Malicious-Secure Structure-Aware Private Set Intersection

Gayathri Garimella
Mike Rosulek
Jaspal Singh

CRYPTO 2023

Private Set Intersection (PSI)

PKC [Mea86, CT10], OT ext [PSZ14, PSSZ15, KKRT16, RR17, PRTY19, CM20, PRTY20], VOLE[RS21, RR22]…

Alice

input A

{p, r, i, v, a, t, e}

special case of two-party secure computation

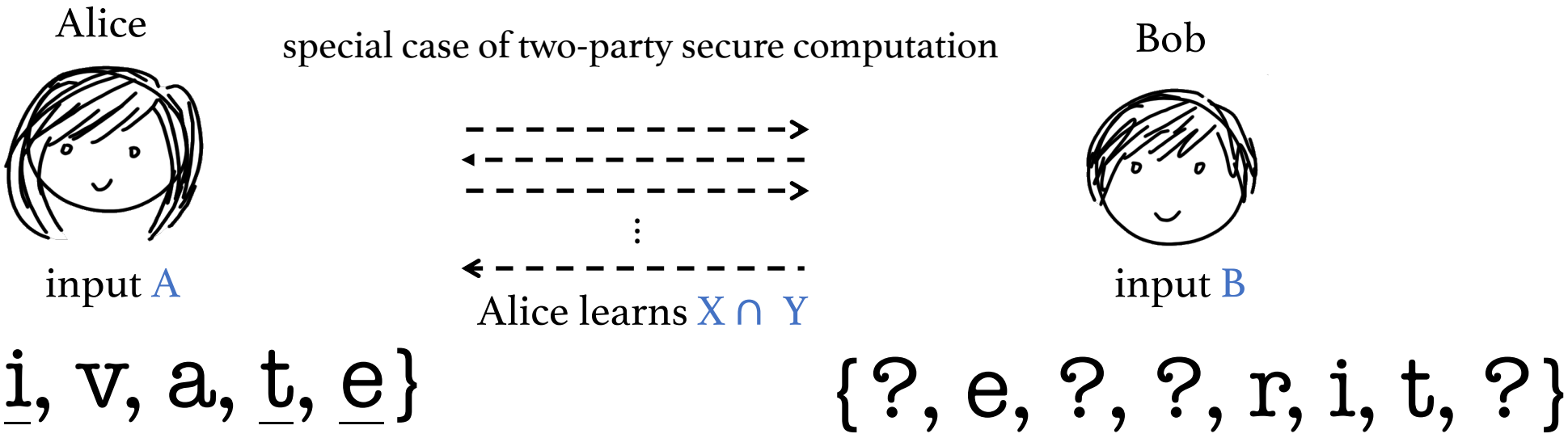
Bob

input B

{s, e, c, u, r, i, t, y}

Private Set Intersection (PSI)

PKC [Mea86, CT10], OT ext [PSZ14, PSSZ15, KKRT16, RR17, PRTY19, CM20, PRTY20], VOLE[RS21, RR22]…



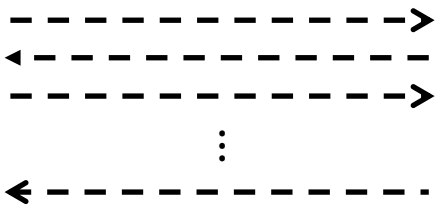
Private Set Intersection (PSI)

PKC [Mea86, CT10], OT ext [PSZ14, PSSZ15, KKRT16, RR17, PRTY19, CM20, PRTY20], VOLE[RS21, RR22]…

Alice

input A

special case of two-party secure computation



Alice learns X ∩ Y

Bob

input B

{p, r, i, v, a, t, e}

{?, e, ?, ?, r, i, t, ?}

Bob learns nothing extra

structure-aware Private Set Intersection (sa-PSI)

a variant of PSI where Alice's input has a publicly known structure

Examples - interval, ball or union of balls in some metric space, ...

input A

input B

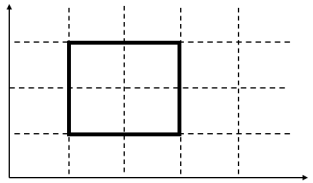
Structure-aware Private Set Intersection, With Applications to Fuzzy Matching. CRYPTO 2022.
Gayathri Garimella, Mike Rosulek and Jaspal Singh

structure-aware Private Set Intersection (sa-PSI)

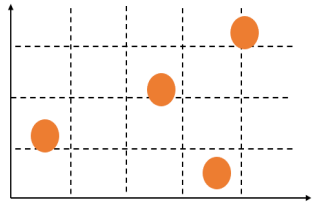
a variant of PSI where Alice's input has a publicly known structure

Examples - interval, ball or union of balls in some metric space, ...

input A



input B



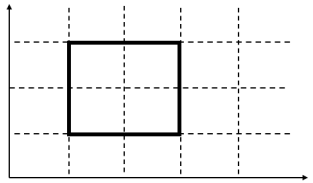
Structure-aware Private Set Intersection, With Applications to Fuzzy Matching. CRYPTO 2022.
Gayathri Garimella, Mike Rosulek and Jaspal Singh

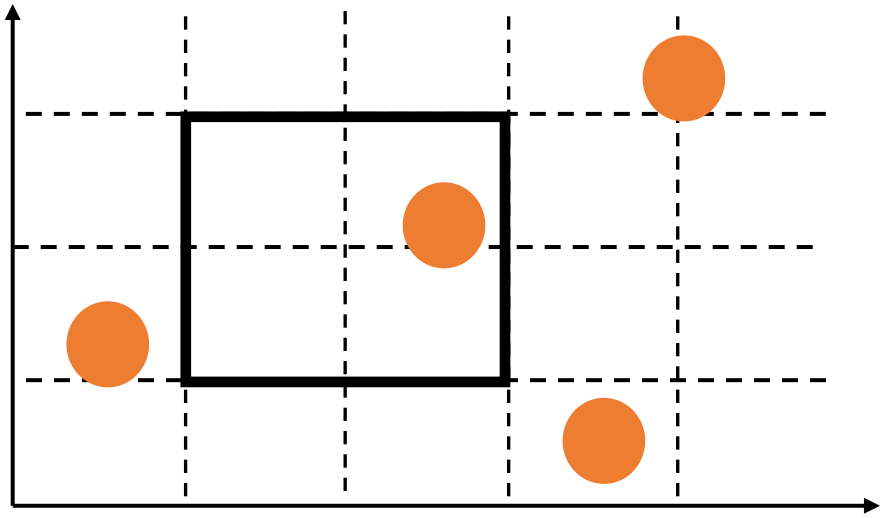
structure-aware Private Set Intersection (sa-PSI)

a variant of PSI where Alice's input has a publicly known structure

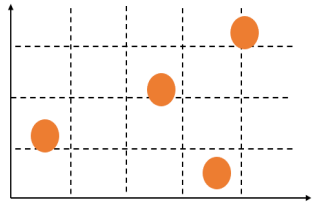
Examples - interval, ball or union of balls in some metric space, …

input A





input B

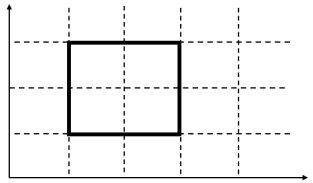


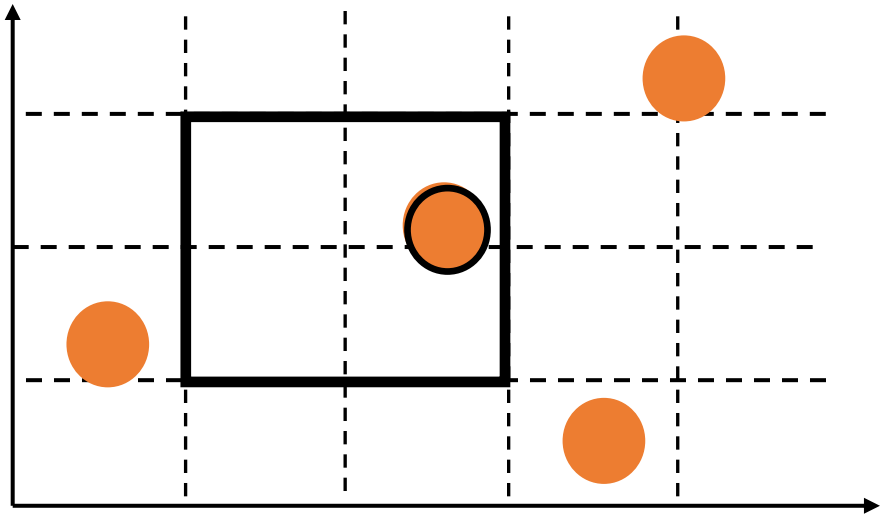
structure-aware Private Set Intersection (sa-PSI)

a variant of PSI where Alice's input has a publicly known structure

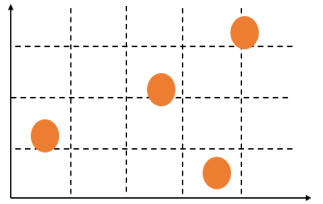
Examples - interval, ball or union of balls in some metric space, …

input A





input B



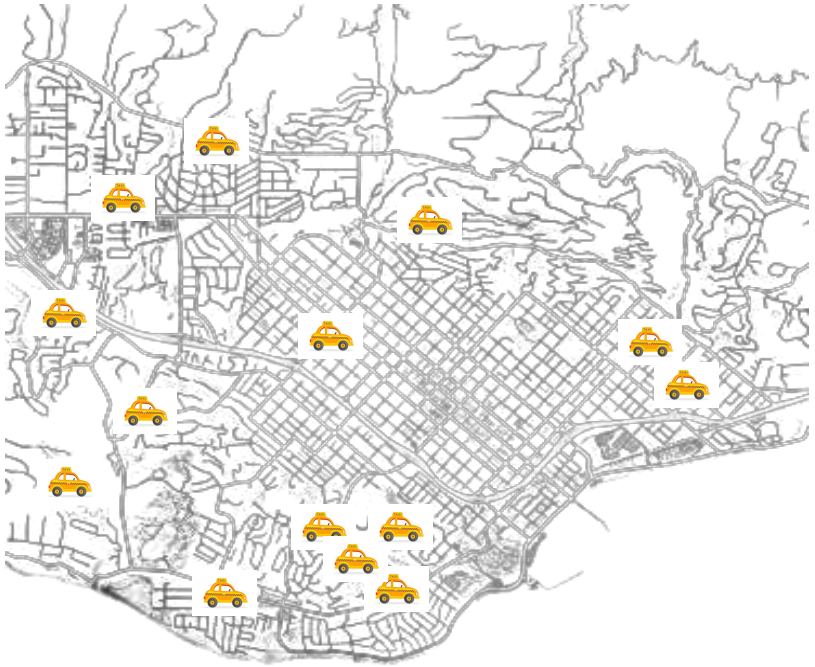
Alice learns X ∩ Y = {all points inside square}

Motivating example

privacy-preserving ride hailing service

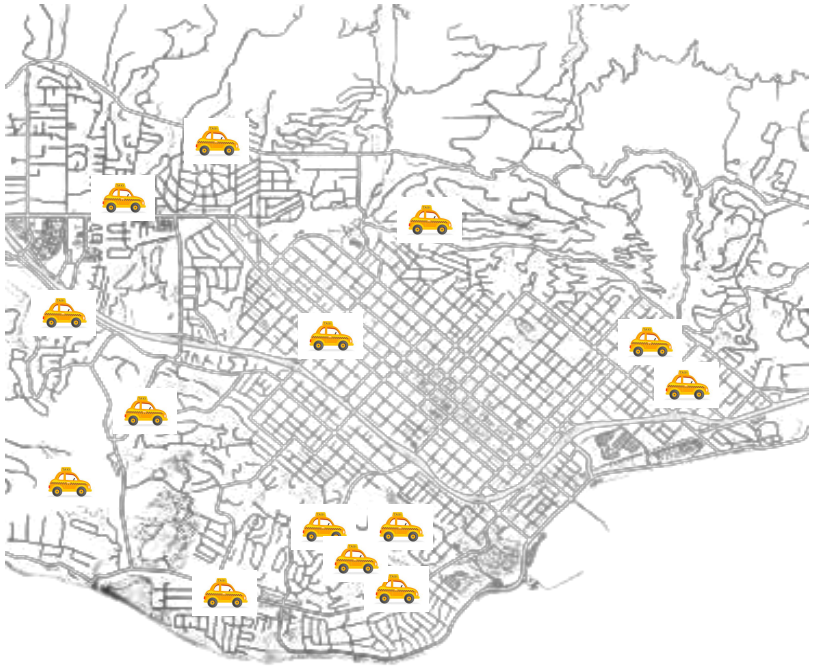
Motivating example

privacy-preserving ride hailing service



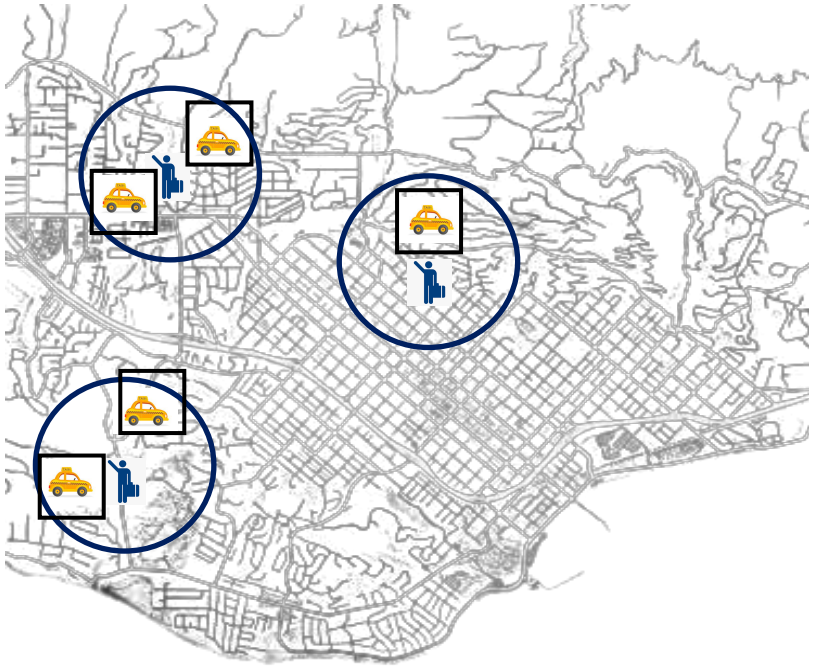
Motivating example

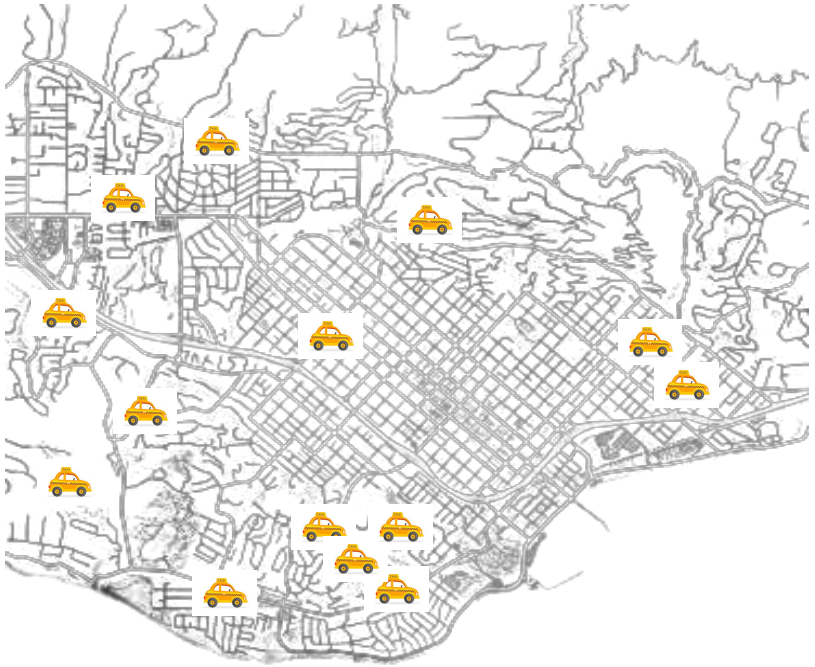
privacy-preserving ride hailing service



Motivating example

privacy-preserving ride hailing service





Previous work

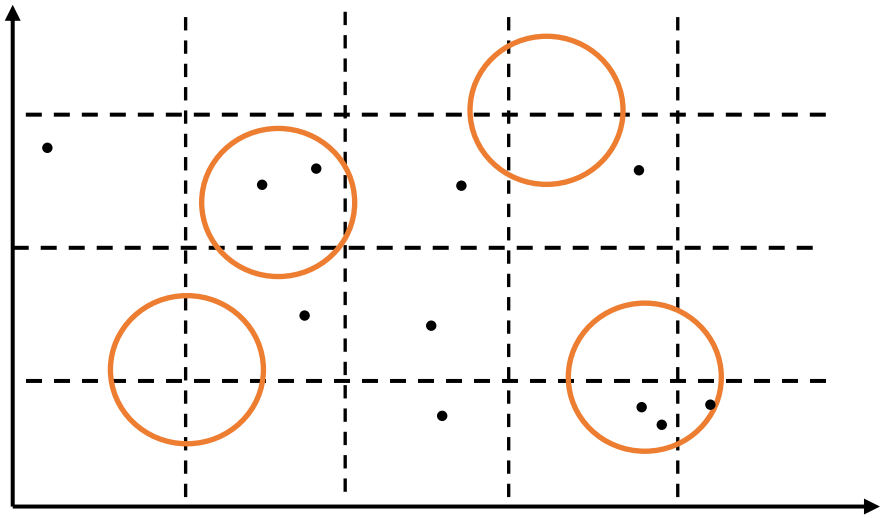
- Naïve solution: Alice's enumerates her structured input A and runs plain PSI

communication cost $O((|A| + |B|).\kappa)$

Previous work

- Naïve solution: Alice's enumerates her structured input A and runs plain PSI

communication cost $O((|A| + |B|).\kappa)$

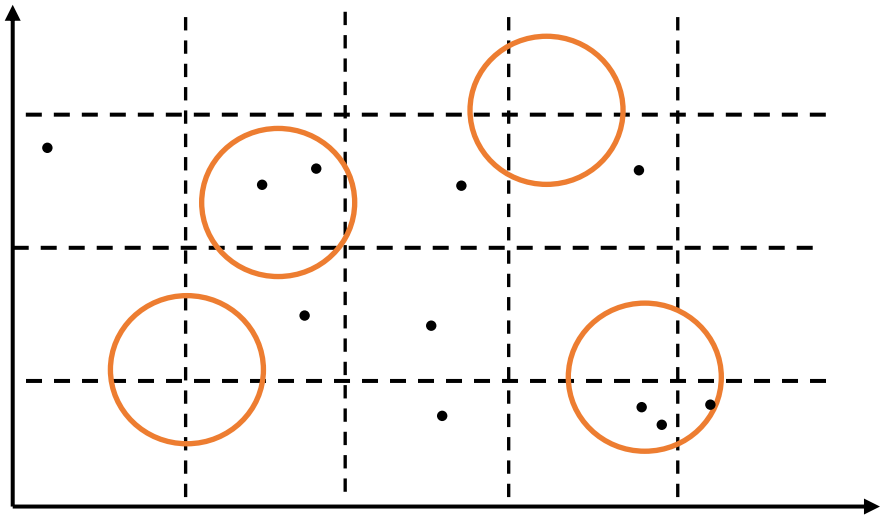


communication scales with total volume $|A|$ of balls in Alice's input

Previous work

- Naïve solution: Alice's enumerates her structured input A and runs plain PSI

communication cost $O((|A| + |B|).\kappa)$



instead, can communication scale with # of balls (description size) of Alice's input?

Previous work

- Naïve solution: Alice's enumerates her structured input A and runs plain PSI

$$\text{communication cost } O((|A| + |B|).\kappa)$$

- [GarimellaRosulekSingh'22]: If Alice's input A can be compactly represented by boolean Function Secret Sharing with share size σ , we get semi-honest structure-aware PSI

$$\text{communication cost } O((\sigma + |B|).\kappa)$$

Previous work

- Naïve solution: Alice's enumerates her structured input A and runs plain PSI

$$\text{communication cost } O((|A| + |B|).\kappa)$$

- [GarimellaRosulekSingh'22]: If Alice's input A can be compactly represented by boolean Function Secret Sharing with share size σ , we get semi-honest structure-aware PSI

$$\text{communication cost } O((\sigma + |B|).\kappa)$$

boolean Function Secret Sharing (bFSS)

[BoyleGilboaIshai15] – style FSS for set membership in A function

Previous work

- Naïve solution: Alice's enumerates her structured input A and runs plain PSI

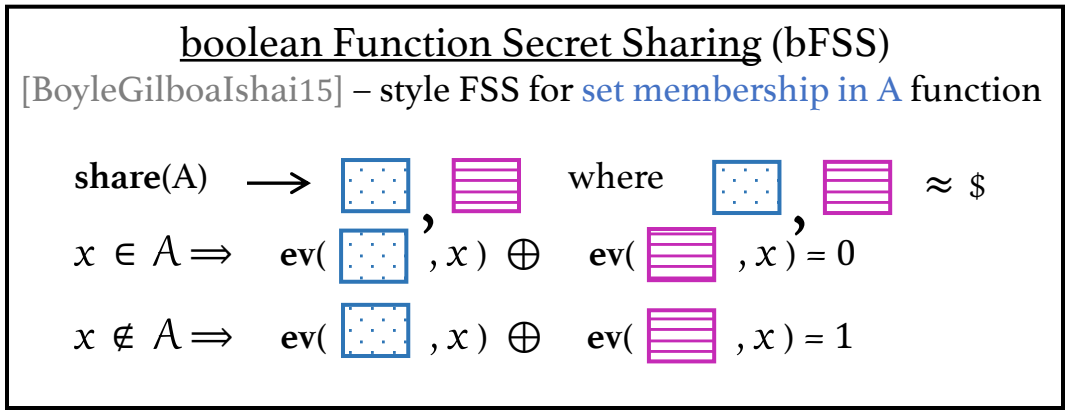
$$\text{communication cost } O((|A| + |B|).\kappa)$$

- [GarimellaRosulekSingh'22]: If Alice's input A can be compactly represented by boolean Function Secret Sharing with share size σ , we get semi-honest structure-aware PSI

$$\text{communication cost } O((\sigma + |B|).\kappa)$$

boolean Function Secret Sharing (bFSS)

[BoyleGilboaIshai15] – style FSS for set membership in A function



Research Question

Can we design malicious-secure structure-aware PSI?

Research Question

Can we design malicious-secure structure-aware PSI?

Why is [GRS'22] not malicious-secure?

How can we fix this?

How does [GRS'22] work?

assumptions: OT-hybrid (Oblivious Transfer[Rabin81]), random oracle model

input A

1. generates κ instances of bFSS shares

input B

How does [GRS'22] work?

assumptions: OT-hybrid (Oblivious Transfer[Rabin81]), random oracle model

input A

1. generates κ instances of bFSS shares

input B

$\mathbf{share}(A) \longrightarrow$ 

$\mathbf{share}(A) \longrightarrow$ 

$\vdots$

$\mathbf{share}(A) \longrightarrow$ 

How does [GRS'22] work?

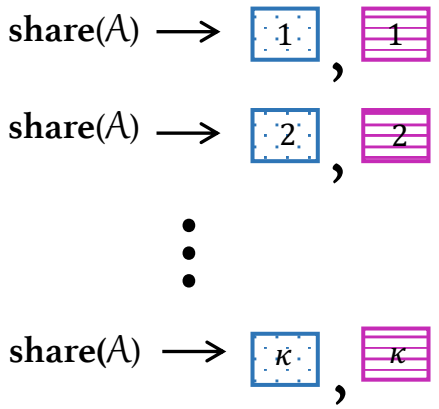
assumptions: OT-hybrid (Oblivious Transfer[Rabin81]), random oracle model

input A

1. generates κ instances of bFSS shares

input B

2. picks κ choice bits to learn  or 



How does [GRS'22] work?

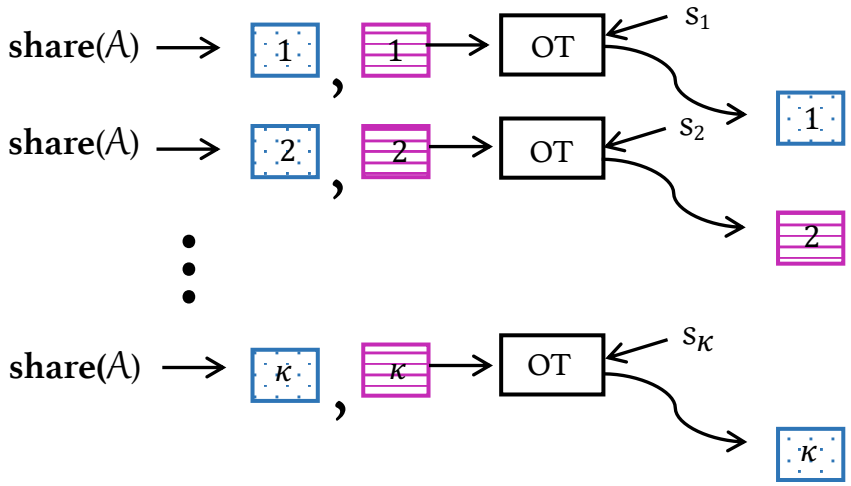
assumptions: OT-hybrid (Oblivious Transfer[Rabin81]), random oracle model

input A

1. generates κ instances of bFSS shares

input B

2. picks κ choice bits to learn  or 



How does [GRS'22] work?

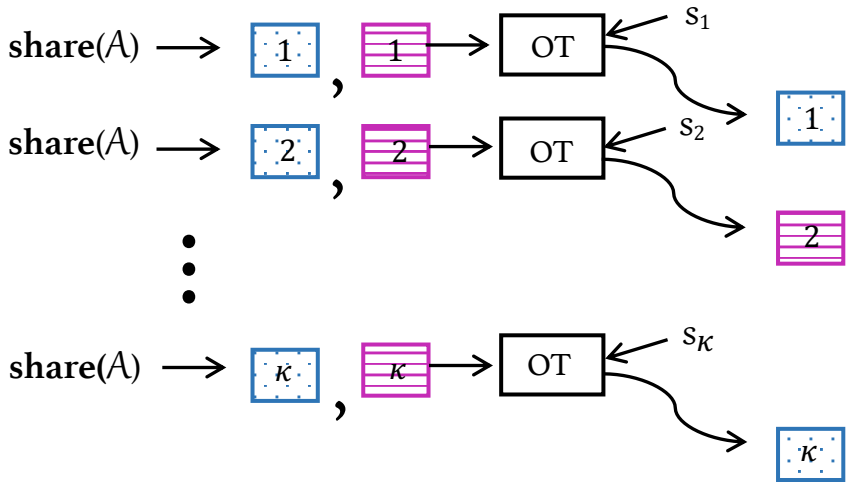
assumptions: OT-hybrid (Oblivious Transfer[Rabin81]), random oracle model

input A

1. generates κ instances of bFSS shares

input B

2. picks κ choice bits to learn  or 



$$F(x) = \mathbf{H}(\mathbf{ev}(\boxed{1}, x) \| \mathbf{ev}(\boxed{2}, x) \| \cdots \mathbf{ev}(\boxed{\kappa}, x))$$

How does [GRS'22] work?

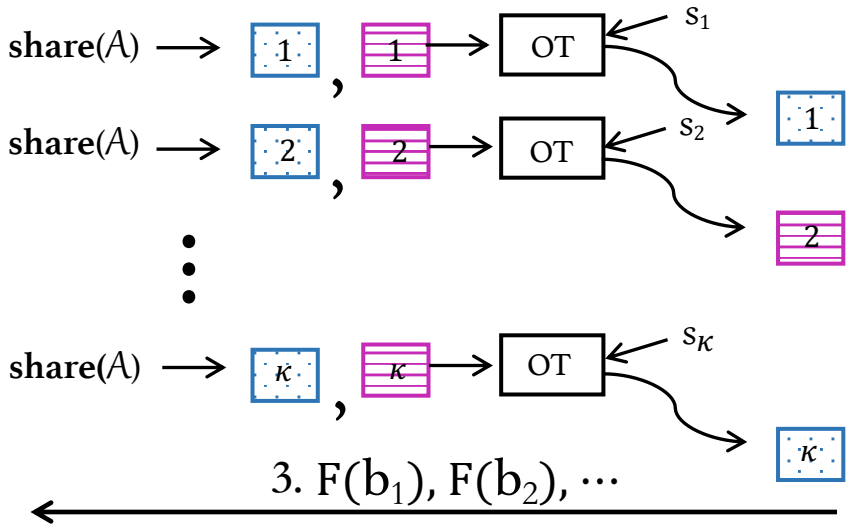
assumptions: OT-hybrid (Oblivious Transfer[Rabin81]), random oracle model

input A

1. generates κ instances of bFSS shares

input B

2. picks κ choice bits to learn  or 



Bob computes $F(x)$ on all his inputs

$$F(x) = \mathbf{H}(\mathbf{ev}(\boxed{1}, x) \| \mathbf{ev}(\boxed{2}, x) \| \cdots \mathbf{ev}(\boxed{\kappa}, x))$$

How does [GRS'22] work?

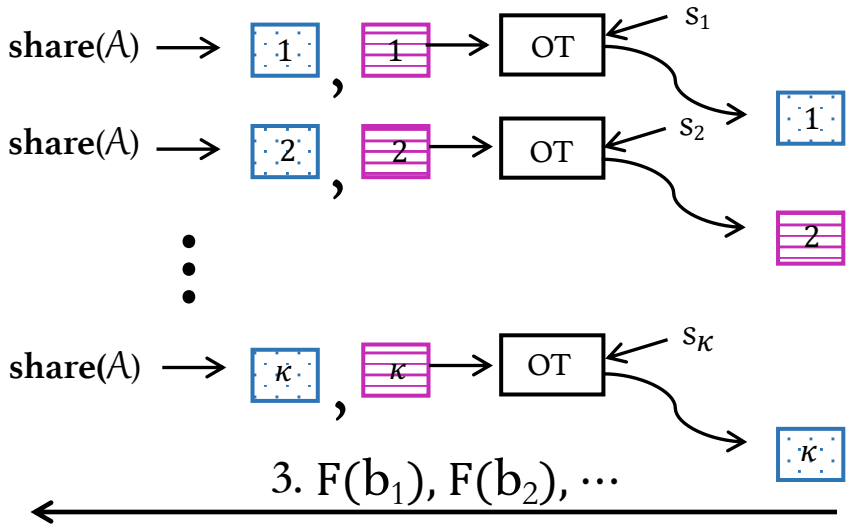
assumptions: OT-hybrid (Oblivious Transfer[Rabin81]), random oracle model

input A

1. generates κ instances of bFSS shares

input B

2. picks κ choice bits to learn  or 



if $(x \in A)$ $\Rightarrow$ Alice can compute $F(x)$

if $(x \notin A)$ $\Rightarrow$ $F(x) \approx \$\$$

$x \in A \implies \mathbf{ev}($  $, x) = \mathbf{ev}($  $, x)$

Bob computes $F(x)$ on all his inputs

$F(x) = \mathbf{H}(\mathbf{ev}($  $, x) \| \mathbf{ev}($  $, x) \| \cdots \mathbf{ev}($  $, x))$

How does [GRS'22] work?

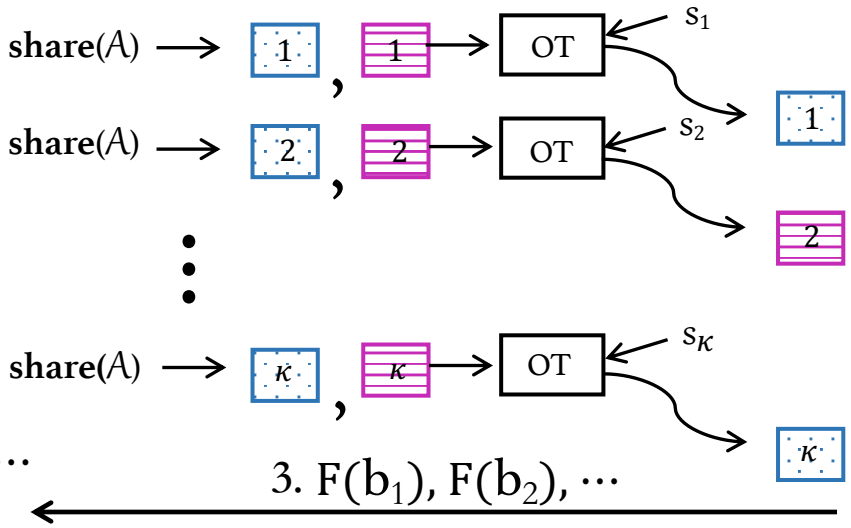
assumptions: OT-hybrid (Oblivious Transfer[Rabin81]), random oracle model

input A

1. generates κ instances of bFSS shares

input B

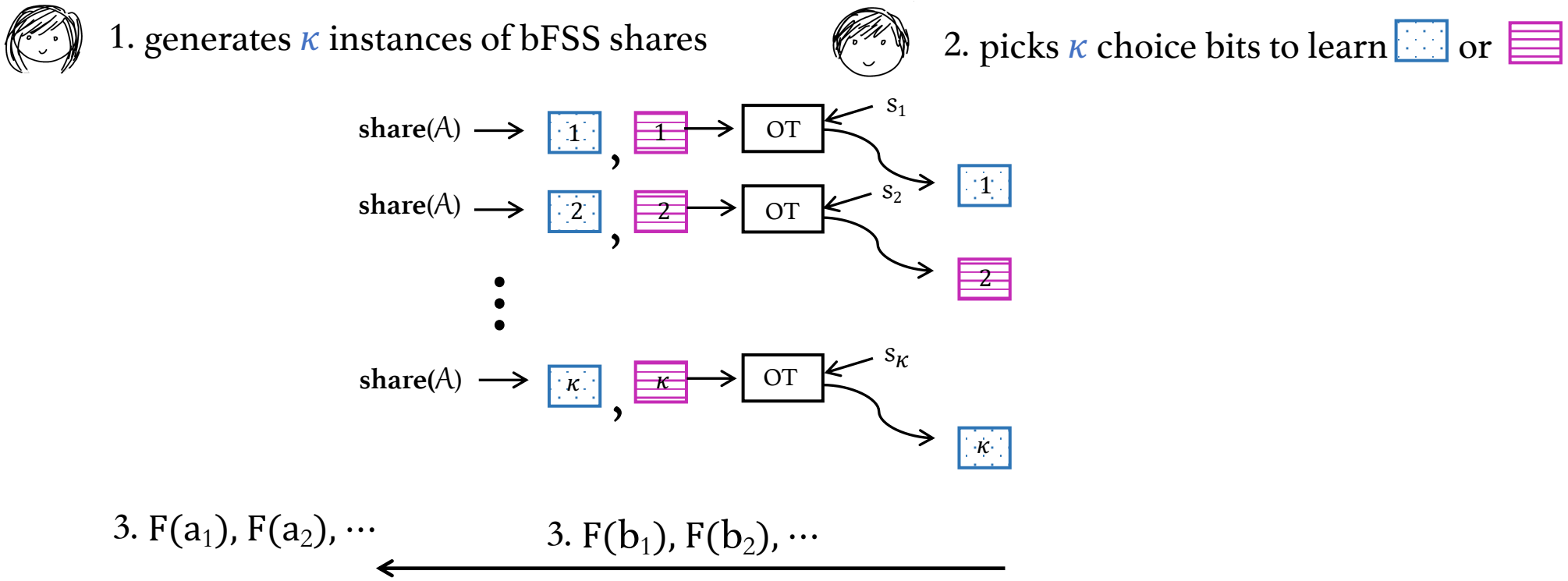
2. picks κ choice bits to learn  or 



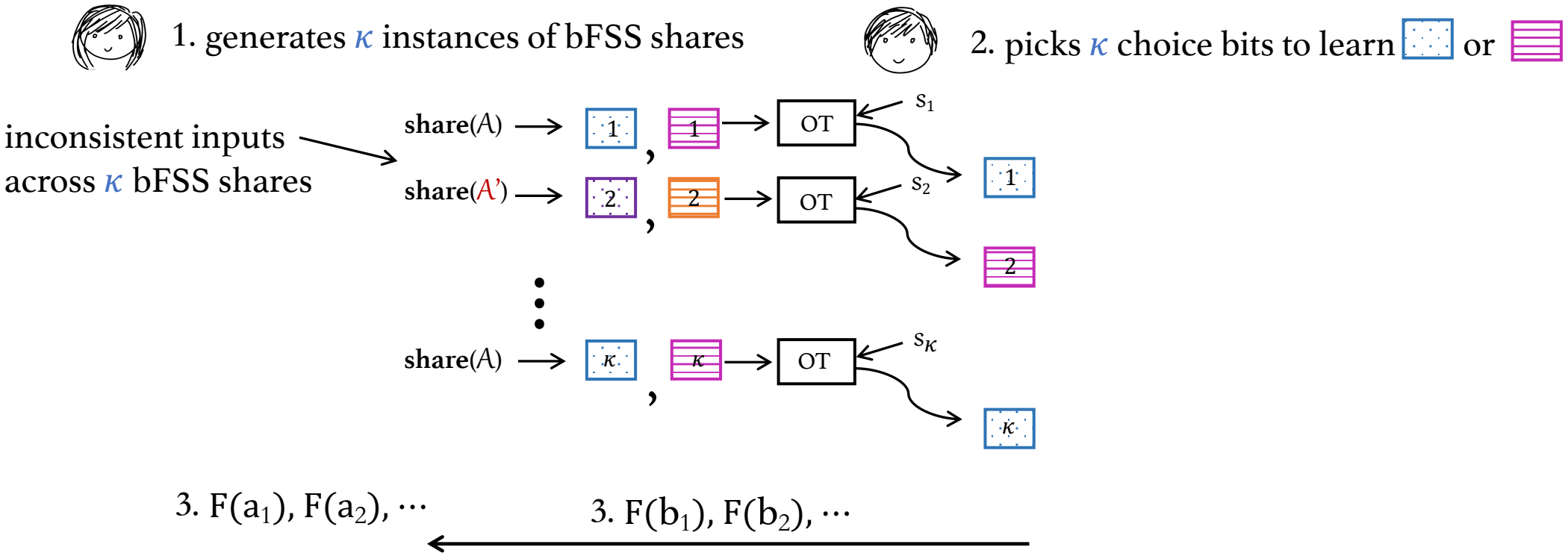
Bob computes $F(x)$ on all his inputs

$$F(x) = \mathbf{H}(\mathbf{ev}(\boxed{1}, x) \| \mathbf{ev}(\boxed{2}, x) \| \cdots \mathbf{ev}(\boxed{\kappa}, x))$$

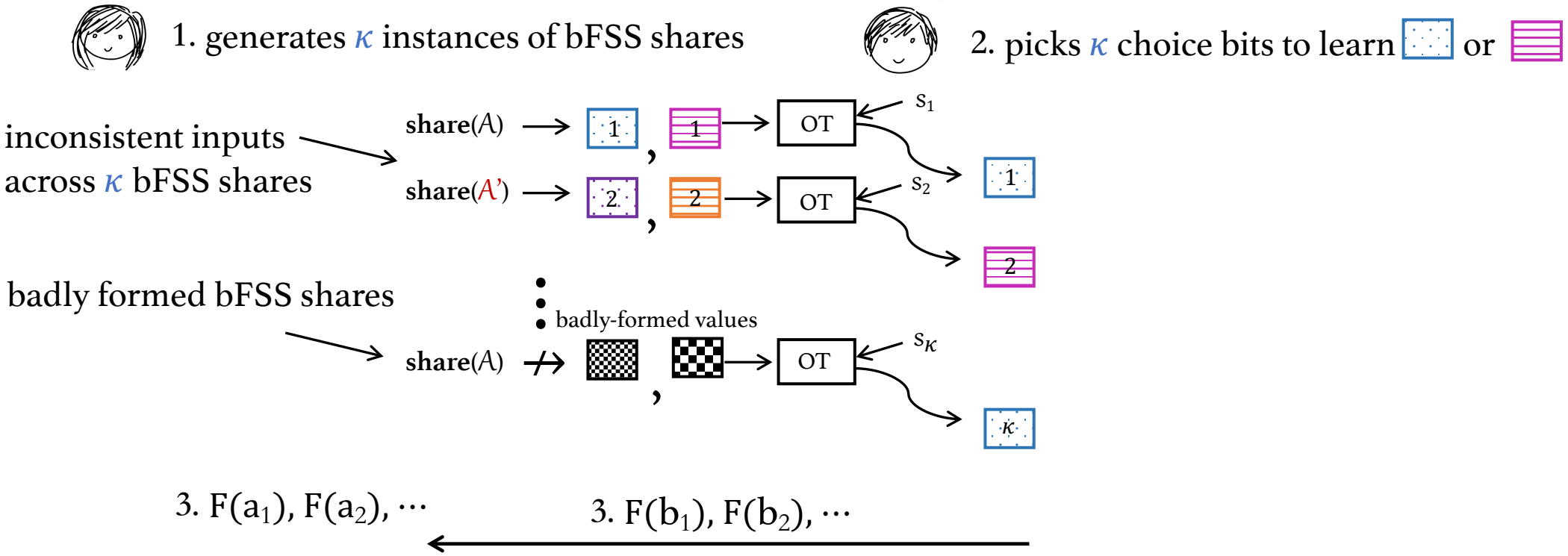
Why is [GRS'22] not malicious-secure?



Why is [GRS'22] not malicious-secure?

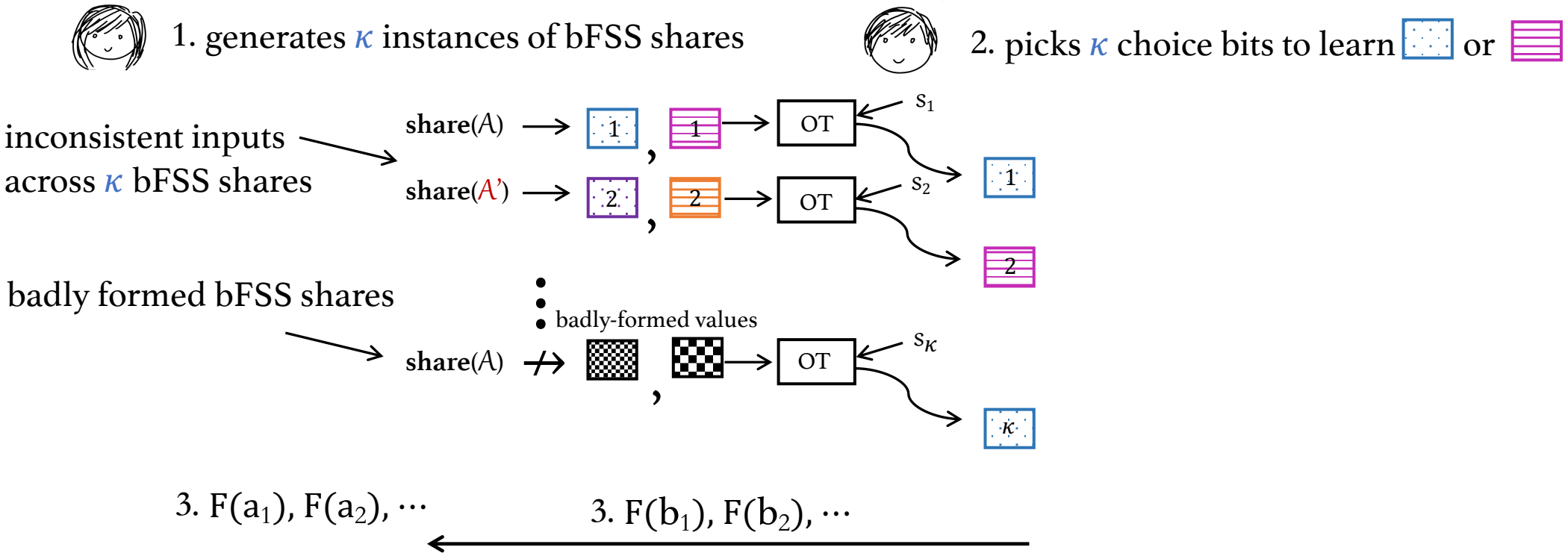


Why is [GRS'22] not malicious-secure?



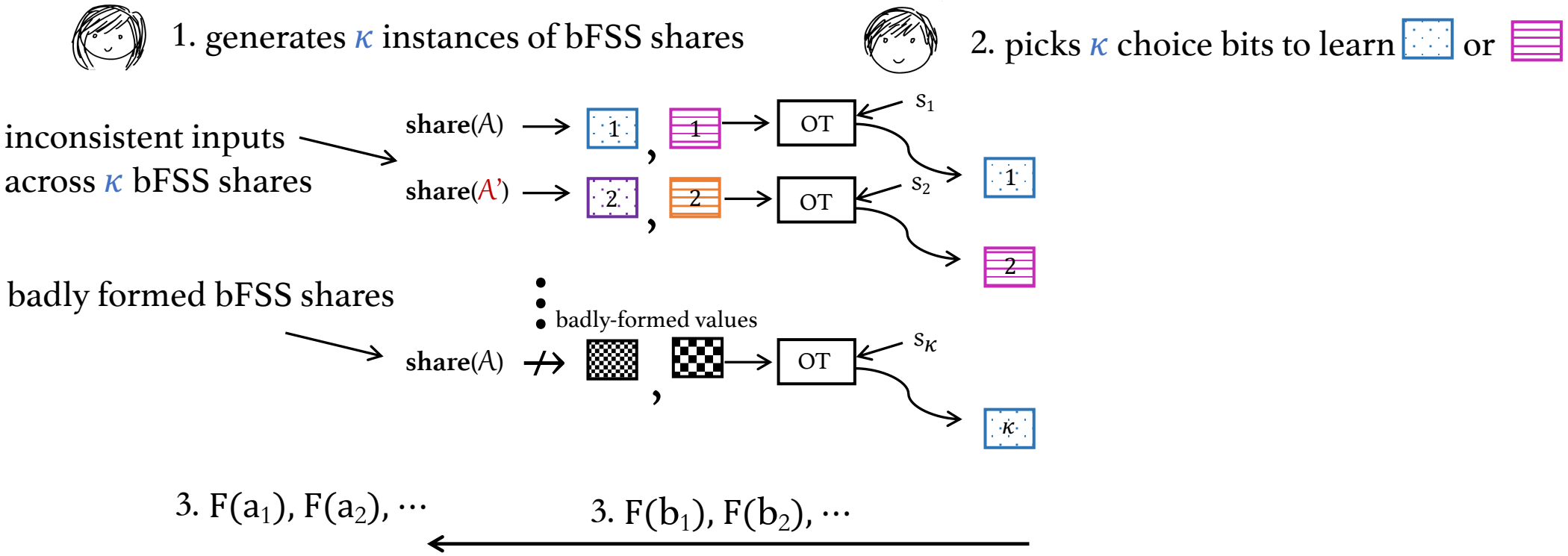
Why is [GRS'22] not malicious-secure?

malicious-secure sa-PSI reduces to consistency-of-bFSS problem



Why is [GRS'22] not malicious-secure?

Challenge: enforce that Alice encodes same valid set across all κ bFSS shares



Attempt 1: Let's Cut-and-Choose

Cut-and-choose: allow Bob to open and verify both bFSS shares for some subset of 't' instances

Attempt 1: Let's Cut-and-Choose

Cut-and-choose: allow Bob to open and verify both bFSS shares for some subset of 't' instances

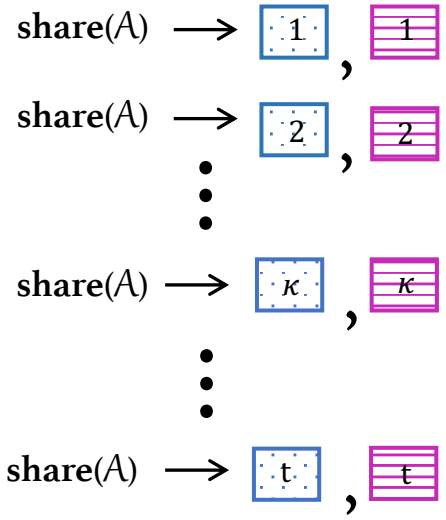
1. pick subset **check** $\subset$ [t]

Attempt 1: Let's Cut-and-Choose

Cut-and-choose: allow Bob to open and verify both bFSS shares for some subset of 't' instances

2. generates t bFSS shares

1. pick subset **check** $\subset$ [t]

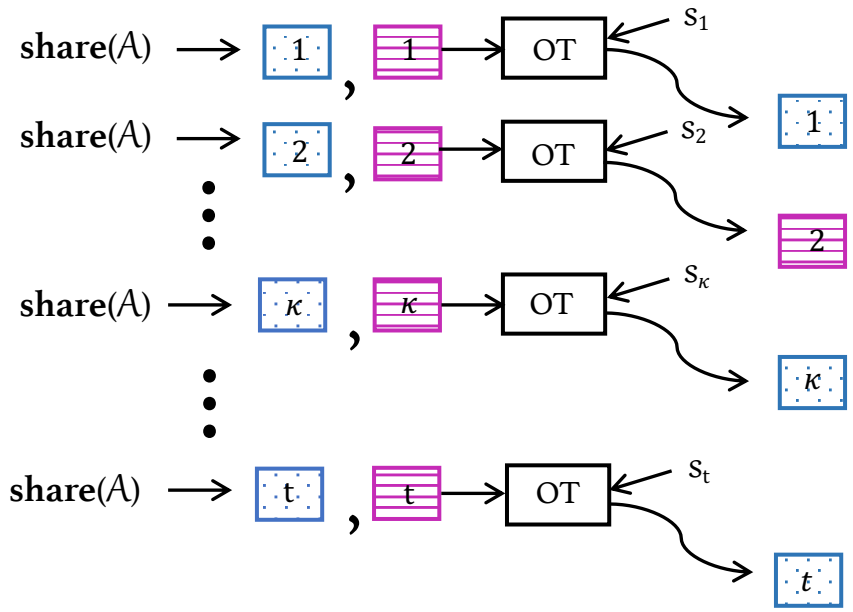


Attempt 1: Let's Cut-and-Choose

Cut-and-choose: allow Bob to open and verify both bFSS shares for some subset of 't' instances

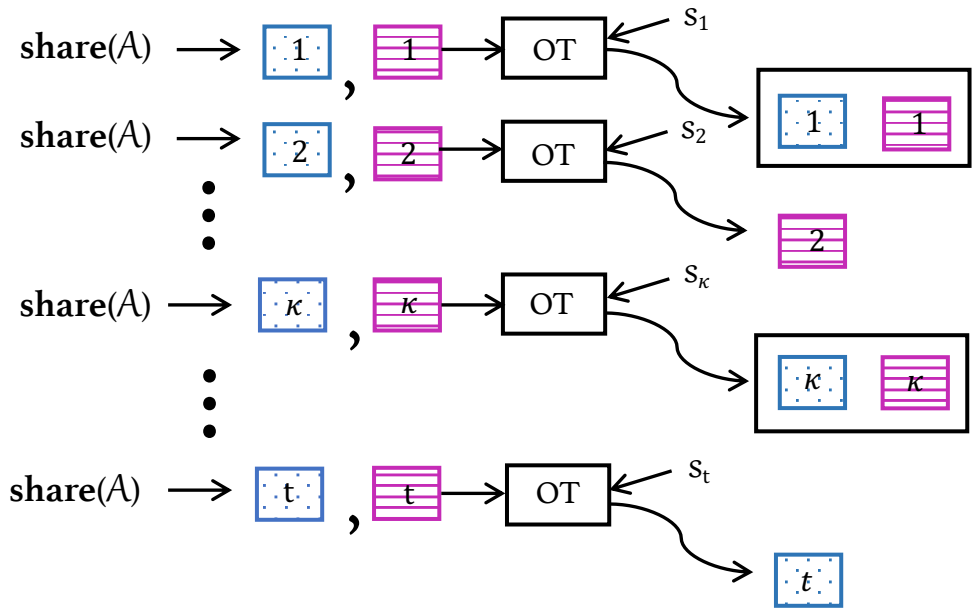

2. generates t bFSS shares


1. pick subset **check** $\subset$ [t]



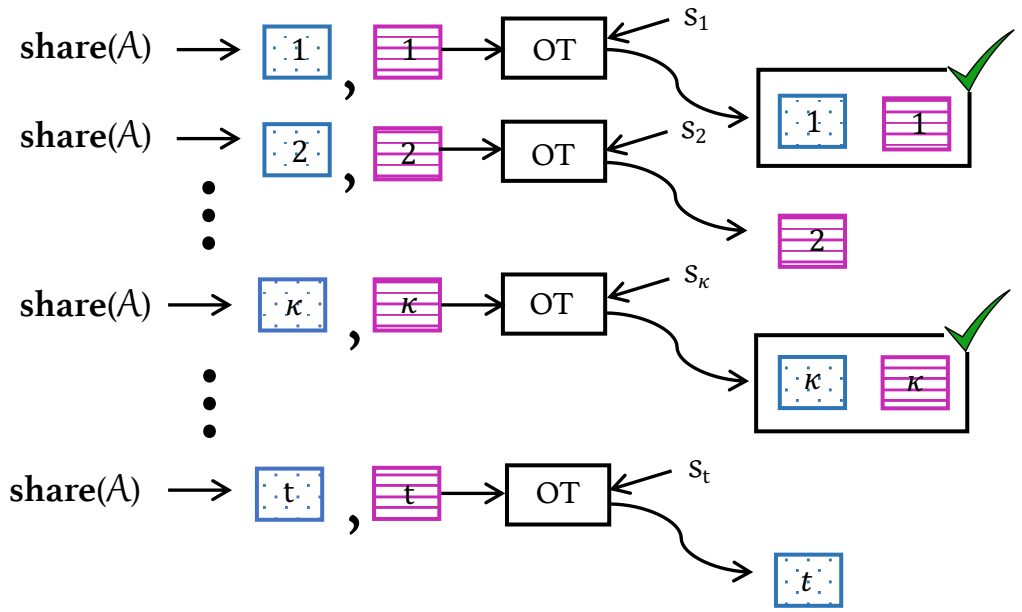
Attempt 1: Let's Cut-and-Choose

Cut-and-choose: allow Bob to open and verify both bFSS shares for some subset of 't' instances



Attempt 1: Let's Cut-and-Choose

Cut-and-choose: allow Bob to open and verify both bFSS shares for some subset of 't' instances



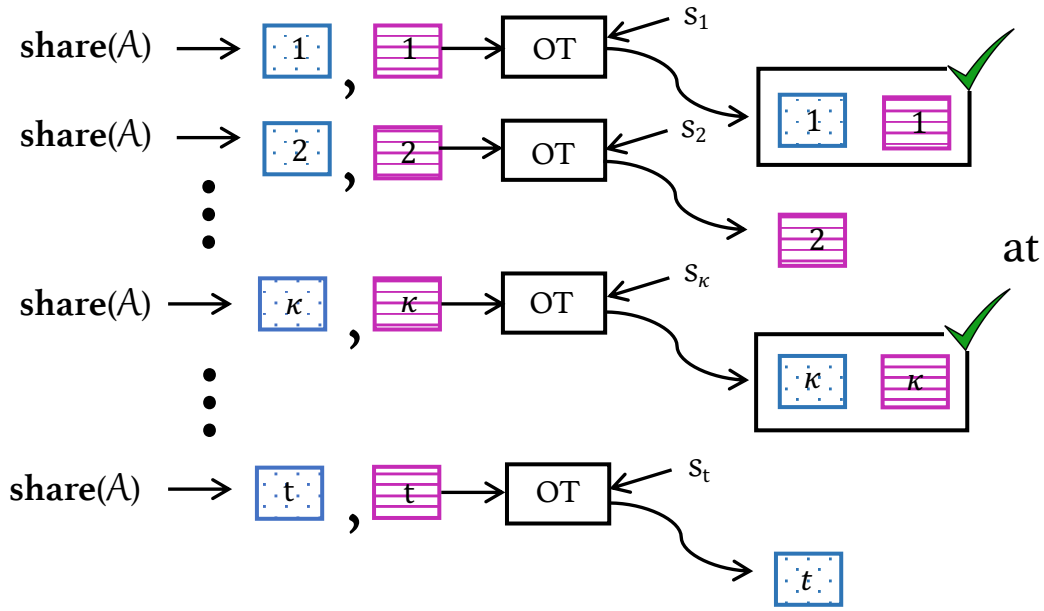
Attempt 1: Let's Cut-and-Choose

Cut-and-choose: allow Bob to open and verify both bFSS shares for some subset of 't' instances


2. generates t bFSS shares

3. reveal **check**


1. pick subset **check** $\subset$ [t]



cut-and-choose parameters:
at least κ unopened shares are valid


4. checks if : { **share**$^{-1}$(1 , 1) = **share**$^{-1}$(i , i) = $\cdots$}
all shares of **check indices** encode the **same** underlying input set

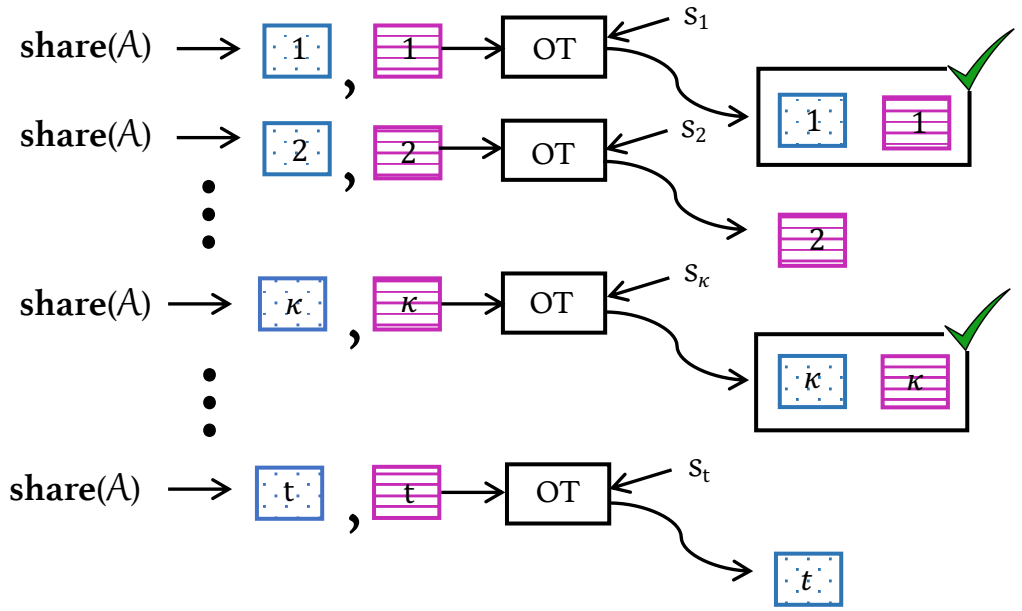
Attempt 1: Let's Cut-and-Choose

Cut-and-choose: allow Bob to open and verify both bFSS shares for some subset of 't' instances


2. generates *t* bFSS shares

3. reveal **check**


1. pick subset **check** ⊂ [t]



Bob can re-construct Alice's input
violates **input privacy!!!**


4. checks if : { **share**$^{-1}$(1 , 1) = **share**$^{-1}$(i , i) = ⋯}
all shares of **check indices** encode the **same** underlying input set

1. How does Bob verify that a single bFSS is valid; without revealing Alice's input?

1. How does Bob verify that a single bFSS is valid; without revealing Alice's input?

2. How does Bob check that different bFSS sharings all encode the same input?

1. How does Bob verify that a single bFSS is valid; without revealing Alice's input?

derandomizable bFSS

2. How does Bob check that different bFSS sharings all encode the same input?

homomorphic commitments

Derandomizable bFSS

we formalize a new primitives derandomizable Function Secret Sharing with following properties

[BoyleGilboaIshai19]: FSS with derandomizability for “offset” families

Derandomizable bFSS

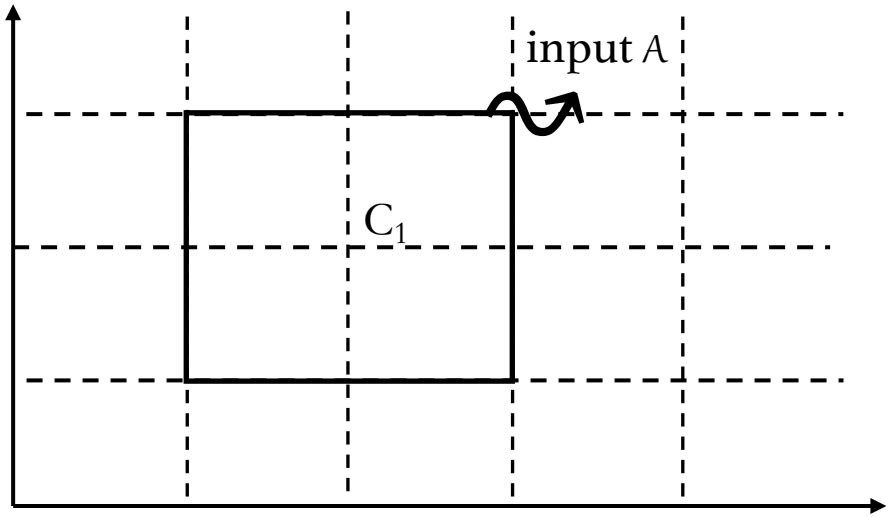
we formalize a new primitives derandomizable Function Secret Sharing with following properties

[BoyleGilboaIshai19]: FSS with derandomizability for "offset" families

derandomizable Function Secret Sharing (drFSS)

given input $A \in \mathcal{S}$ from a class of structured sets

structured input example: square



Derandomizable bFSS

we formalize a new primitives derandomizable Function Secret Sharing with following properties

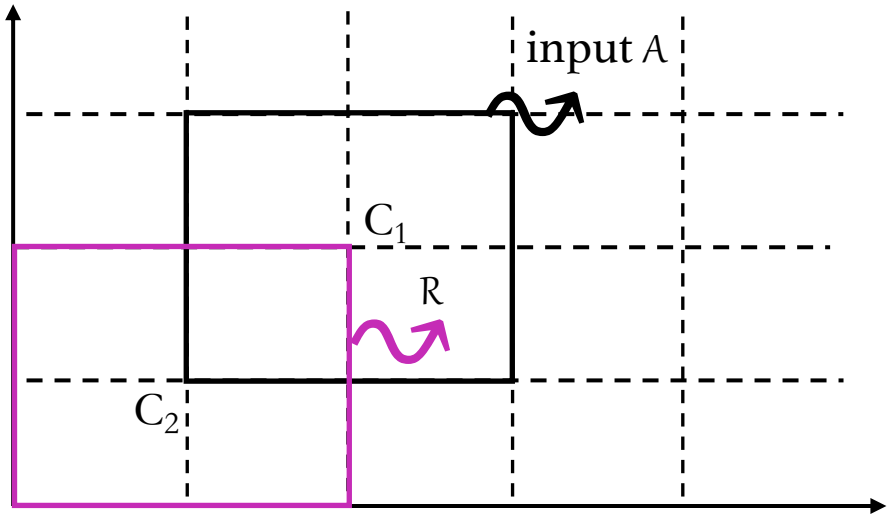
[BoyleGilboaIshai19]: FSS with derandomizability for "offset" families

derandomizable Function Secret Sharing (drFSS)

given input $A \in \mathcal{S}$ from a class of structured sets

Rshare() $\longrightarrow$  ,  , $\mathbb{R}$ where  ,  $\approx$ \$

structured input example: square



Derandomizable bFSS

we formalize a new primitives derandomizable Function Secret Sharing with following properties

[BoyleGilboaIshai19]: FSS with derandomizability for "offset" families

derandomizable Function Secret Sharing (drFSS)

given input $A \in \mathcal{S}$ from a class of structured sets

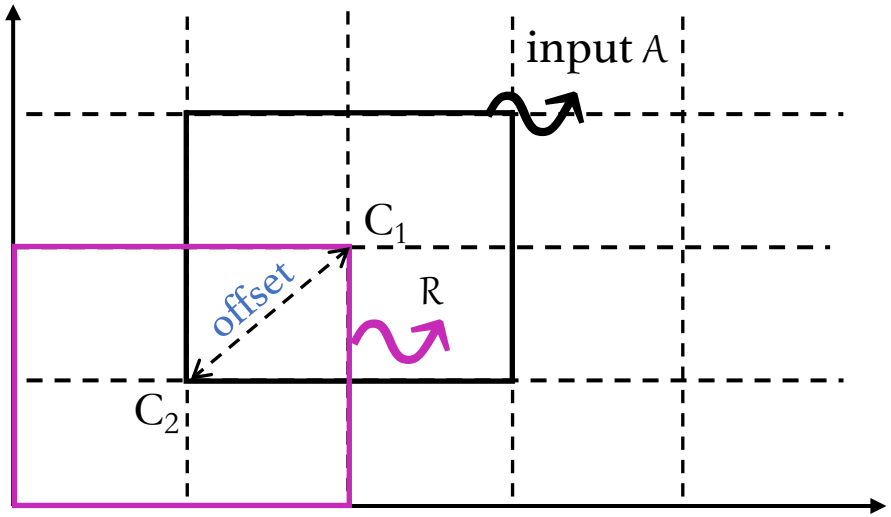
Rshare() $\longrightarrow$  , R where $\approx$ \$

offset = A - R

structured input example: square

encode(A) → center of square C_1

offset = C_1 - C_2



Derandomizable bFSS

we formalize a new primitives derandomizable Function Secret Sharing with following properties

[BoyleGilboaIshai19]: FSS with derandomizability for "offset" families

derandomizable Function Secret Sharing (drFSS)

given input $A \in \mathcal{S}$ from a class of structured sets

Rshare() $\longrightarrow$  ,  , $\mathbb{R}$ where  ,  $\approx$ \$

offset = A - $\mathbb{R}$

$x \in A \Longrightarrow$ **Deval**( , offset , x) $\oplus$ **Deval**( , offset , x) = 0

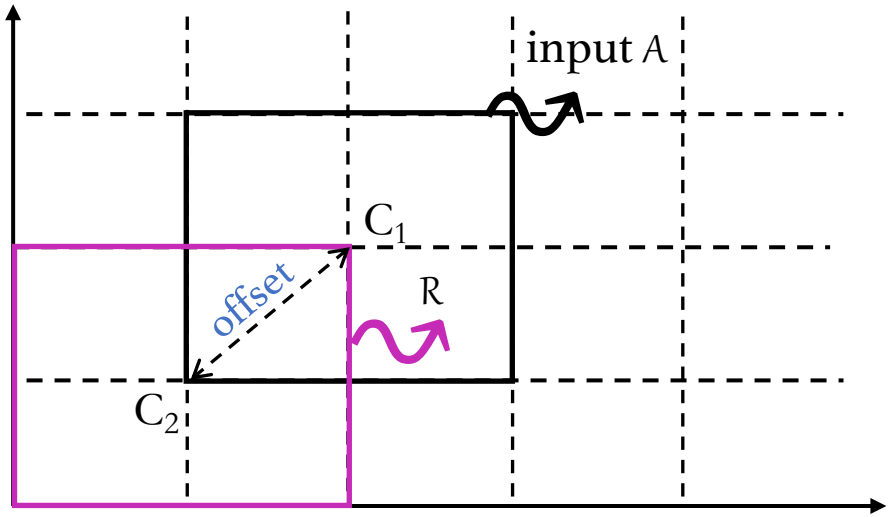
$x \notin A \Longrightarrow$ **Deval**( , offset , x) $\oplus$ **Deval**( , offset , x) = 1

Privacy guarantee:  , offset $\approx$ \$ and  , offset $\approx$ \$

structured input example: square

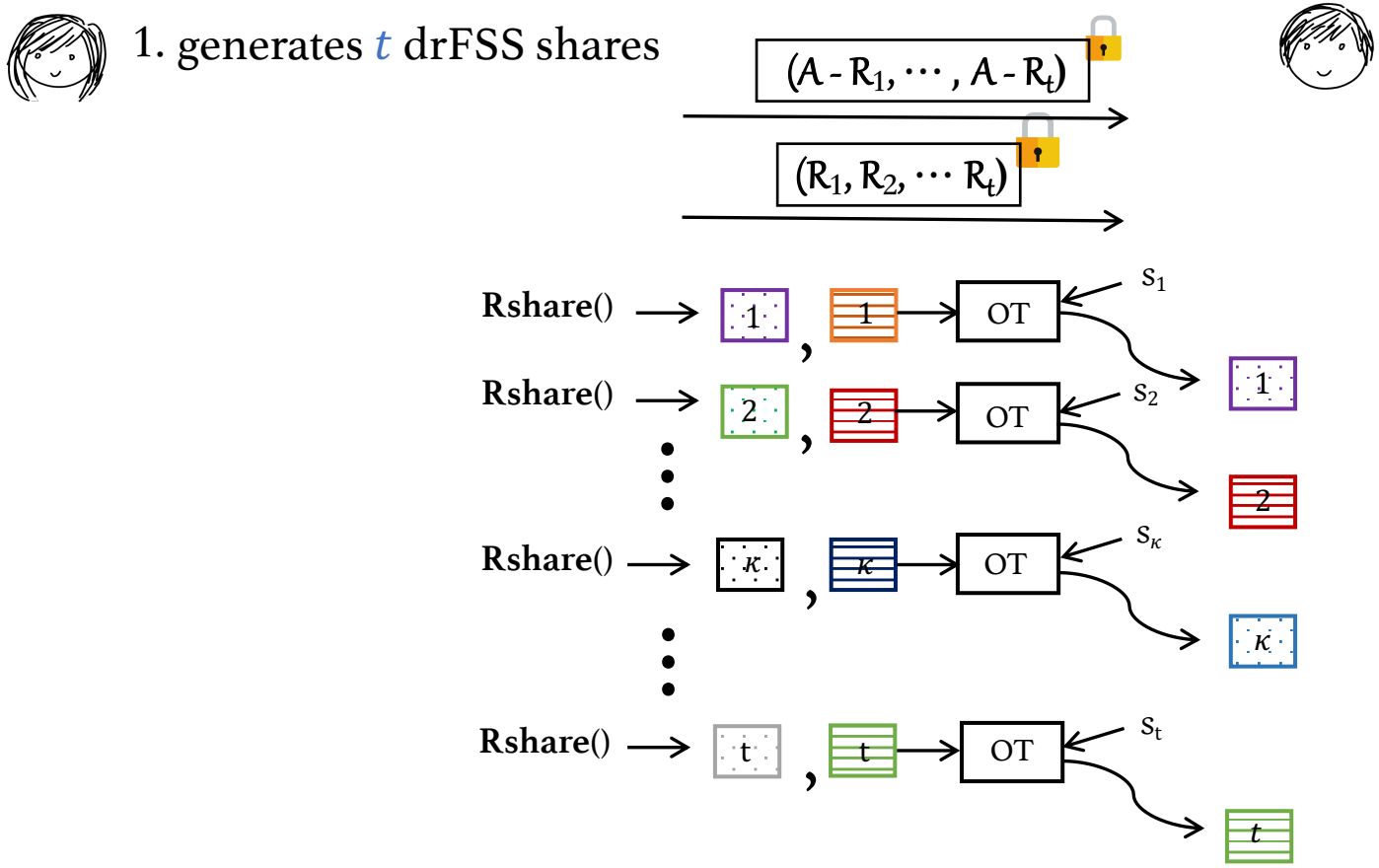
encode(A) $\rightarrow$ center of square C_1

offset = C_1 - C_2

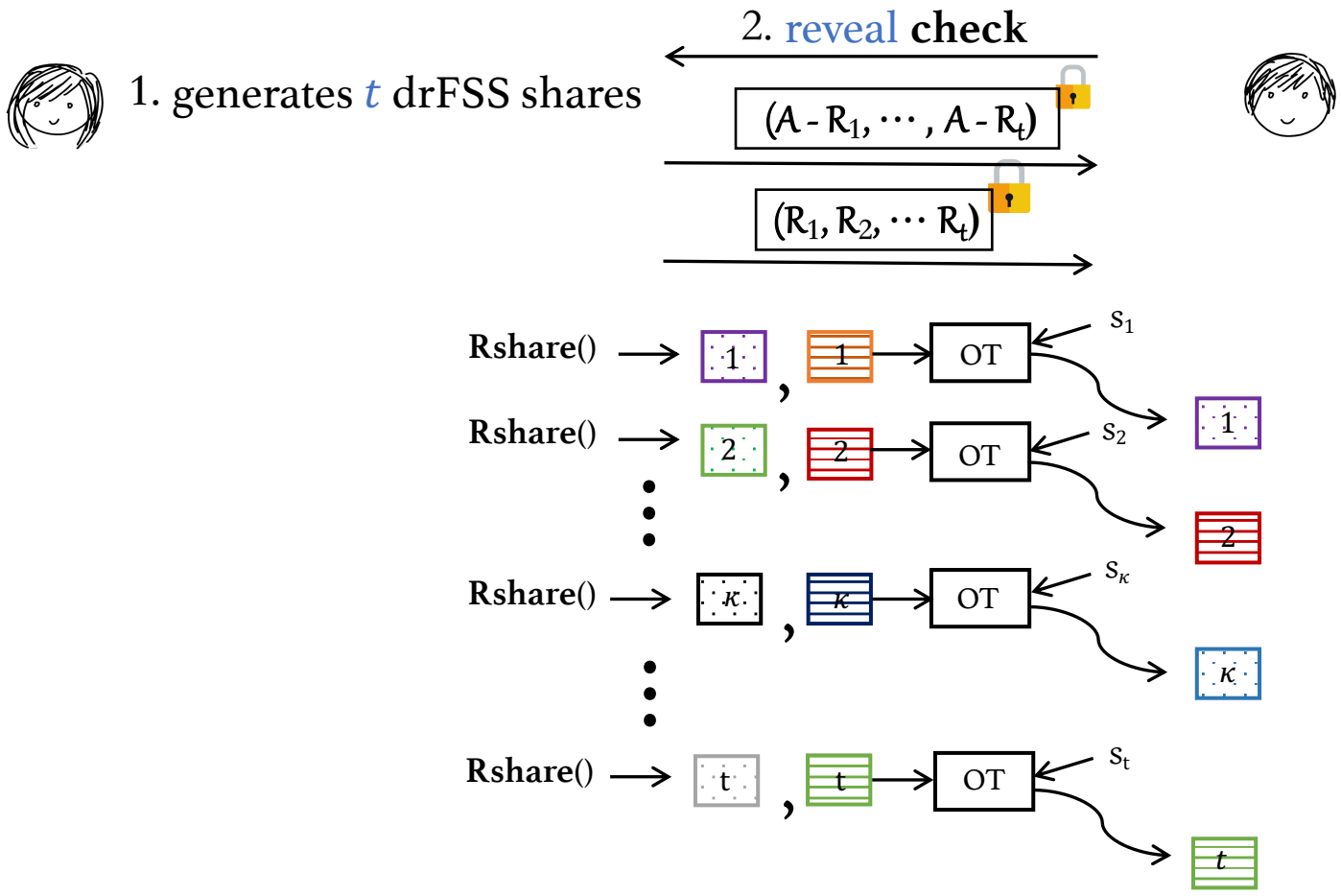


How can Bob check for valid encodings?

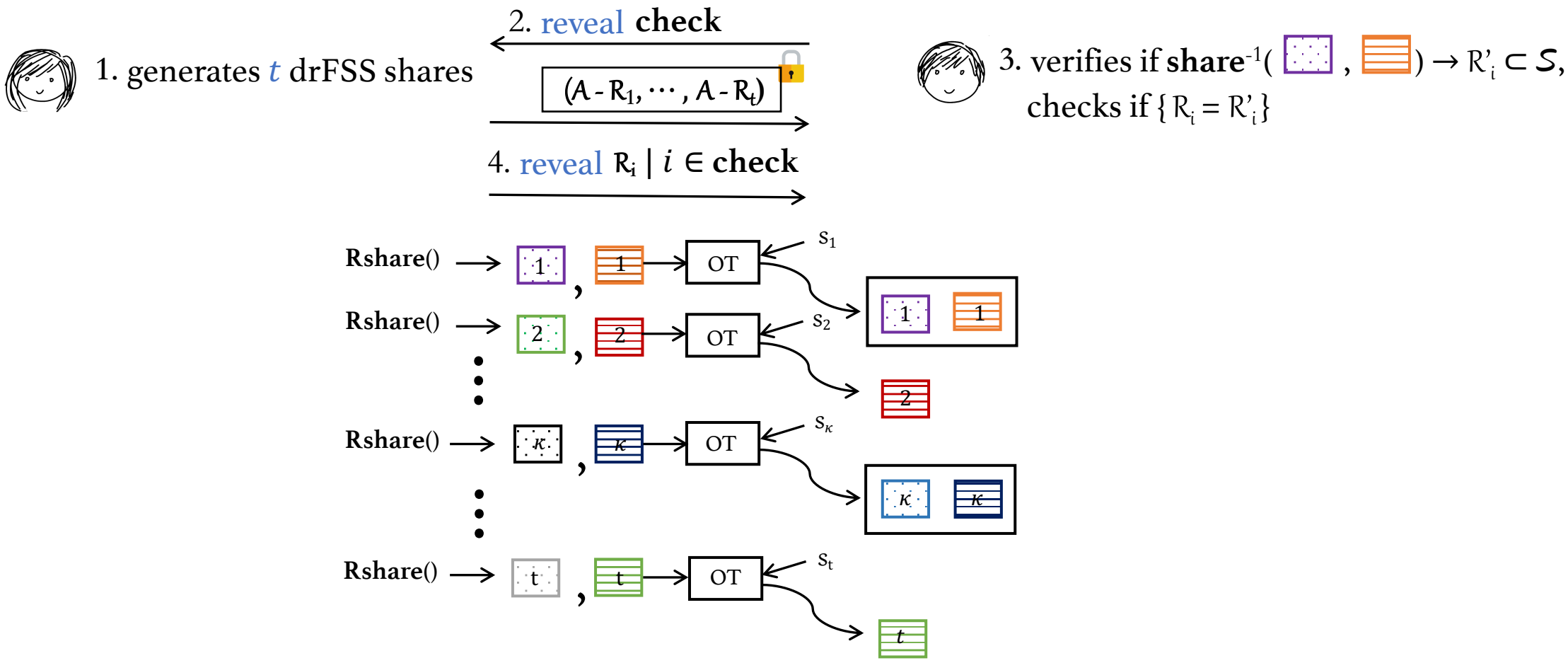
How can Bob check for valid encodings?



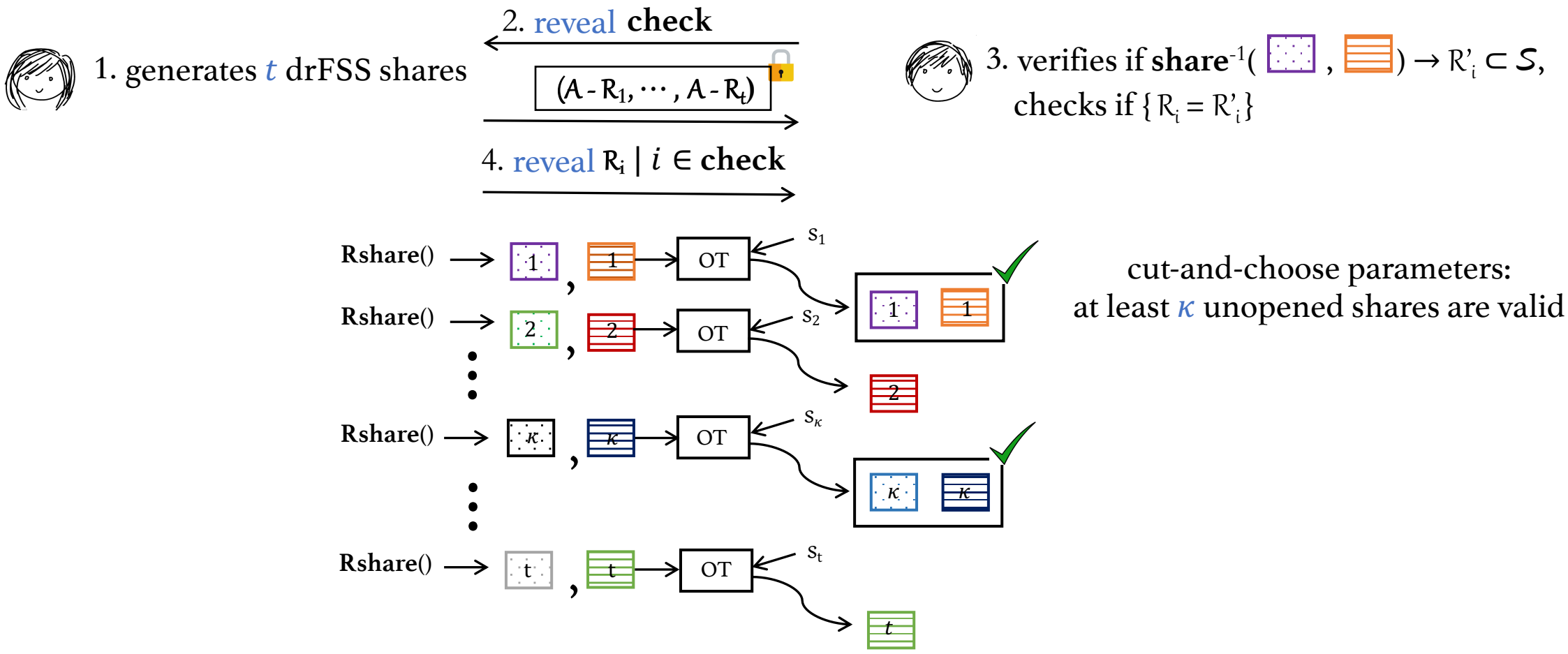
How can Bob check for valid encodings?



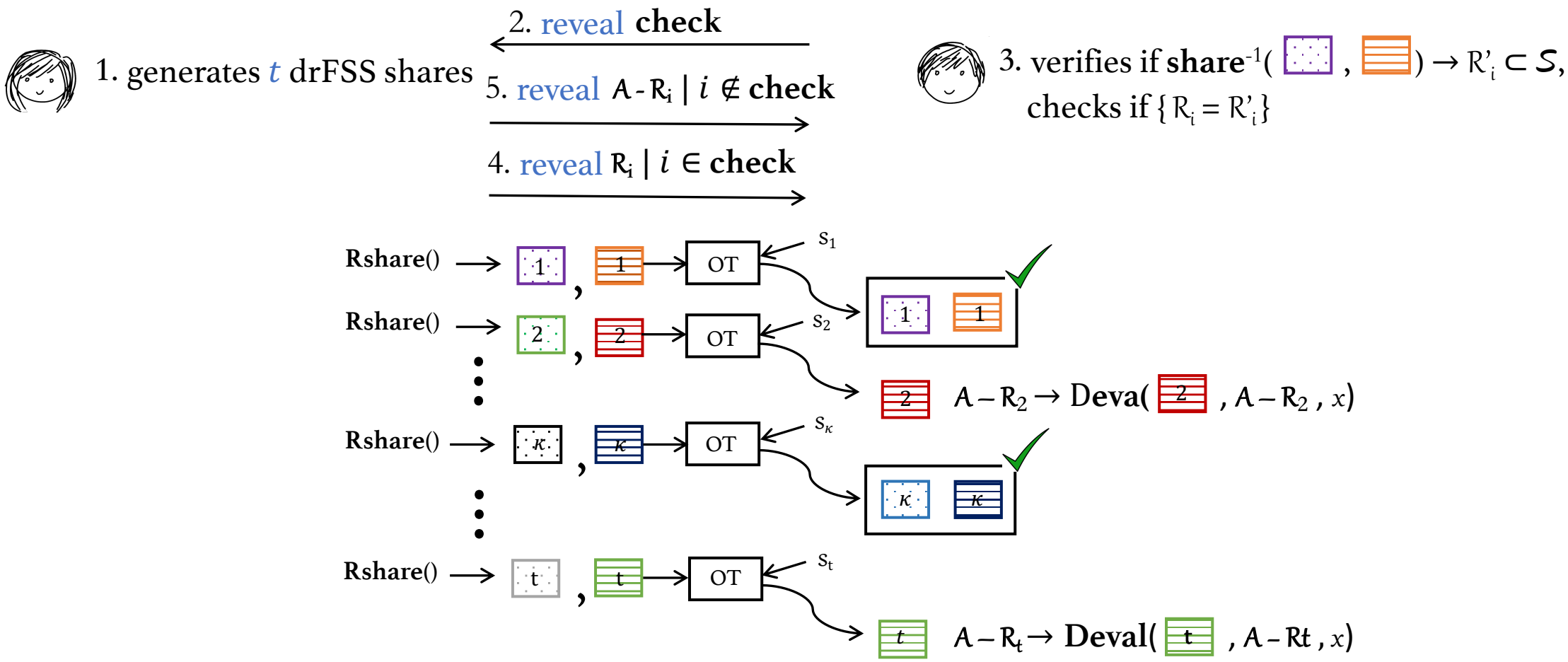
How can Bob check for valid encodings?



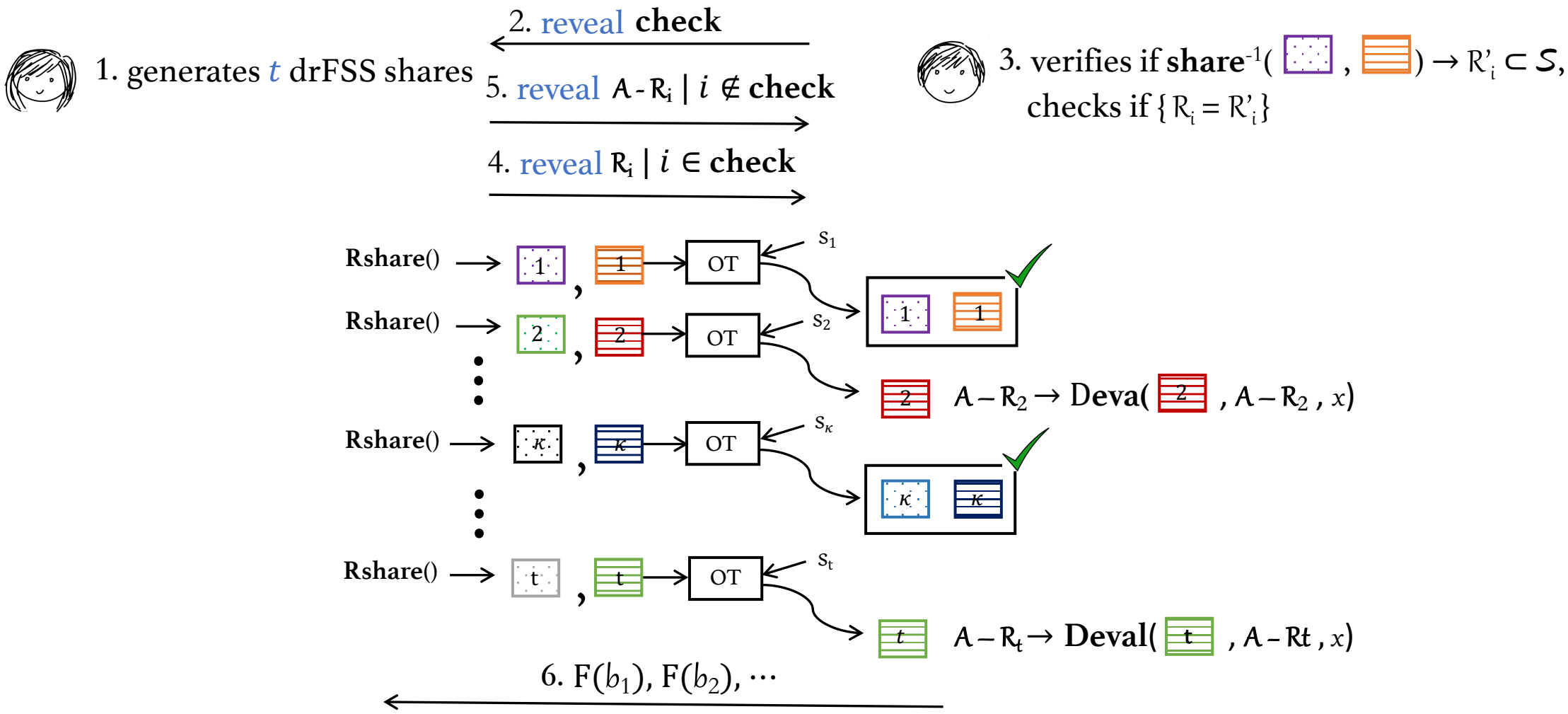
How can Bob check for valid encodings?



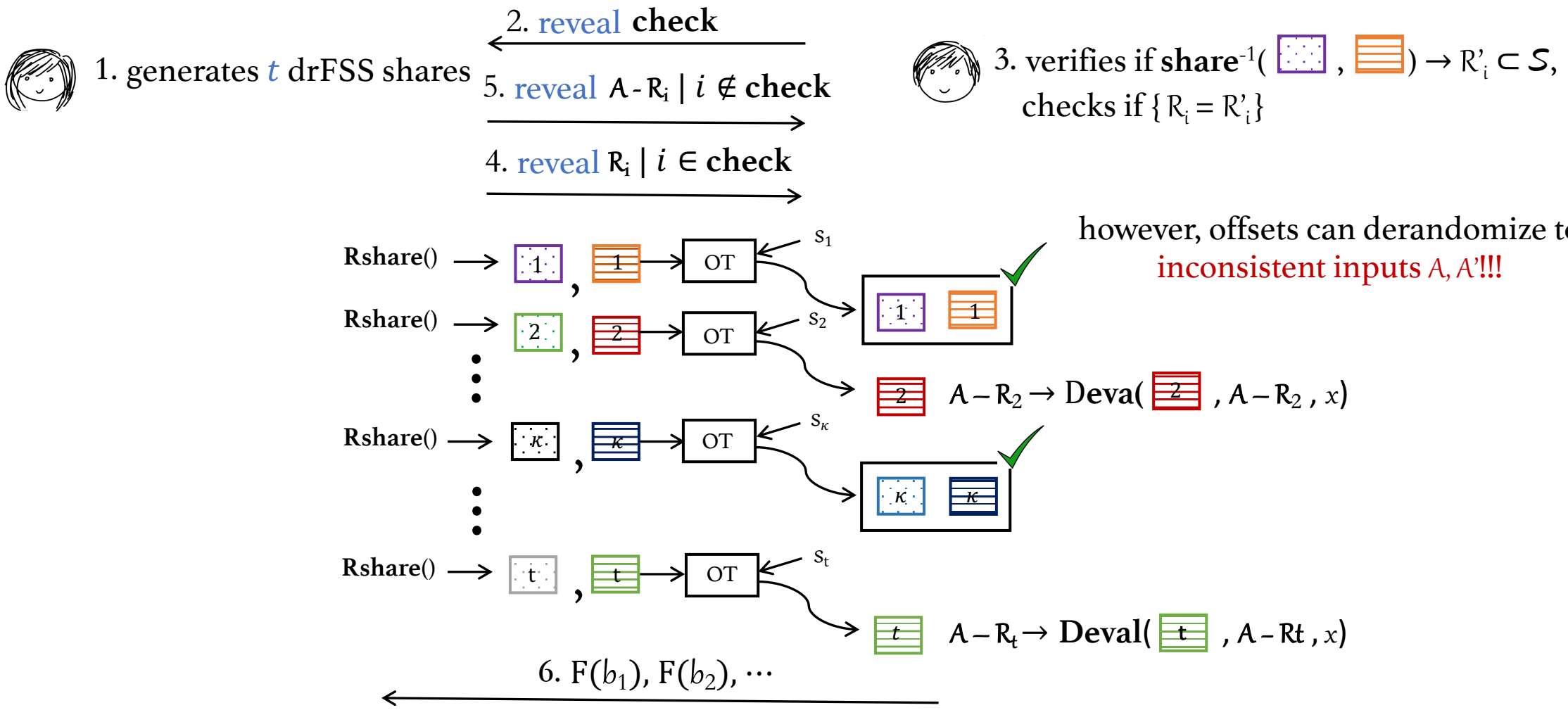
How can Bob check for valid encodings?



How can Bob check for valid encodings?

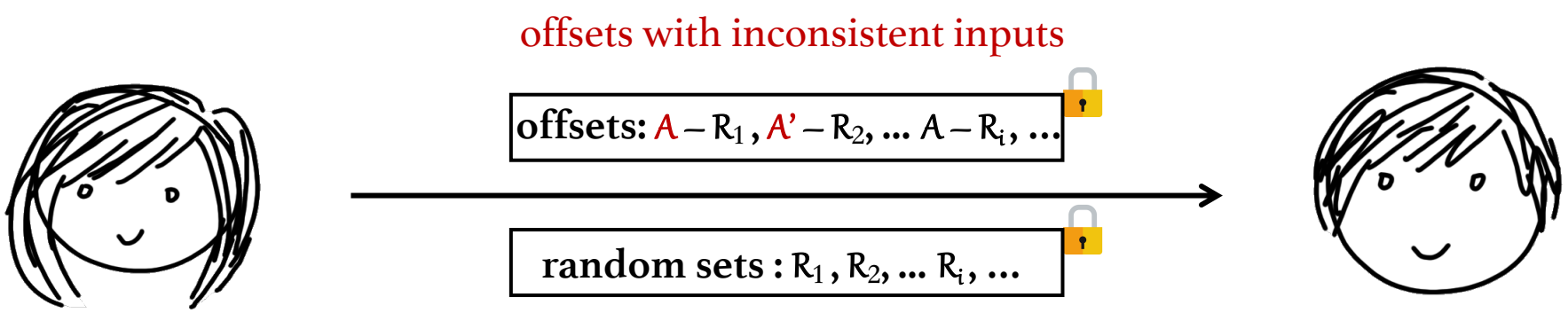


How can Bob check for valid encodings?



How can Bob check for consistent input set?

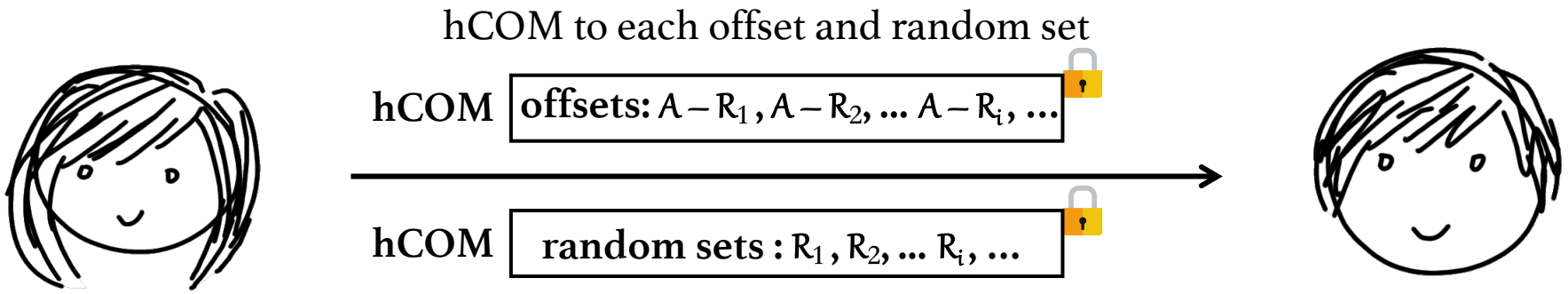
How can we ensure that offsets de-randomize evaluation back to the same input?



How can Bob check for consistent input set?

How can we ensure that offsets de-randomize evaluation back to the same input?

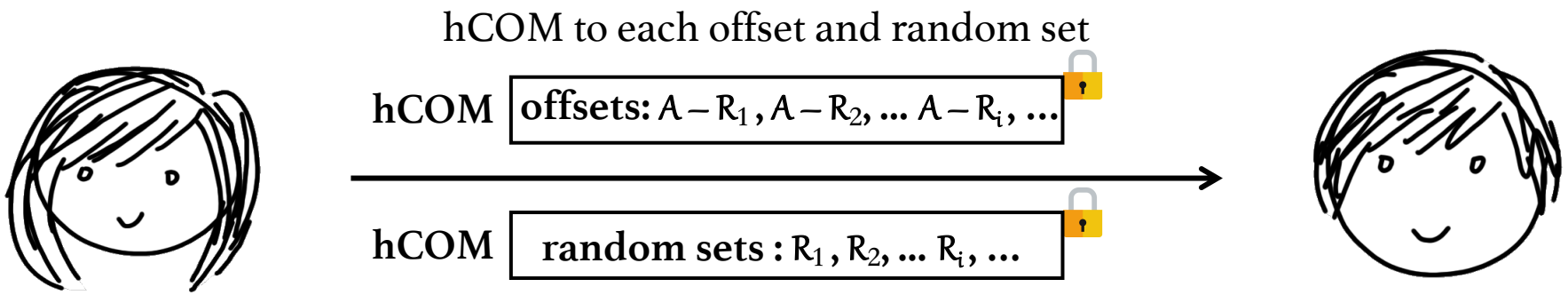
Homomorphic Commitments: allows Bob to check if offsets de-randomize to same valid input set



How can Bob check for consistent input set?

How can we ensure that offsets de-randomize evaluation back to the same input?

Homomorphic Commitments: allows Bob to check if offsets de-randomize to same valid input set

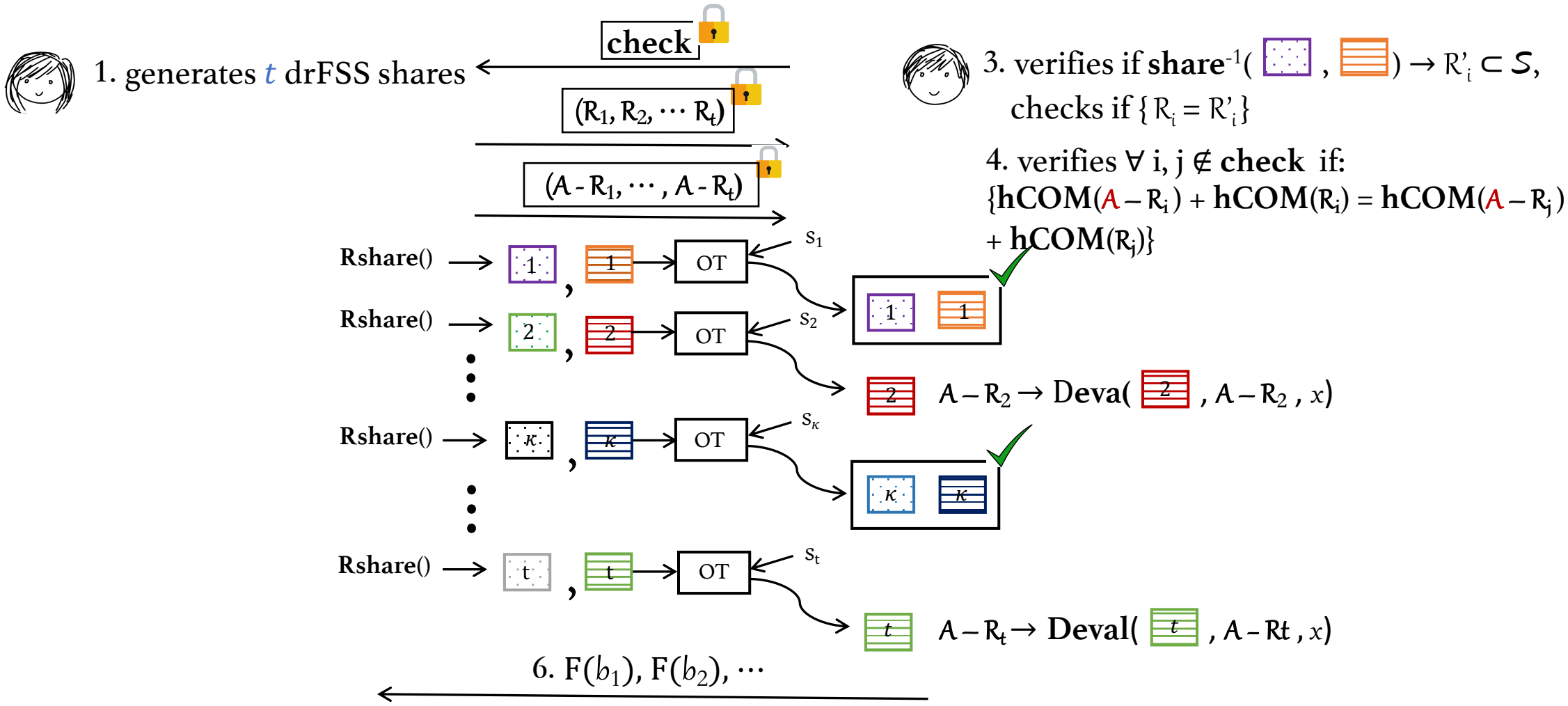


Before evaluating Bob verifies $\forall$ i, j $\notin$ **check** if:

$$\{\mathbf{hCOM}(A - R_i) + \mathbf{hCOM}(R_i) = \mathbf{hCOM}(A - R_j) + \mathbf{hCOM}(R_j)\}$$

Protocol Snapshot

assumptions: drFSS, committed OT, hCOM, random oracle model



Summary of our contributions

✓ new framework for malicious-secure structure-aware PSI - **communication cost** $O((\sigma + |B|).\kappa)$

~6-7x communication overhead over the semi-honest protocol

Summary of our contributions

✓ new framework for malicious-secure structure-aware PSI - **communication cost** $O((\sigma + |B|).\kappa)$

~6-7x communication overhead over the semi-honest protocol

✓ formalize derandomizable FSS and present constructions for following sets

- singleton
- interval
- d-dimensional interval

Summary of our contributions

✓ new framework for malicious-secure structure-aware PSI - **communication cost** $O((\sigma + |B|).\kappa)$

~6-7x communication overhead over the semi-honest protocol

✓ formalize derandomizable FSS and present constructions for following sets

- singleton
- interval
- d-dimensional interval
- union of L_∞ balls

✓ new spatial hashing technique for union of L_∞ balls, improves previous known semi-honest construction

Summary of our contributions

✓ new framework for malicious-secure structure-aware PSI - **communication cost** $O((\sigma + |B|).\kappa)$

~6-7x communication overhead over the semi-honest protocol

✓ formalize derandomizable FSS and present constructions for following sets

- singleton
- interval
- d-dimensional interval
- union of L_∞ balls

✓ new spatial hashing technique for union of L_∞ balls, improves previous known semi-honest construction

✓ more generally, show that derandomizable FSS can instantiate semi-honest structure-aware PSI

Future directions

- Can you make Alice's computation scale with description size? (both semi-honest and malicious)

Future directions

- Can you make Alice's computation scale with description size? (both semi-honest and malicious)

- Can we have structure-aware PSI where party with unstructured input learns result?

Future directions

- Can you make Alice's computation scale with description size? (both semi-honest and malicious)

- Can we have structure-aware PSI where party with unstructured input learns result?

- What if both Alice and Bob have structured inputs?

Future directions

- Can you make Alice's computation scale with description size? (both semi-honest and malicious)
- Can we have structure-aware PSI where party with unstructured input learns result?
- What if both Alice and Bob have structured inputs?
- Can we design drFSS for union of balls in other metric spaces like hamming distance, L-2 ?

Questions?

How does [GRS'22] work?

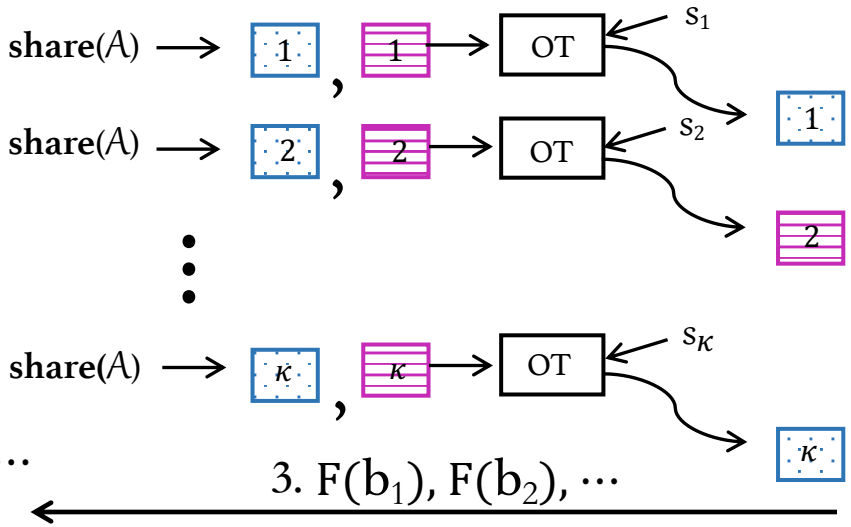
assumptions: OT-hybrid (Oblivious Transfer[Rabin81]), random oracle model

input A

1. generates κ instances of bFSS shares

input B

2. picks κ choice bits to learn  or 



$$(x \in A \implies \mathbf{ev}(\,\square_1\,, x) = \mathbf{ev}(\,\square_2\,, x))$$

$$F(x) = \mathbf{H}(\mathbf{ev}(\,\square_1\,, x) \| \mathbf{ev}(\,\square_2\,, x) \| \cdots \mathbf{ev}(\,\square_\kappa\,, x))$$
$$= \mathbf{H}(\mathbf{ev}(\,\square_1\,, x) \| \mathbf{ev}(\,\square_2\,, x) \| \cdots \mathbf{ev}(\,\square_\kappa\,, x))$$

Bob computes $F(x)$ on all his inputs

$$F(x) = \mathbf{H}(\mathbf{ev}(\,\square_1\,, x) \| \mathbf{ev}(\,\square_2\,, x) \| \cdots \mathbf{ev}(\,\square_\kappa\,, x))$$

construction	dFSS?	share size	eval cost
disjoint balls: basic FSS [BGI16]	yes	$O(n\kappa u^d)$	$O(\boldsymbol{n}u^d)$
disjoint balls: [GRS22]	no	$O(n(4\log\delta)^d\kappa)$	$O((2\log\delta)^d)$
disjoint balls: ours	yes	$O(n\kappa(ud+(\log\delta)^d))$	$O((2\log\delta)^d)$
balls with centers $> 4\delta$ apart: [GRS22]	no	$O(nd2^d\kappa\log\delta)$	$O(d\log\delta)$
balls with centers $> 8\delta$ apart: ours	yes	$O(nd\kappa\log\delta)$	$O(d\log\delta)$
axis-disjoint balls: [GRS22]	no	$O(nd\kappa\log\delta)$	$O(d\log\delta)$

Figure 1: FSS share size for n balls (ℓ_∞ norm) of radius δ in d dimensions, over u-bit integers. Evaluation time is for evaluating on one point.

structured input set	Communication cost (in GB)	
	[GRS22] (semi-honest secure)	Ours (malicious secure)
disjoint union of ℓ_∞ balls	101.1	662.4
union of balls with centers $> 8\delta$ apart	13.2	85.6

Figure 2: PSI concrete communication cost comparison with [GRS22]. Here Alice's structured set (of size $\approx 10^7$) contains $n = 2700$ ℓ_∞ balls in 2 dimensions, each with radius $\delta = 30$ in universe of size 2^{32} along each dimension, and Bob inputs an unstructured set of size 10^6.

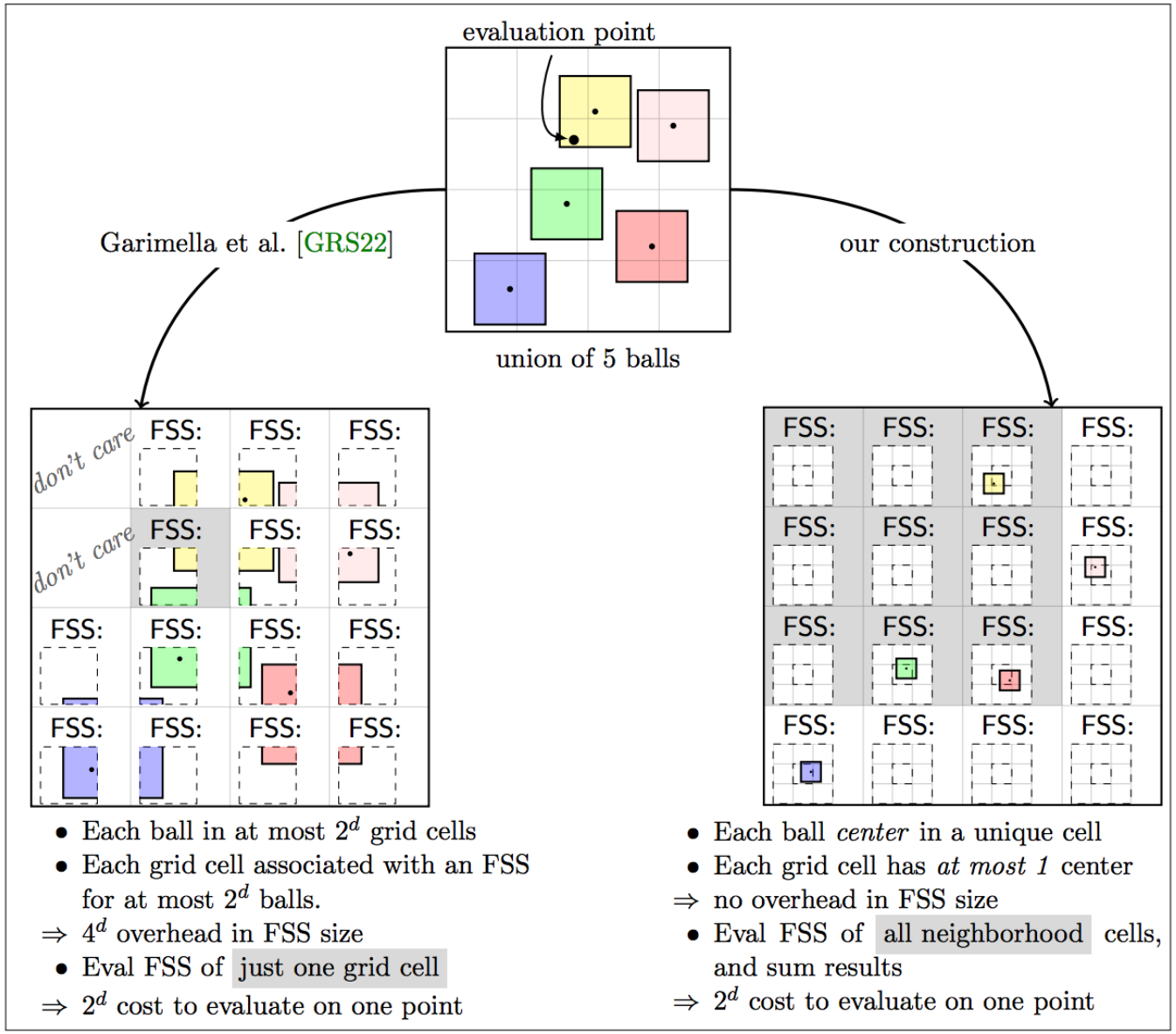


Figure 8: Illustration of the construction of Garimella et al. [GRS22] and our improved construction, for the union of n balls.