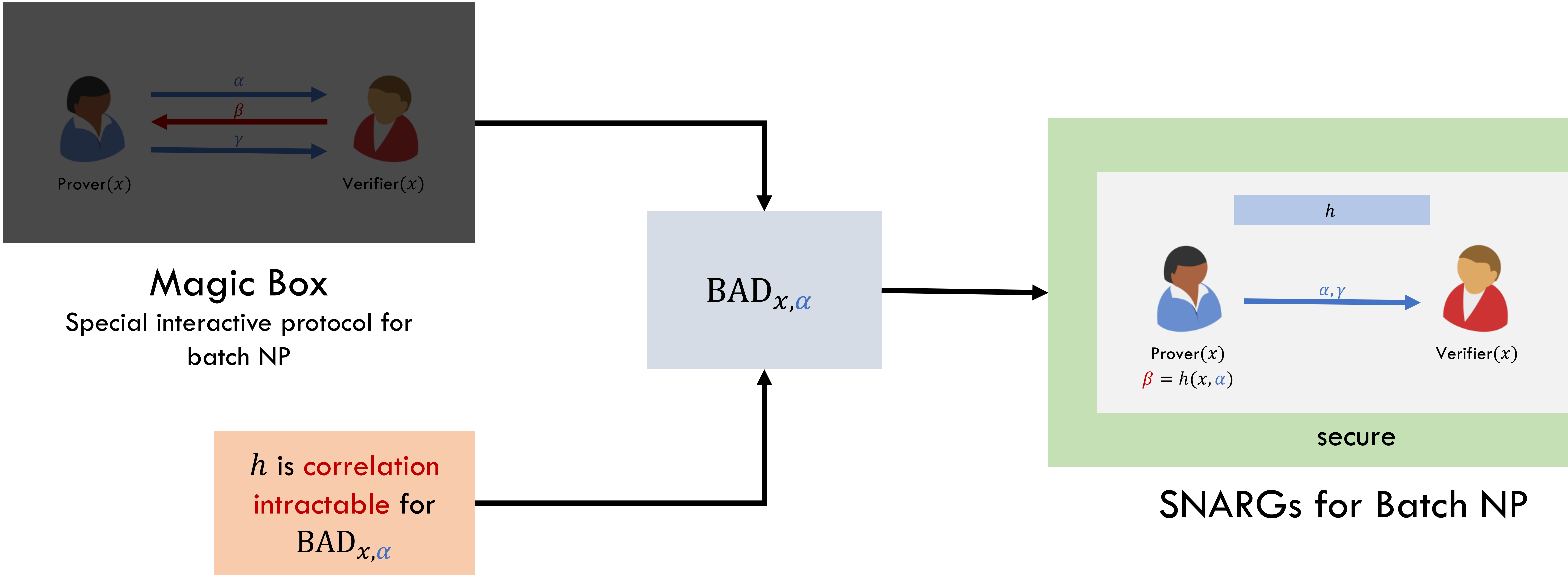
see paper for details

[C-Jain-Jin'21] Methodology



see paper for details

[C-Jain-Jin'21] Methodology



What properties does $\mathrm{BAD}_{x,\alpha}$ have?

see paper for details

Properties of $\mathrm{BAD}_{x,\alpha}$

$\mathrm{BAD}_{x,\alpha}$ is product verifiable.

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

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

Properties of $\mathrm{BAD}_{x,\alpha}$

$$\mathrm{BAD}_{x,\alpha} = \mathrm{BAD}_{x,\alpha}^{(1)} \times \mathrm{BAD}_{x,\alpha}^{(2)} \times \mathrm{BAD}_{x,\alpha}^{(3)} \times \mathrm{BAD}_{x,\alpha}^{(4)}$$

$\mathrm{BAD}_{x,\alpha}$ is product verifiable.

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

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

Properties of $\mathrm{BAD}_{x,\alpha}$

$\mathrm{BAD}_{x,\alpha} = \mathrm{BAD}_{x,\alpha}^{(1)} \times \mathrm{BAD}_{x,\alpha}^{(2)} \times \mathrm{BAD}_{x,\alpha}^{(3)} \times \mathrm{BAD}_{x,\alpha}^{(4)}$

$\mathrm{BAD}_{x,\alpha}$ is product verifiable.

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

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

Exponentially many bad challenges even when β sampled from polynomial size challenge space.

Properties of $\mathrm{BAD}_{x,\alpha}$

$\mathrm{BAD}_{x,\alpha} = \mathrm{BAD}^{(1)}_{x,\alpha} \times \mathrm{BAD}^{(2)}_{x,\alpha} \times \mathrm{BAD}^{(3)}_{x,\alpha} \times \mathrm{BAD}^{(4)}_{x,\alpha}$

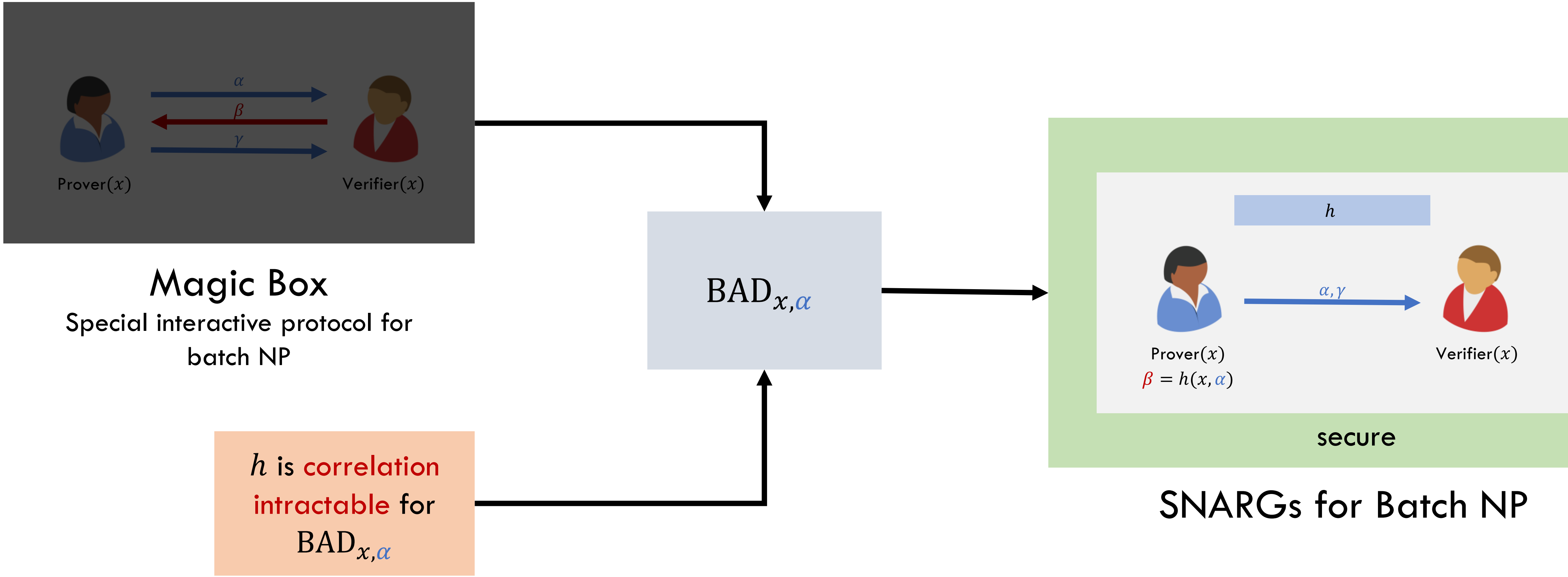
$\mathrm{BAD}_{x,\alpha}$ is product verifiable.

Each $\mathrm{BAD}^{(i)}_{x,\alpha}$ is efficiently verifiable

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

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

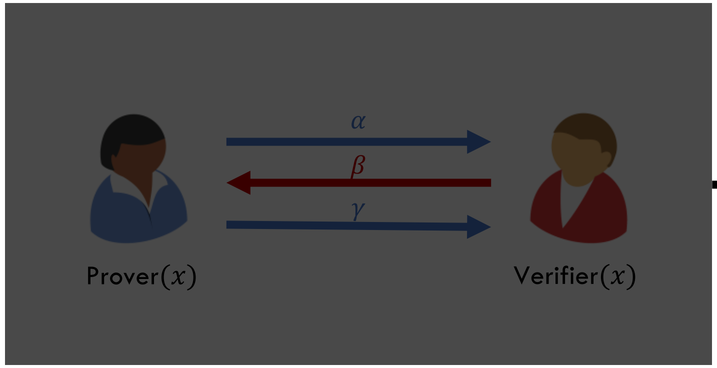
[C-Jain-Jin'21] Methodology



What properties does $\mathrm{BAD}_{x,\alpha}$ have?

see paper for details

[C-Jain-Jin'21] Methodology



Magic Box

Special interactive protocol for batch NP

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

$\mathrm{BAD}_{x,\alpha}$

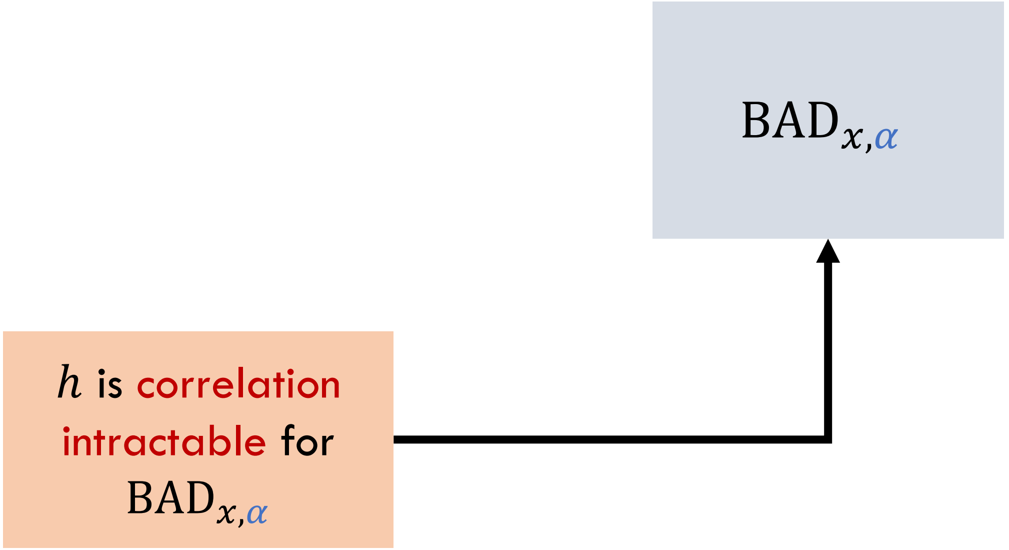
$\mathrm{BAD}_{x,\alpha}$ properties

1. Bad challenges are a product set
2. Challenge space is of polynomial size
3. Bad challenges are product verifiable in TC^0

TC^0 - Constant depth polynomial-size threshold circuits

see paper for details

[C-Jain-Jin'21] Methodology

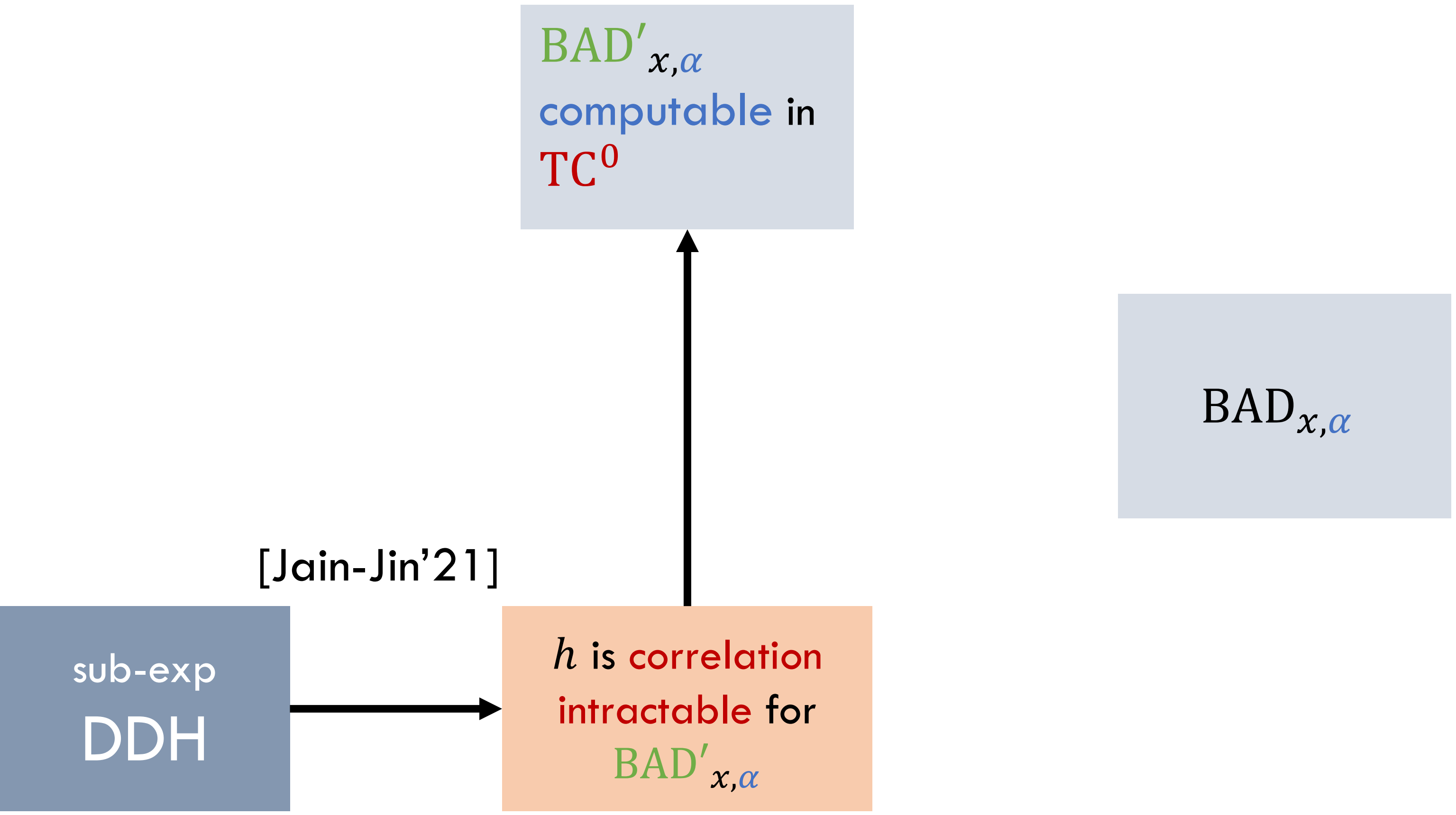


$\mathrm{BAD}_{x,\alpha}$ properties

1. Bad challenges are a product set
2. Challenge space is of polynomial size
3. Bad challenges are product verifiable in TC^0

TC^0 - Constant depth polynomial-size threshold circuits

[C-Jain-Jin'21] Methodology

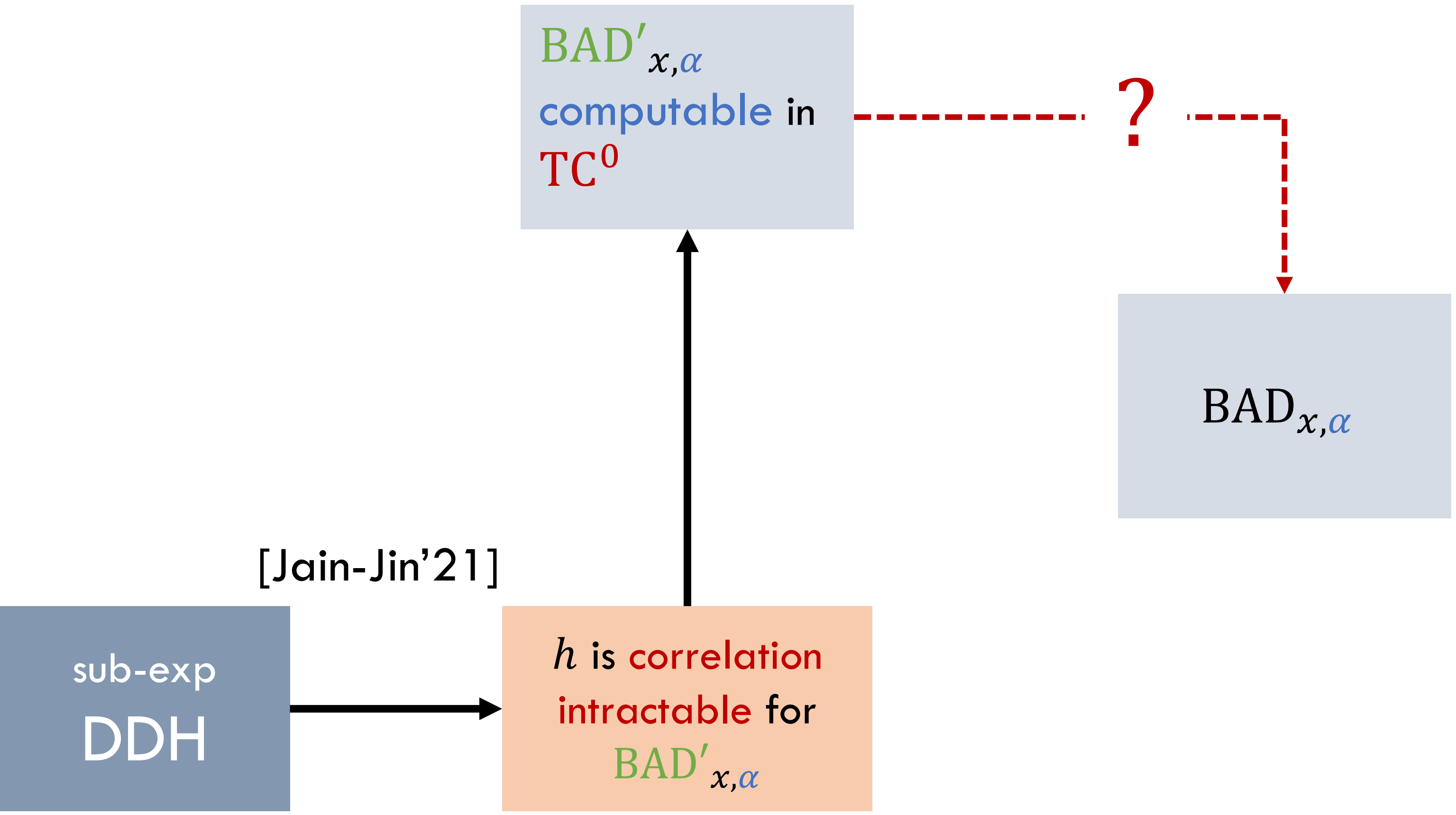


$\mathrm{BAD}_{x,\alpha}$ properties

1. Bad challenges are a product set
2. Challenge space is of polynomial size
3. Bad challenges are product verifiable in TC^0

TC^0 - Constant depth polynomial-size threshold circuits

[C-Jain-Jin'21] Methodology



$\mathrm{BAD}_{x,\alpha}$ properties

1 Bad challenges are a product set

2 Challenge space is of polynomial size

3 Bad challenges are product verifiable in TC^0

TC^0 - Constant depth polynomial-size threshold circuits

[C-Jain-Jin'21] Methodology

$\mathrm{BAD}'_{x,\alpha}$ computable in TC^0

?

$\mathrm{BAD}_{x,\alpha}$

[Jain-Jin'21]

Difficulty [Holmgren-Lombardi-Rothblum'21]:

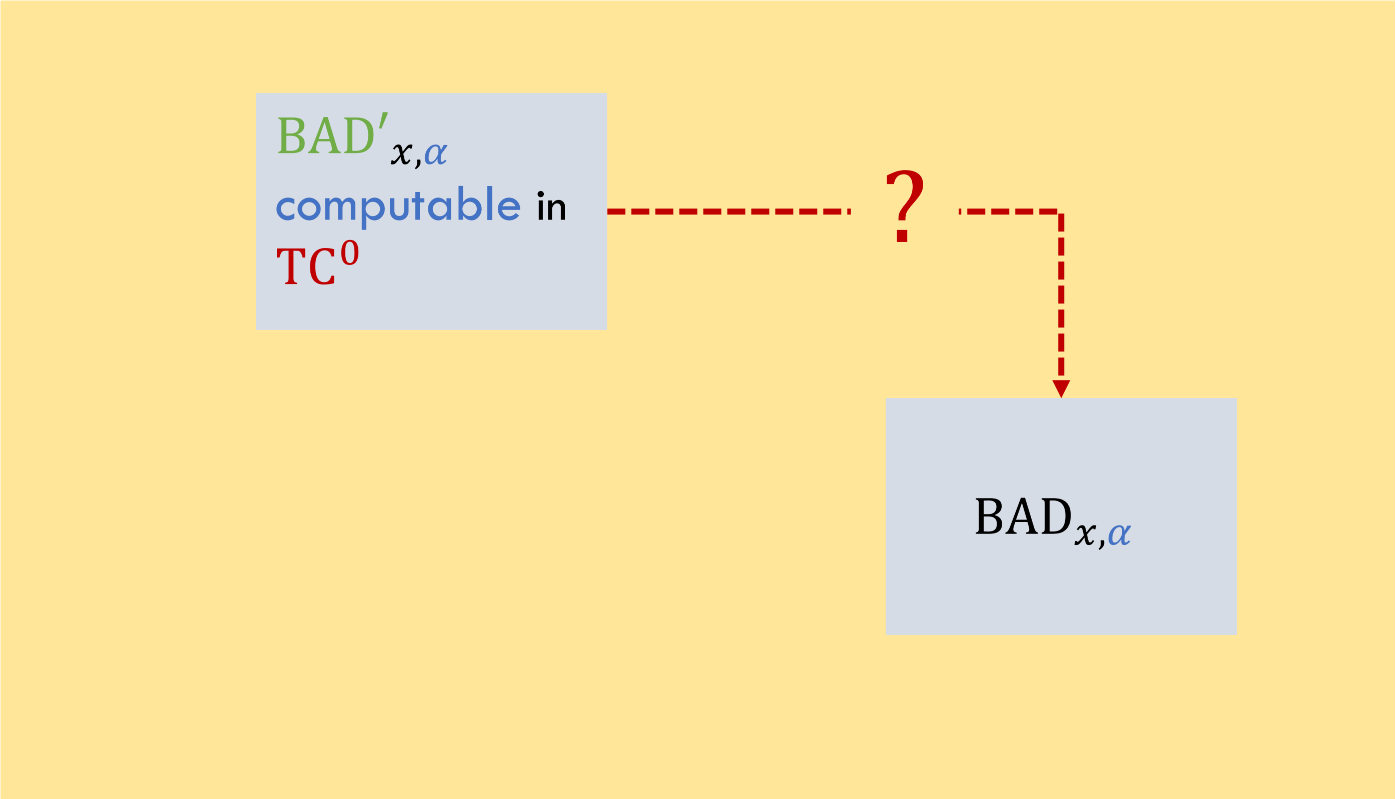
$\mathrm{BAD}_{x,\alpha}$ has exponentially many bad challenges.

$\mathrm{BAD}_{x,\alpha}$ properties

1. Bad challenges are a product set
2. Challenge space is of polynomial size
3. Bad challenges are product verifiable in TC^0

TC^0 - Constant depth polynomial-size threshold circuits

[C-Jain-Jin'21] Methodology

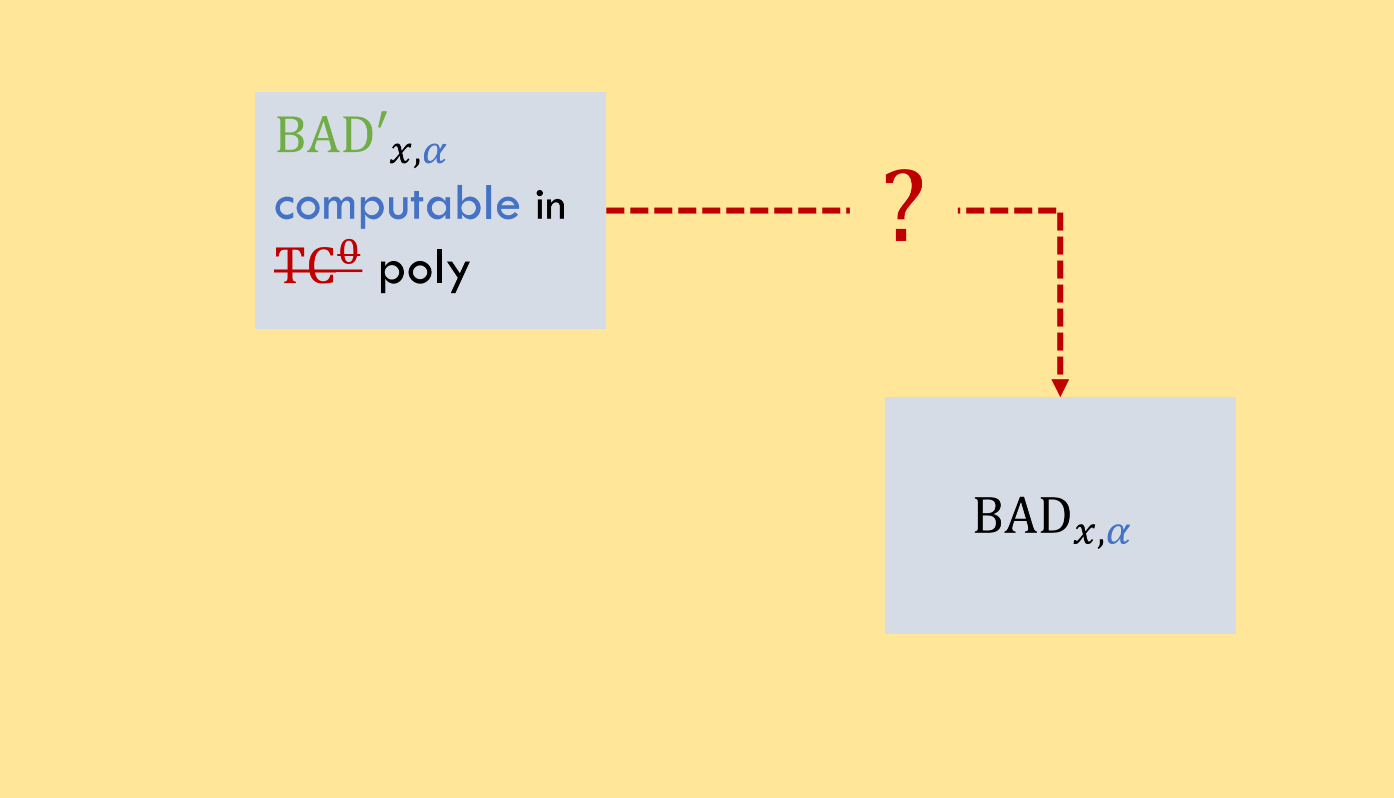


$\mathrm{BAD}_{x,\alpha}$ properties

1. Bad challenges are a product set
2. Challenge space is of polynomial size
3. Bad challenges are product verifiable in TC^0

TC^0 - Constant depth polynomial-size threshold circuits

[C-Jain-Jin'21] Methodology



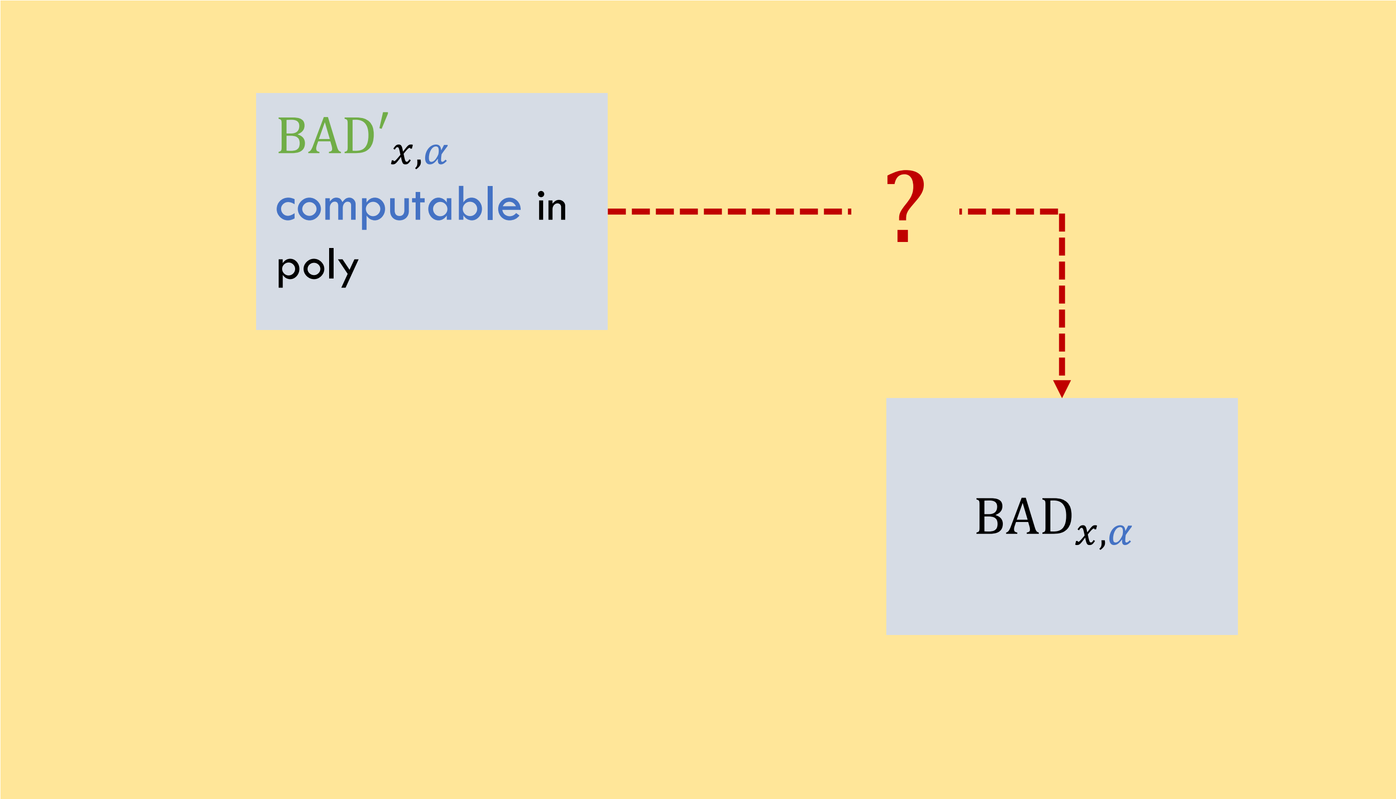
$\text{BAD}_{x,\alpha}$ properties

1. Bad challenges are a product set
2. Challenge space is of polynomial size
3. Bad challenges are product verifiable in ~~TC^0~~ poly

TC^0 - Constant depth polynomial-size threshold circuits

For this talk

[C-Jain-Jin'21] Methodology



$\mathrm{BAD}_{x,\alpha}$ properties

1. Bad challenges are a product set
2. Challenge space is of polynomial size
3. Bad challenges are product verifiable in poly

For this talk

Easy Case: Verifiable Unique Bad Challenge

$$\mathrm{BAD}_{x,\alpha}^{(1)}$$

Easy Case: Verifiable Unique Bad Challenge

$\mathrm{BAD}_{x,\alpha}^{(1)}$

Compute Bad Challenge

for $\beta \in \mathrm{ChallengeSpace}$

  if $\beta \in \mathrm{BAD}_{x,\alpha}^{(1)}$

    return β

ChallengeSpace polynomial size + $\mathrm{BAD}_{x,\alpha}^{(1)}$ efficiently verifiable $\Rightarrow$ $\mathrm{BAD}_{x,\alpha}^{(1)}$ efficiently computable.

Easy Case: Verifiable Unique Bad Challenge

$$\mathrm{BAD}_{x,\alpha} = \mathrm{BAD}_{x,\alpha}^{(1)} \times \mathrm{BAD}_{x,\alpha}^{(2)} \times \cdots \times \mathrm{BAD}_{x,\alpha}^{(d)}$$

Easy Case: Verifiable Unique Bad Challenge

$$\text{BAD}_{x,\alpha} = \text{BAD}_{x,\alpha}^{(1)} \times \text{BAD}_{x,\alpha}^{(2)} \times \cdots \times \text{BAD}_{x,\alpha}^{(d)}$$

Compute Bad Challenge

for $i \in [d]$

  for $\beta^{(i)} \in \text{ChallengeSpace}$

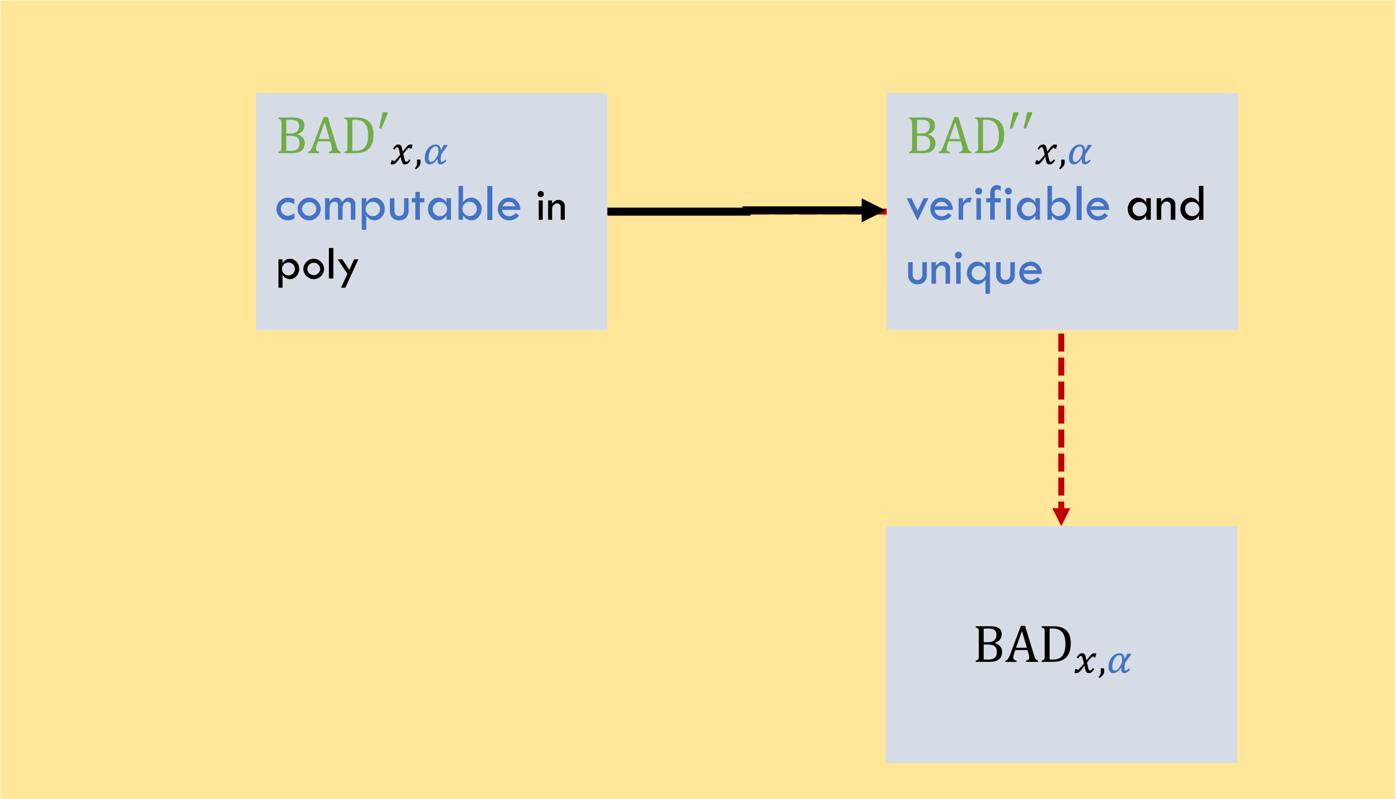
    if $\beta^{(i)} \in \text{BAD}_{x,\alpha}^{(i)}$

      store $\beta^{(i)}$

return $(\beta^{(1)}, \cdots, \beta^{(d)})$

poly repetitions + ChallengeSpace polynomial size + $\text{BAD}_{x,\alpha}^{(i)}$ efficiently verifiable $\Rightarrow$ $\text{BAD}_{x,\alpha}$ efficiently computable.

Overview So Far



$\mathrm{BAD}_{x,\alpha}$ properties

1. Bad challenges are a product set
2. Challenge space is of polynomial size
3. Bad challenges are product verifiable in poly

Reducing to Verifiable Unique Bad Challenge

No parallel repetition

$\leftarrow \ell = \log_2 |\text{ChallengeSpace}| \rightarrow$

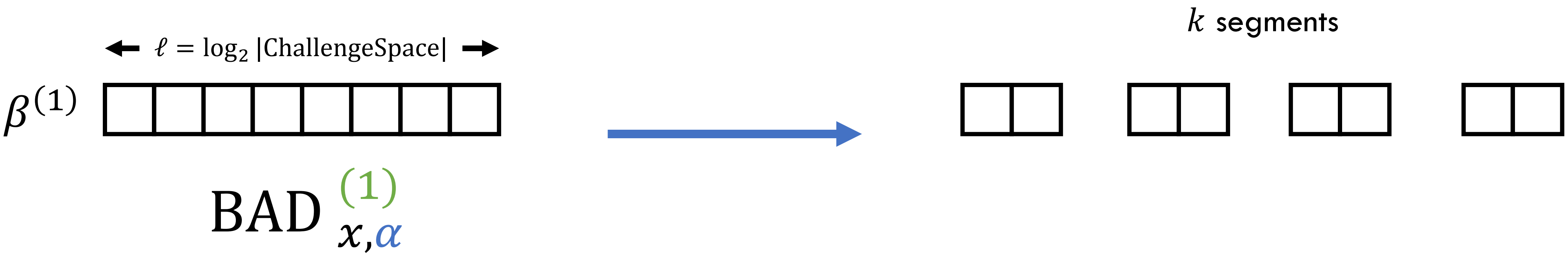
$\beta^{(1)}$ $\square\square\square\square\square\square\square\square$

$$\text{BAD}_{x,\alpha}^{(1)}$$

No restriction on number of bad challenges

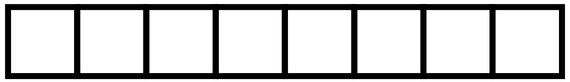
Reducing to Verifiable Unique Bad Challenge

No parallel repetition

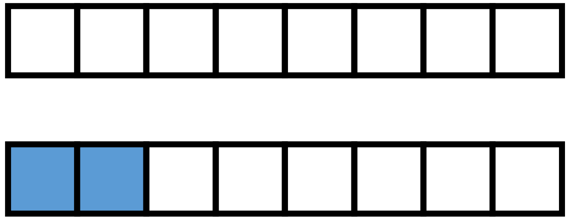


Each segment has ℓ/k bits

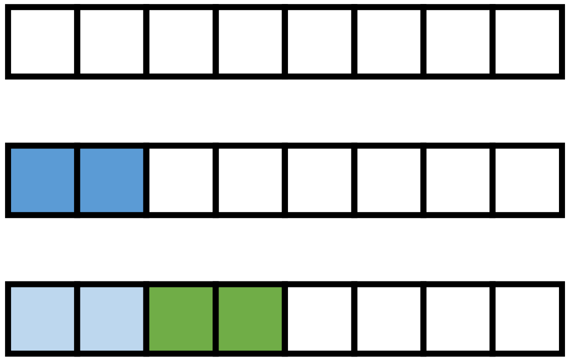
Sampling Challenges via Segments



Sampling Challenges via Segments



Sampling Challenges via Segments



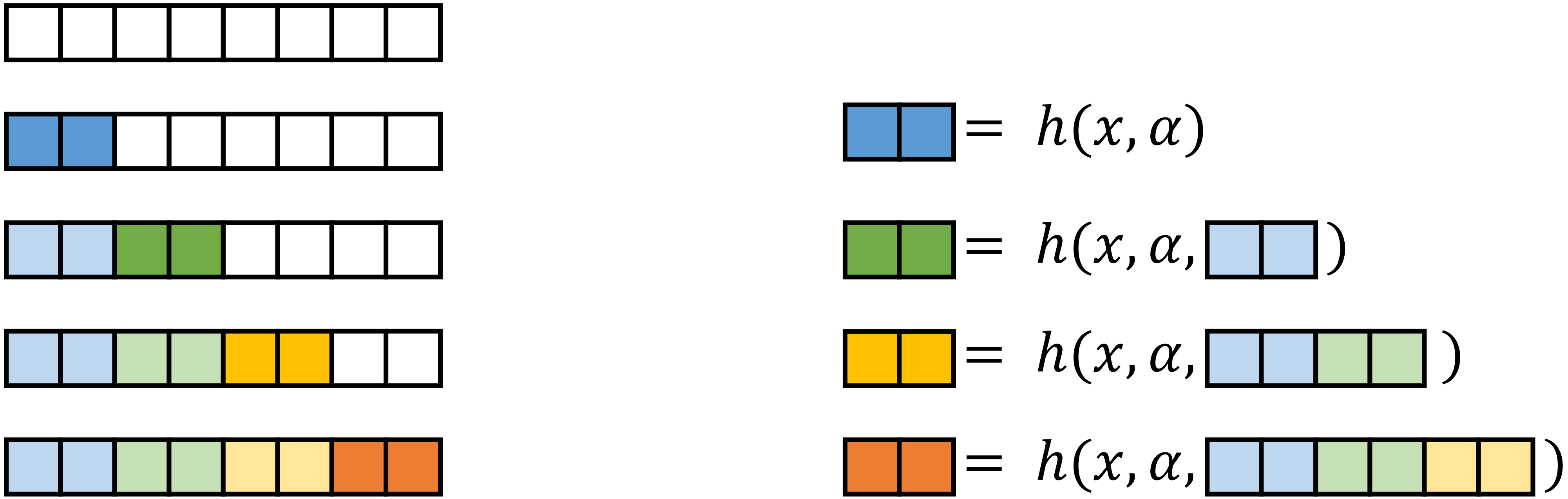
Sampling Challenges via Segments

h is correlation intractable for efficiently verifiable unique bad challenge relations.

Sampling Challenges via Segments

h is correlation intractable for efficiently verifiable unique bad challenge relations.

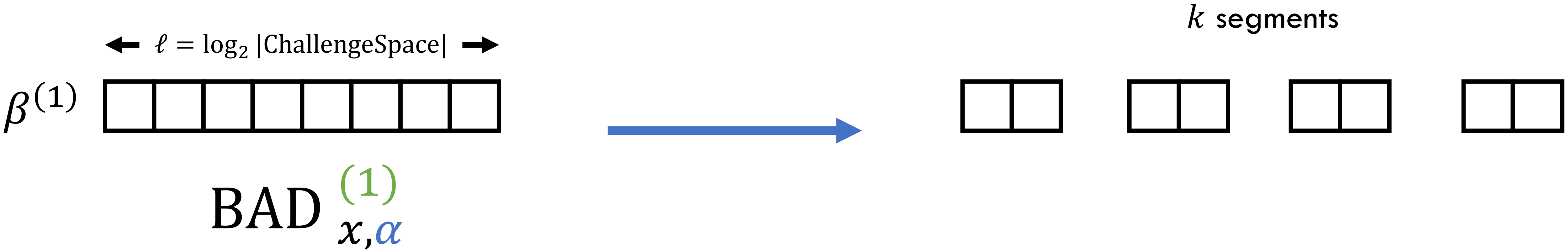
Sampling Challenges via Segments



h is correlation intractable for efficiently verifiable unique bad challenge relations.

Reducing to Verifiable Unique Bad Challenge

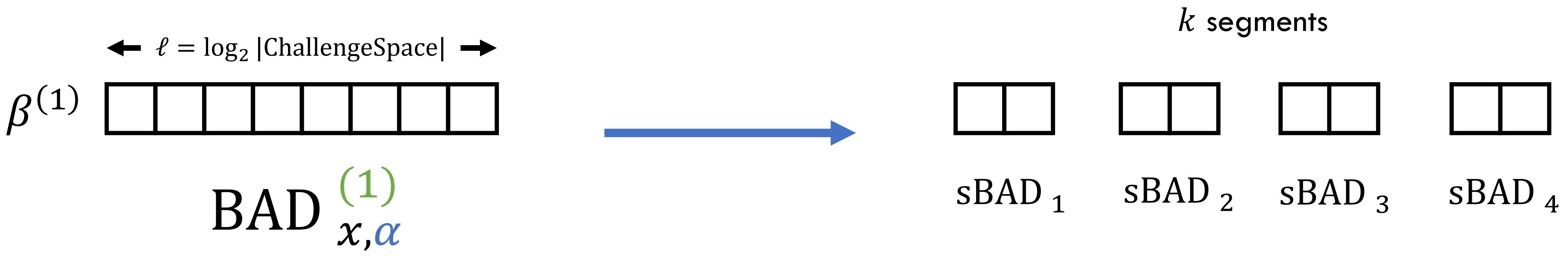
No parallel repetition



Each segment has ℓ / k bits

Reducing to Verifiable Unique Bad Challenge

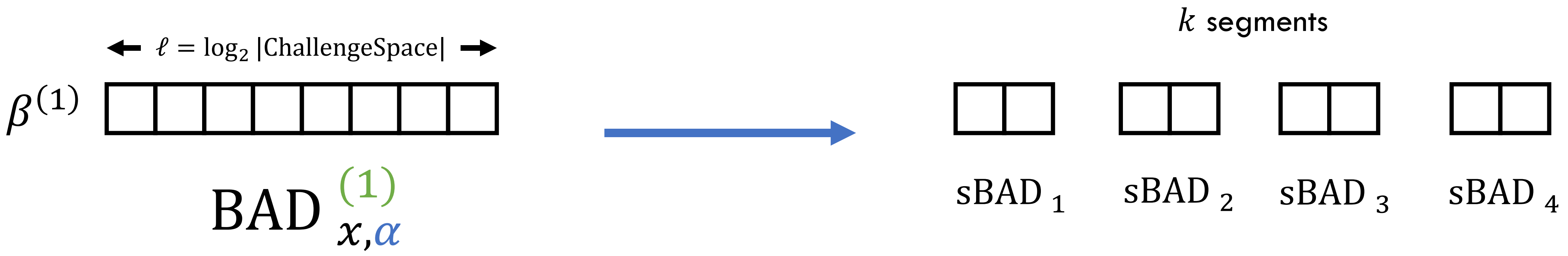
No parallel repetition



Each segment has ℓ/k bits

Reducing to Verifiable Unique Bad Challenge

No parallel repetition



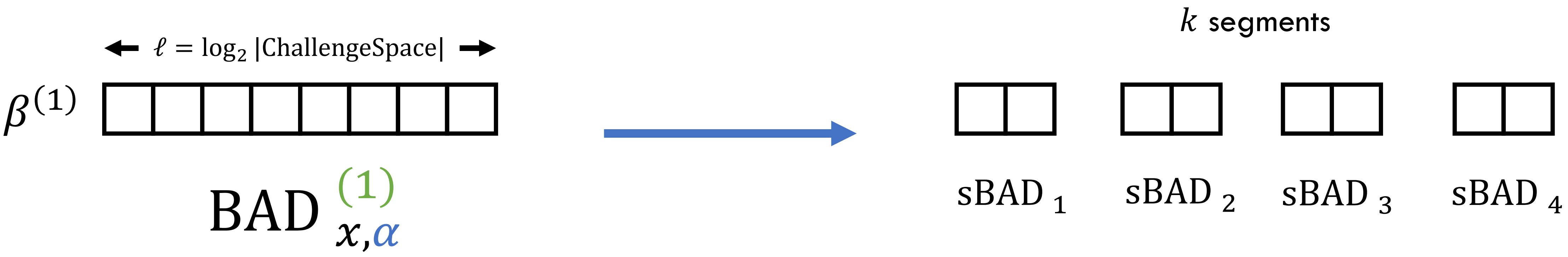
Requirements:

1. Each sBAD_j must be efficiently verifiable unique bad challenge relations.

Each segment has ℓ/k bits

Reducing to Verifiable Unique Bad Challenge

No parallel repetition

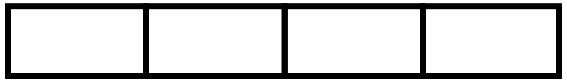


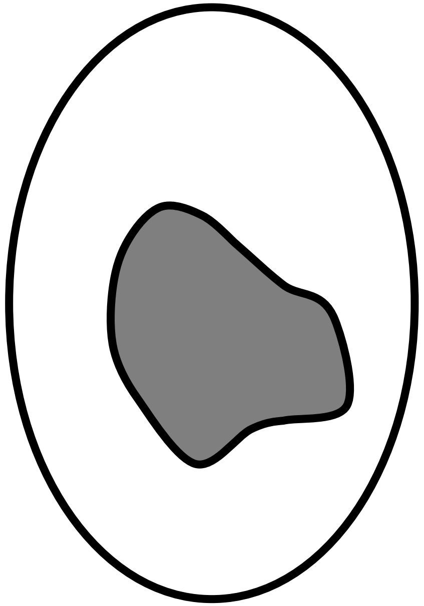
Requirements:

1. Each sBAD_j must be efficiently verifiable unique bad challenge relations.
2. If a challenge is bad, then there must exist a bad segment.

Each segment has ℓ/k bits

Defining Bad Segments

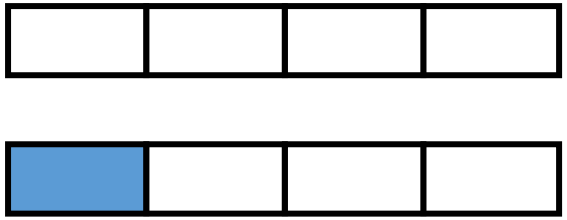


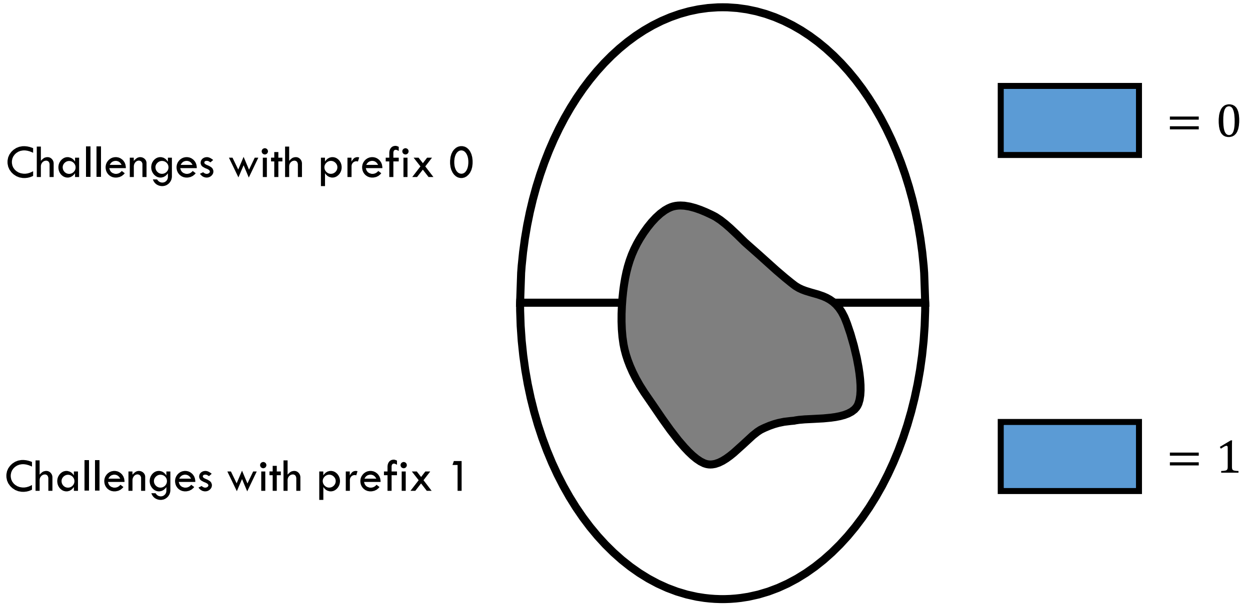


Challenge space

All bad challenges for $\mathrm{BAD}_{x,\alpha}^{(1)}$

Defining Bad Segments



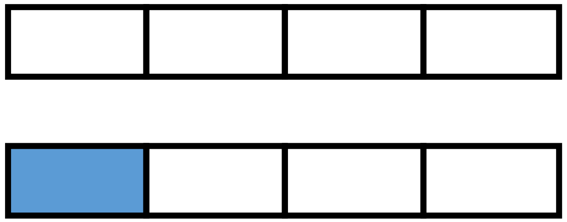


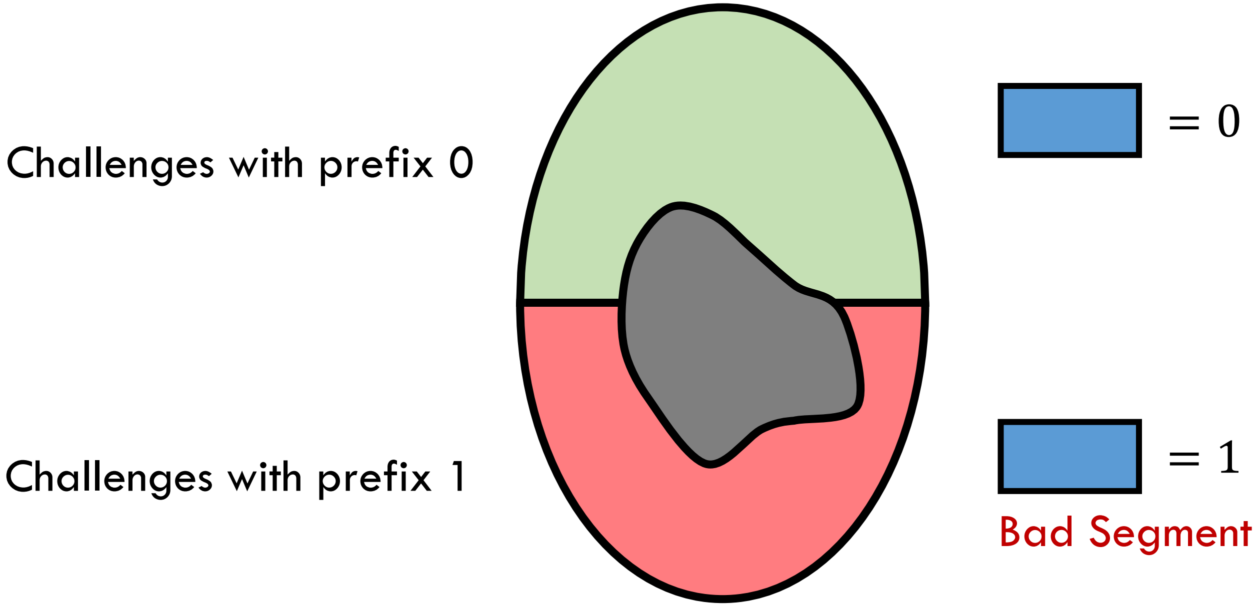
sBAD_1

is bad if

#bad challenges with prefix $>$ #bad challenges/2

Defining Bad Segments



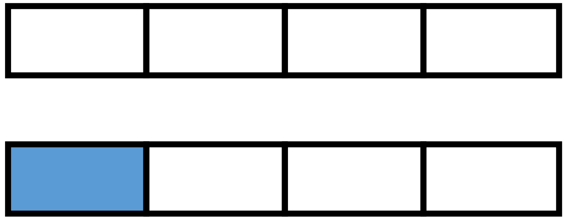


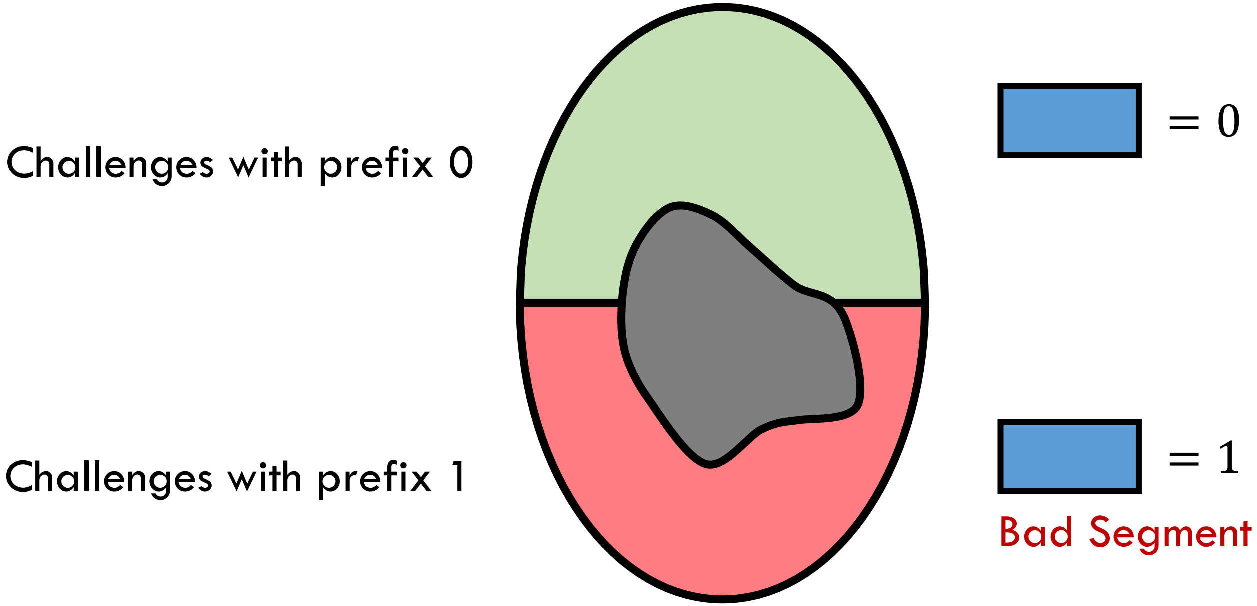
$sBAD_1$

is bad if

#bad challenges with prefix $>$ #bad challenges/2

Defining Bad Segments





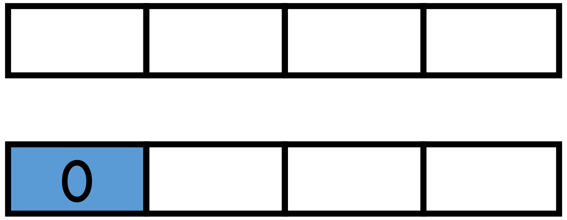
1. By pigeonhole principle, unique bad
2. ChallengeSpace polynomial size + $\mathrm{BAD}_{x,\alpha}^{(1)}$ efficiently verifiable $\Rightarrow$ sBAD_1 efficiently verifiable

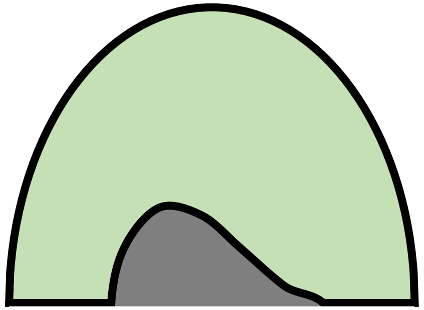
sBAD_1

is bad if

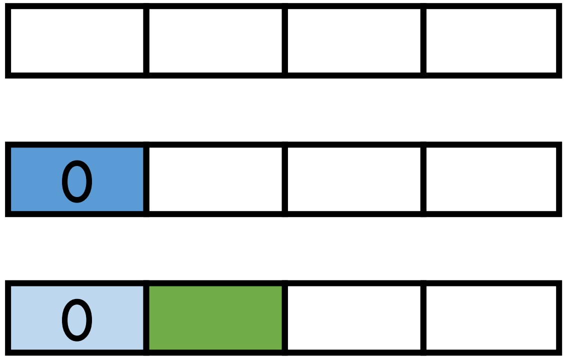
#bad challenges with prefix $>$ #bad challenges$/2$

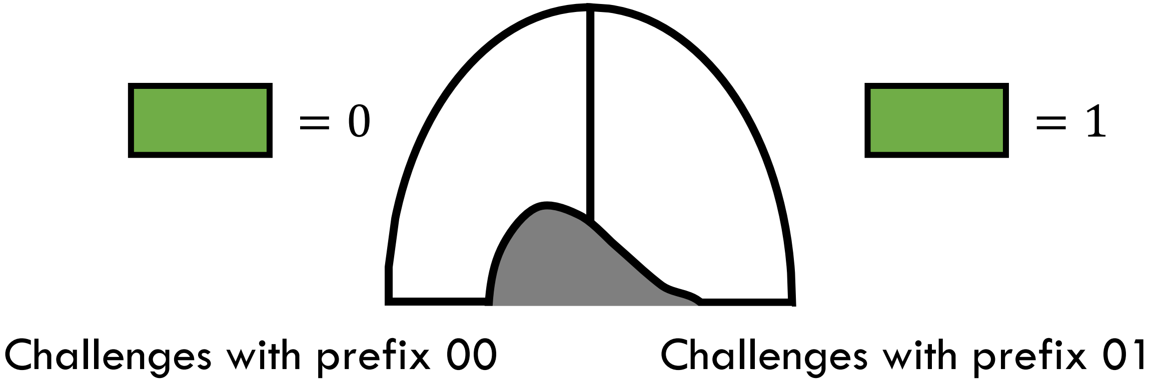
Defining Bad Segments



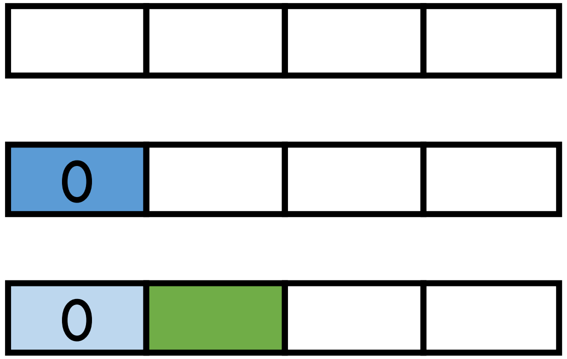


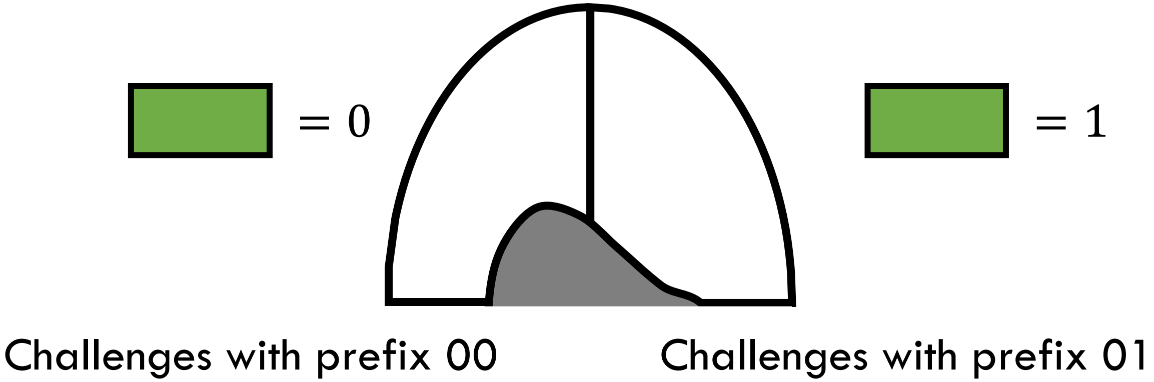
Defining Bad Segments





Defining Bad Segments

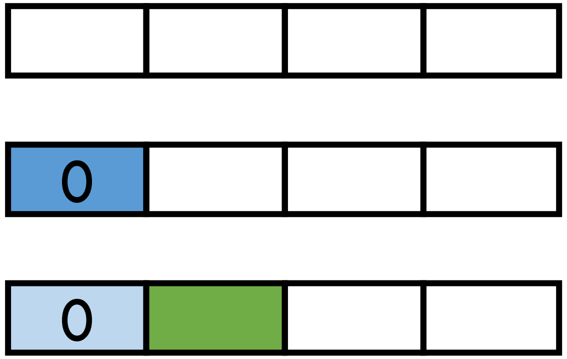


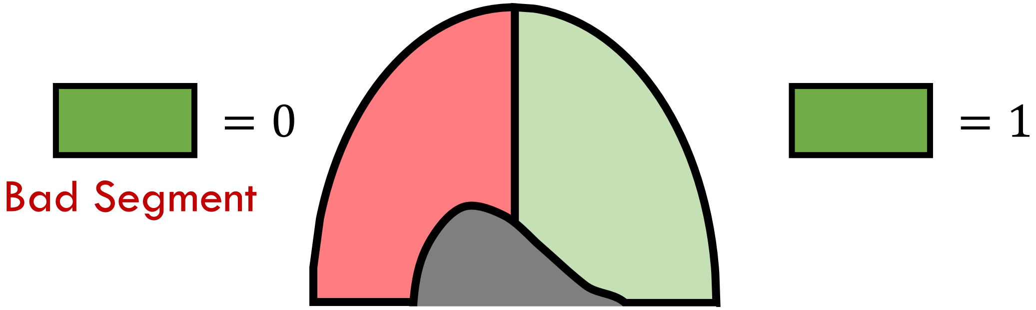


sBAD $_2$

[ ] is bad given [0] if
#bad challenges with prefix [0][ ]
> (#bad challenges with prefix [0])/2

Defining Bad Segments





sBAD_2

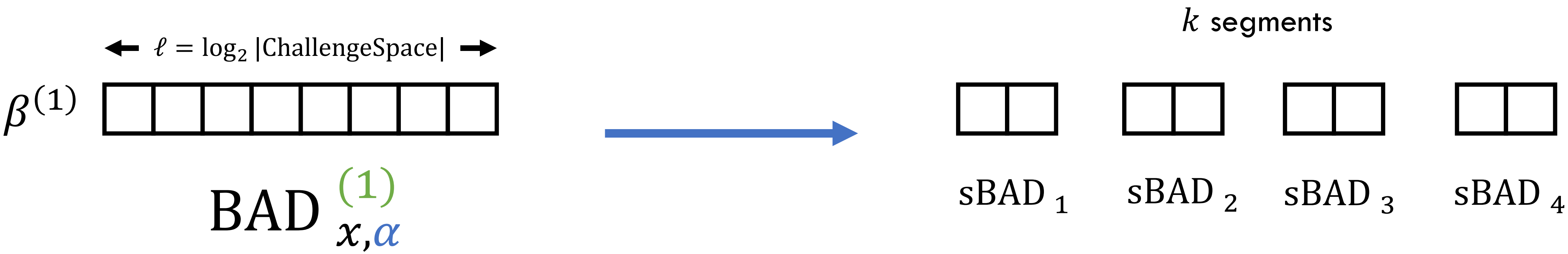
is bad given 0 if

#bad challenges with prefix 0

> (#bad challenges with prefix 0)/2

Reducing to Verifiable Unique Bad Challenge

No parallel repetition



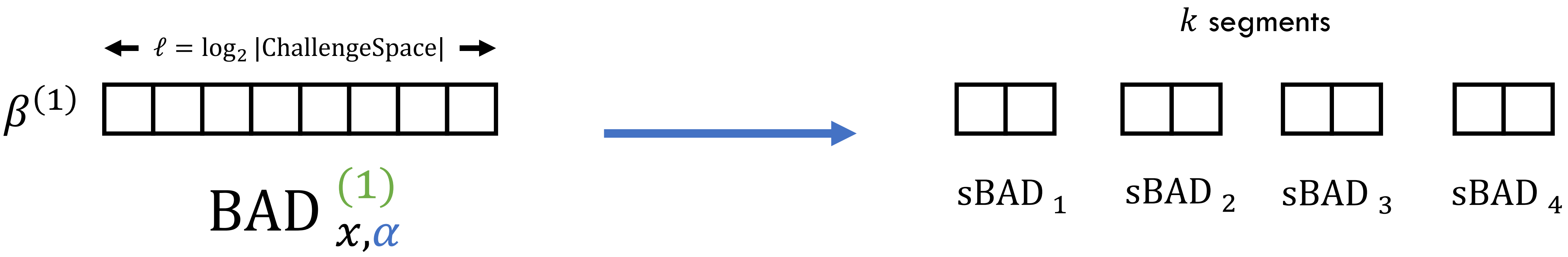
Requirements:

1. Each sBAD_j must be efficiently verifiable unique bad challenge relations.
2. If a challenge is bad, then there must exist a bad segment.

Each segment has ℓ/k bits

Reducing to Verifiable Unique Bad Challenge

No parallel repetition



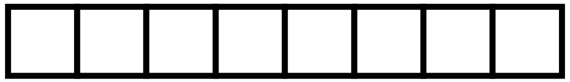
Requirements:

1. Each sBAD_j must be efficiently verifiable unique bad challenge relations.
2. If a challenge is bad, then there must exist a bad segment.

Each segment has ℓ / k bits

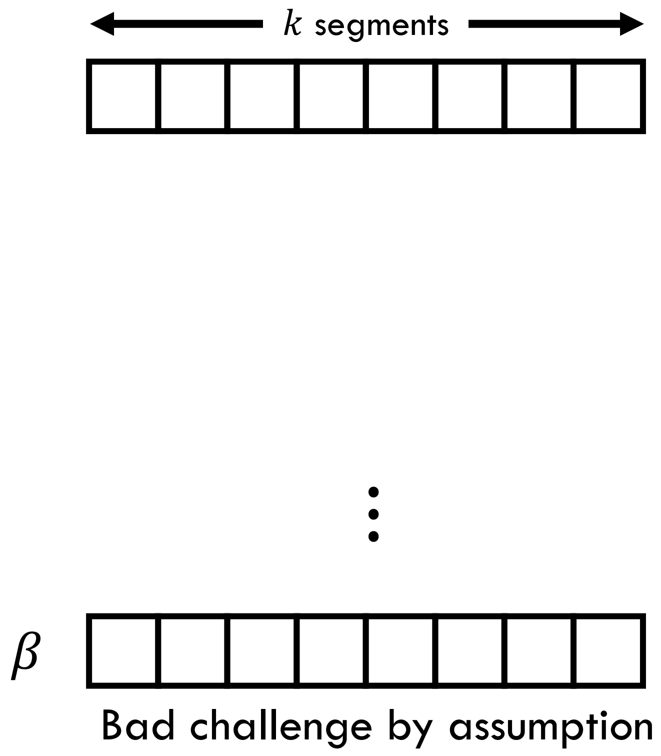
Existence of a bad segment

β



Bad challenge by assumption

Existence of a bad segment



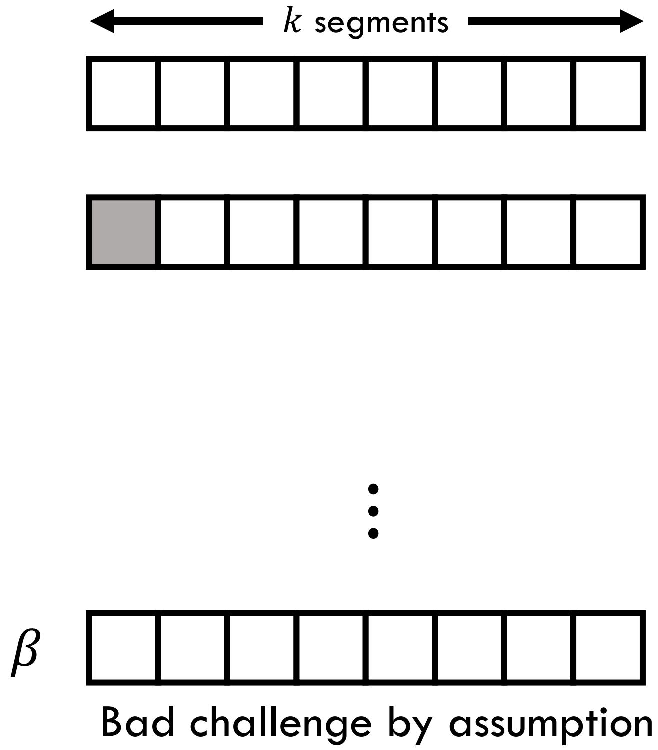
#bad challenges remaining

T

T = #bad challenges $\mathrm{BAD}_{x,\alpha}^{(1)}$

k such that $2^k > T$

Existence of a bad segment



#bad challenges remaining

T

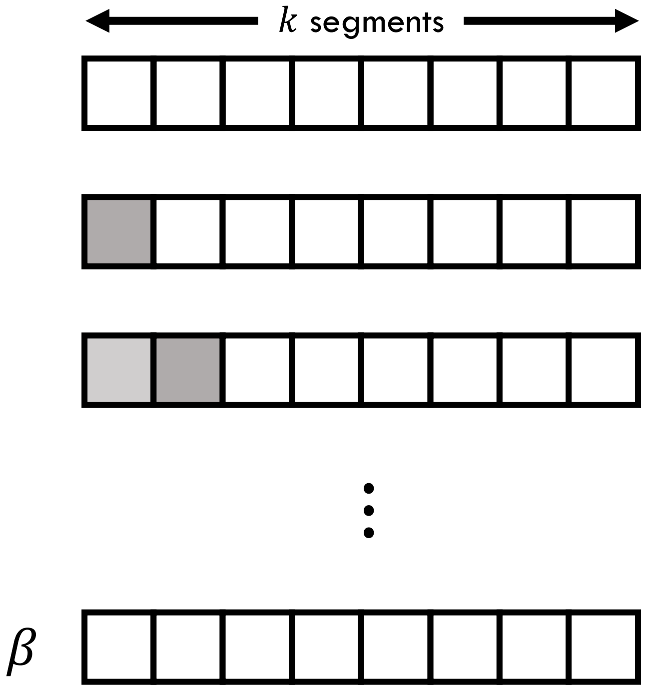
$< T/2$

If each segment is good

$T = $ #bad challenges $\mathrm{BAD}_{x,\alpha}^{(1)}$

k such that $2^k > T$

Existence of a bad segment



Bad challenge by assumption

#bad challenges remaining

T

$< T/2$

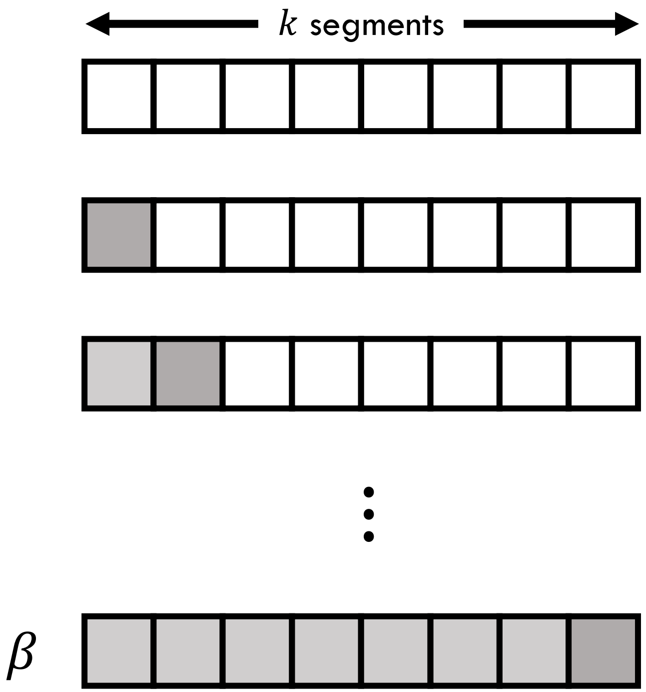
$< T/4$

If each segment is good

T = #bad challenges $\mathrm{BAD}_{x,\alpha}^{(1)}$

k such that $2^k > T$

Existence of a bad segment



Bad challenge by assumption

#bad challenges remaining

T

$< T/2$

$< T/4$

⋮

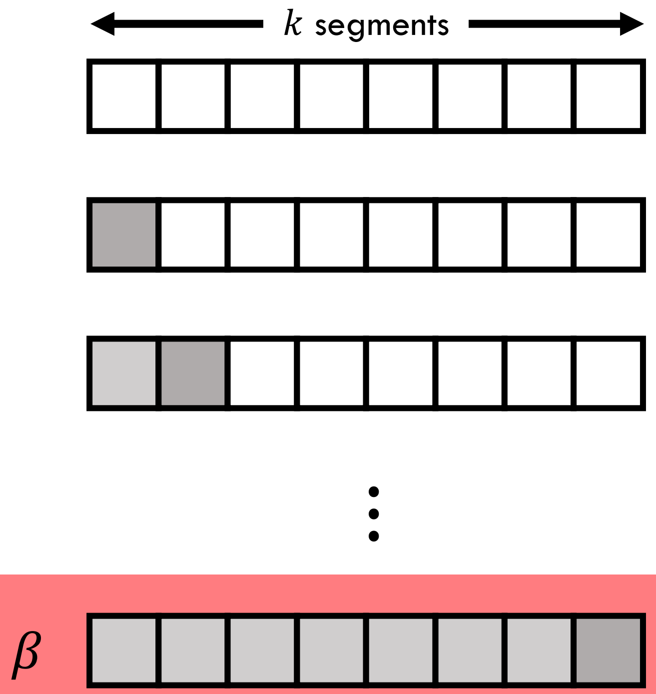
< 1

If each segment is good

$T =$ #bad challenges $\mathrm{BAD}_{x,\alpha}^{(1)}$

k such that $2^k > T$

Existence of a bad segment



Bad challenge by assumption

#bad challenges remaining

T

$< T/2$

$< T/4$

⋮

< 1

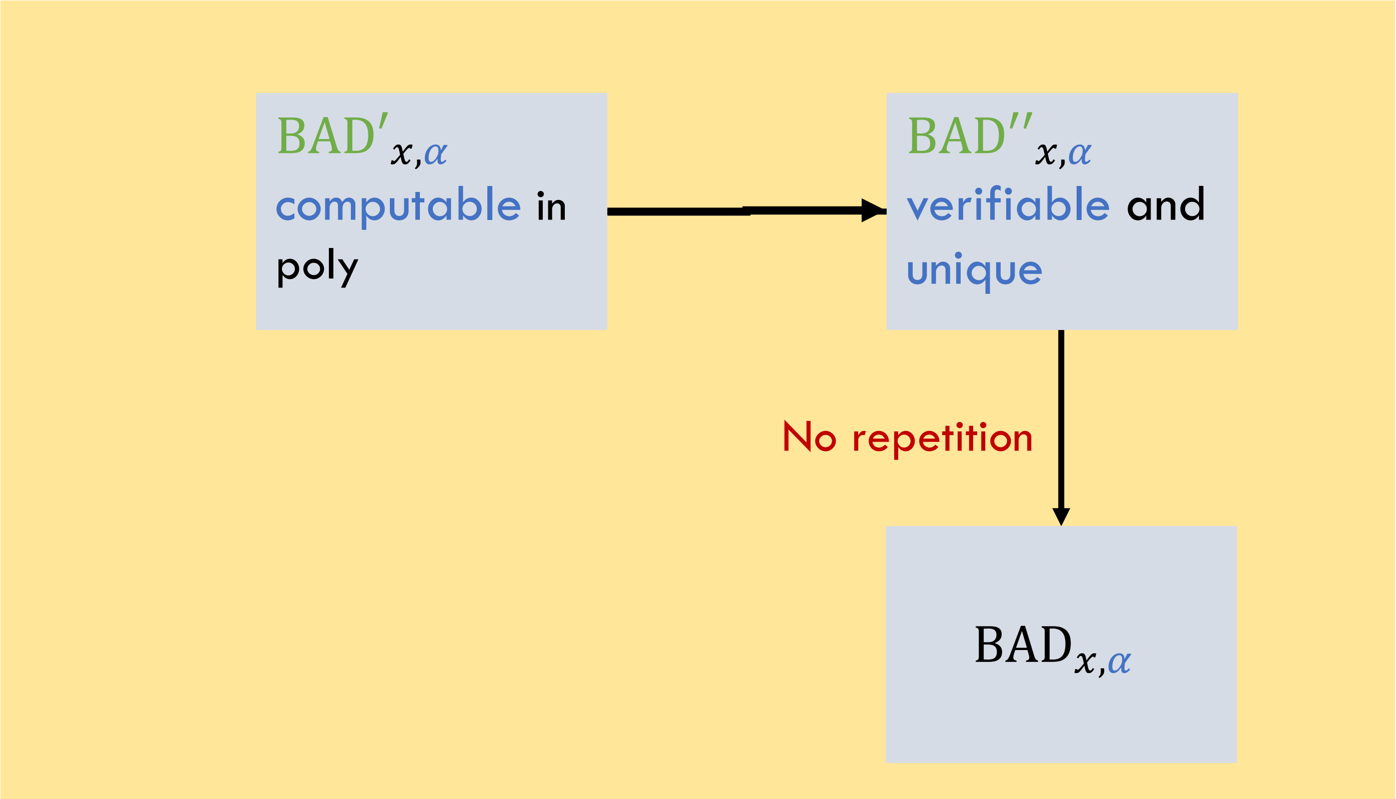
$T = \#\text{bad challenges } \mathrm{BAD}_{x,\alpha}^{(1)}$

k such that $2^k > T$

contradiction

If each segment is good

Overview So Far



$\mathrm{BAD}_{x,\alpha}$ properties

1. Bad challenges are a product set
2. Challenge space is of polynomial size
3. Bad challenges are product verifiable in poly

Concluding Remarks

See paper for:

1. Extension to parallel repetition.
2. Choice of parameters for size of segments, number of repetitions.
3. New somewhere extractable hash scheme necessary for “Magic box”.

Recap: Our Results

Theorem 1

Assuming sub-exponential hardness of DDH, there exists SNARGs for batch NP where

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

Theorem 2

Assuming sub-exponential hardness of DDH, there exists SNARGs for P where

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

Thank you. Questions?

Arka Rai Choudhuri

arkarai.choudhuri@ntt-research.com

ia.cr/2022/1486