Non-Interactive Distributed Point Functions

PKC 2025

Elette Boyle Lalita Devadas **Sacha Servan-Schreiber**

Point Functions

$$P_t\left(x\right)$$

Point Functions

$$P_t(x) = \begin{cases} 1 & x = t \\ 0 & x \neq t \end{cases}$$

Point Functions

$$P_t(x) = \begin{cases} 1 & x = t \\ 0 & x \neq t \end{cases}$$

$P_5(0)$

$\boxed{0}$

Point Functions

$$P_t(x) = \begin{cases} 1 & x = t \\ 0 & x \neq t \end{cases}$$

$P_5(0)$	$P_5(1)$
$\boxed{0}$	$\boxed{0}$

Point Functions

$$P_t(x) = \begin{cases} 1 & x = t \\ 0 & x \neq t \end{cases}$$

$P_5(0)$	$P_5(1)$	$P_5(2)$	$P_5(3)$	$P_5(4)$	$P_5(5)$
$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{1}$

Point Functions

$$P_t(x) = \begin{cases} 1 & x = t \\ 0 & x \neq t \end{cases}$$

$P_5(0)$	$P_5(1)$	$P_5(2)$	$P_5(3)$	$P_5(4)$	$P_5(5)$	$P_5(6)$	$P_5(7)$	$P_5(8)$
$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{1}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$

Point Functions

$$P_t(x) = \begin{cases} 1 & x = t \\ 0 & x \neq t \end{cases}$$

$P_5(0)$	$P_5(1)$	$P_5(2)$	$P_5(3)$	$P_5(4)$	$P_5(5)$	$P_5(6)$	$P_5(7)$	$P_5(8)$
$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{1}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$

Distributed Point Functions [GI'14]

Alice

Bob

Distributed Point Functions [GI'14]

Alice

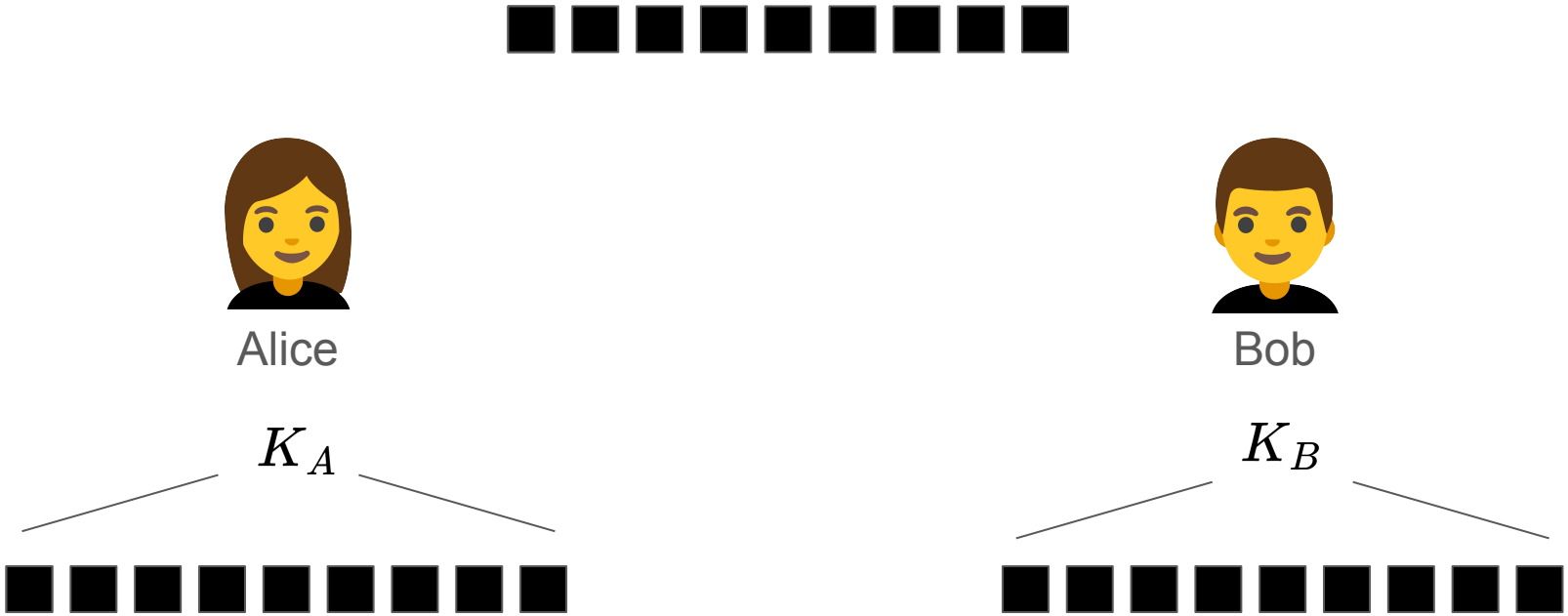
K_A

Bob

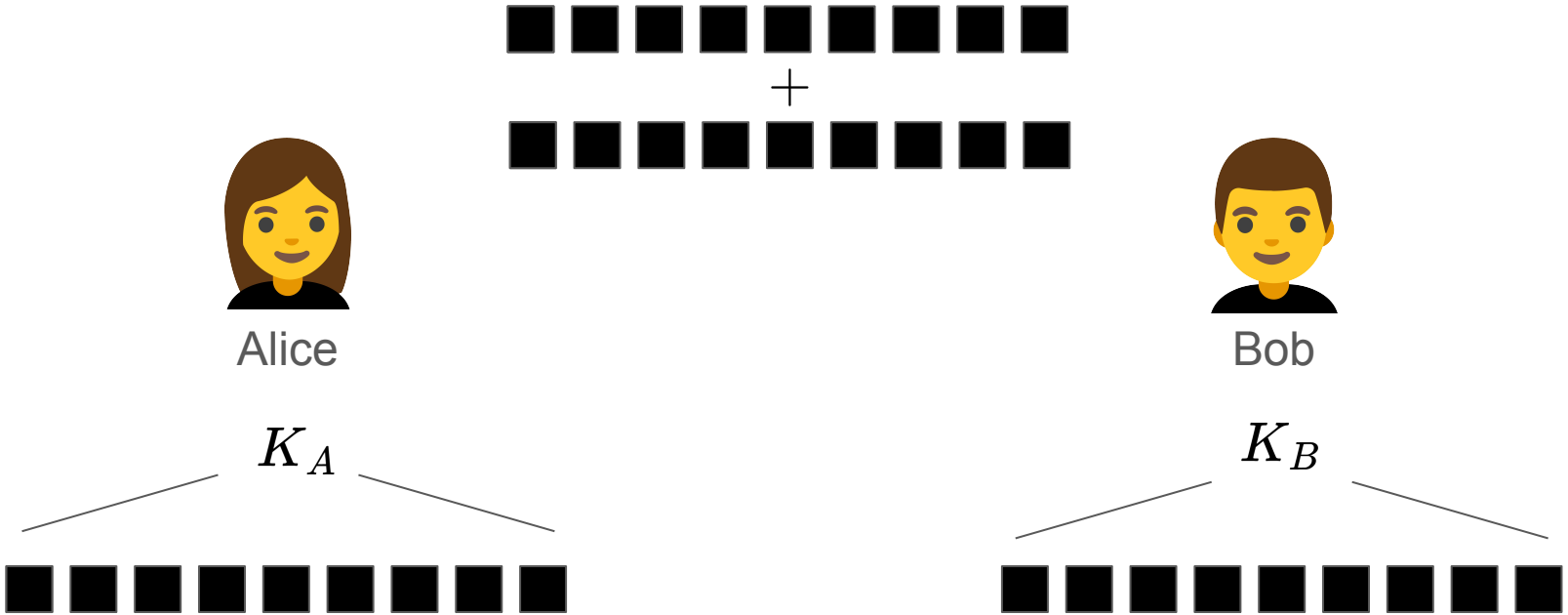
K_B

Distributed Point Functions [GI'14]

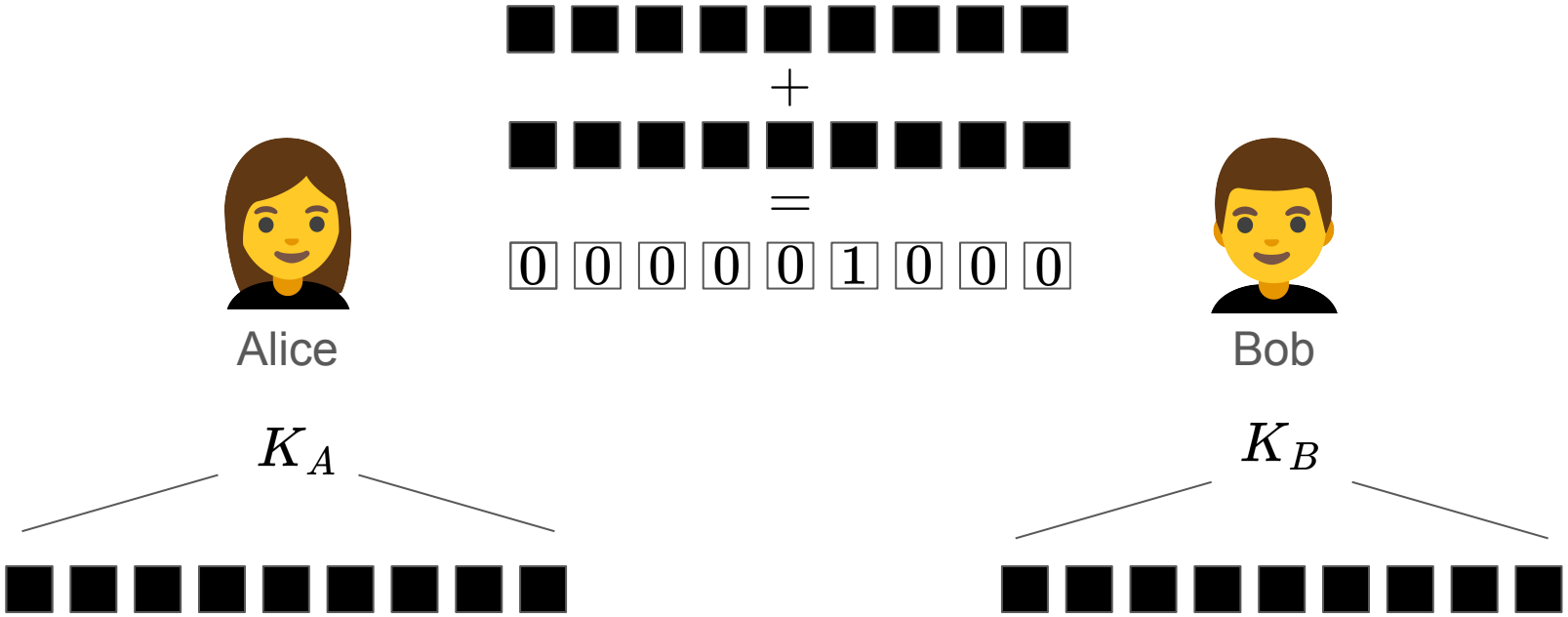
Distributed Point Functions [GI'14]



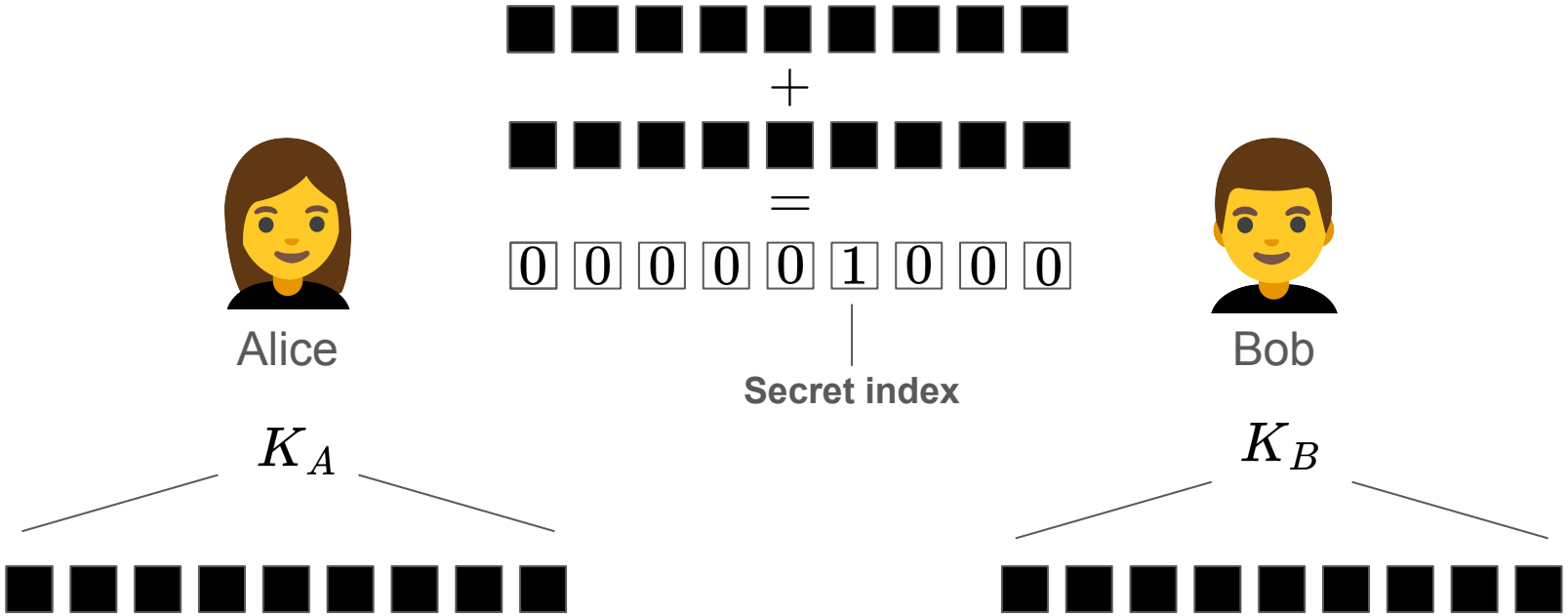
Distributed Point Functions [GI'14]



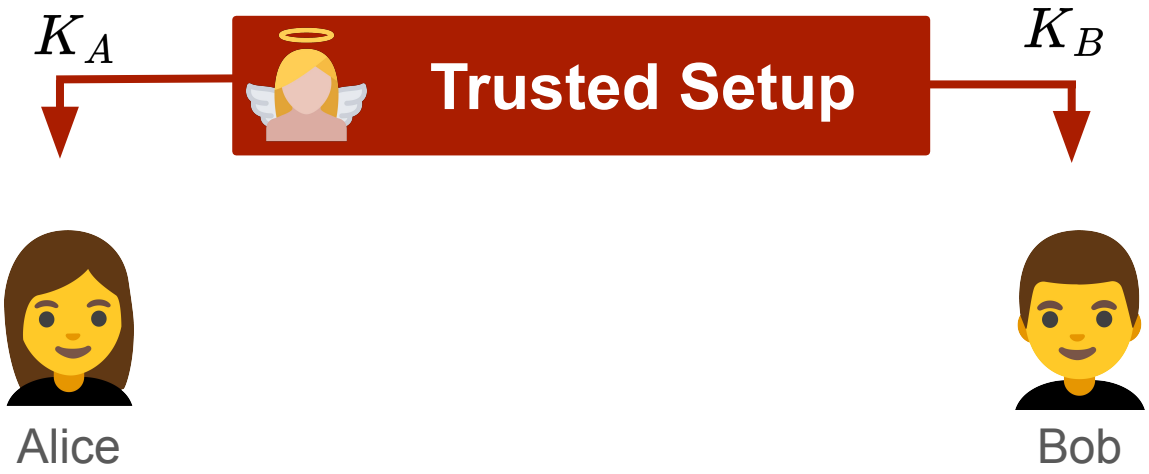
Distributed Point Functions [GI'14]



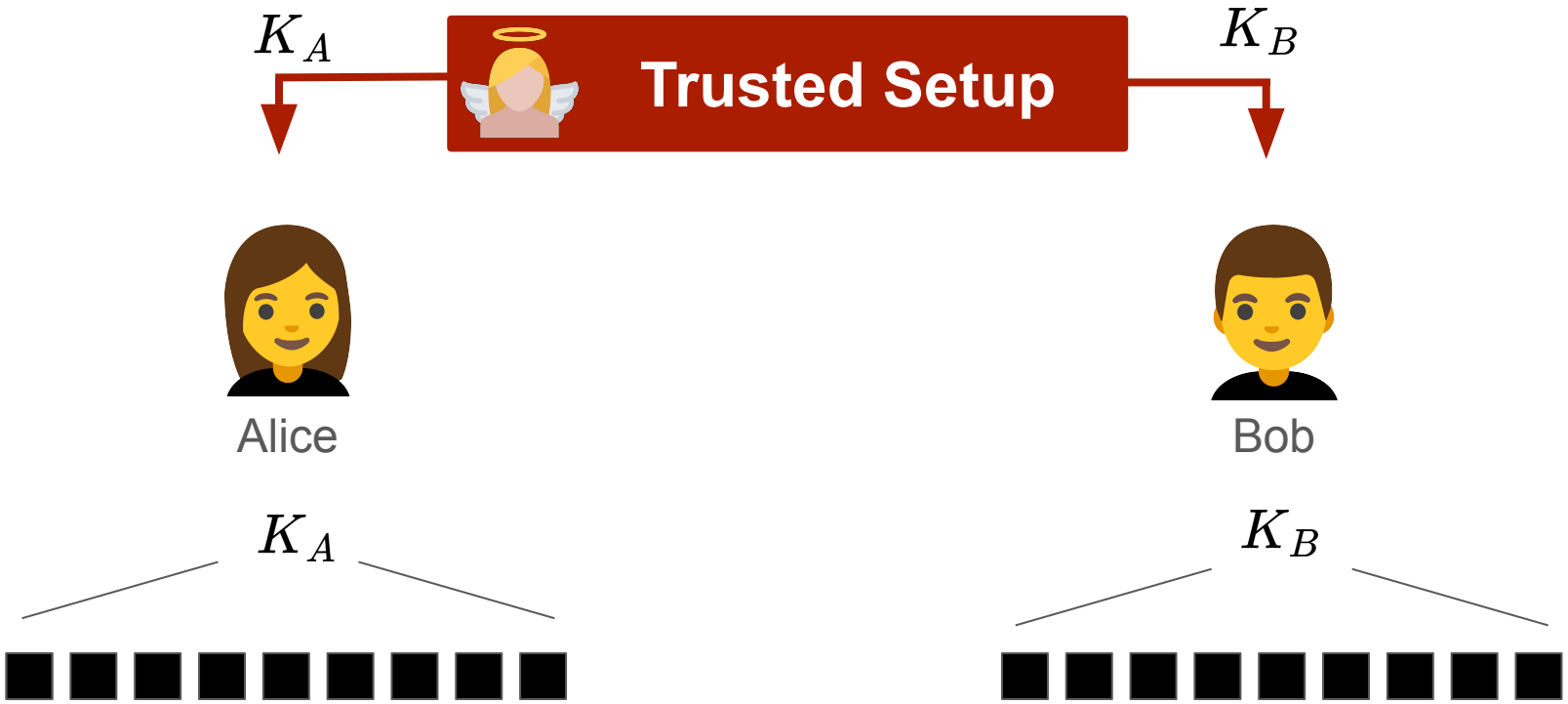
Distributed Point Functions [GI'14]



Distributed Point Functions [GI'14]



Distributed Point Functions [GI'14]



Applications of Distributed Point Functions

Private Information Retrieval and Search [GI'14, BGI'15, DPKY'20]

Applications of Distributed Point Functions

Private Information Retrieval and Search [GI'14, BGI'15, DPKY'20]

Distributed Oblivious RAM [Ds'17 + follow-up work]

Applications of Distributed Point Functions

Private Information Retrieval and Search [GI'14, BGI'15, DPKY'20]

Distributed Oblivious RAM [Ds'17 + follow-up work]

Preprocessing multi-party computation [BCGI'18 + follow-up work]

Applications of Distributed Point Functions

Private Information Retrieval and Search [GI'14, BGI'15, DPKY'20]

Distributed Oblivious RAM [Ds'17 + follow-up work]

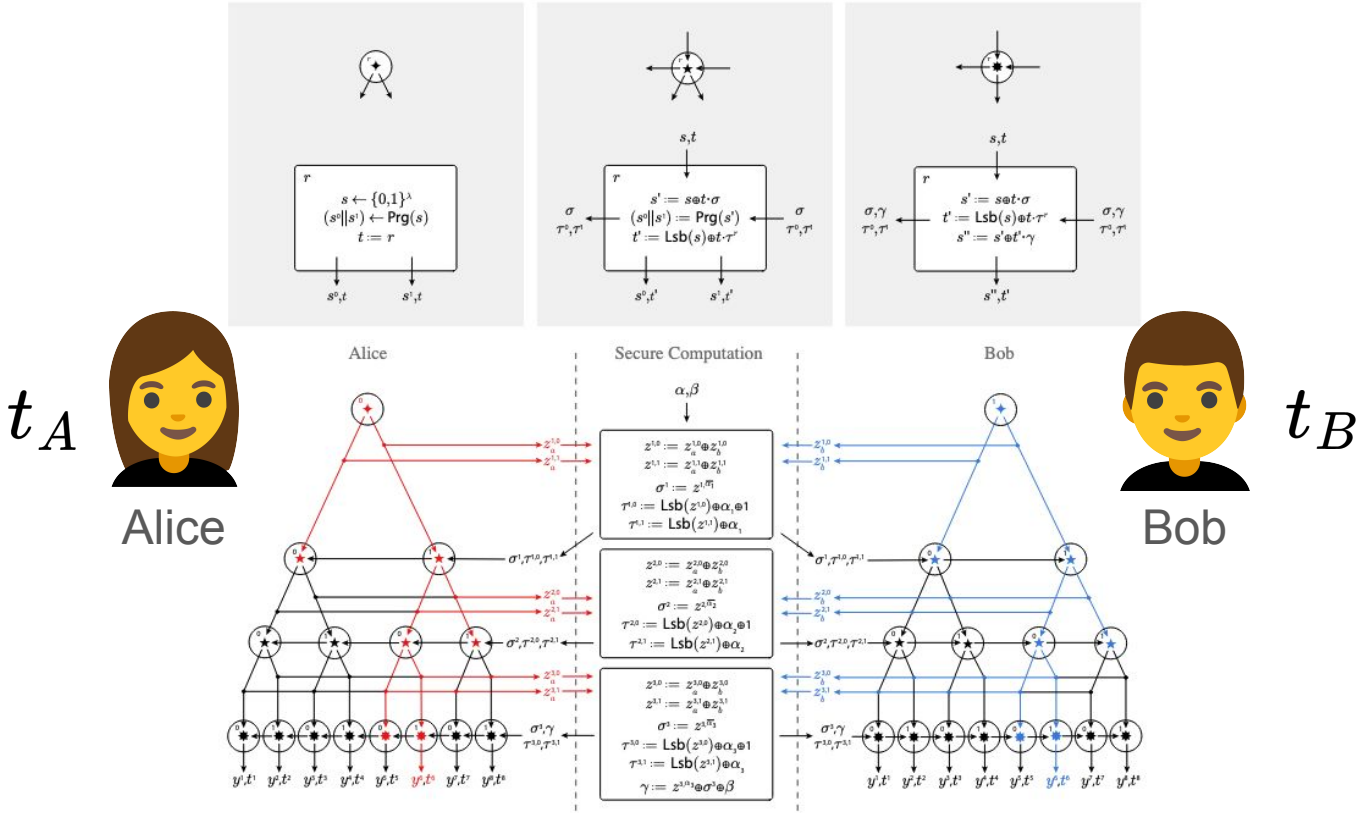
Preprocessing multi-party computation [BCGI'18 + follow-up work]

More efficient secure computation [BGIK'21 + follow-up work]

Can we remove the trusted setup?

Removing the Trusted Setup

The Doerner-shelat Protocol



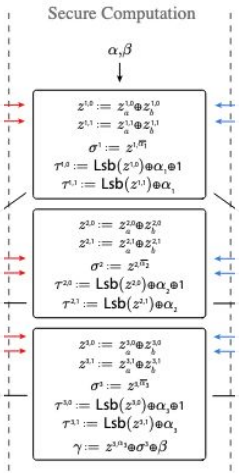
Removing the Trusted Setup

The Doerner-shelat Protocol

log(N) rounds of communication for domain of size N

t_A

Alice



Bob

t_B

Can we remove interaction?

Inspiration

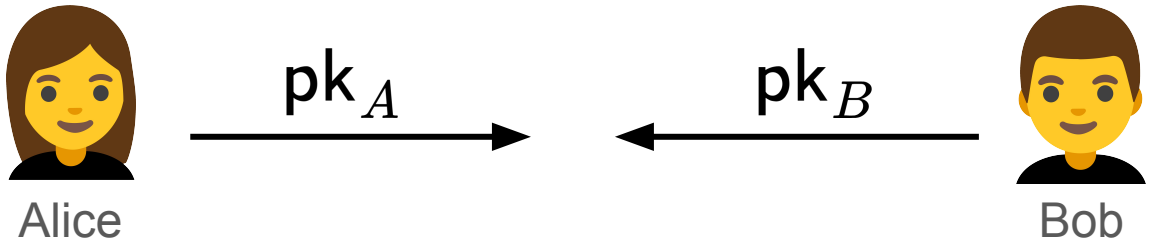
Diffie–Hellman Key Exchange

Diffie–Hellman Key Exchange [DH'76]

Alice

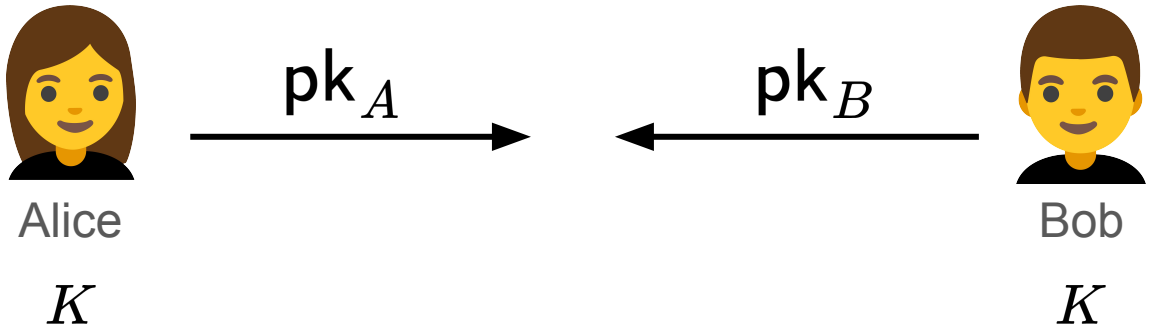
Bob

Diffie–Hellman Key Exchange [DH'76]

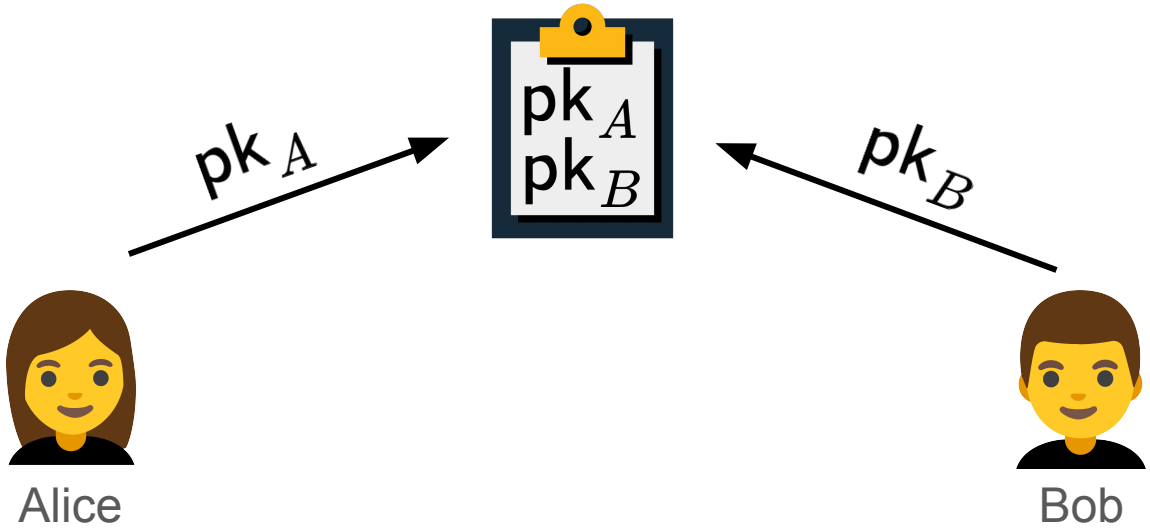


Diffie–Hellman Key Exchange [DH'76]

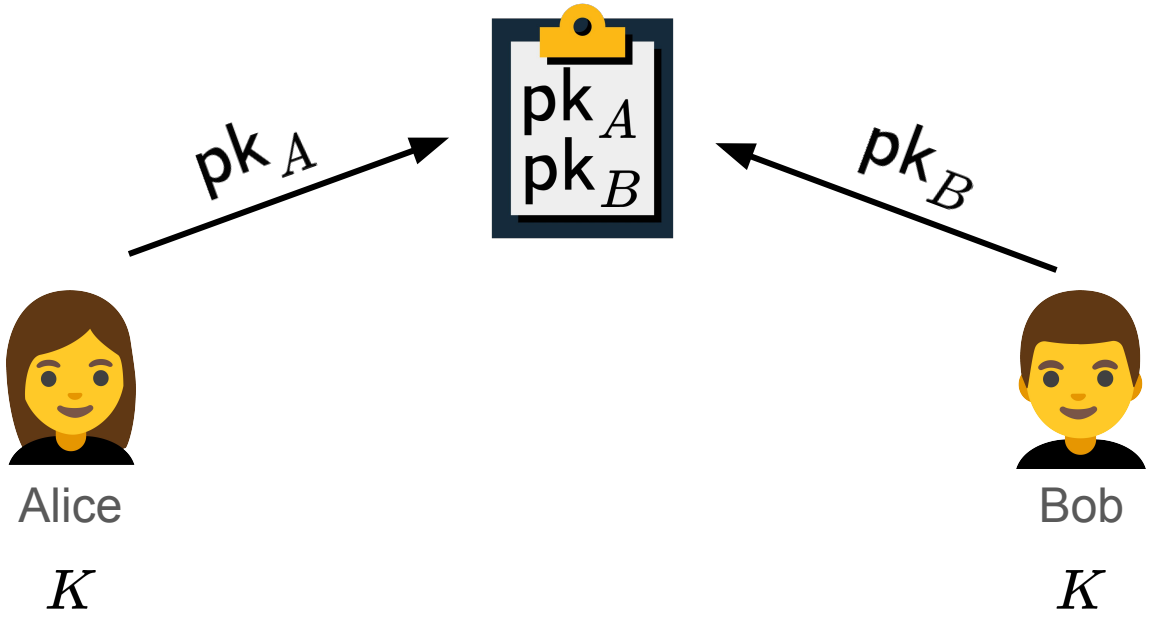
$$z_A + z_B = f(x, y)$$



Diffie–Hellman Key Exchange [DH'76]



Diffie–Hellman Key Exchange [DH'76]



“Diffie–Hellman” for DPF keys?

Non-Interactive
Distributed Point Functions

Non-Interactive Distributed Point Functions

No Trusted Setup

t_A 

Alice

 t_B

Bob

Non-Interactive Distributed Point Functions

No Trusted Setup*

t_A  Alice

 t_B Bob

*We still allow for a common reference string

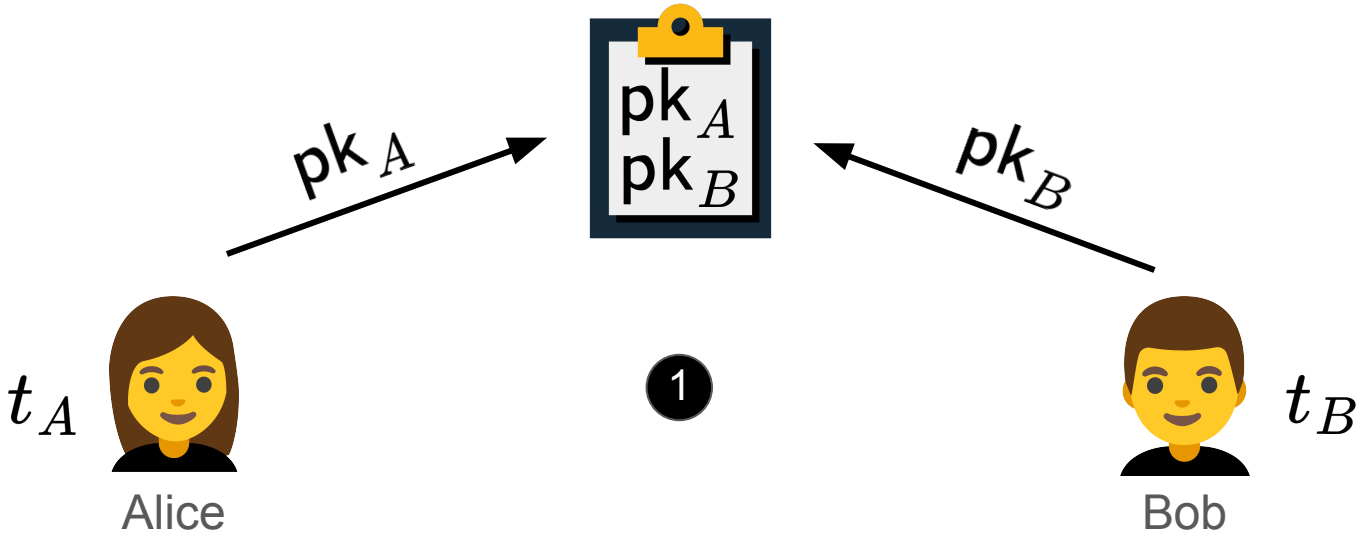
Non-Interactive Distributed Point Functions

No Trusted Setup

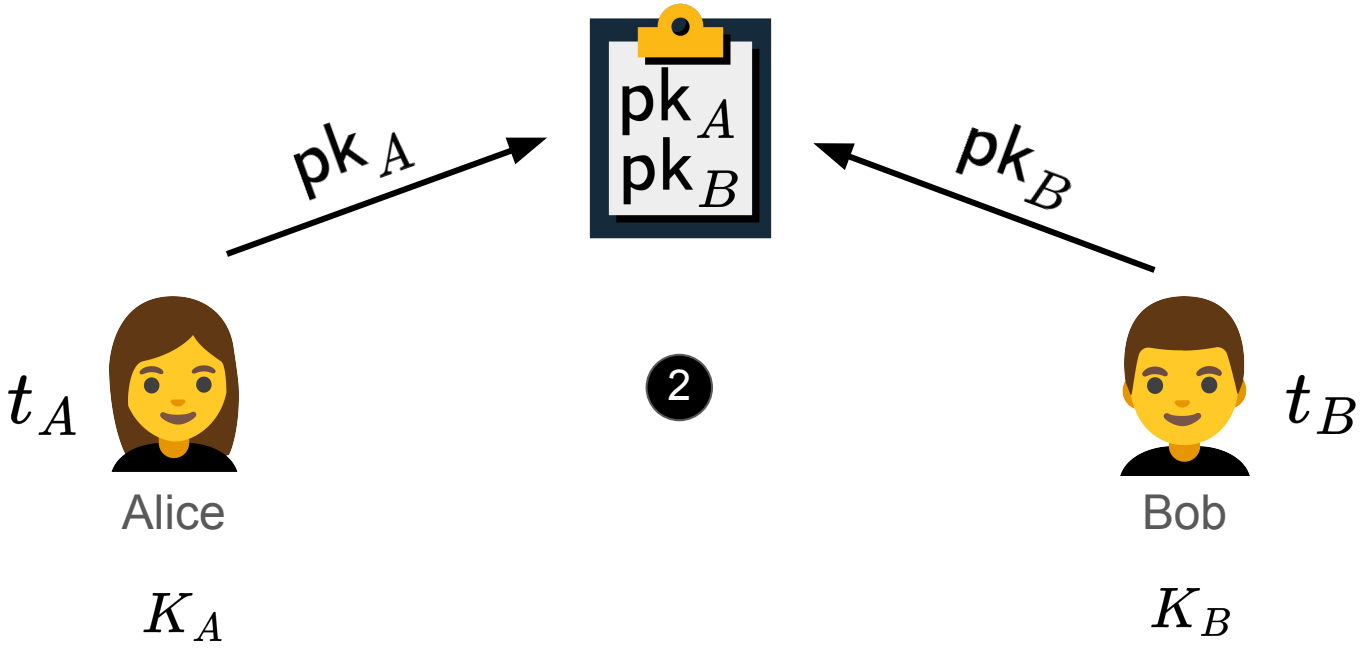
Goal

Get DPF keys for a point function with **secret index** $t = t_A + t_B$.

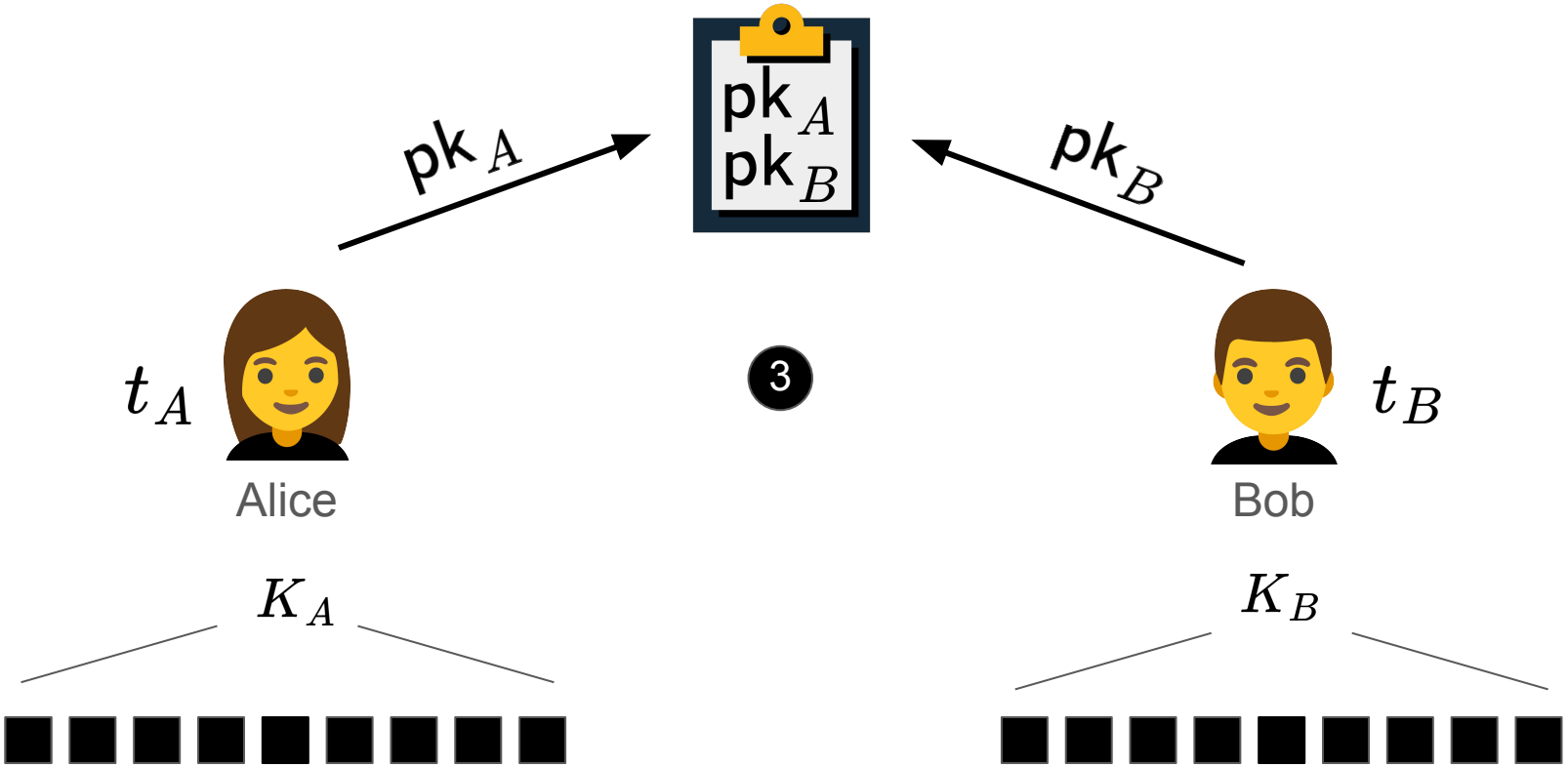
Non-Interactive Distributed Point Functions



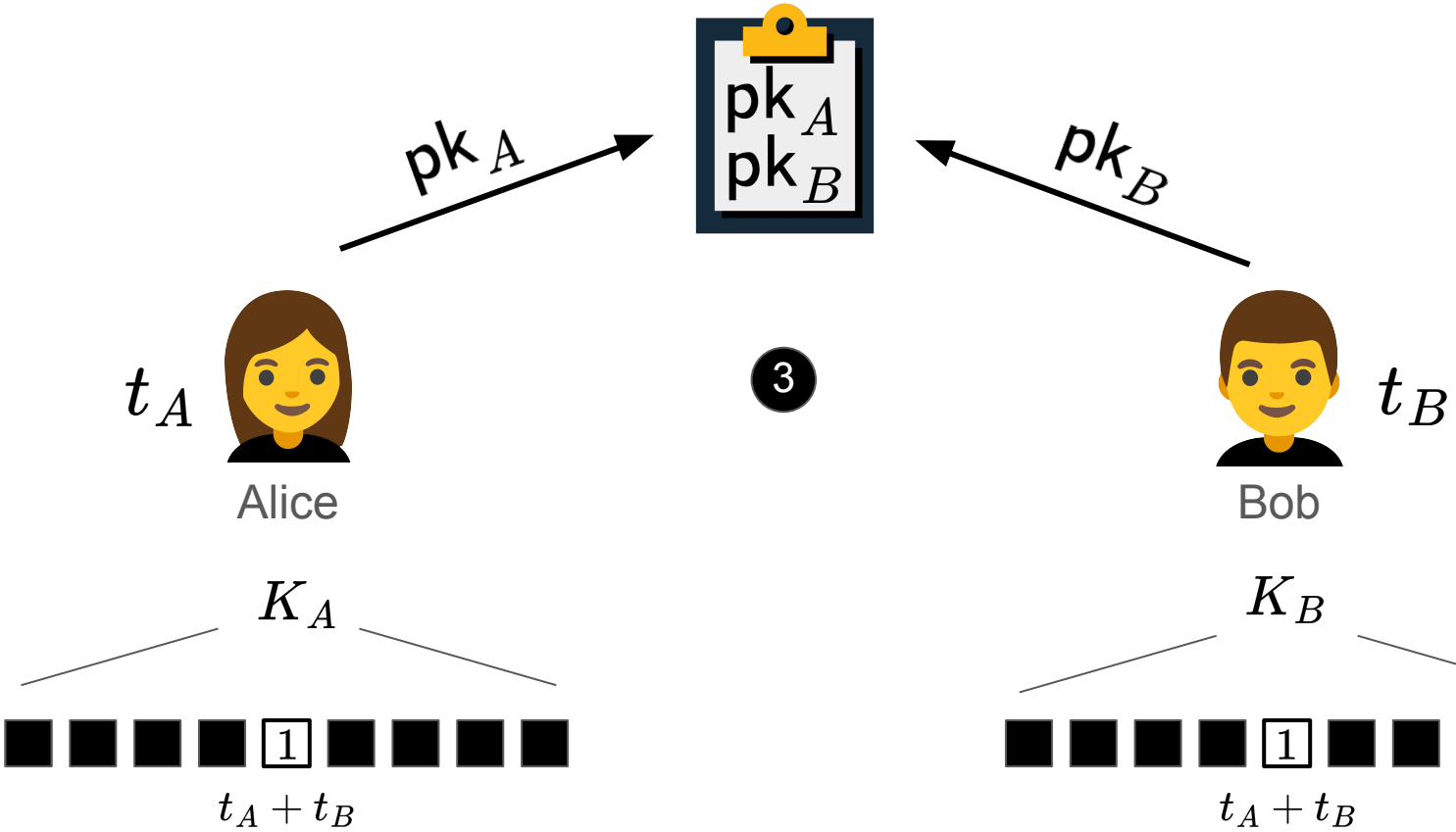
Non-Interactive Distributed Point Functions



Non-Interactive Distributed Point Functions



Non-Interactive Distributed Point Functions

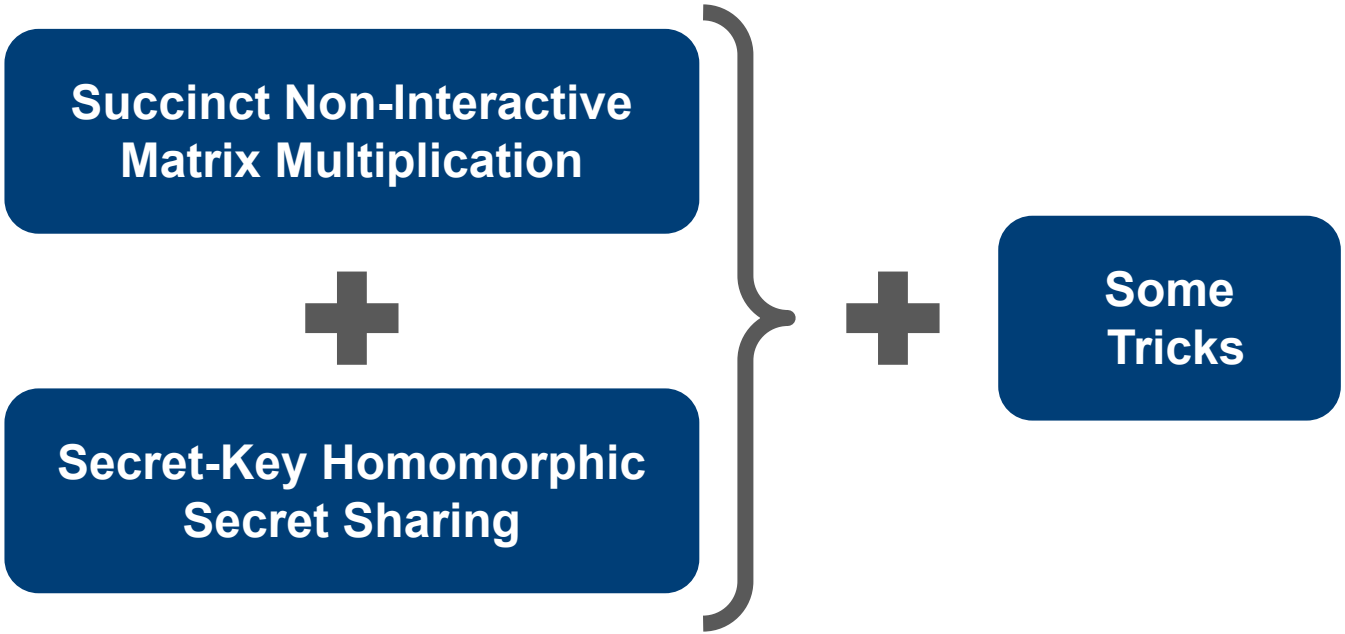


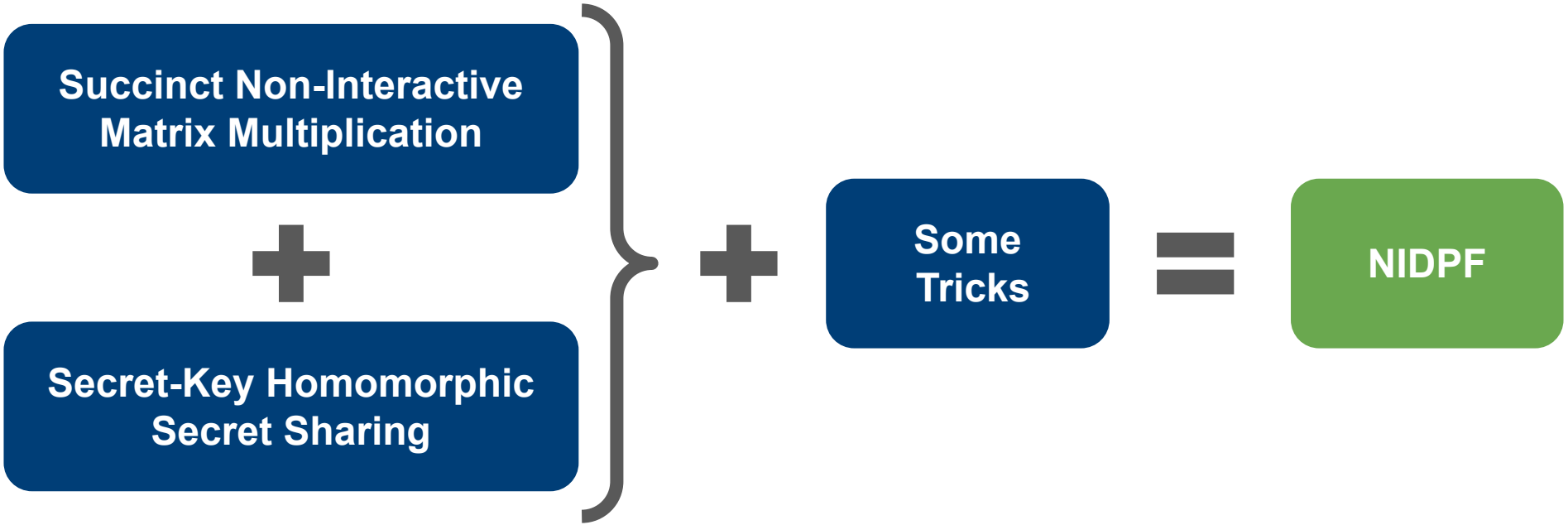
Building NIDPFs

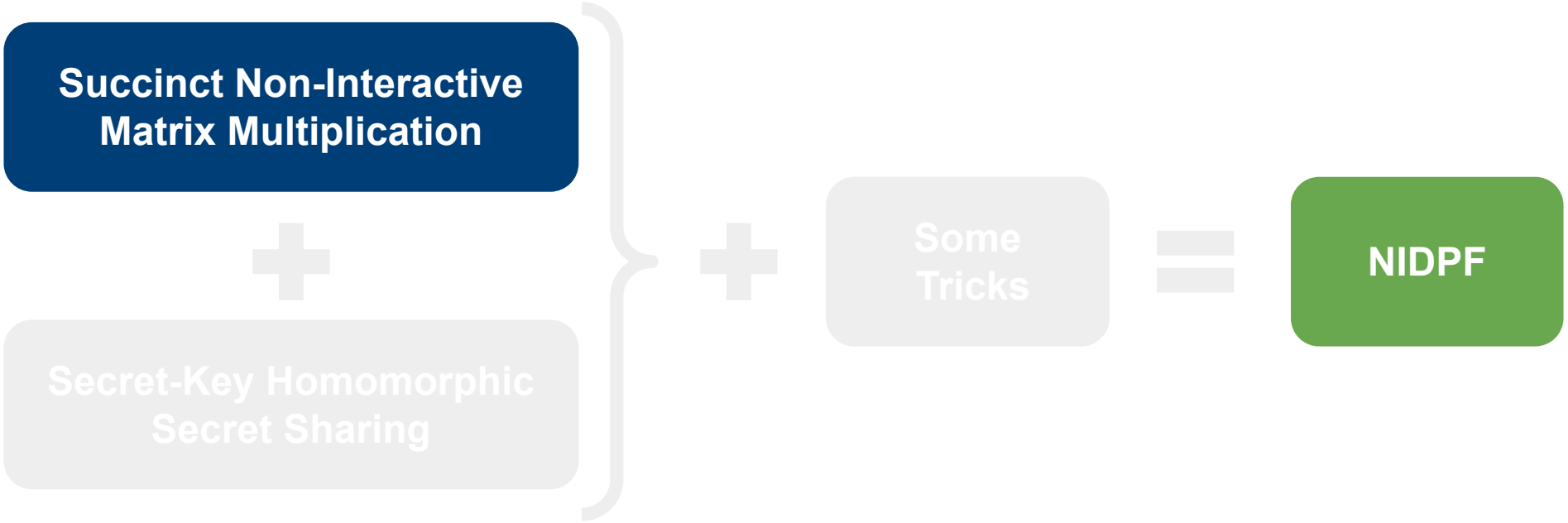
Succinct Non-Interactive Matrix Multiplication

Succinct Non-Interactive Matrix Multiplication

Secret-Key Homomorphic Secret Sharing


Succinct Non-Interactive Matrix Multiplication
+
Secret-Key Homomorphic Secret Sharing
+
Some Tricks


Succinct Non-Interactive Matrix Multiplication
+
Secret-Key Homomorphic Secret Sharing
+
Some Tricks
=
NIDPF


Succinct Non-Interactive Matrix Multiplication
+
Secret-Key Homomorphic Secret Sharing
+
Some Tricks
=
NIDPF

Tool: Succinct Non-Interactive Matrix Multiplication

Adapted from protocols described in [ARS'24, BCMPR'24]

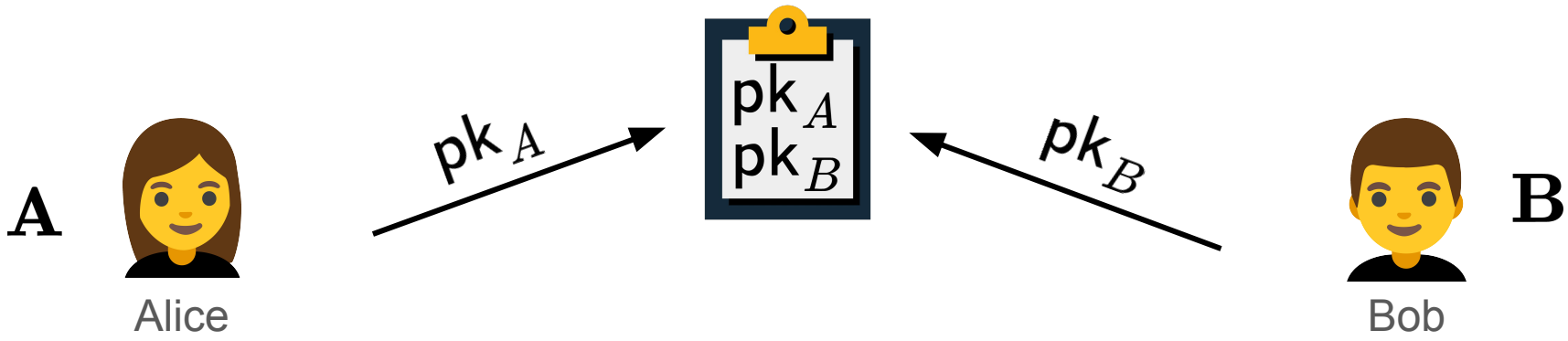
A

Alice

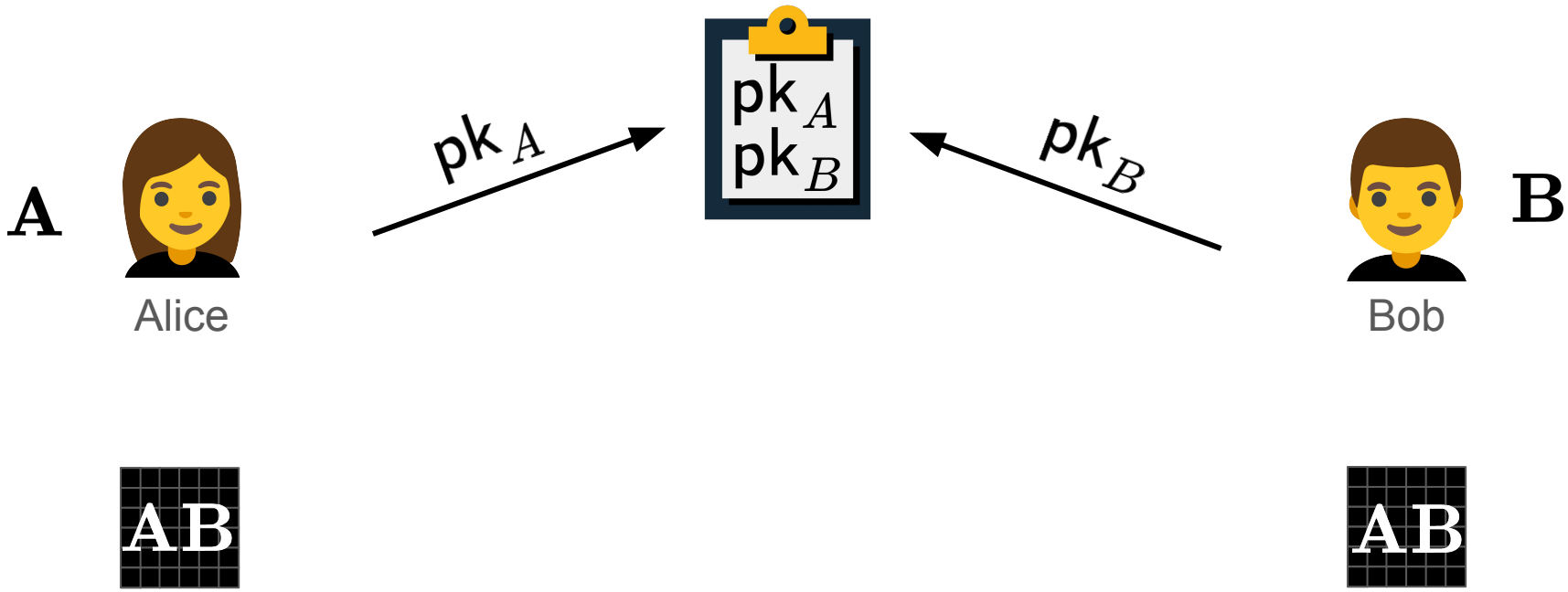
Bob

B

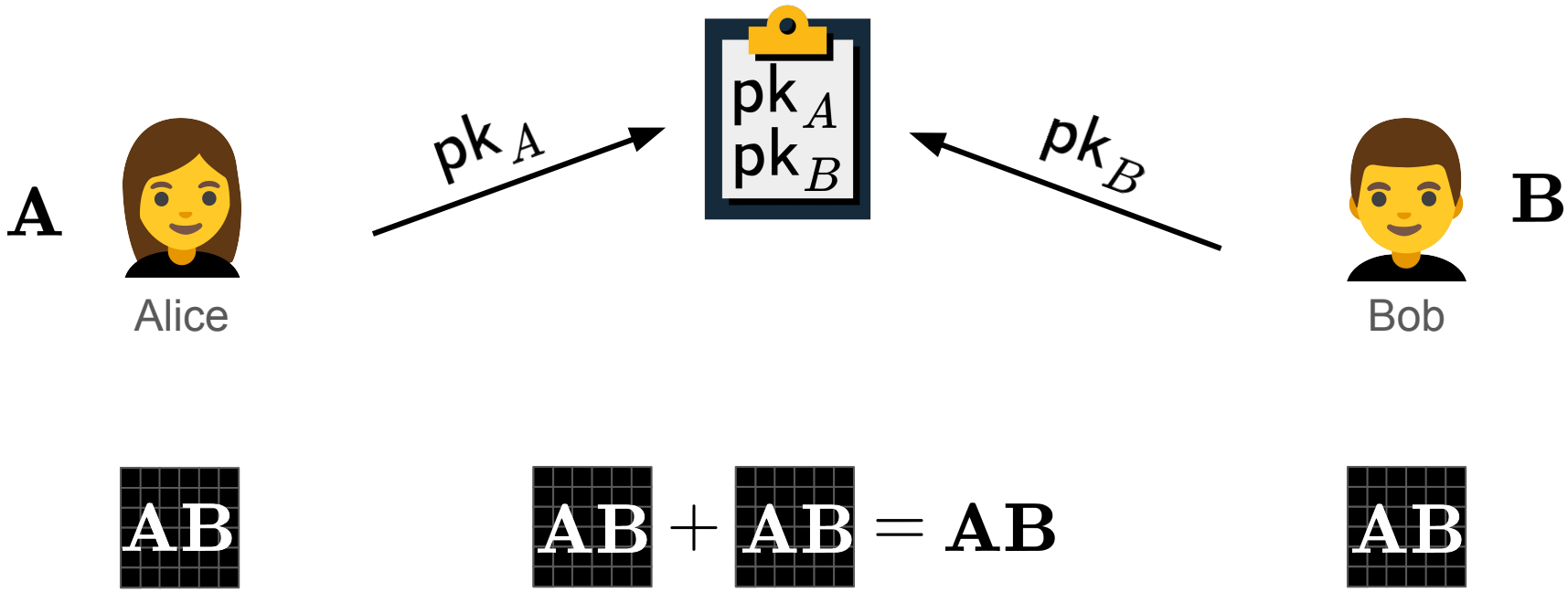
Tool: Succinct Non-Interactive Matrix Multiplication



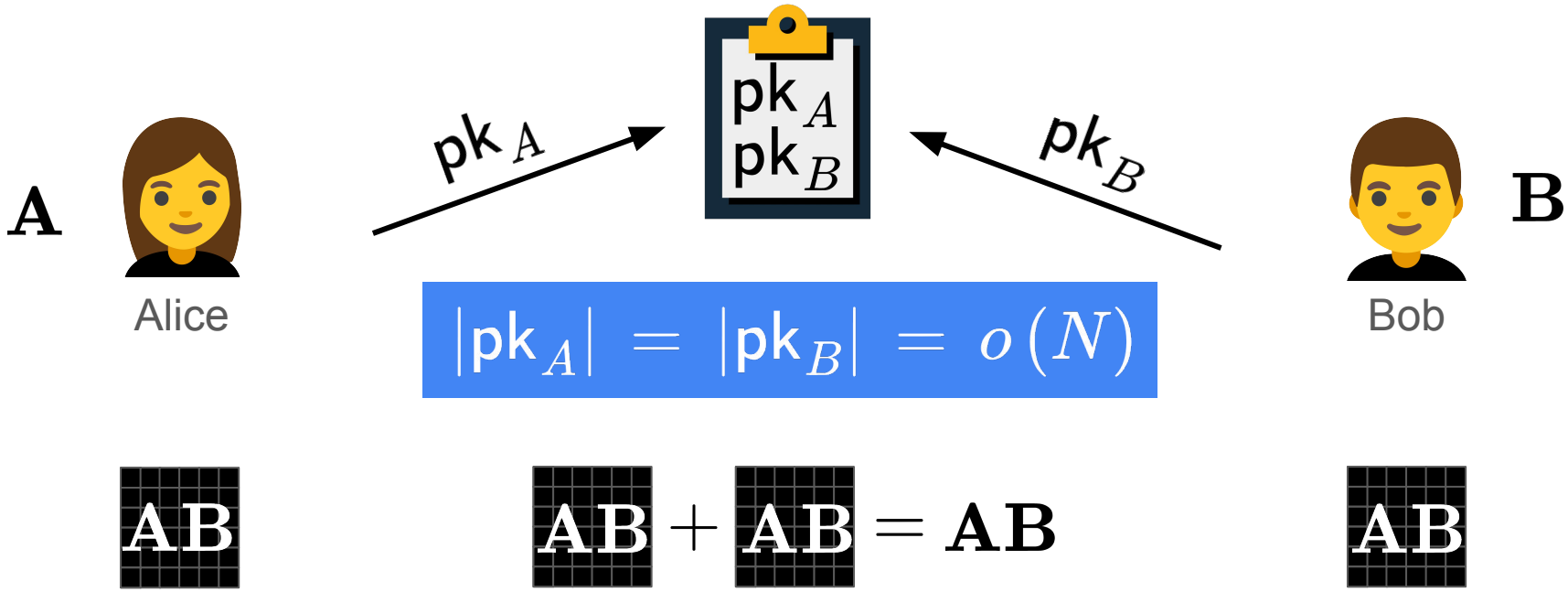
Tool: Succinct Non-Interactive Matrix Multiplication



Tool: Succinct Non-Interactive Matrix Multiplication



Tool: Succinct Non-Interactive Matrix Multiplication



Using succinct matrix multiplication to realize a NIDPF

Step 1: Reinterpret Indices via CRT

t_A 

Alice

 t_B

Bob

Step 1: Reinterpret Indices via CRT

t_A 

Alice

$$t_A \rightarrow t_A = i_A \cdot \ell + j_A$$

t_B

Bob

Step 1: Reinterpret Indices via CRT

t_A  Alice

$$t_A \rightarrow t_A = i_A \cdot \ell + j_A$$

$$t_B \rightarrow t_B = i_B \cdot \ell + j_B$$

 t_B

Bob

Step 1: Reinterpret Indices via CRT

t_A  Alice

$$t_A \rightarrow t_A = i_A \cdot \ell + j_A$$

$$t_B \rightarrow t_B = i_B \cdot \ell + j_B$$

$$\ell \approx \sqrt{N}$$

 t_B

Bob

Step 2: Define matrices representing secret indices

t_A 

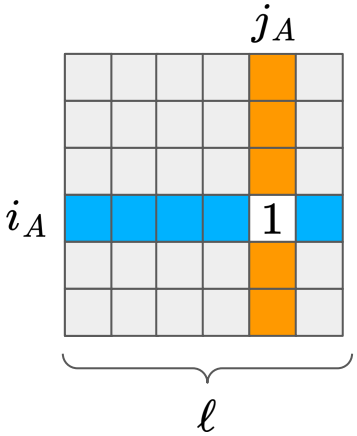
Alice

$$t_A \rightarrow t_A = 3 \cdot \ell + 4$$

$$t_B \rightarrow t_B = i_B \cdot \ell + j_B$$

Bob t_B

$$\mathbf{A} =$$



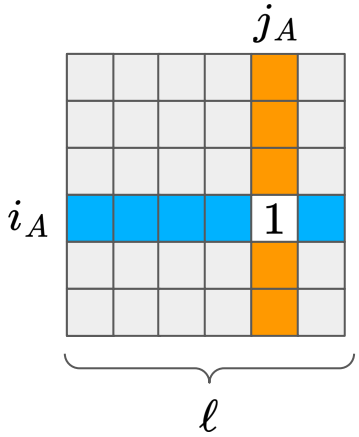
Step 2: Define matrices representing secret indices

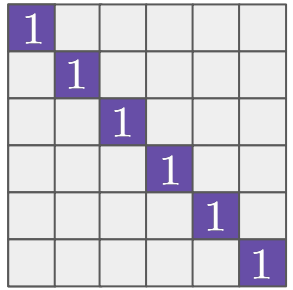
t_A  Alice

$$t_A \rightarrow t_A = 3 \cdot \ell + 4$$

$$t_B \rightarrow t_B = i_B \cdot \ell + 0$$

 t_B Bob

$\mathbf{A} =$ 

$\mathbf{B} =$ 

Cyclic Shift Matrix

Step 2: Define matrices representing secret indices

t_A 

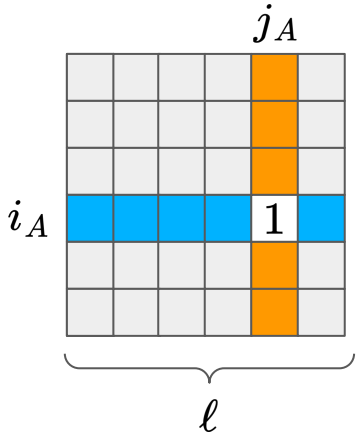
Alice

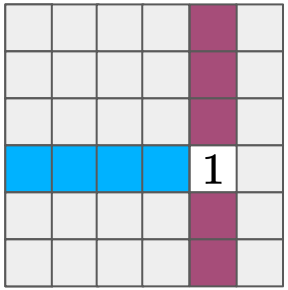
$$t_A \rightarrow t_A = 3 \cdot \ell + 4$$

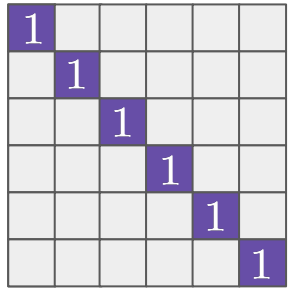
$$t_B \rightarrow t_B = i_B \cdot \ell + 0$$

 t_B

Bob

$\mathbf{A} =$ 

$\mathbf{AB} =$ 

$\mathbf{B} =$ 

Cyclic Shift Matrix

Step 2: Define matrices representing secret indices

t_A 

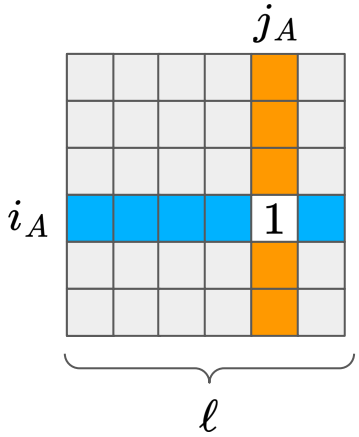
Alice

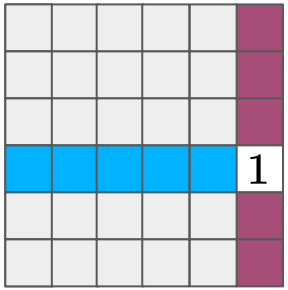
$$t_A \rightarrow t_A = 3 \cdot \ell + 4$$

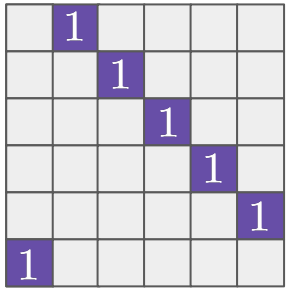
$$t_B \rightarrow t_B = i_B \cdot \ell + 1$$

 t_B

Bob

$\mathbf{A} =$ 

$\mathbf{AB} =$ 

$\mathbf{B} =$ 

Cyclic Shift Matrix

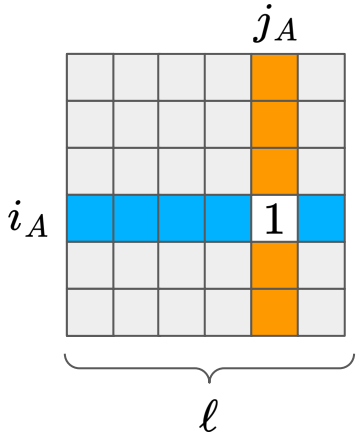
Step 2: Define matrices representing secret indices

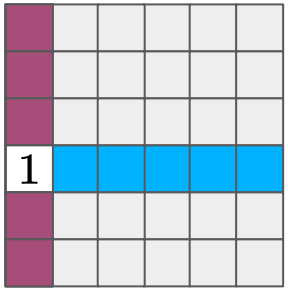
t_A 
Alice

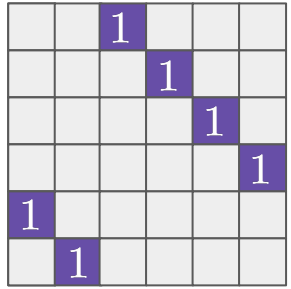
$$t_A \rightarrow t_A = 3 \cdot \ell + 4$$

$$t_B \rightarrow t_B = i_B \cdot \ell + 2$$


Bob t_B

$\mathbf{A} =$ 

$\mathbf{AB} =$ 

$\mathbf{B} =$ 

Cyclic Shift Matrix

Step 2: Define matrices representing secret indices

t_A 

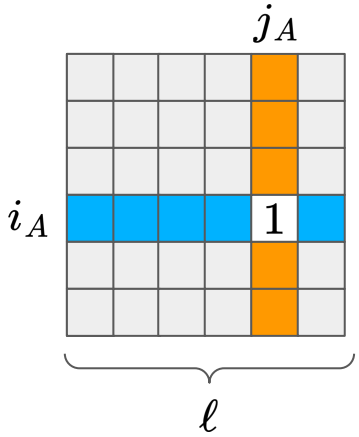
Alice

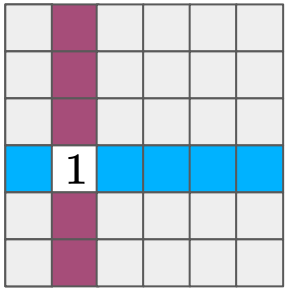
$$t_A \rightarrow t_A = 3 \cdot \ell + 4$$

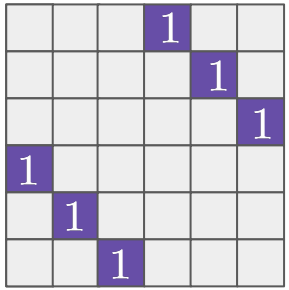
$$t_B \rightarrow t_B = i_B \cdot \ell + 3$$

 t_B

Bob

$\mathbf{A} =$ 

$\mathbf{AB} =$ 

$\mathbf{B} =$ 

Cyclic Shift Matrix

Step 2: Define matrices representing secret indices

t_A 

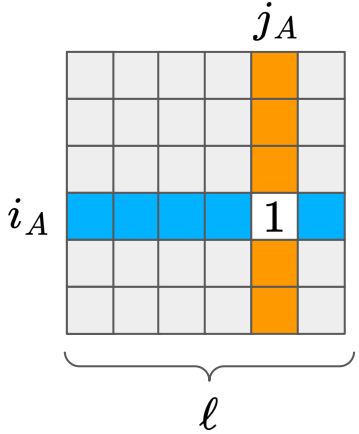
Alice

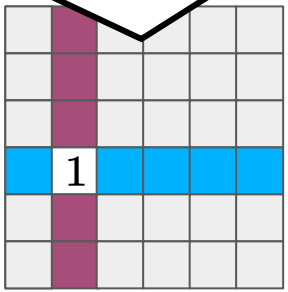
$$t_A \rightarrow t_A = 3 \cdot \ell + 4$$

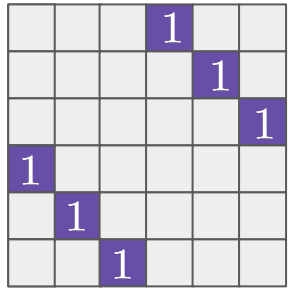
$$t_B \rightarrow t_B = i_B \cdot \ell + 3$$

Bob t_B

$$t = i_A \cdot \ell + j_A + j_B$$

$\mathbf{A} =$ 

$\mathbf{AB} =$ 

$\mathbf{B} =$ 

Cyclic Shift Matrix

Step 2: Define matrices representing secret indices

t_A 

Alice

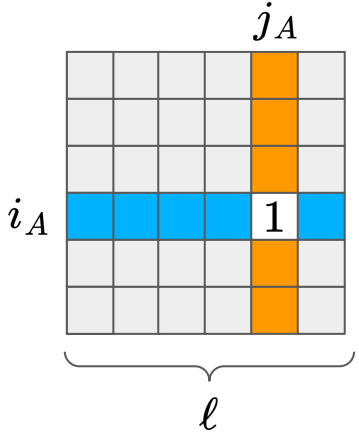
$$t_A \rightarrow t_A = 3 \cdot \ell + 4$$

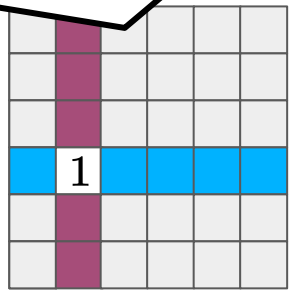
$$t_B \rightarrow t_B = i_B \cdot \ell + 3$$

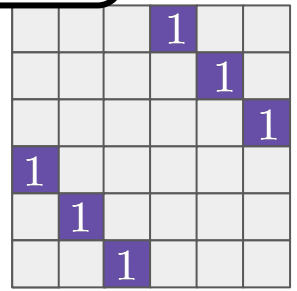
 t_B

Bob

Want: $t = (i_A + i_B) \cdot \ell + j_A + j_B$

$\mathbf{A} =$ 

$\mathbf{AB} =$ 

$\mathbf{B} =$ 

Cyclic Shift Matrix

Step 2: Define matrices representing secret indices

t_A 

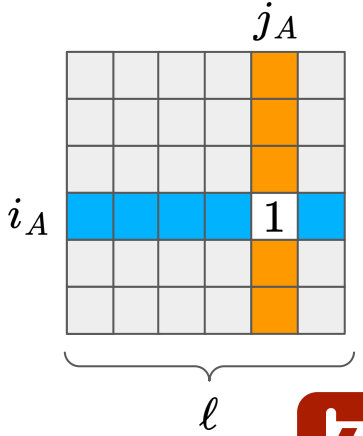
Alice

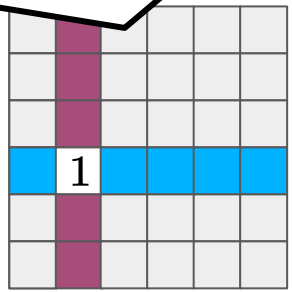
$$t_A \rightarrow t_A = 3 \cdot \ell + 4$$

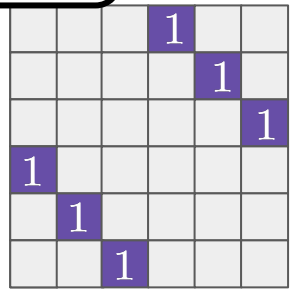
$$t_B \rightarrow t_B = i_B \cdot \ell + 3$$

Bob t_B

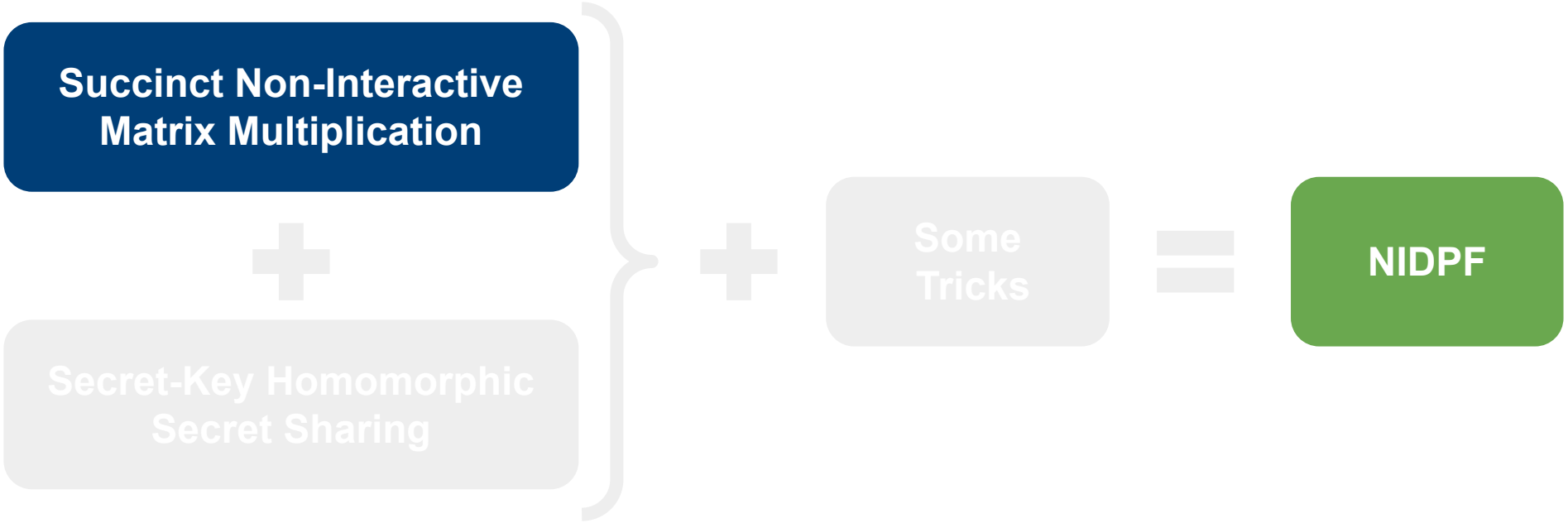
Want: $t = (i_A + i_B) \cdot \ell + j_A + j_B$

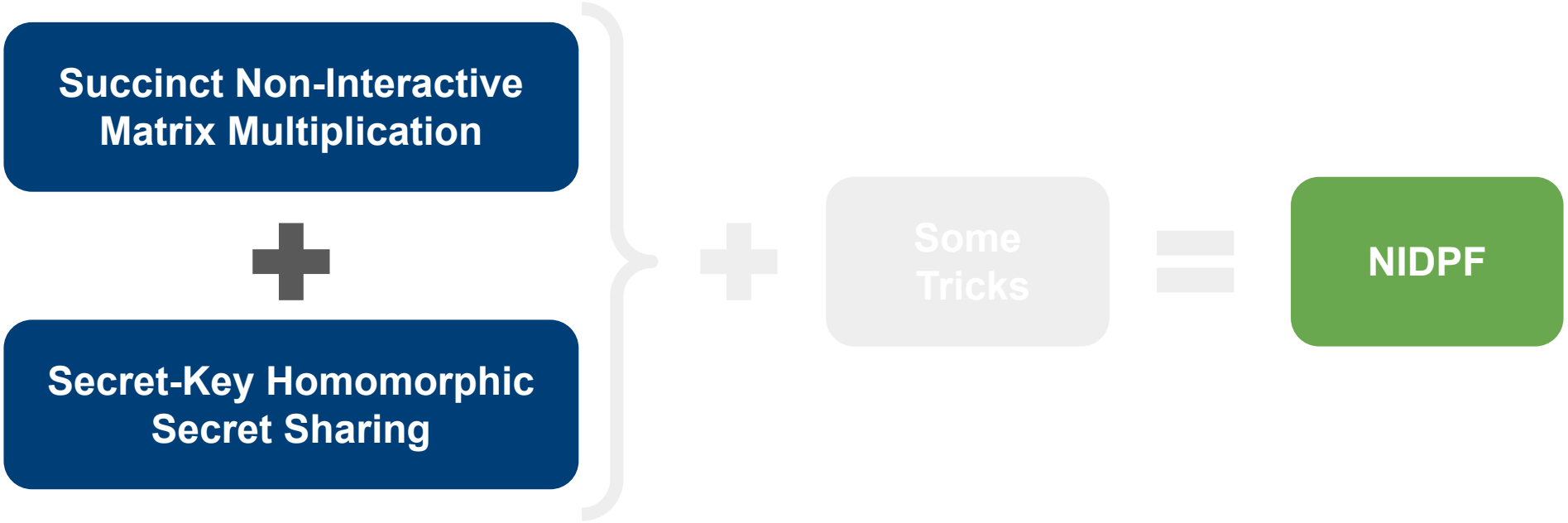
$\mathbf{A} =$ 

$\mathbf{AB} =$ 

$\mathbf{B} =$ 

Problem: Matrix multiplication just shifts the columns


Succinct Non-Interactive Matrix Multiplication
+
Secret-Key Homomorphic Secret Sharing
+
Some Tricks
=
NIDPF


Succinct Non-Interactive Matrix Multiplication
Secret-Key Homomorphic Secret Sharing
Some Tricks
NIDPF

Tool: Homomorphic Secret Sharing [BGI'16]

Alice

Bob

Tool: Homomorphic Secret Sharing [BGI'16]

$$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, x)$$

Trusted Setup

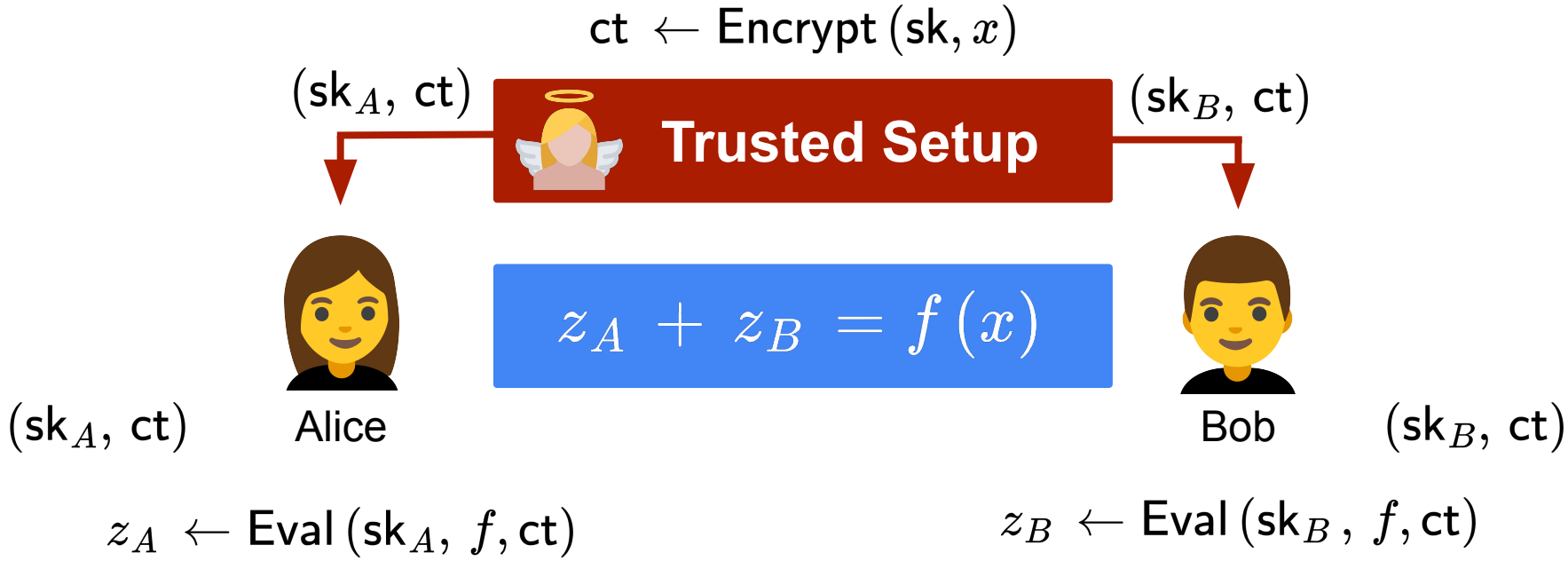
Alice

Bob

Tool: Homomorphic Secret Sharing [BGI'16]

Tool: Homomorphic Secret Sharing [BGI'16]

Tool: Homomorphic Secret Sharing [BGI'16]




ct ← Encrypt (sk, $\mathbf{B_{row}}$)
Trusted Setup
(sk_A, ct)
(sk_B, ct)
t_A
t_B
Alice
Bob

$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{B_{row}})$

Trusted Setup

$(\mathsf{sk}_A, \mathsf{ct})$

$(\mathsf{sk}_B, \mathsf{ct})$

t_A

Alice

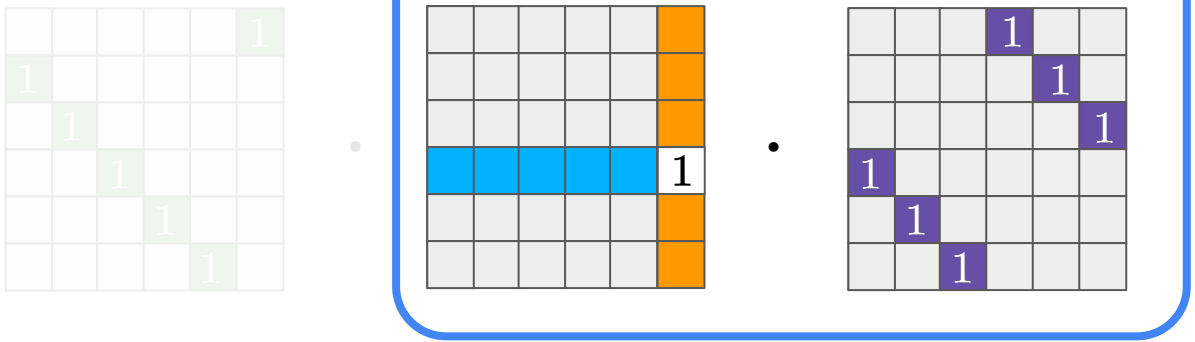
t_B

Bob

Compute using succinct matrix multiplication protocol

Row Shift Matrix

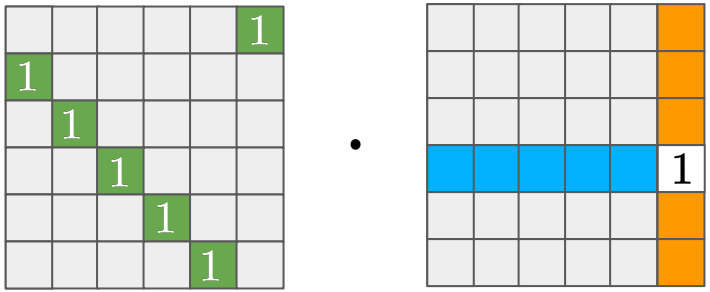
Col Shift Matrix



$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{B_{row}})$

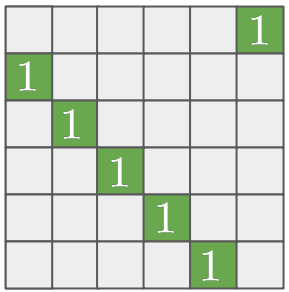
Compute using HSS

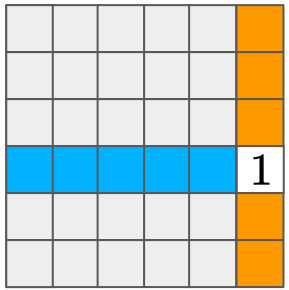
Row Shift Matrix



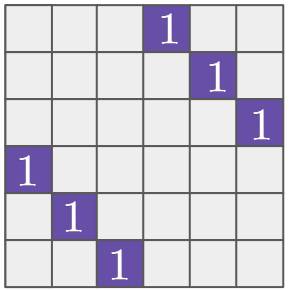
Col Shift Matrix

Row Shift Matrix





Col Shift Matrix



$$\mathsf{ct} \leftarrow \mathsf{Encrypt}\,(\mathsf{sk}, \mathbf{B_{row}})$$

Trusted Setup

Alice

t_A

$(\mathsf{sk}_A,\ \mathsf{ct})$

$(\mathsf{sk}_B,\ \mathsf{ct})$

Bob

t_B

Problem 1: who runs the trusted setup?

Row Shift Matrix

Col Shift Matrix

$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{B_{row}})$

Trusted Setup

$(\mathsf{sk}_A, \mathsf{ct})$ $(\mathsf{sk}_B, \mathsf{ct})$

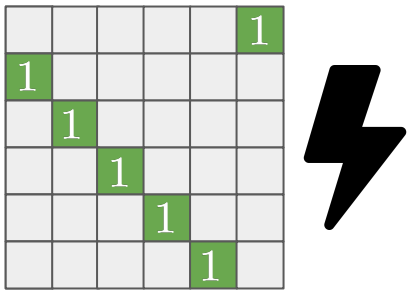
t_A t_B

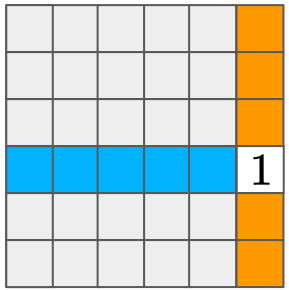
Alice Bob

Problem 1: who runs the trusted setup?

Problem 2: cannot evaluate HSS over secret shares

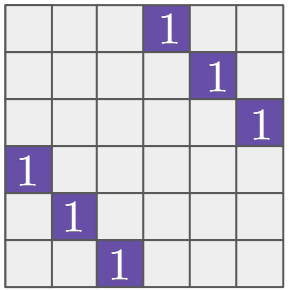
Row Shift Matrix

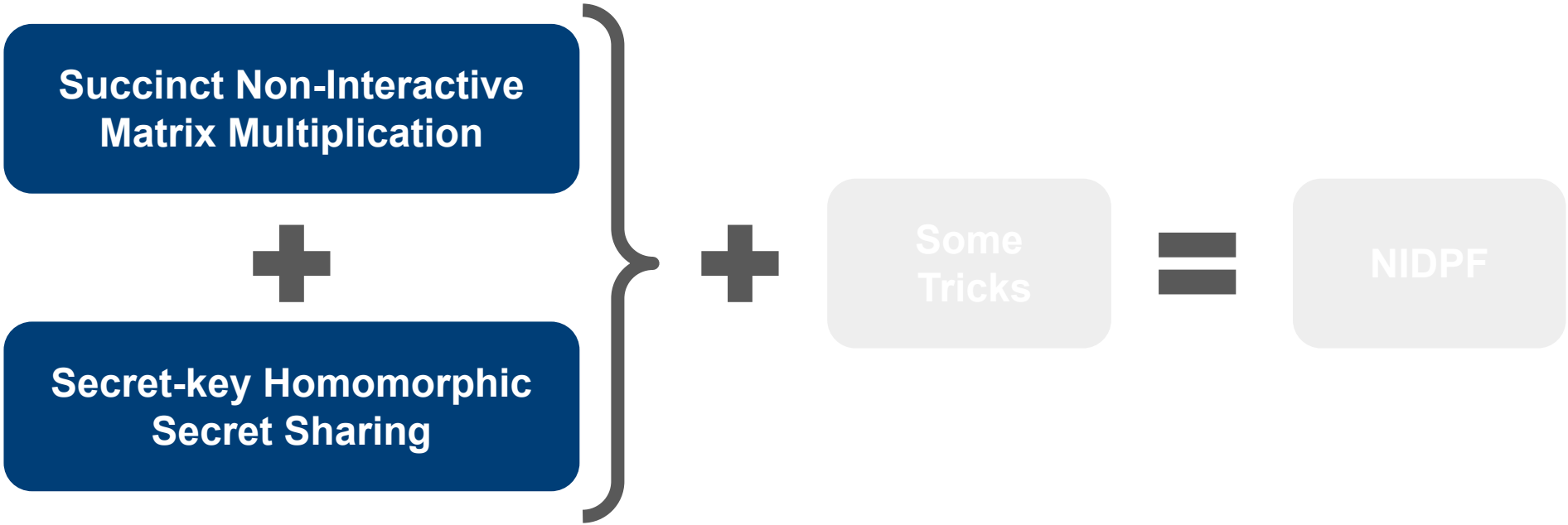


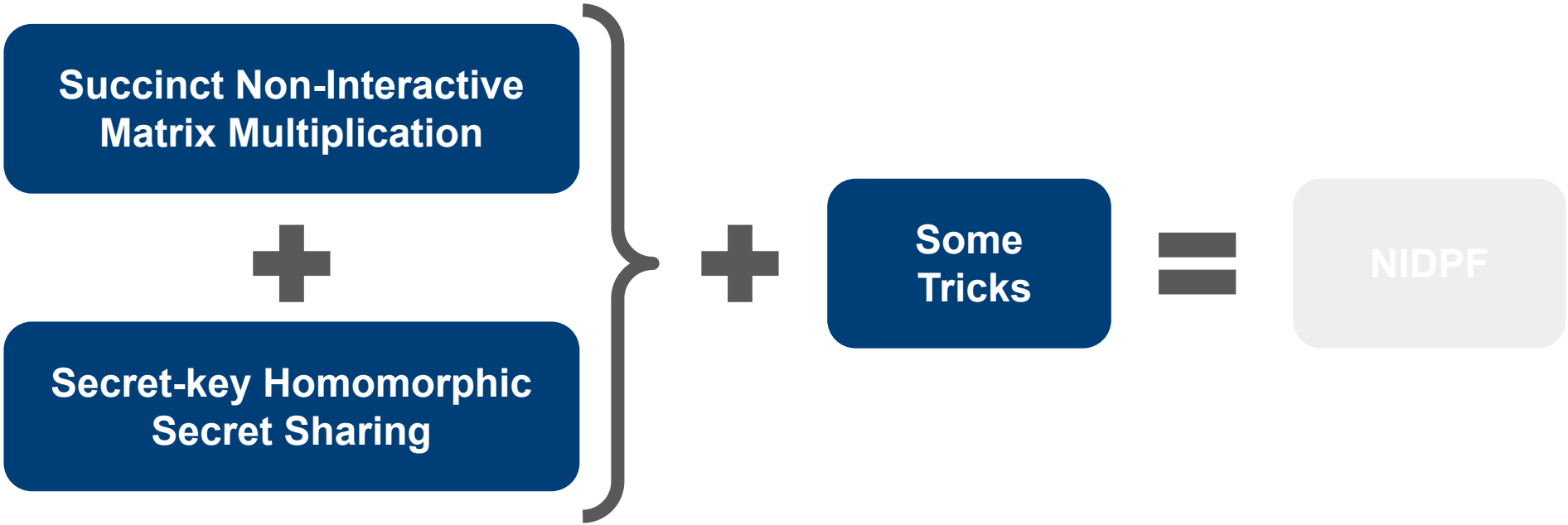


·

Col Shift Matrix



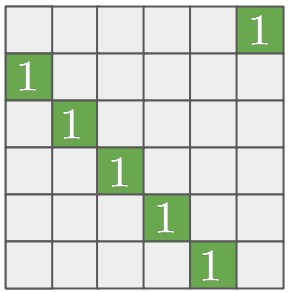

Succinct Non-Interactive Matrix Multiplication
+
Secret-key Homomorphic Secret Sharing
+
Some Tricks
=
NIDPF

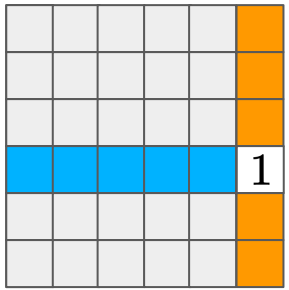

Succinct Non-Interactive Matrix Multiplication
+
Secret-key Homomorphic Secret Sharing
+
Some Tricks
=
NIDPF

$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{B_{row}})$

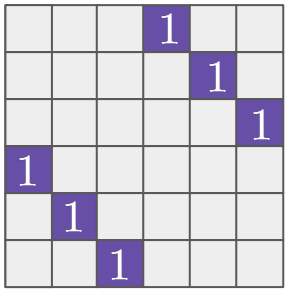
Problem 1: who runs the trusted setup?

Row Shift Matrix





Col Shift Matrix



Trick 1: Bob runs the setup!

$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{B_{row}})$

$(\mathsf{sk}_A, \mathsf{ct})$ $(\mathsf{sk}_B, \mathsf{ct})$

t_A

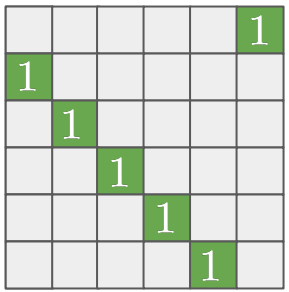
Alice

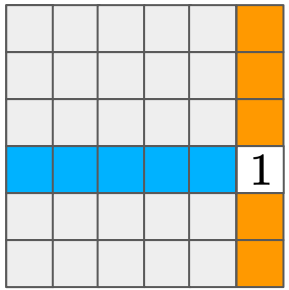
$(\mathsf{sk}_A, \mathsf{ct})$

Bob

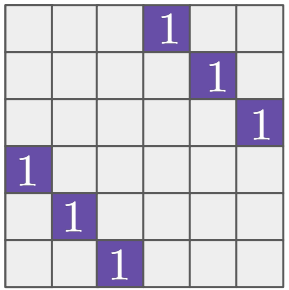
t_B

Row Shift Matrix





Col Shift Matrix



$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{B_{row}})$

$(\mathsf{sk}_A, \mathsf{ct})$ $(\mathsf{sk}_B, \mathsf{ct})$

t_A

Alice

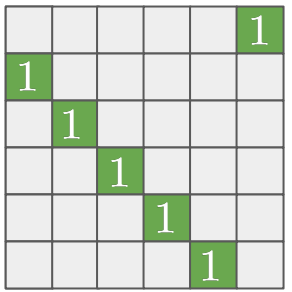
$(\mathsf{sk}_A, \mathsf{ct})$

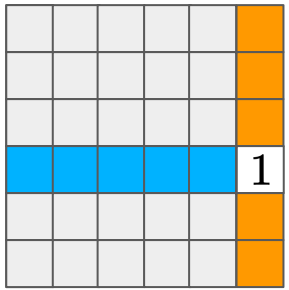
Bob

t_B

Problem 2: cannot evaluate HSS over secret shares

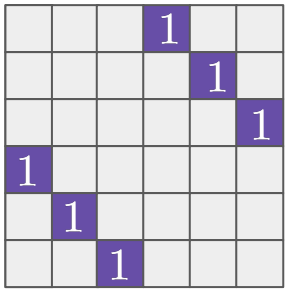
Row Shift Matrix





·

Col Shift Matrix



Informal Theorem ([CMPR'23])

Existing homomorphic secret sharing schemes under DDH, DCR, QR, and LWE have input shares and memory shares where:

Informal Theorem ([CMPR'23])

Existing homomorphic secret sharing schemes under DDH, DCR, QR, and LWE have input shares and memory shares where:

- **Input shares** are additively-homomorphic ciphertexts encrypted with key sk

Informal Theorem ([CMPR'23])

Existing homomorphic secret sharing schemes under DDH, DCR, QR, and LWE have input shares and memory shares where:

- **Input shares** are additively-homomorphic ciphertexts encrypted with key sk
- **Memory shares** of x are additive secret shares of the tuple $(x, \mathsf{sk} \cdot x)$

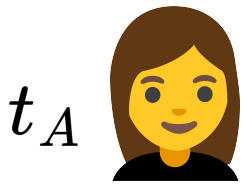
Informal Theorem ([CMPR'23])

Existing homomorphic secret sharing schemes under DDH, DCR, QR, and LWE have input shares and memory shares where:

- **Input shares** are additively-homomorphic ciphertexts encrypted with key sk
- **Memory shares** of x are additive secret shares of the tuple $(x, \mathsf{sk} \cdot x)$

There exists a Mult algorithm that computes additive shares of the product between an input share and a memory share.

Trick 2: Convert additive shares into “memory shares”



t_A

Alice

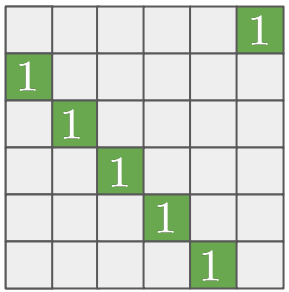
$(\mathsf{sk}_A, \mathsf{ct})$

t_B

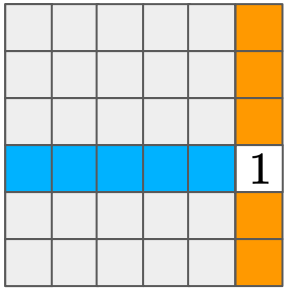
Bob

$\mathsf{sk} = \mathsf{sk}_A + \mathsf{sk}_B$

Row Shift Matrix

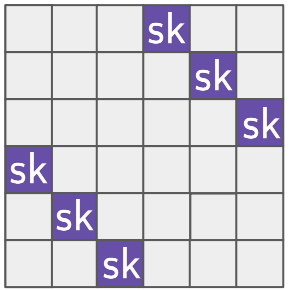


·

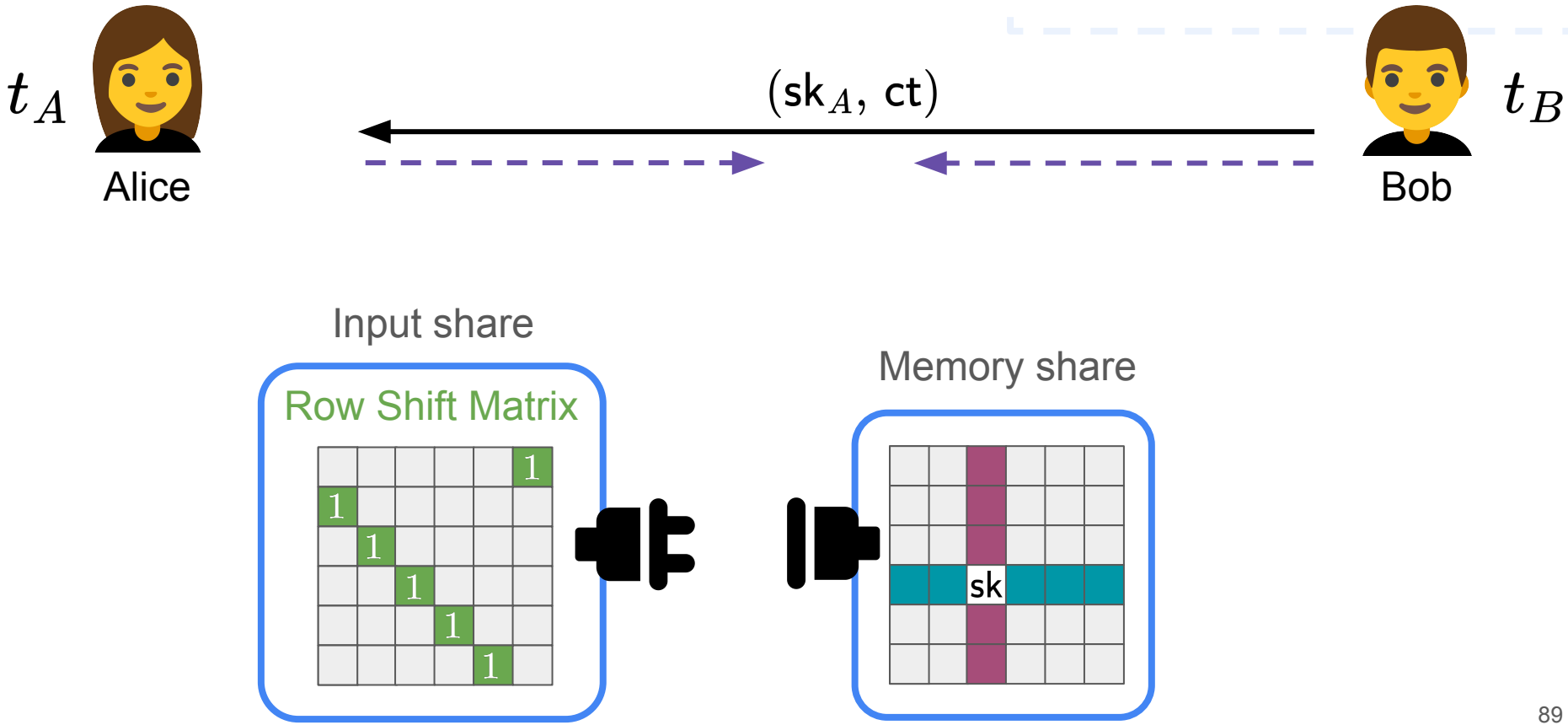


·

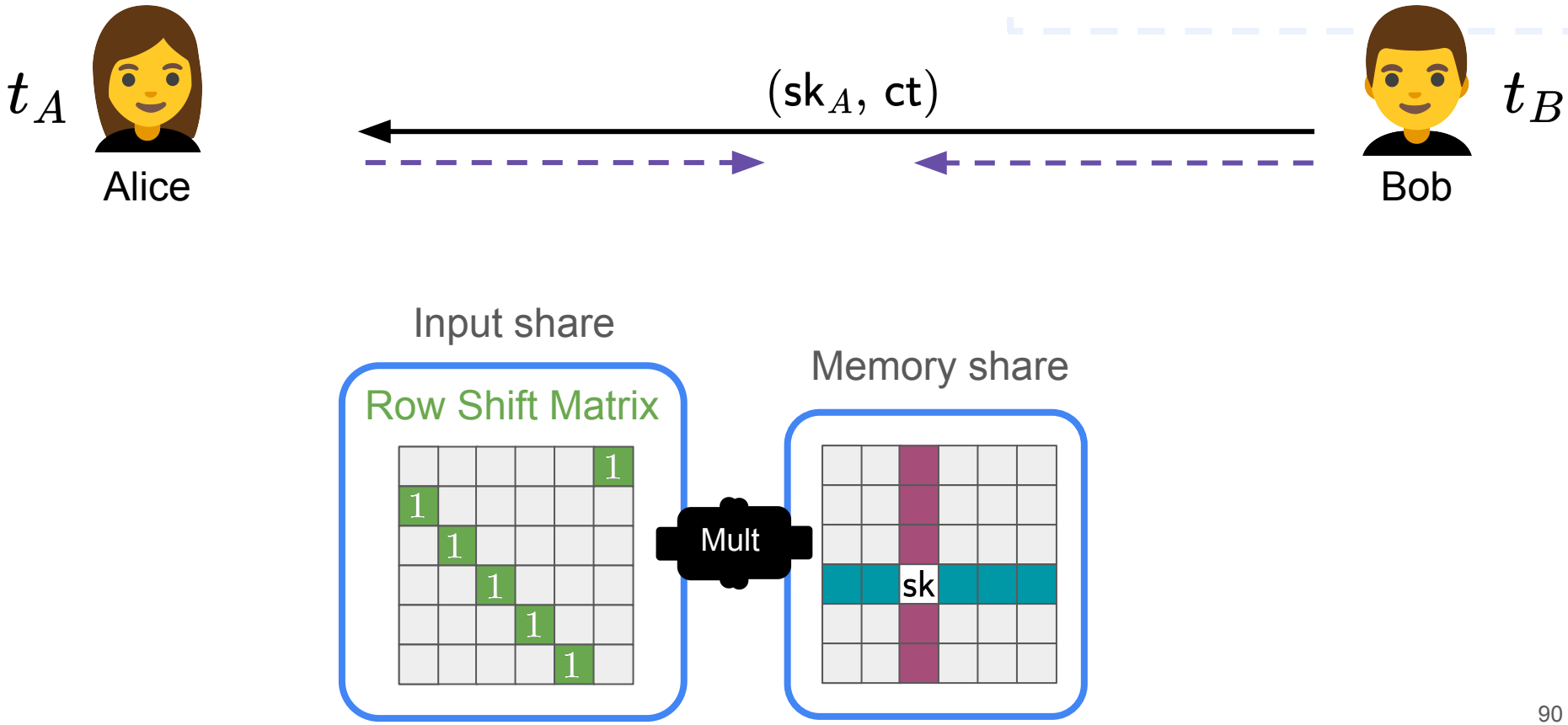
Col Shift Matrix



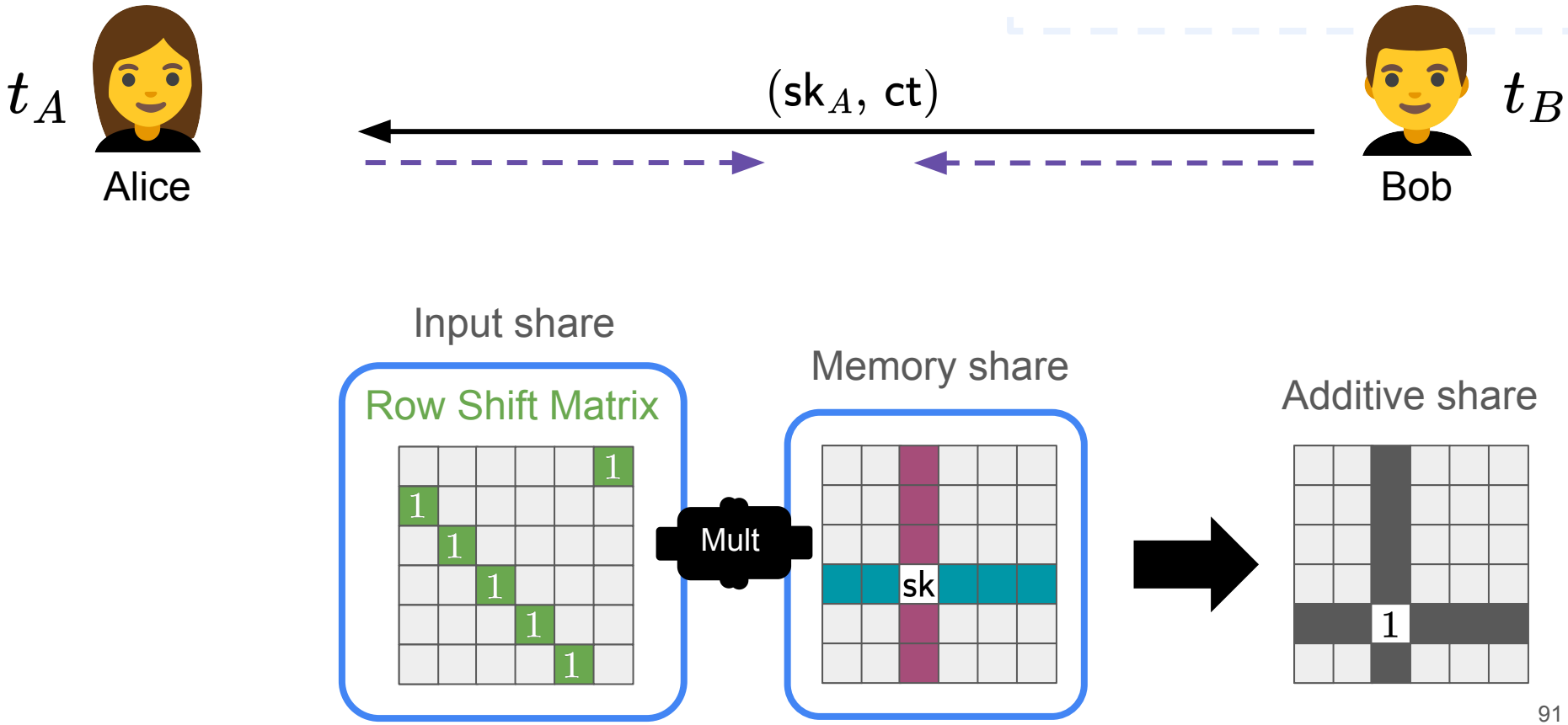
Trick 2: Convert additive shares into “memory shares”



Trick 2: Convert additive shares into “memory shares”



Trick 2: Convert additive shares into “memory shares”



Putting things together

Putting things together

1

t_A  Alice

$$t_A \rightarrow t_A = i_A \cdot \ell + j_A$$

$$t_B \rightarrow t_B = i_B \cdot \ell + j_B$$

 t_B

Bob

Putting things together

2

t_A 

Alice

 t_B

Bob

$\mathsf{pk}_A := \mathsf{pk}_A^{\mathsf{matmul}}$

Putting things together

2

t_A 

Alice

 t_B

Bob

$\mathsf{pk}_A := \mathsf{pk}_A^{\mathsf{matmul}}$

$$\mathbf{A} = \begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & 1 \\ & & & & & \\ & & & & & \end{bmatrix}$$

Putting things together

2

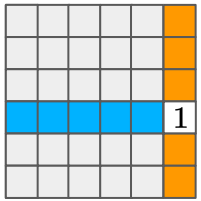
t_A 

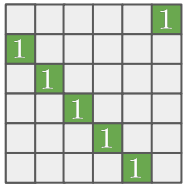
Alice

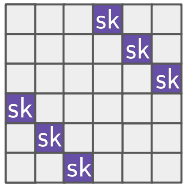
 t_B

Bob

$\mathsf{pk}_A := \mathsf{pk}_A^{\mathsf{matmul}}$

$\mathbf{A} =$ 

$\mathbf{B_{row}} =$ 

$\mathbf{B_{col}} =$ 

Putting things together

$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{B_{row}})$

$(\mathsf{sk}_A, \mathsf{ct})$ $(\mathsf{sk}_B, \mathsf{ct})$

2

t_A

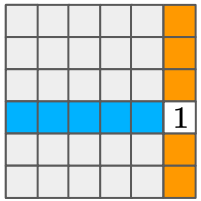
Alice

t_B

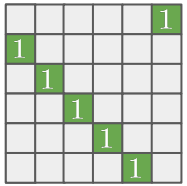
Bob

$\mathsf{pk}_A := \mathsf{pk}_A^{\mathsf{matmul}}$

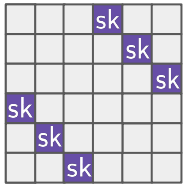
$\mathbf{A} =$



$\mathbf{B_{row}} =$



$\mathbf{B_{col}} =$



Putting things together

$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{B_{row}})$

$(\mathsf{sk}_A, \mathsf{ct})$ $(\mathsf{sk}_B, \mathsf{ct})$

2

t_A 

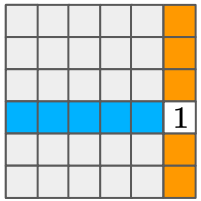
Alice

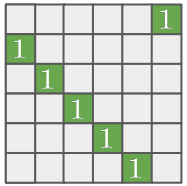
 t_B

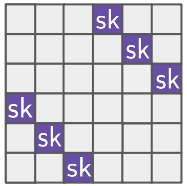
Bob

$\mathsf{pk}_A := \mathsf{pk}_A^{\mathsf{matmul}}$

$\mathsf{pk}_B := (\mathsf{pk}_B^{\mathsf{matmul}}, \mathsf{sk}_A, \mathsf{ct})$

$\mathbf{A} =$ 

$\mathbf{B_{row}} =$ 

$\mathbf{B_{col}} =$ 

Putting things together

$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{B_{row}})$

$(\mathsf{sk}_A, \mathsf{ct})$ $(\mathsf{sk}_B, \mathsf{ct})$

2

t_A

Alice

pk_A

pk_B

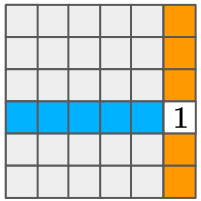
t_B

Bob

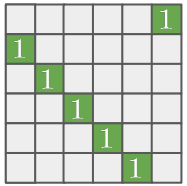
$\mathsf{pk}_A := \mathsf{pk}_A^{\mathsf{matmul}}$

$\mathsf{pk}_B := \left(\mathsf{pk}_B^{\mathsf{matmul}}, \mathsf{sk}_A, \mathsf{ct}\right)$

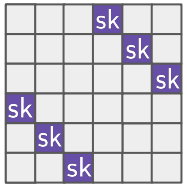
$\mathbf{A} =$



$\mathbf{B_{row}} =$

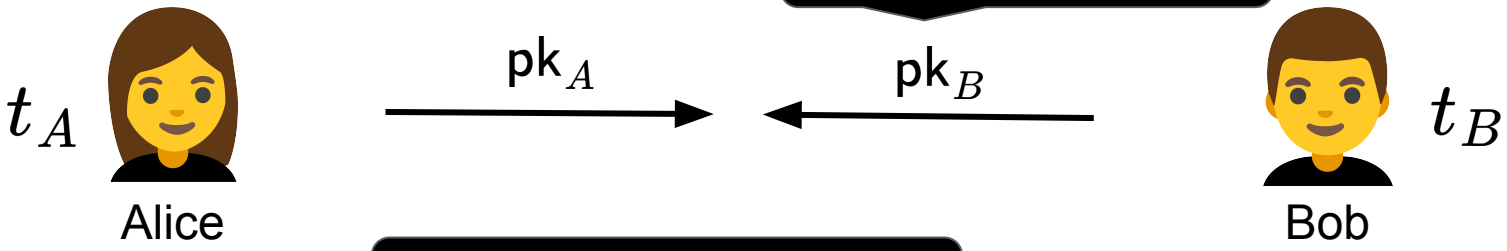


$\mathbf{B_{col}} =$

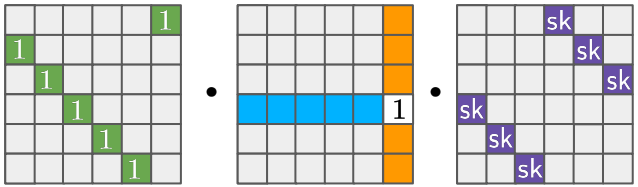


Putting things together

$|\mathsf{pk}_B| = o(N)$

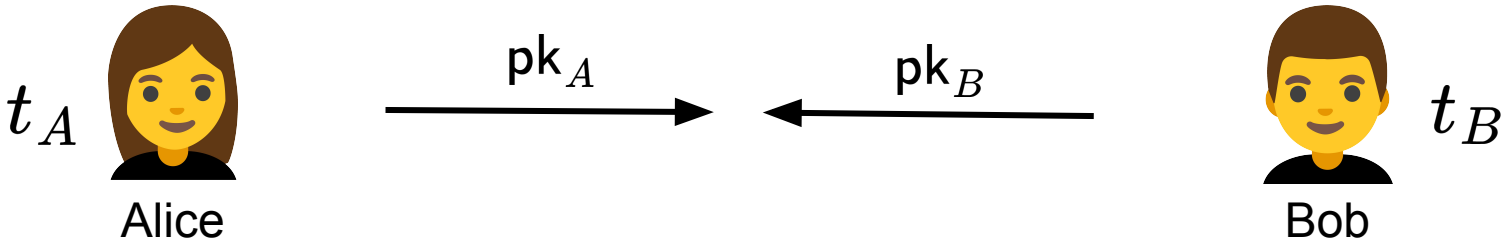


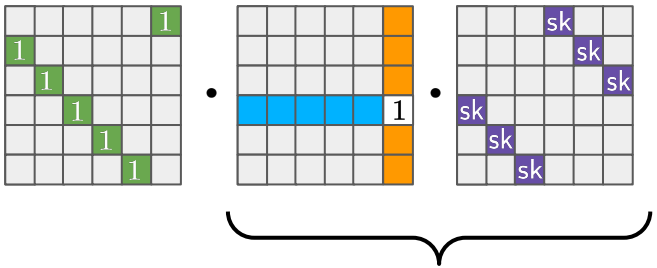
Sparse and compressible



Putting things together

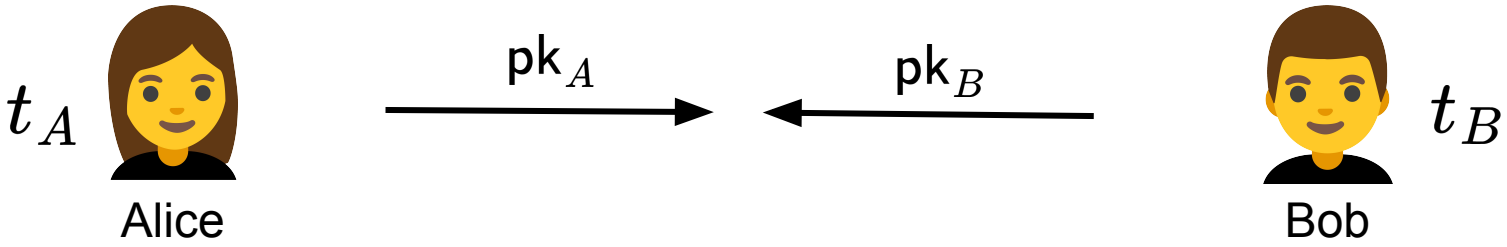
3

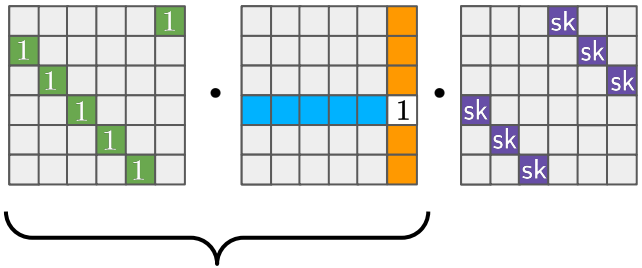




Putting things together

3

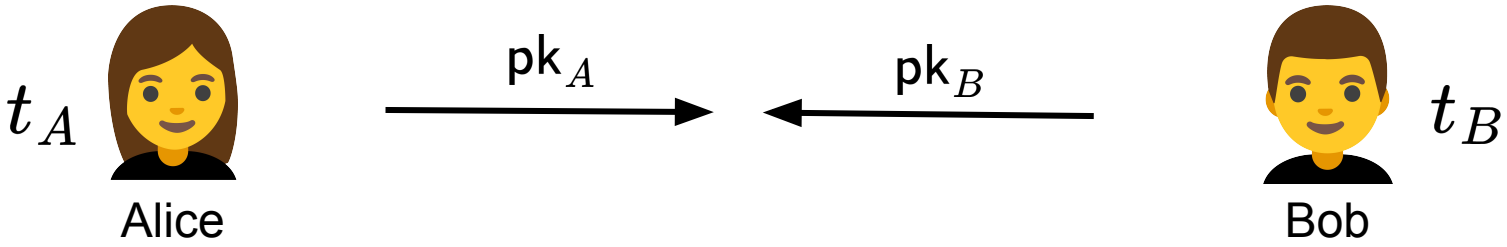


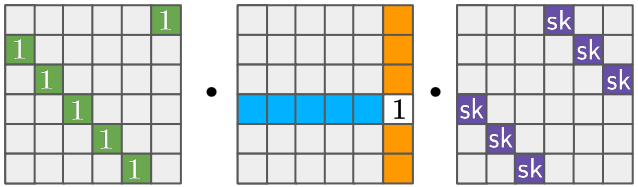


Multiply using Homomorphic Secret Sharing

Putting things together

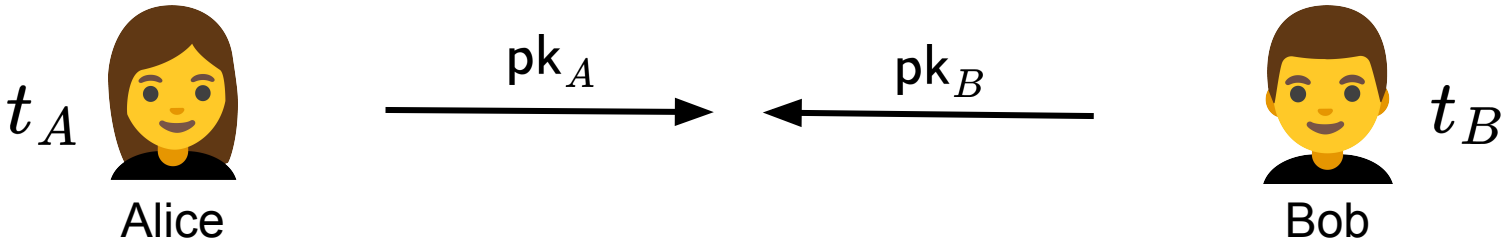
3

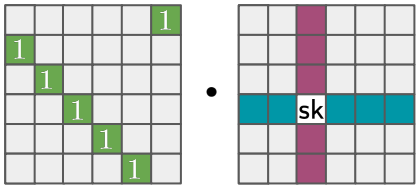




Putting things together

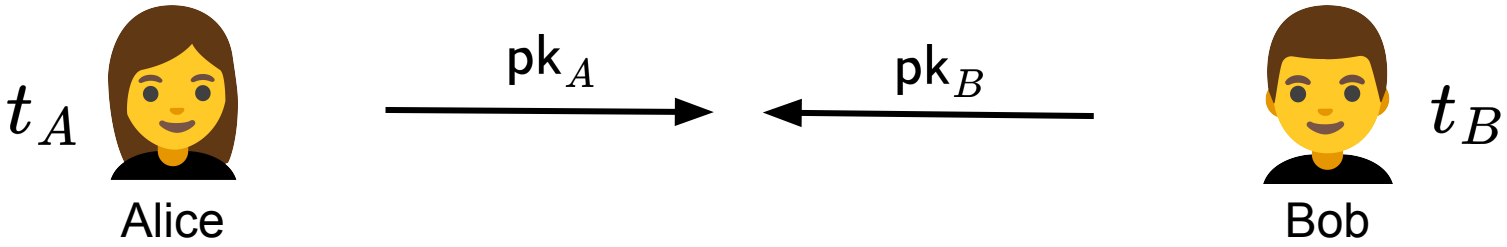
3

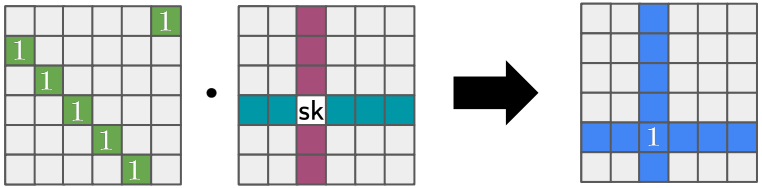




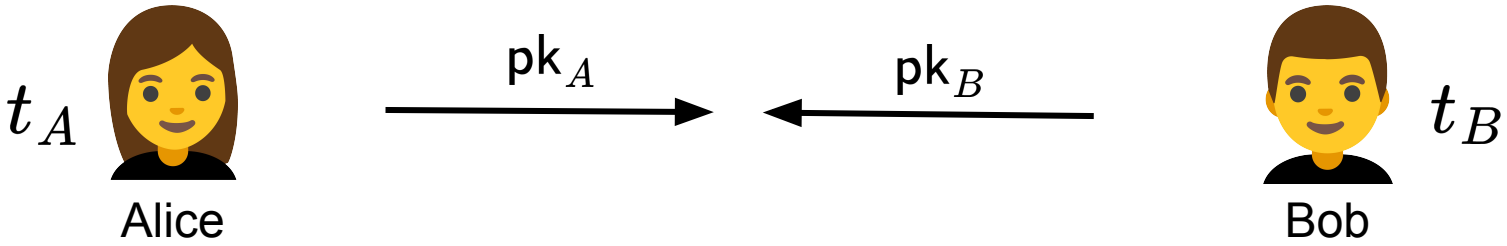
Putting things together

3





Putting things together



Results

NIDPF with domain size N

	Assumptions	Communication	Comments
Spooky [DHRW'16]	LWE OR iO+DDH	log(N)	Requires multi-key FHE

Results

NIDPF with domain size N

	Assumptions	Communication	Comments
Spooky [DHRW'16]	LWE OR iO+DDH	log(N)	Requires multi-key FHE
This work	**DCR**	$N^{2/3}$	

Results

NIDPF with domain size N

	Assumptions	Communication	Comments
Spooky [DHRW'16]	LWE OR iO+DDH	log(N)	Requires multi-key FHE
This work	**DCR**	$N^{2/3}$	
This work	**QR**	$N^{2/3}$	

Results

NIDPF with domain size N

	Assumptions	Communication	Comments
Spooky [DHRW'16]	LWE OR iO+DDH	log(N)	Requires multi-key FHE
This work	**DCR**	$N^{2/3}$	
This work	**QR**	$N^{2/3}$	
This work	**LWE**	$N^{2/3}$	**LWE but "without FHE"**

Results

NIDPF with domain size N

	Assumptions	Communication	Comments
Spooky [DHRW'16]	LWE OR iO+DDH	log(N)	Requires multi-key FHE
This work	**DCR**	$N^{2/3}$	
This work	**QR**	$N^{2/3}$	
This work	**LWE**	$N^{2/3}$	**LWE but "without FHE"**
This work	**SXDH**	$N^{2/3}$	**Random payload DPF**

Results

NIDPF with domain size N

	Assumptions	Communication	Comments
Spooky [DHRW'16]	LWE OR iO+DDH	log(N)	Requires multi-key FHE
This work	**DCR**	$N^{2/3}$	
This work	**QR**	$N^{2/3}$	
This work	**LWE**	$N^{2/3}$	**LWE but "without FHE"**
This work	**SXDH**	$N^{2/3}$	**Random payload DPF**

Still only modestly sublinear. **Open problem: $\sqrt{N}$ or better**

Thank you!

Email: 3s@mit.edu
ePrint: ia.cr/**2025/095**

Non-Interactive Distributed Point Functions

Elette Boyle[1], Lalita Devadas[2], and Sacha Servan-Schreiber[2⋆]

[1] NTT Research and Reichman University
[2] MIT

References

[GI'14]: N. Gilboa and Y. Ishai. "Distributed point functions and their applications."

[BGI'15]: E. Boyle, N. Gilboa, and Y. Ishai. "Function secret sharing."

[BGI'16]: E. Boyle, N. Gilboa, and Y. Ishai. "Breaking the Circuit Size Barrier for Secure Computation Under DDH."

[DHRW'16]: D. Dodis, S. Halevi, R. D. Rothblum, and D. Wichs. "Spooky Encryption and Its Applications."

[Ds'17]: J. Doerner and A. Shelat. "Scaling ORAM for secure computation."

[BCGI'18]: E. Boyle, et al. "Compressing vector OLE."

[DPKY'20]: E. Dauterman, et al. "DORY: An encrypted search system with distributed trust."

References

[BGIK'21]: E. Boyle, et al. "Function secret sharing for mixed-mode and fixed-point secure computation."

[ARS'24]: D. Abram, L. Roy, and P. Scholl. "Succinct Homomorphic Secret Sharing."

[BCMPR'24]: D. Bui, G. Couteau, P. Meyer, A. Passelègue, and M. Riahinia. "Fast Public-Key Silent OT and More from Constrained Naor-Reingold."

[CDHJS'24]: G. Couteau, L. Devadas, A. Hegde, A. Jain, and S. Servan-Schreiber. "Multi-key Homomorphic Secret Sharing."