Non-Interactive Distributed Point Functions

MIT CSAIL

Sacha Servan-Schreiber

Tinfoil

Joint work with
Elette Boyle and Lalita Devadas

NTT Research

Reichman University

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

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

$P_5\left(0\right)$

$\boxed{0}$

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

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

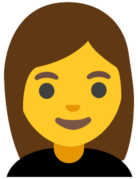
$P_5(0)$	$P_5(1)$	$P_5(2)$	$P_5(3)$	$P_5(4)$	$P_5(5)$	$P_5(6)$	$P_5(7)$	$P_5(8)$
0	0	0	0	0	1	0	0	0

Point Functions

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

$P_5(0)$	$P_5(1)$	$P_5(2)$	$P_5(3)$	$P_5(4)$	$P_5(5)$	$P_5(6)$	$P_5(7)$	$P_5(8)$
0	0	0	0	0	1	0	0	0

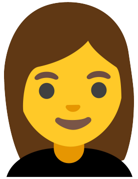
Distributed Point Functions [GI’14]



Alice

Bob

Distributed Point Functions [GI'14]



Alice

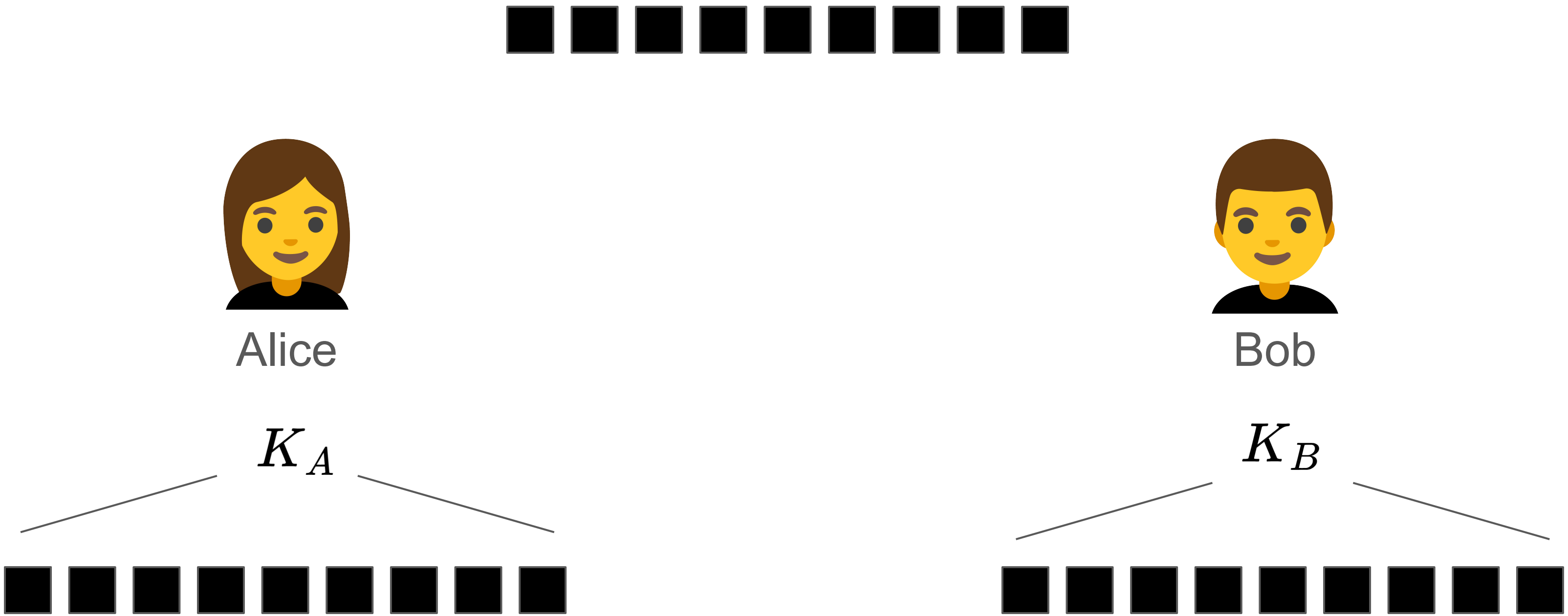
K_A

Bob

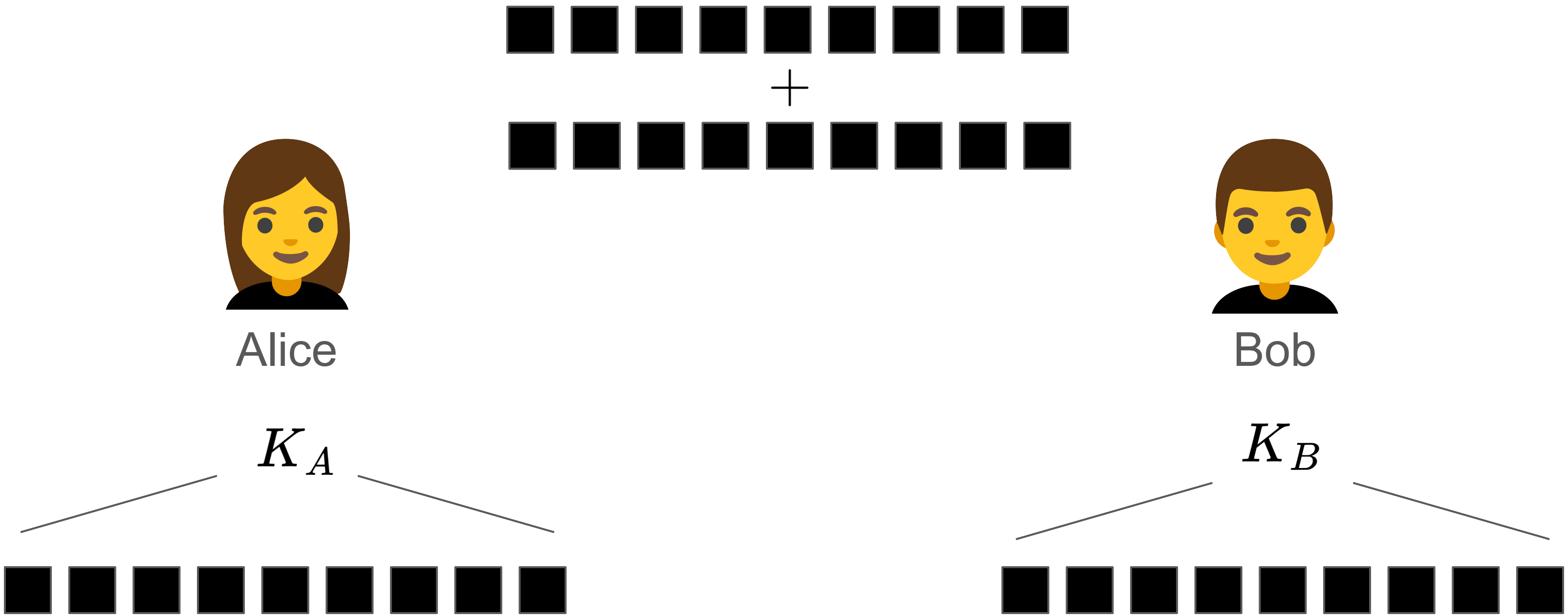
K_B

Distributed Point Functions [GI'14]

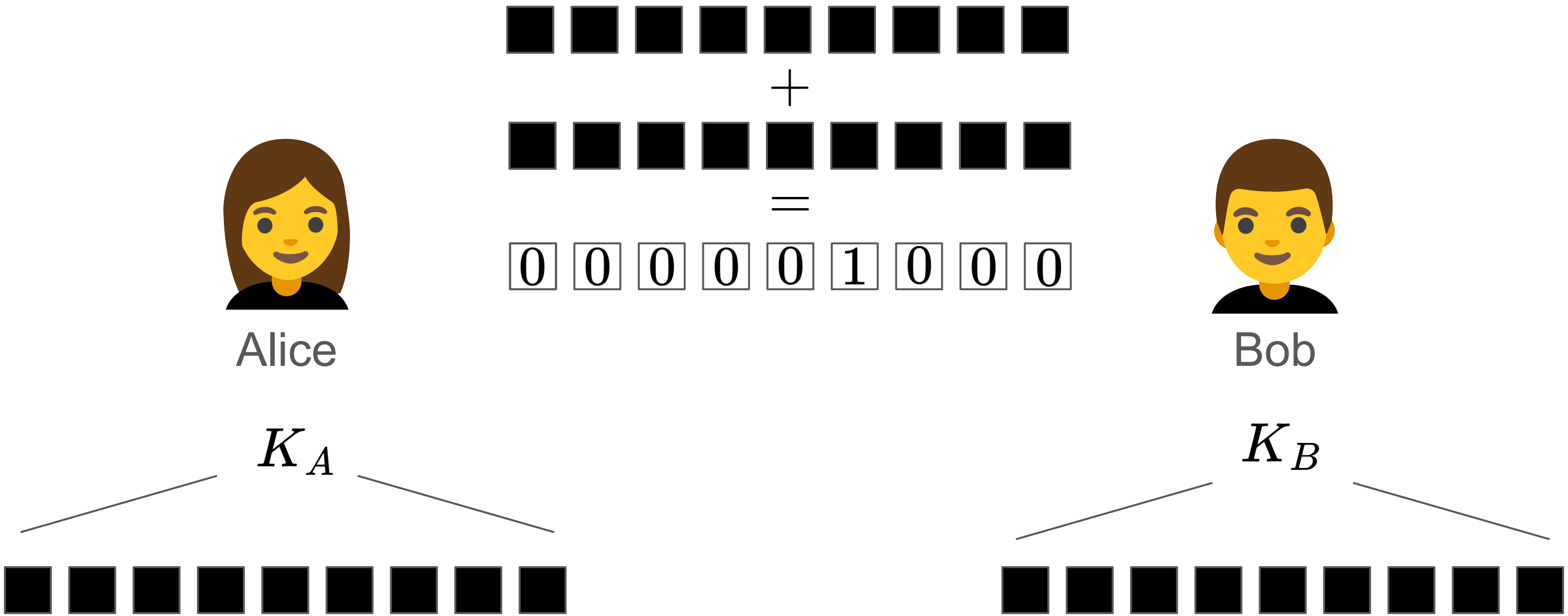
Distributed Point Functions [GI'14]



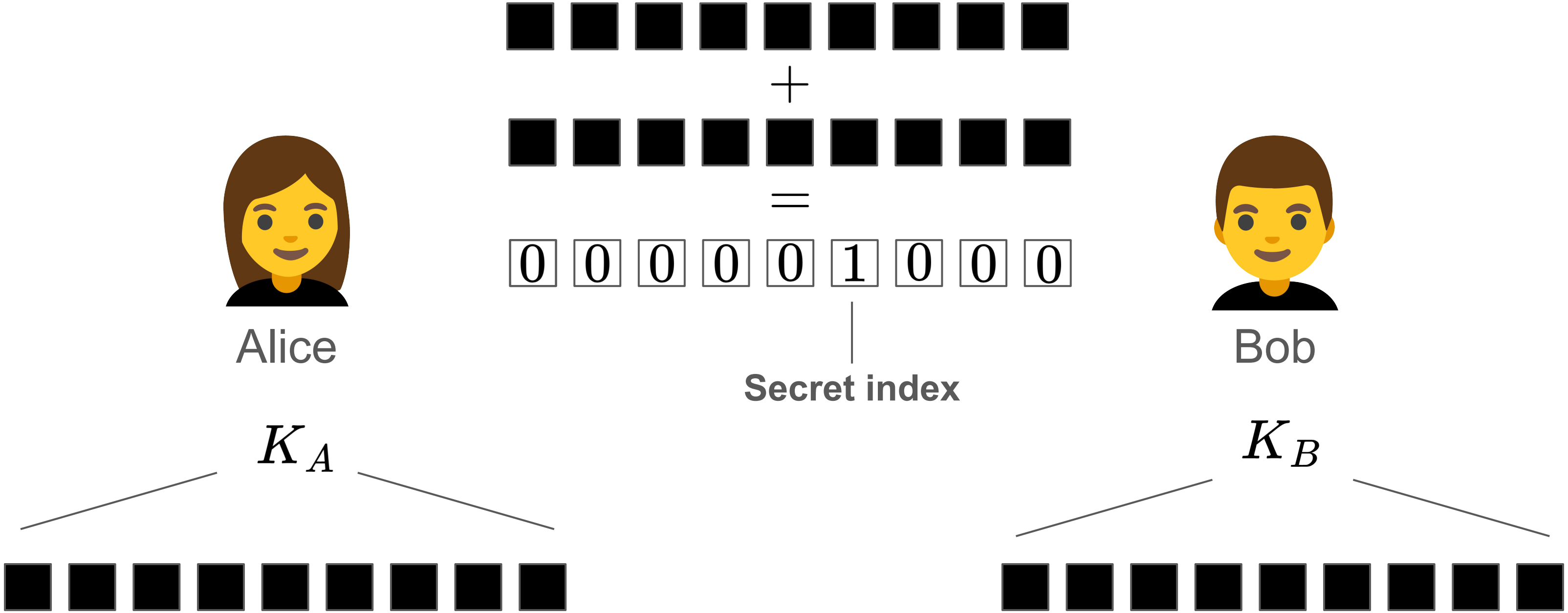
Distributed Point Functions [GI'14]



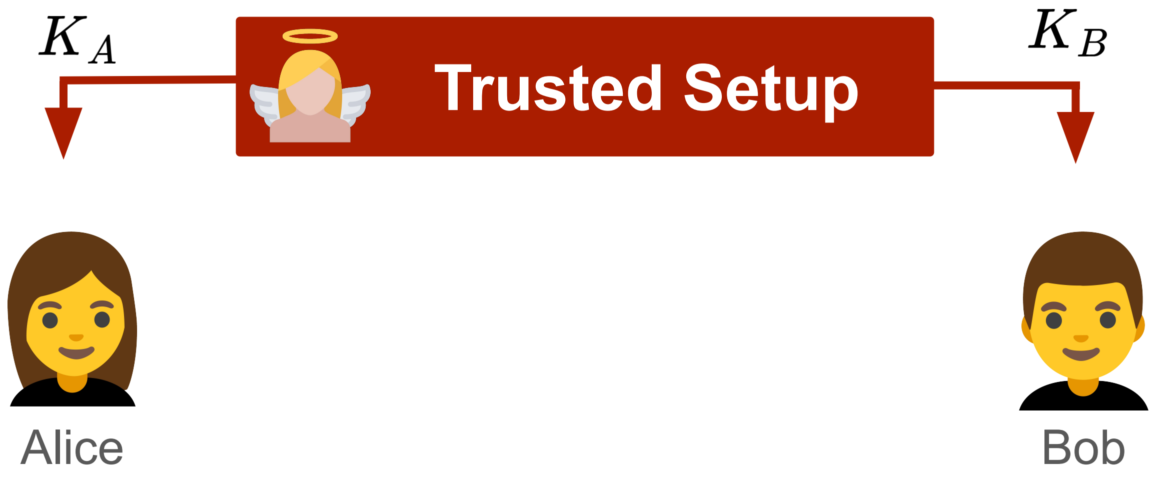
Distributed Point Functions [GI'14]



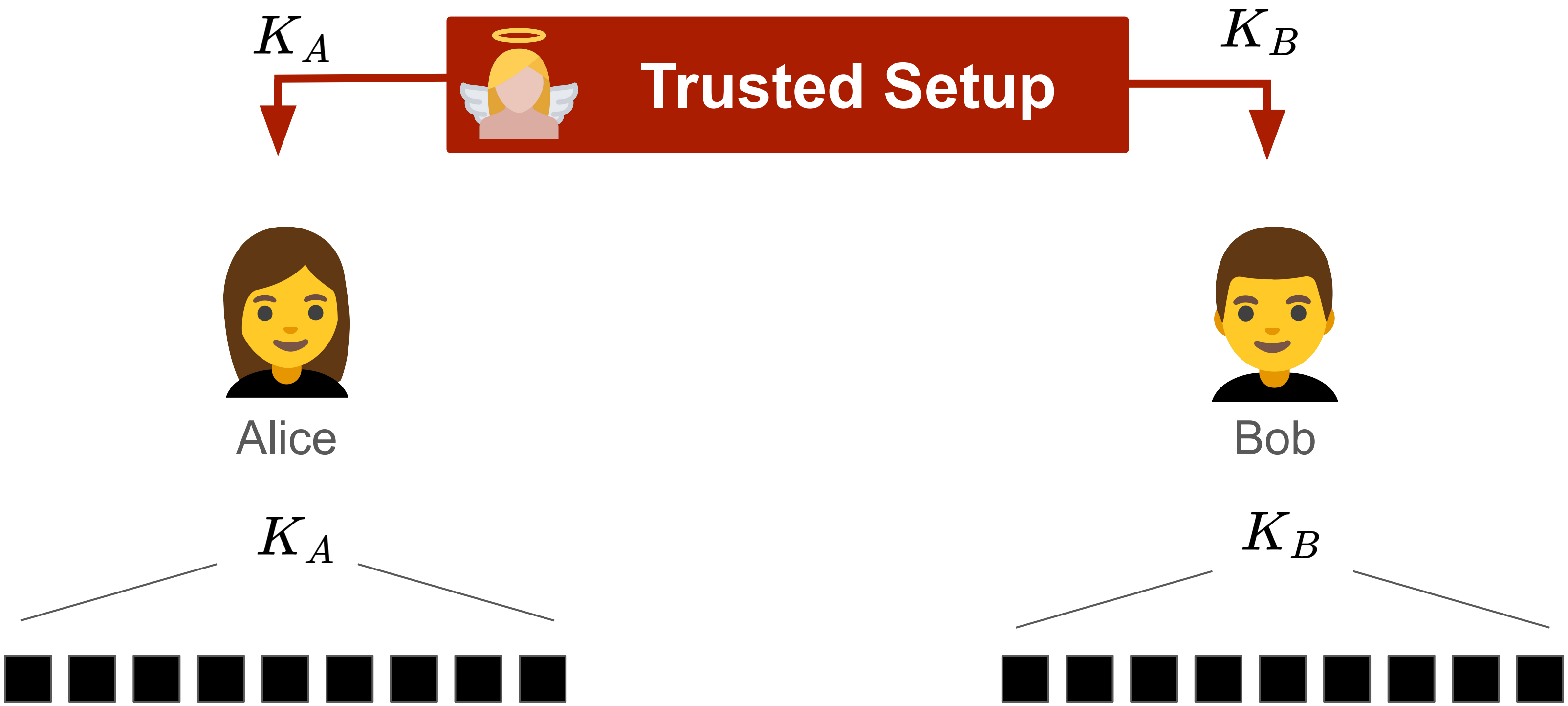
Distributed Point Functions [GI'14]



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]

Applications of Distributed Point Functions

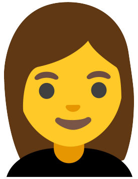
Private Information Retrieval and Search [GI'14, BGI'15, DPKY'20]

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]

Preprocessing in MPC

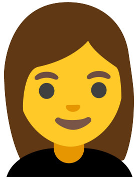

Alice


Bob

Preprocessing in MPC

Offline Phase

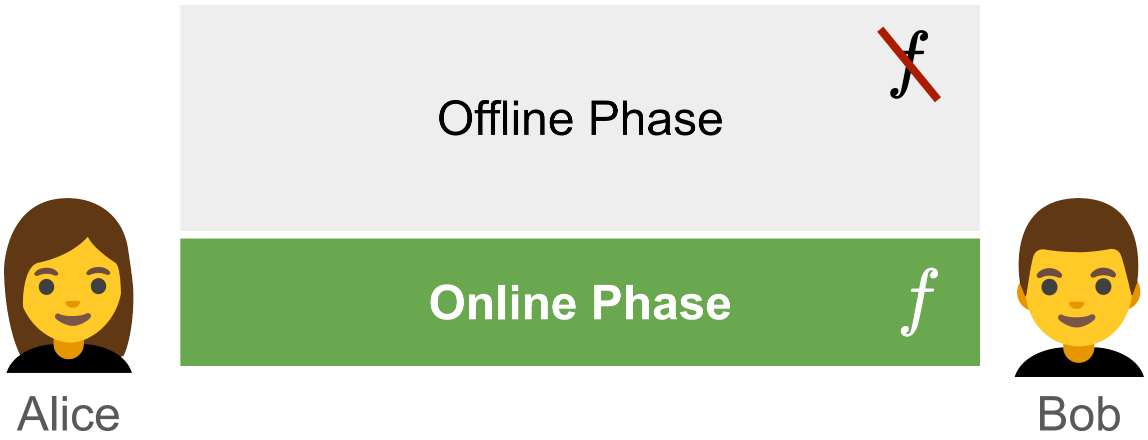
$\not{f}$



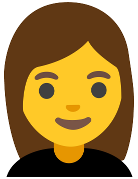
Alice

Bob

Preprocessing in MPC



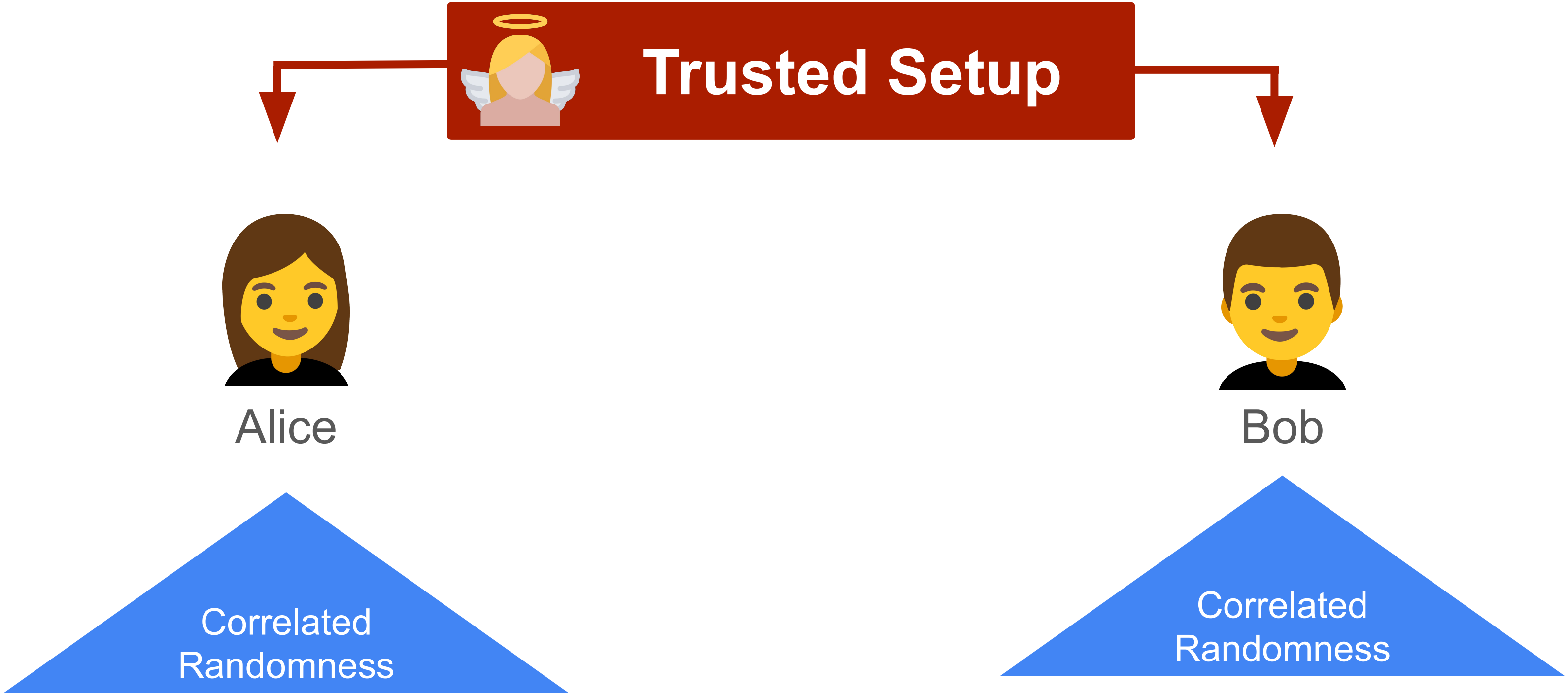
Preprocessing in MPC: “Offline” Phase



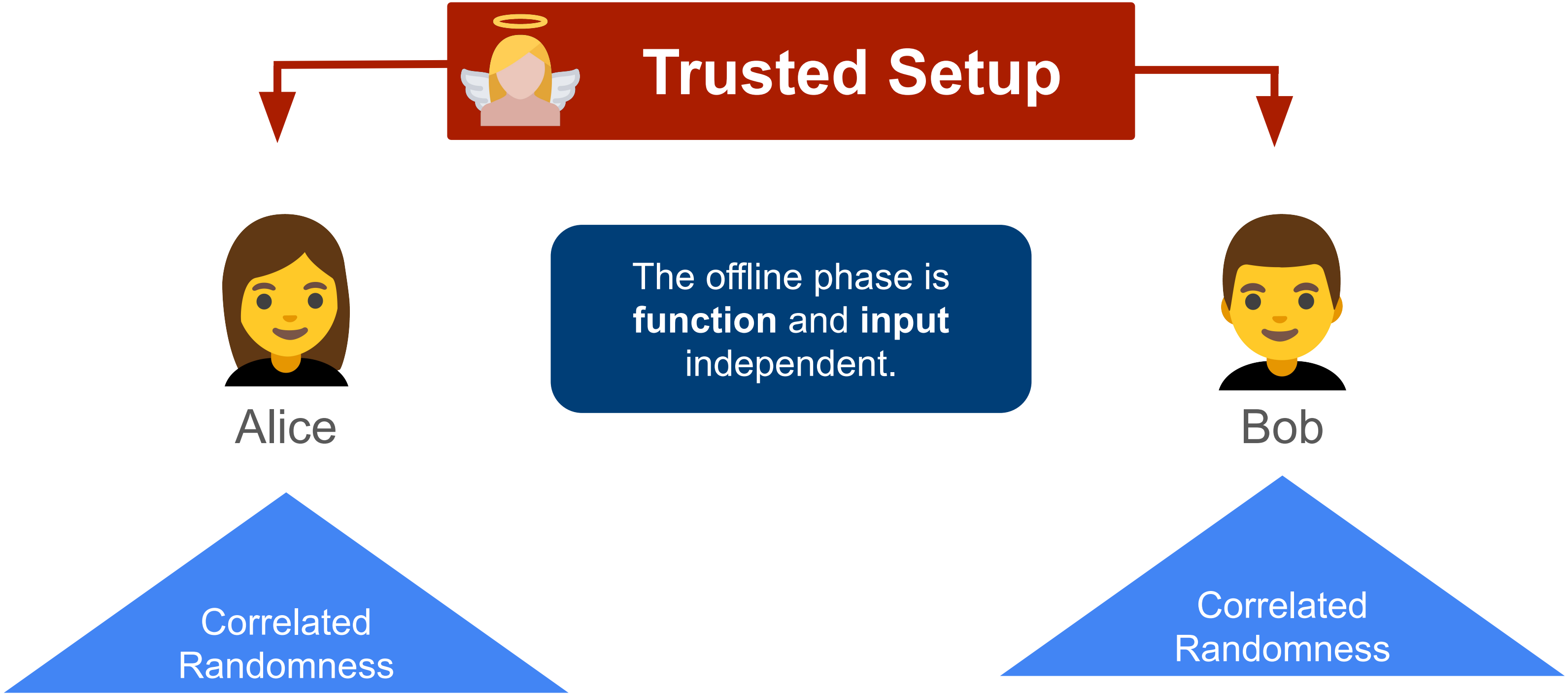
Alice

Bob

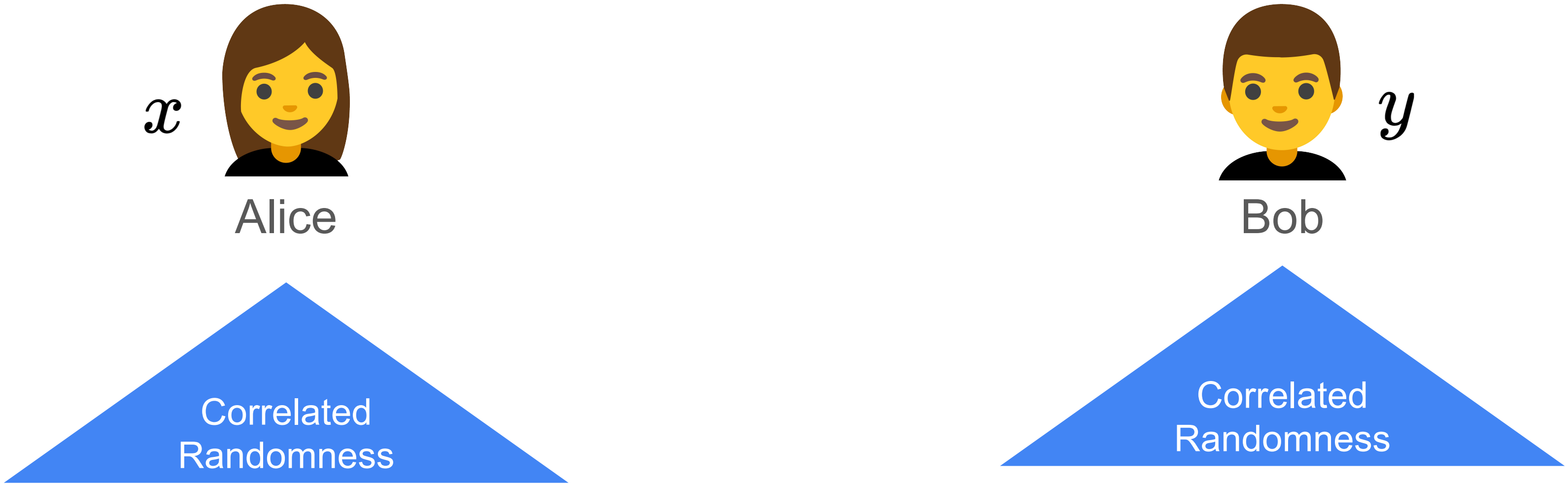
Preprocessing in MPC: “Offline” Phase



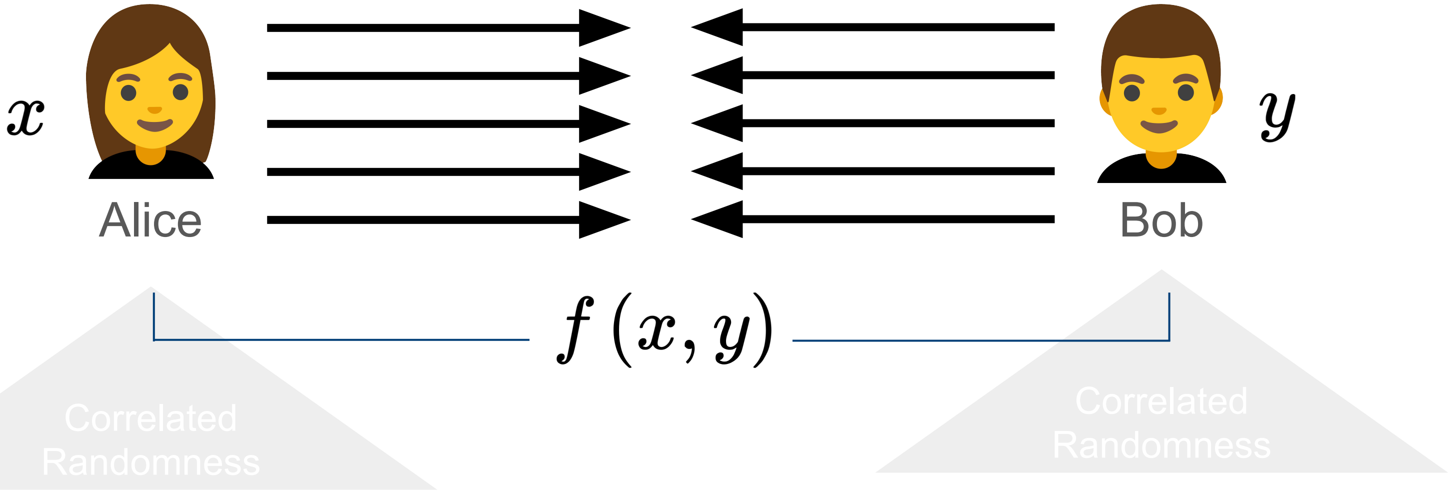
Preprocessing in MPC: “Offline” Phase



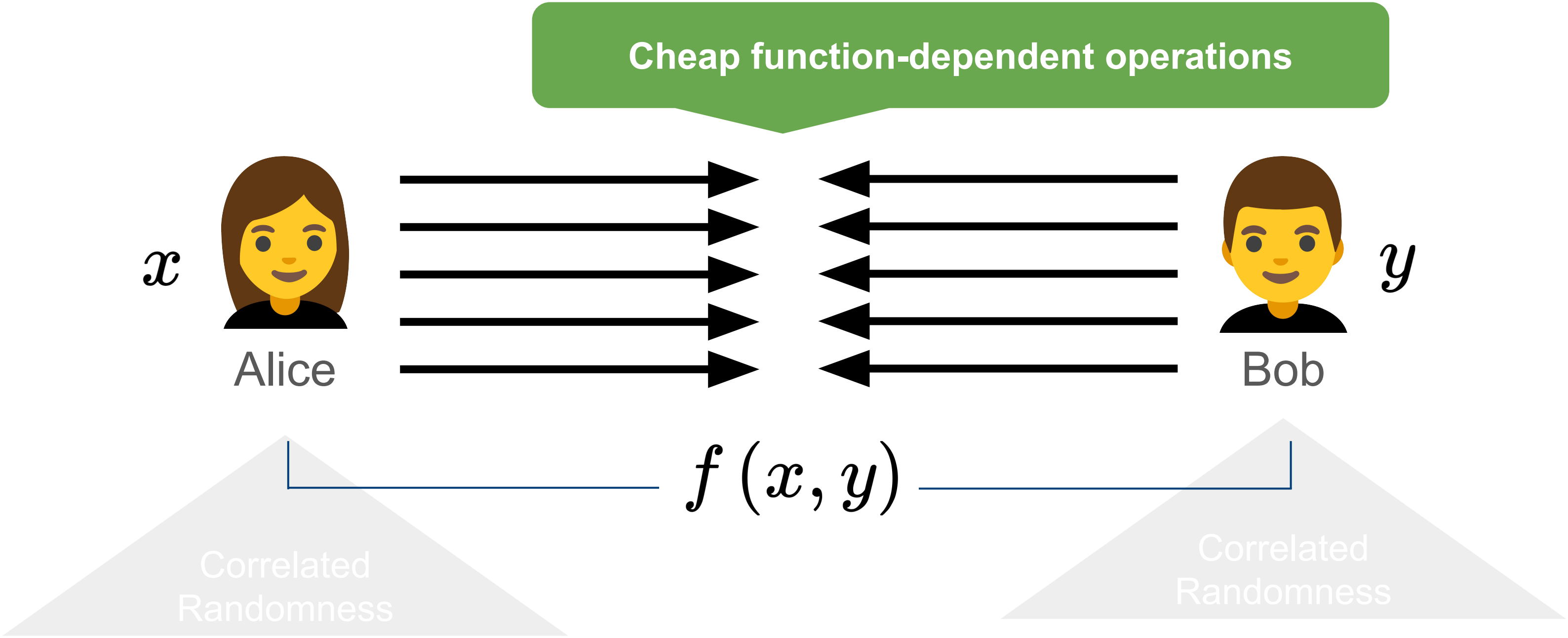
Preprocessing in MPC: “Online” Phase



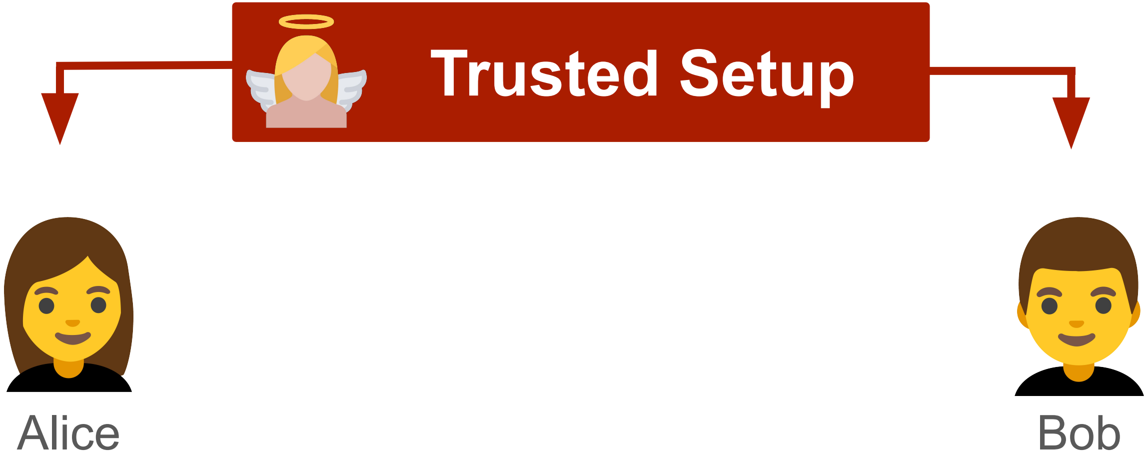
Preprocessing in MPC: “Online” Phase



Preprocessing in MPC: "Online" Phase



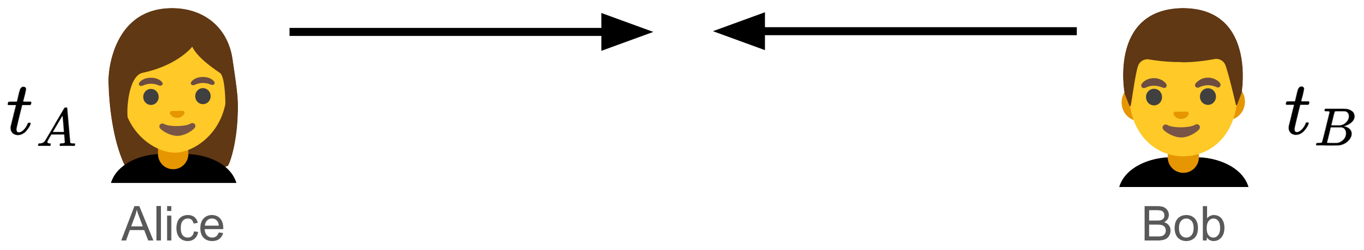
Preprocessing in MPC: “Offline” Phase



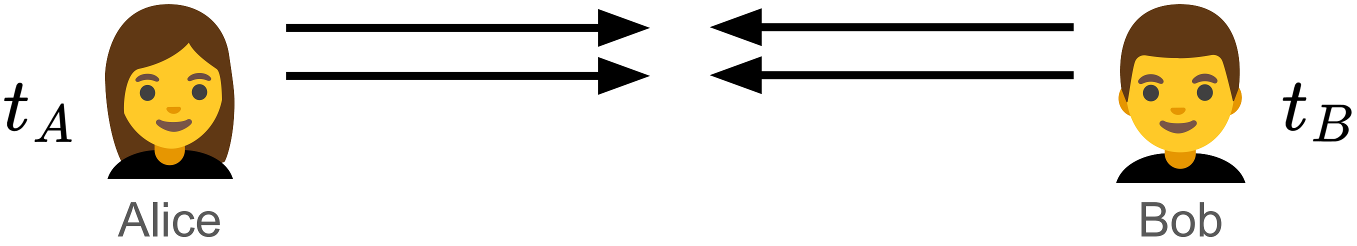
Preprocessing in MPC: “Offline” Phase

Preprocessing in MPC: “Offline” Phase

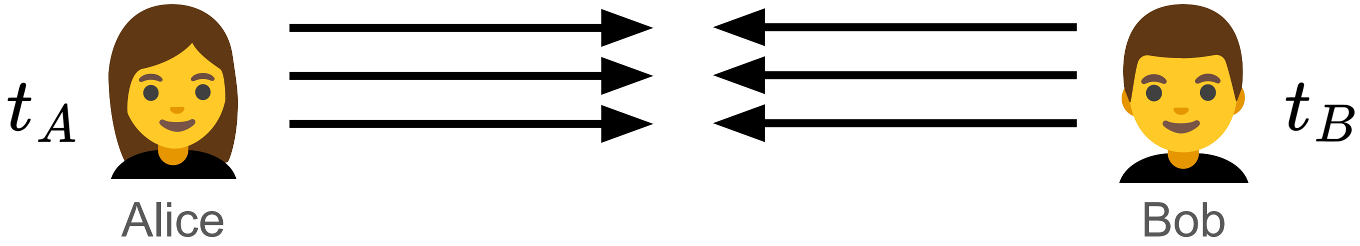
Removing the Trusted Setup



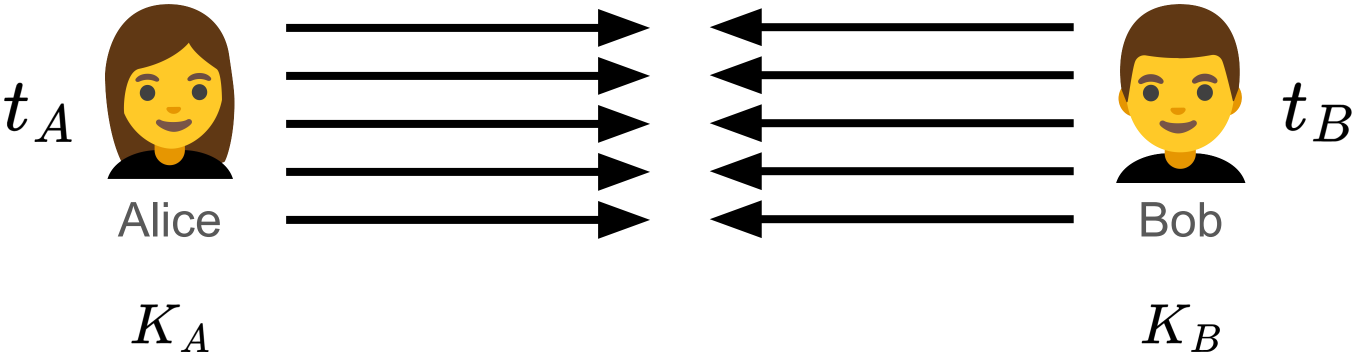
Removing the Trusted Setup



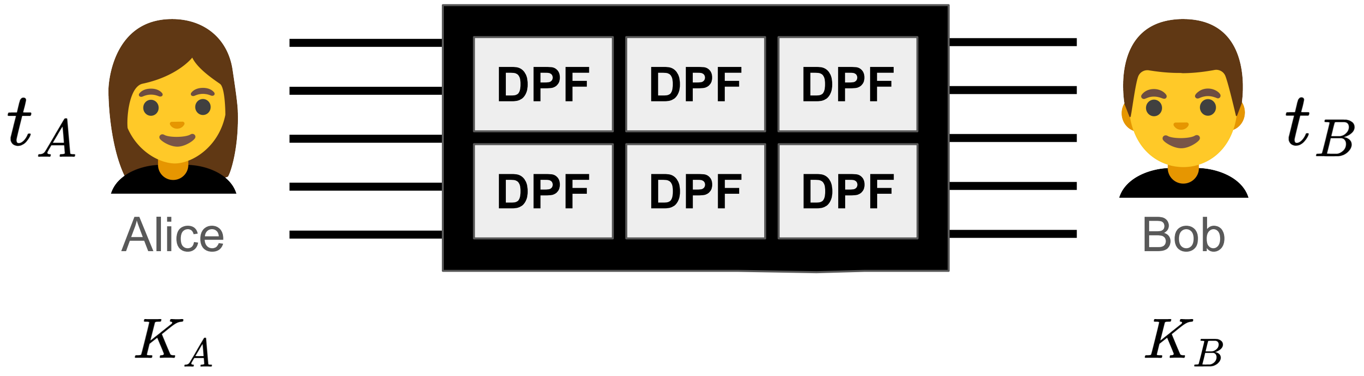
Removing the Trusted Setup



Removing the Trusted Setup

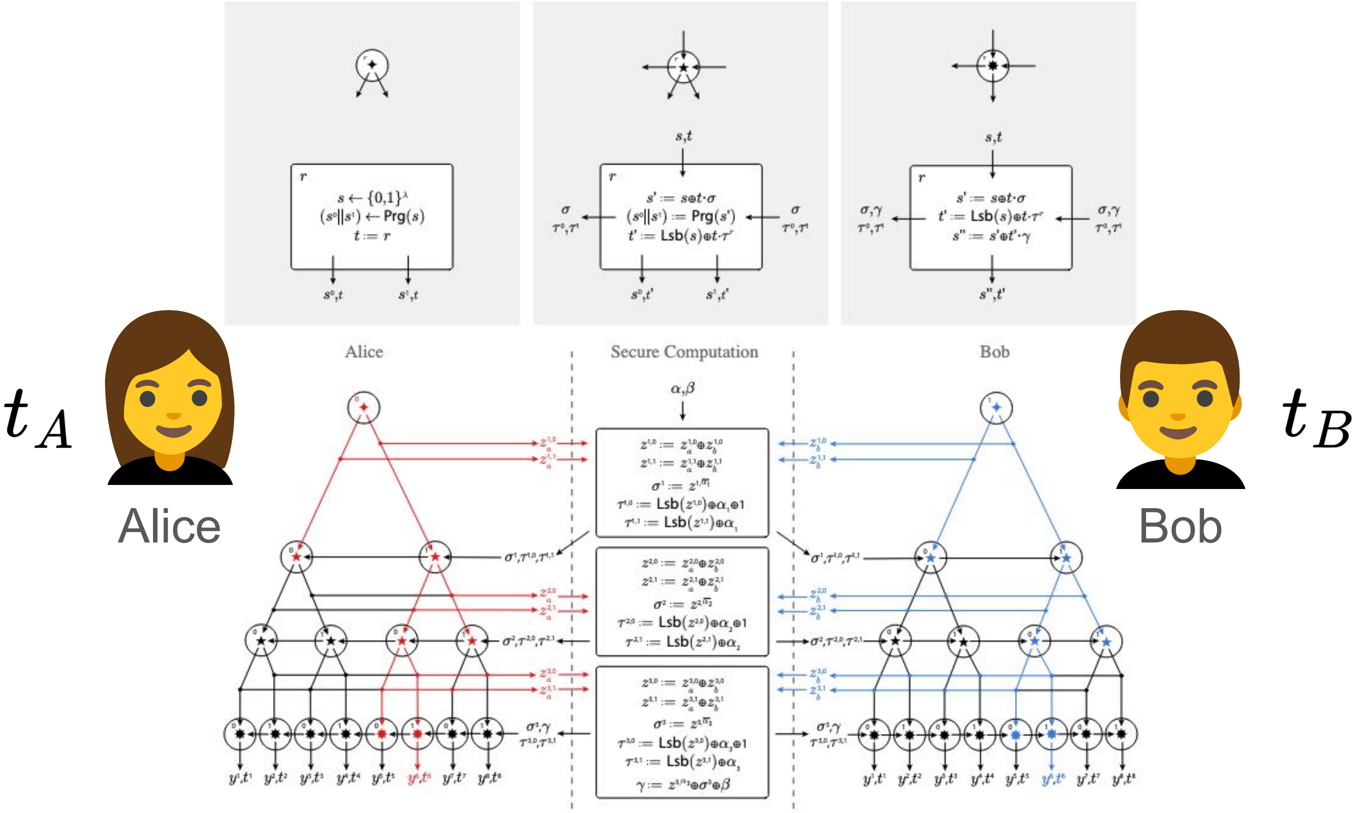


Removing the Trusted Setup



Removing the Trusted Setup

The Doerner-shelat Protocol

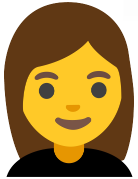


Removing the Trusted Setup

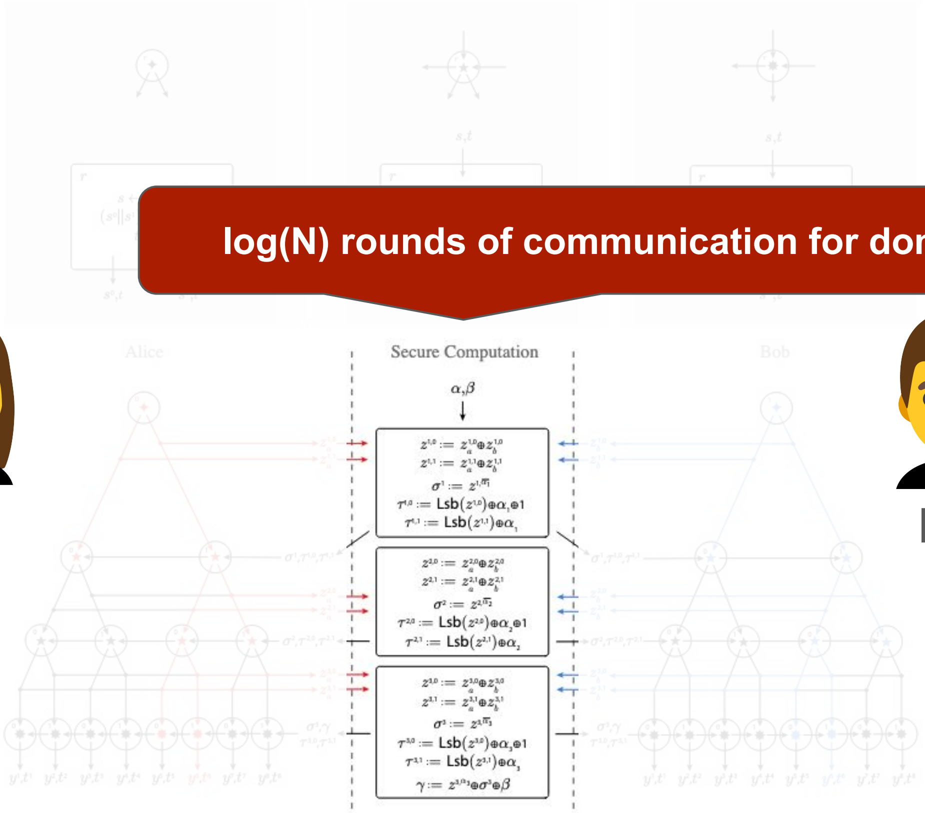
The Doerner-shelat Protocol

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

t_A



Alice



Bob

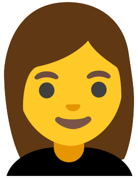
t_B

Can we remove interaction?

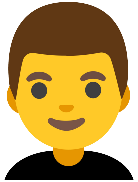
Inspiration

Diffie–Hellman Key Exchange

Diffie–Hellman Key Exchange [DH'76]

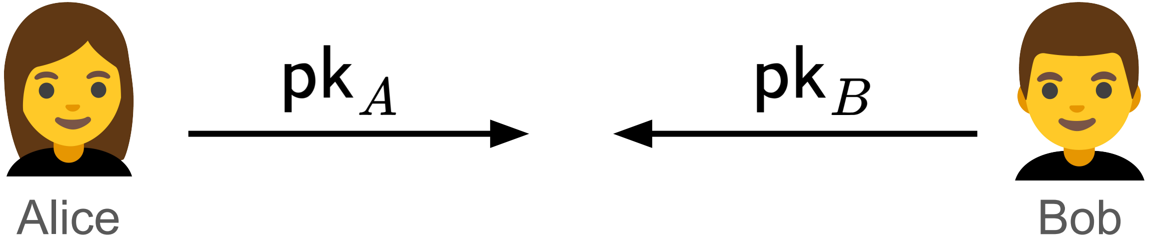


Alice



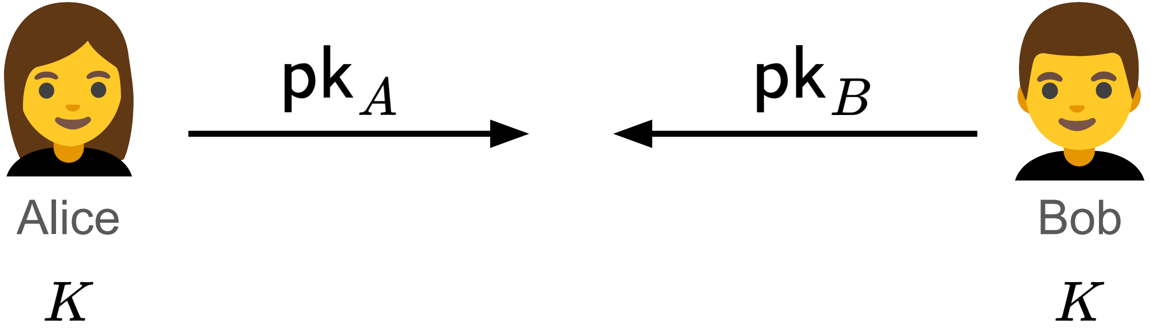
Bob

Diffie–Hellman Key Exchange [DH'76]

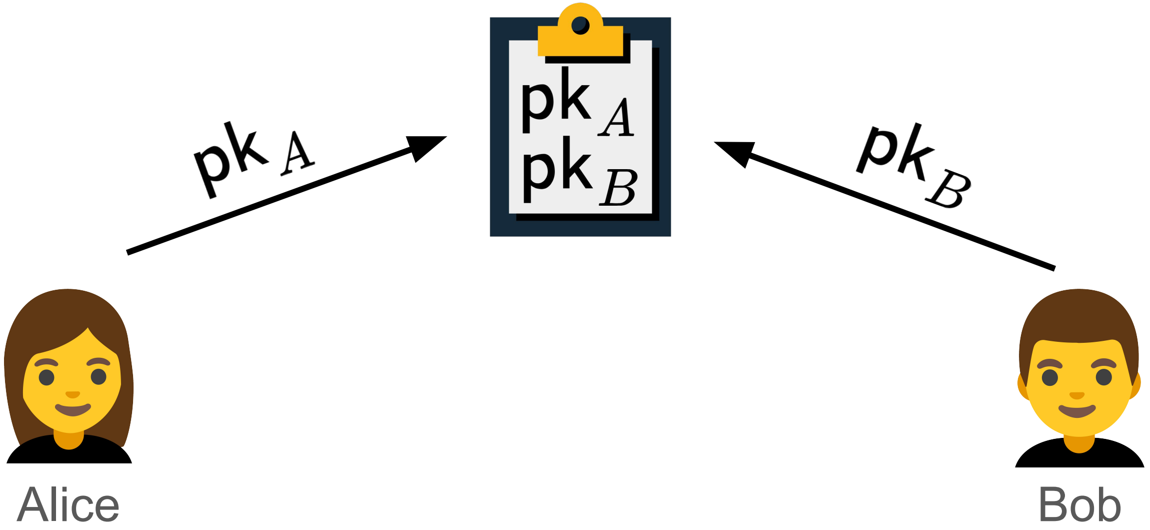


Diffie–Hellman Key Exchange [DH'76]

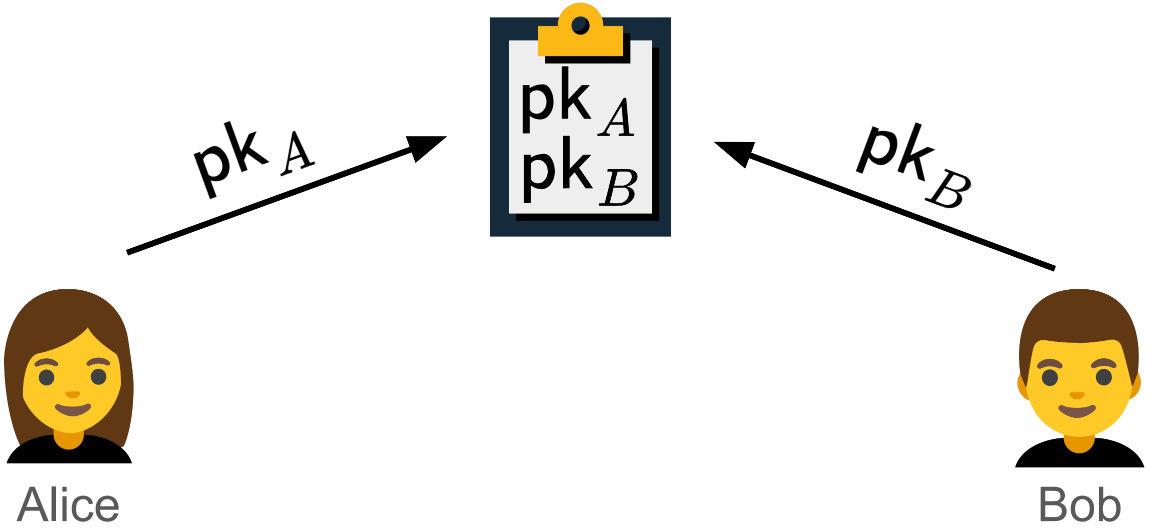
$$z_A + z_B = f(x, y)$$



Diffie–Hellman Key Exchange [DH'76]



Diffie–Hellman Key Exchange [DH'76]



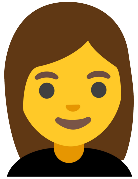
K K

“Diffie–Hellman” for DPF keys?

Non-Interactive

Distributed Point Functions

Distributed Point Functions

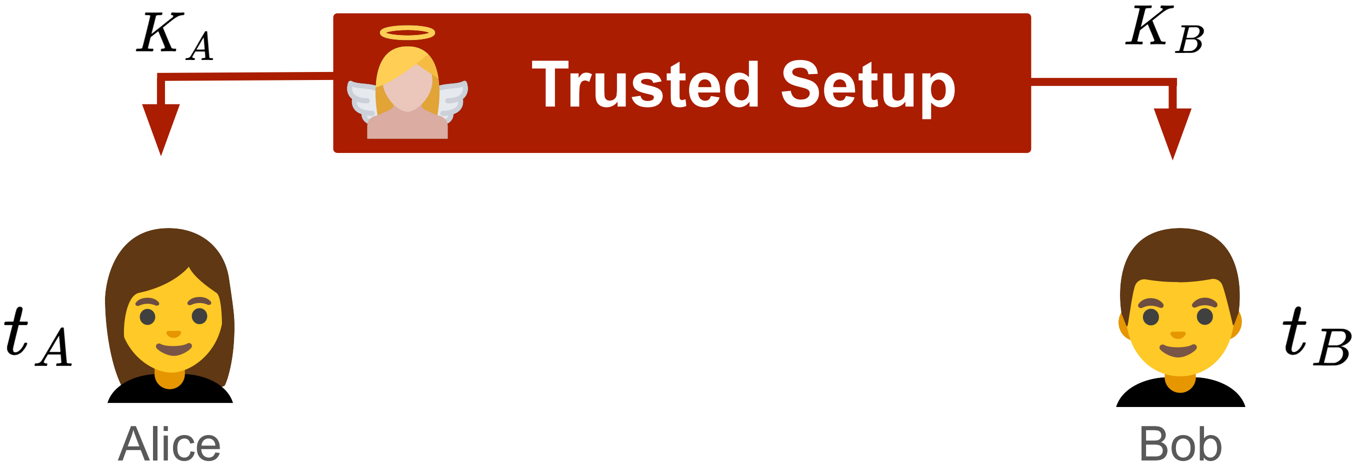
t_A 

Alice

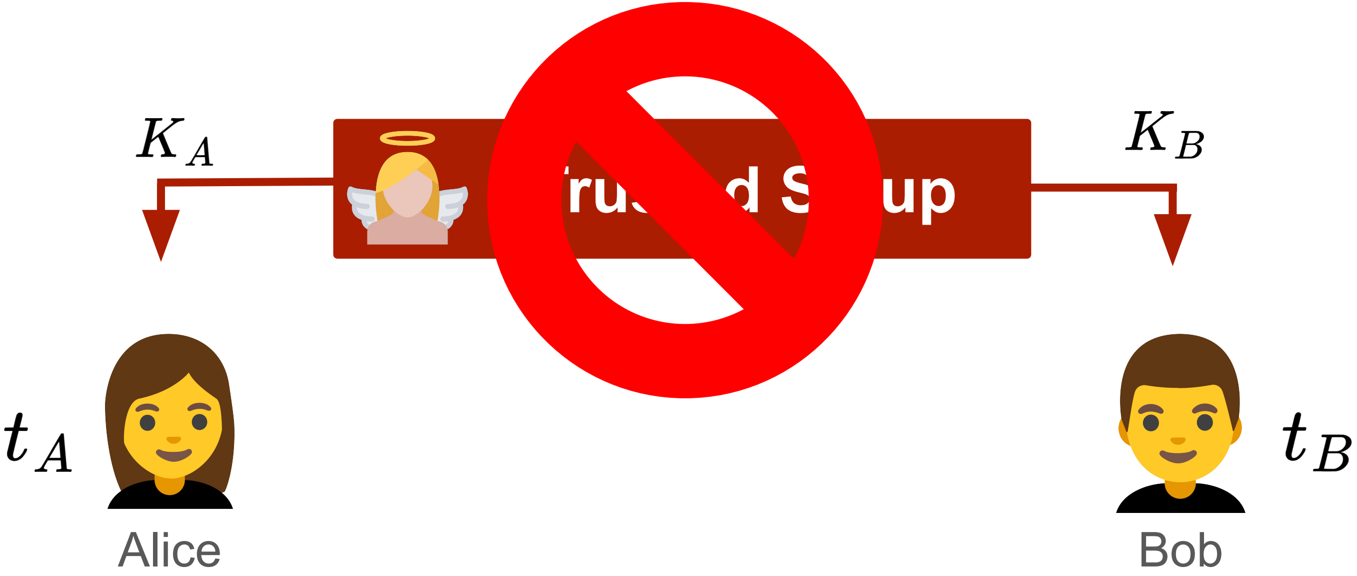
 t_B

Bob

Distributed Point Functions

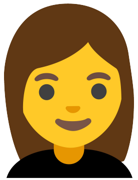


Distributed Point Functions

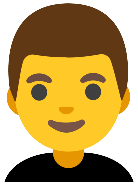


Non-Interactive Distributed Point Functions

No Trusted Setup

t_A 

Alice

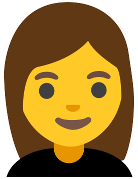
 t_B

Bob

Non-Interactive Distributed Point Functions

No Trusted Setup*

t_A



Alice

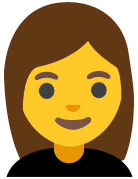
t_B

Bob

*We still allow for a common reference string

Non-Interactive Distributed Point Functions

No Trusted Setup

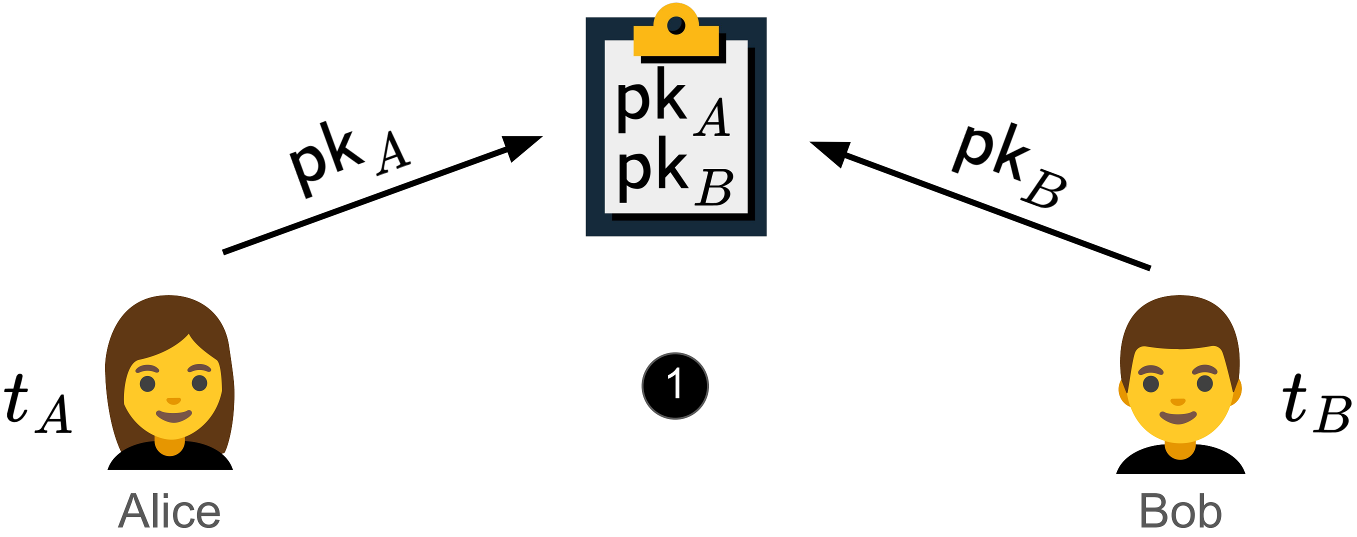
t_A  Alice

Bob  t_B

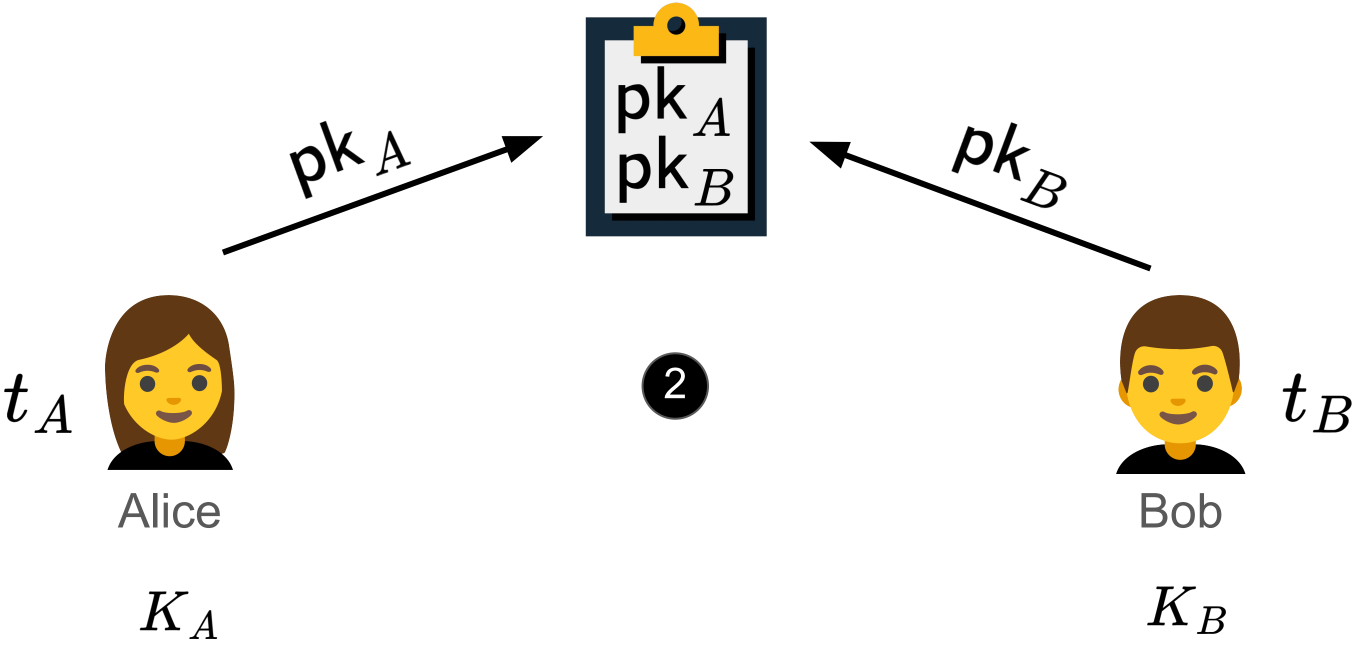
Goal

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

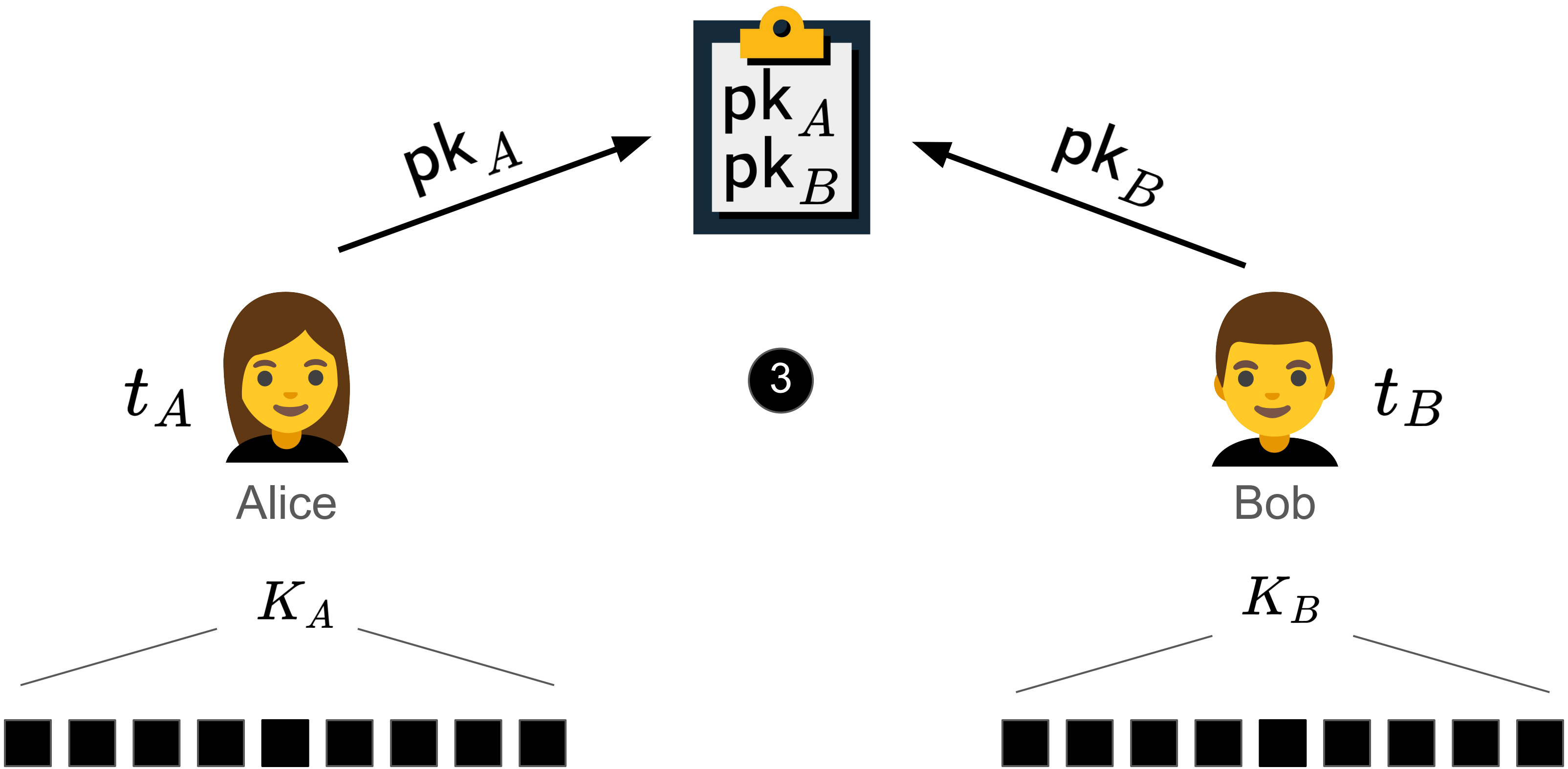
Non-Interactive Distributed Point Functions



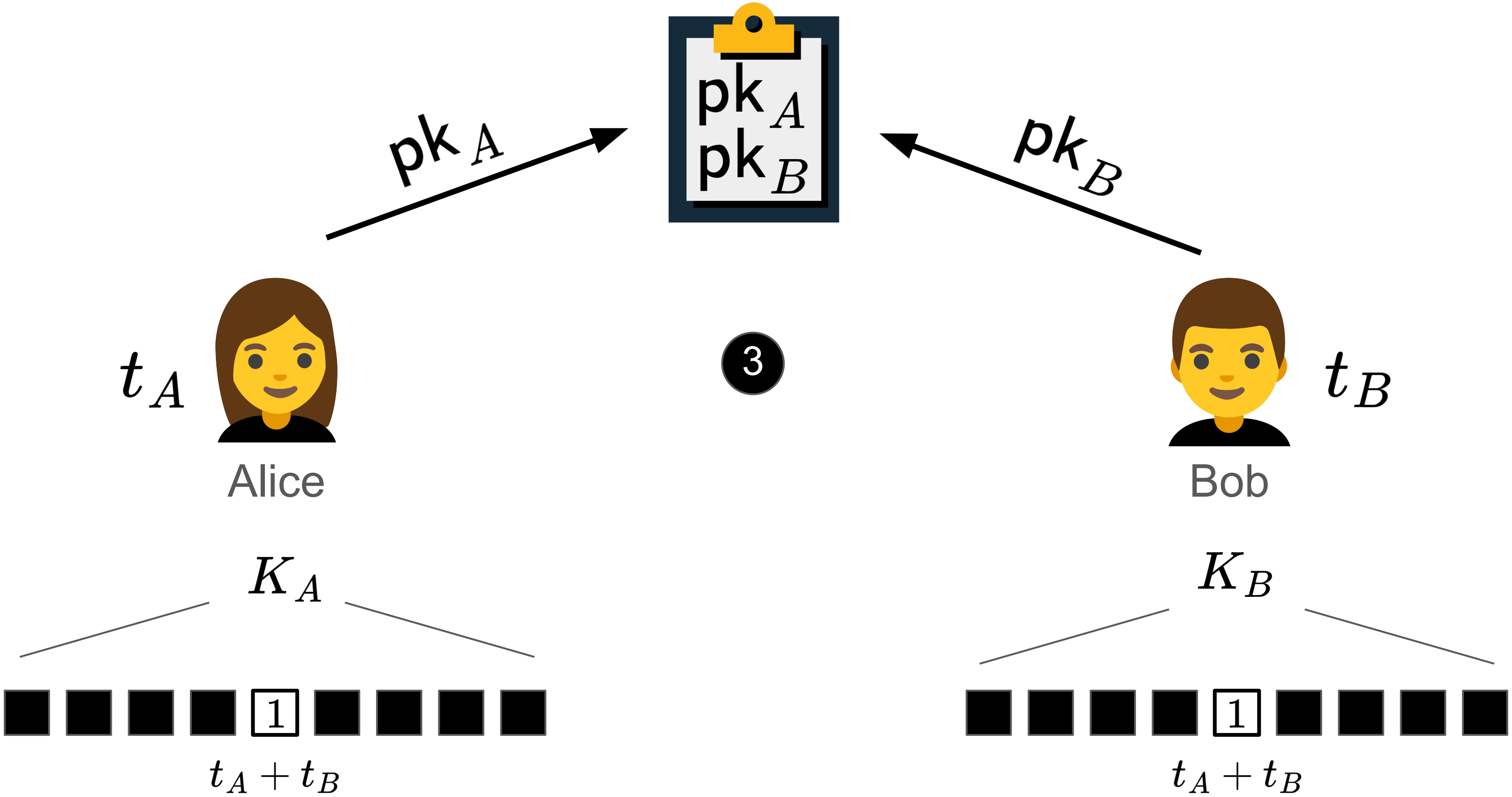
Non-Interactive Distributed Point Functions



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

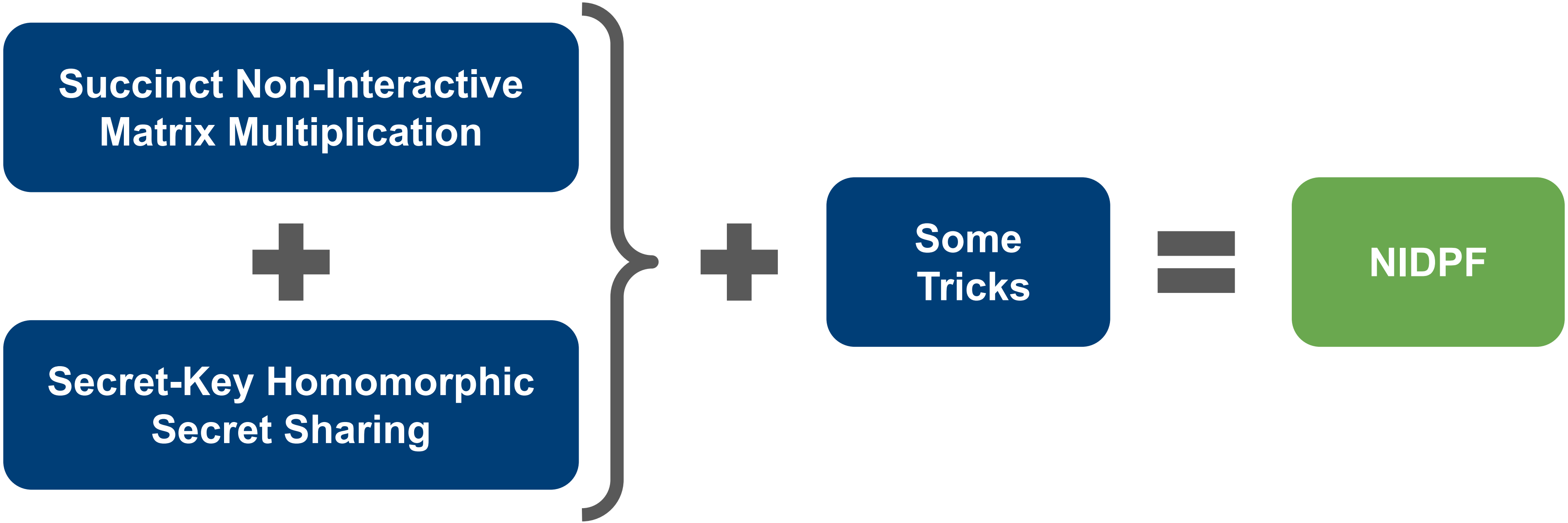
Succinct Non-Interactive Matrix Multiplication

+

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

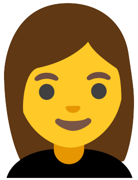
=

NIDPF

Tool: Succinct Non-Interactive Matrix Multiplication

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

$\mathbf{A}$

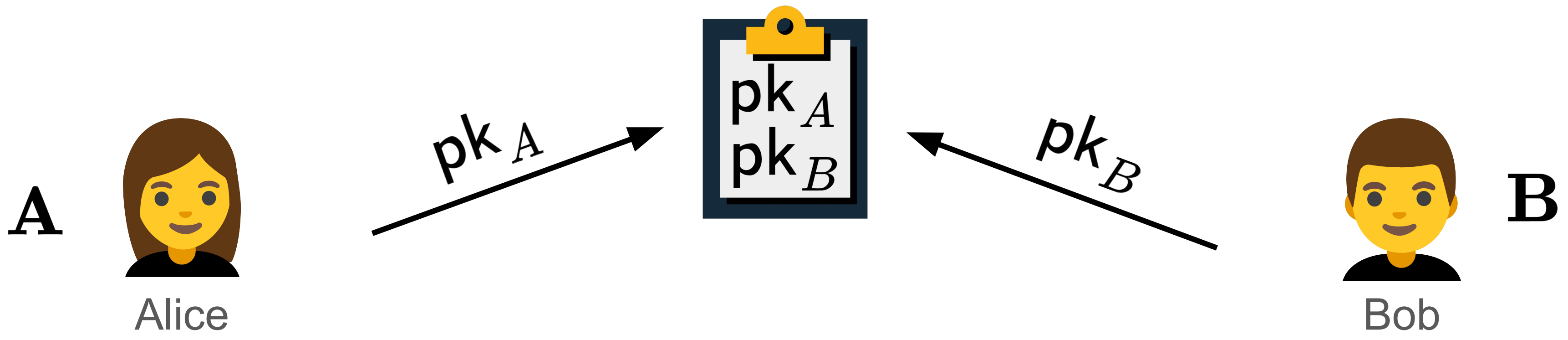


Alice

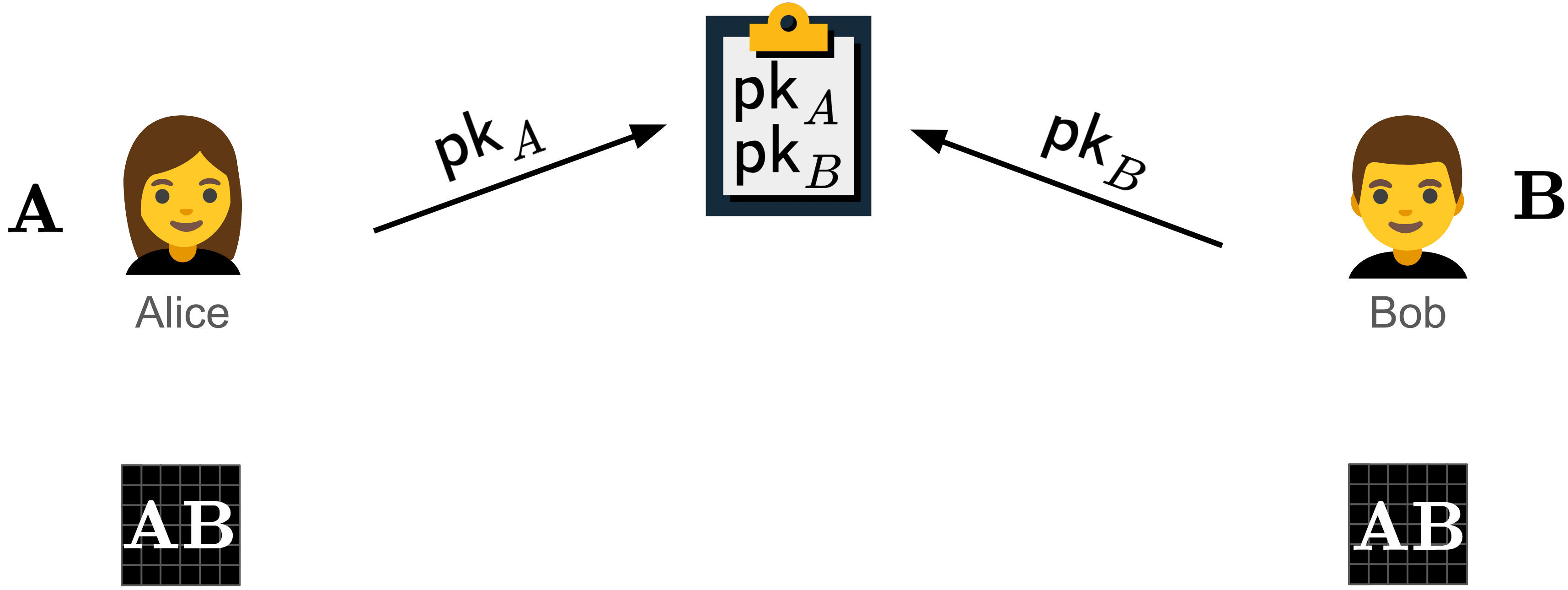
$\mathbf{B}$

Bob

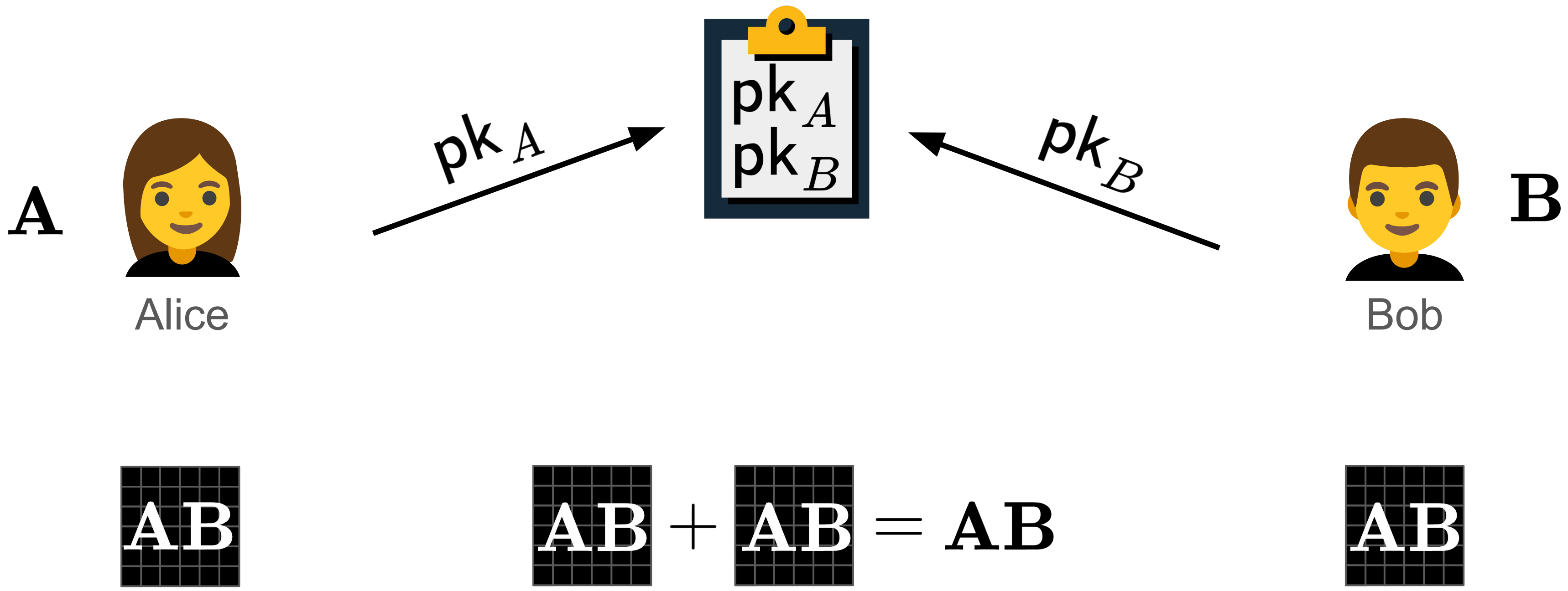
Tool: Succinct Non-Interactive Matrix Multiplication



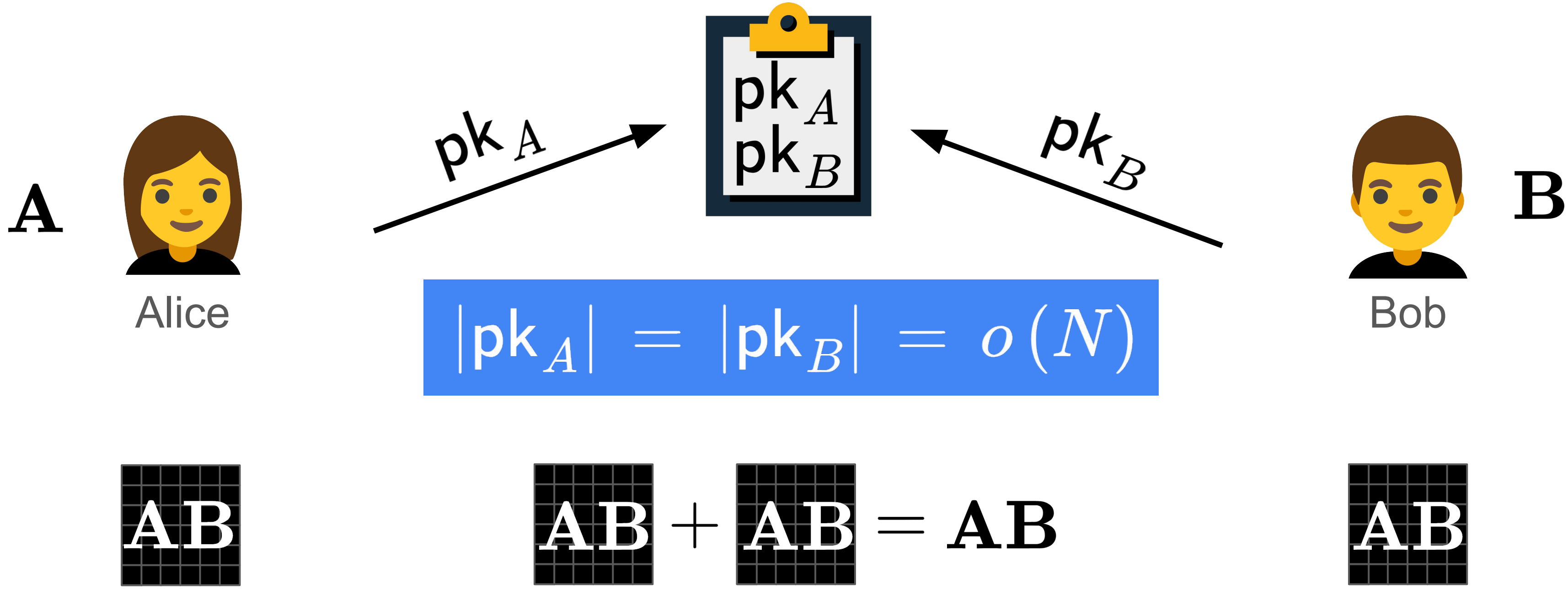
Tool: Succinct Non-Interactive Matrix Multiplication



Tool: Succinct Non-Interactive Matrix Multiplication



Tool: Succinct Non-Interactive Matrix Multiplication



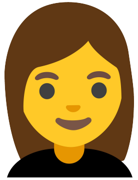
Informal Theorem (Implicit in [ARS’24])

There exists a two-party **succinct, non-interactive matrix multiplication protocol*** under any of the following assumptions: DDH, DCR, QR, or LWE.

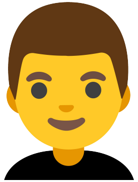
*For suitable matrix dimensions and suitable finite fields.

Using succinct matrix multiplication to realize a NIDPF

Step 1: Reinterpret Indices via CRT

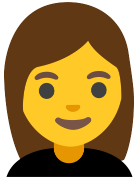
t_A 

Alice

 t_B

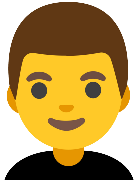
Bob

Step 1: Reinterpret Indices via CRT

t_A 

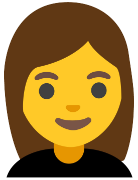
Alice

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

 t_B

Bob

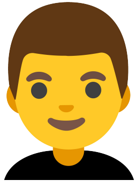
Step 1: Reinterpret Indices via CRT

t_A 

Alice

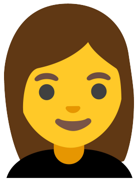
$$t_A \Rightarrow t_A = i_A \cdot \ell + j_A$$

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

 t_B

Bob

Step 1: Reinterpret Indices via CRT

t_A 

Alice

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

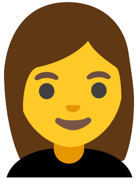
$$t_B \Rightarrow t_B = i_B \cdot \ell + j_B$$

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

 t_B

Bob

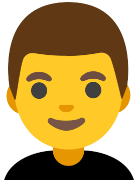
Step 2: Define matrices representing secret indices

t_A 

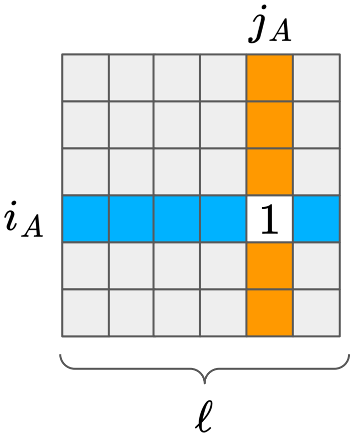
Alice

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

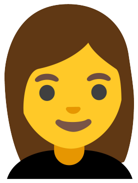
$$t_B \Rightarrow t_B = i_B \cdot \ell + j_B$$

 t_B

Bob

$\mathbf{A} =$ 

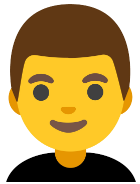
Step 2: Define matrices representing secret indices

t_A 

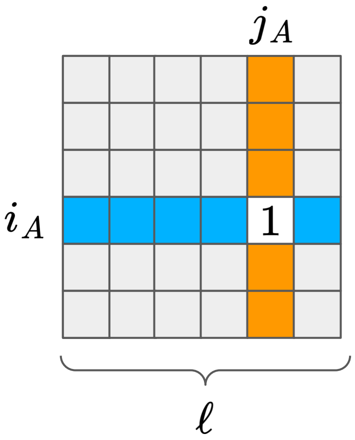
Alice

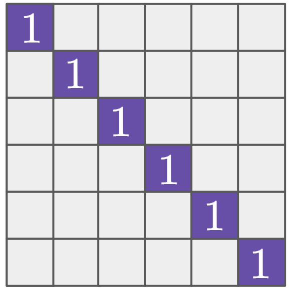
$$t_A \Rightarrow t_A = 3 \cdot \ell + 4$$

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

 t_B

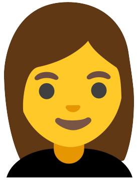
Bob

$\mathbf{A} =$ 

$\mathbf{B} =$ 

Cyclic Shift Matrix

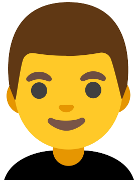
Step 2: Define matrices representing secret indices

t_A 

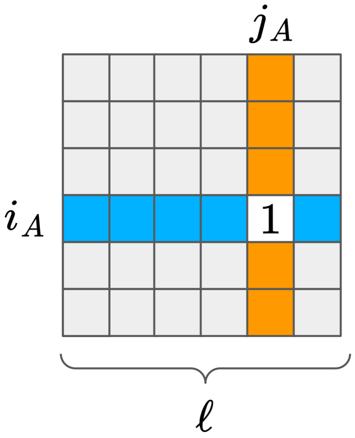
Alice

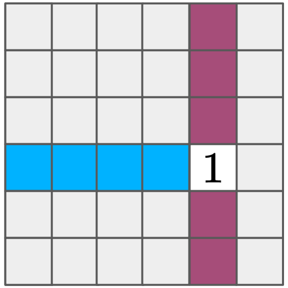
$$t_A \Rightarrow t_A = 3 \cdot \ell + 4$$

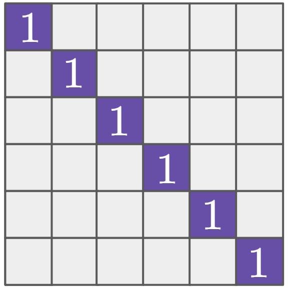
$$t_B \Rightarrow t_B = i_B \cdot \ell + 0$$

 t_B

Bob

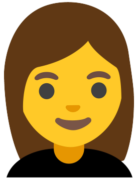
$\mathbf{A} =$ 

$\mathbf{AB} =$ 

$\mathbf{B} =$ 

Cyclic Shift Matrix

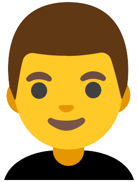
Step 2: Define matrices representing secret indices

t_A 

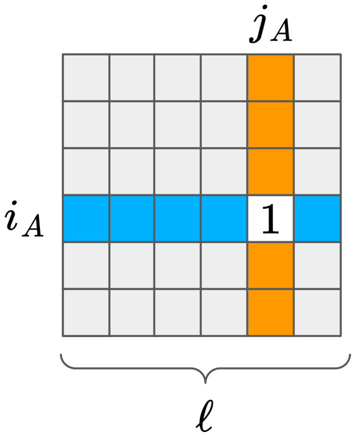
Alice

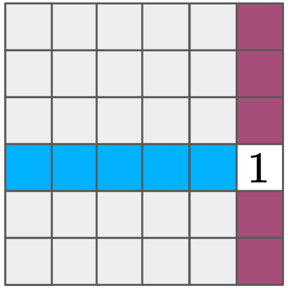
$$t_A \Rightarrow t_A = 3 \cdot \ell + 4$$

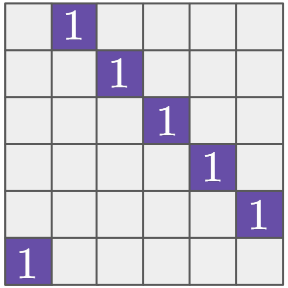
$$t_B \Rightarrow t_B = i_B \cdot \ell + 1$$

 t_B

Bob

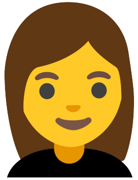
$\mathbf{A} =$ 

$\mathbf{AB} =$ 

$\mathbf{B} =$ 

Cyclic Shift Matrix

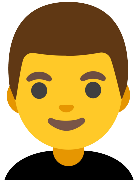
Step 2: Define matrices representing secret indices

t_A 

Alice

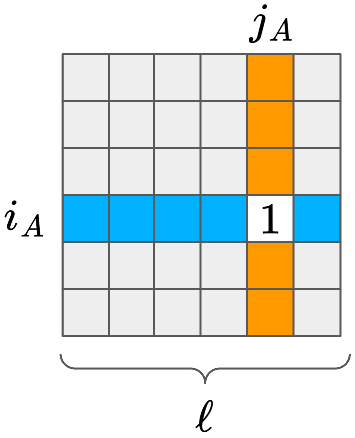
$$t_A \rightarrow t_A = 3 \cdot \ell + 4$$

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

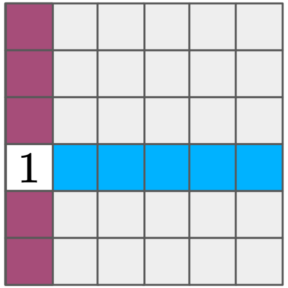
 t_B

Bob

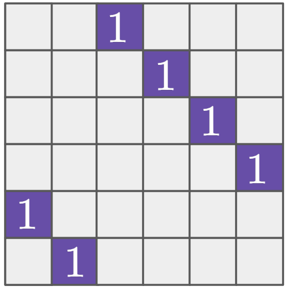
$\mathbf{A} =$



$\mathbf{AB} =$

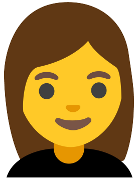


$\mathbf{B} =$



Cyclic Shift Matrix

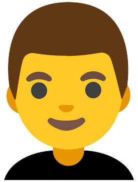
Step 2: Define matrices representing secret indices

t_A 

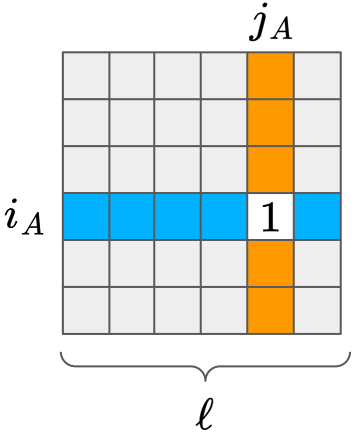
Alice

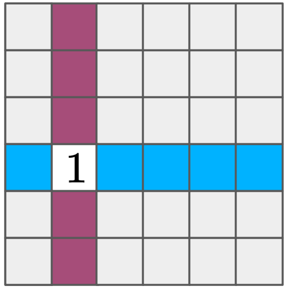
$$t_A \Rightarrow t_A = 3 \cdot \ell + 4$$

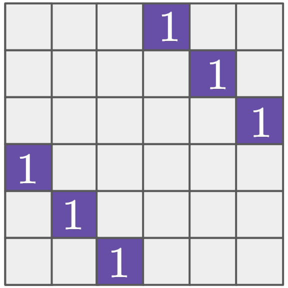
$$t_B \Rightarrow t_B = i_B \cdot \ell + 3$$

 t_B

Bob

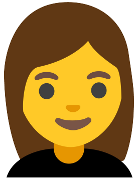
$\mathbf{A} =$ 

$\mathbf{AB} =$ 

$\mathbf{B} =$ 

Cyclic Shift Matrix

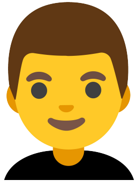
Step 2: Define matrices representing secret indices

t_A 

Alice

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

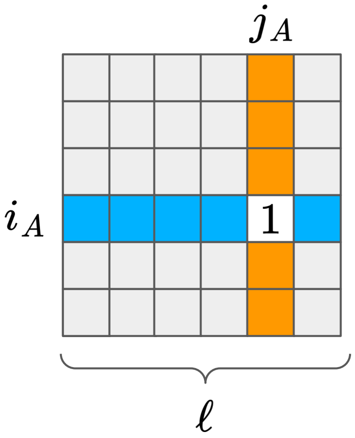
$t_B \Rightarrow t_B = i_B \cdot \ell + 3$

 t_B

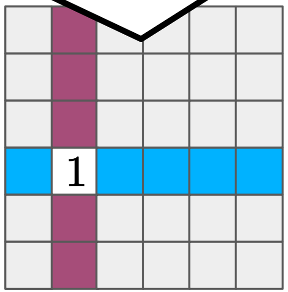
Bob

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

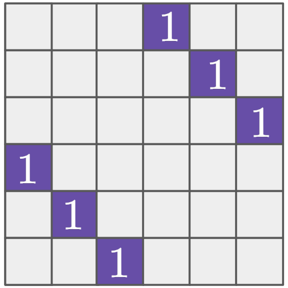
$\mathbf{A} =$



$\mathbf{AB} =$

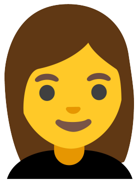


$\mathbf{B} =$



Cyclic Shift Matrix

Step 2: Define matrices representing secret indices

t_A 

Alice

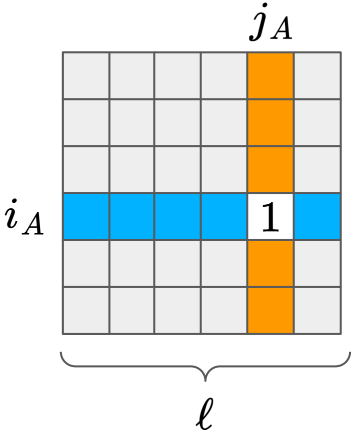
$$t_A \Rightarrow t_A = 3 \cdot \ell + 4$$

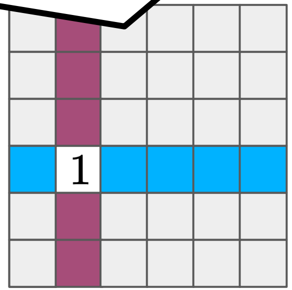
$$t_B \Rightarrow t_B = i_B \cdot \ell + 3$$

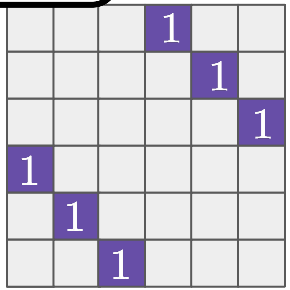
 t_B

Bob

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

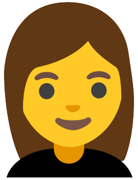
$\mathbf{A} =$ 

$\mathbf{AB} =$ 

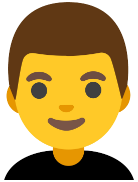
$\mathbf{B} =$ 

Cyclic Shift Matrix

Step 2: Define matrices representing secret indices

t_A 

Alice

 t_B

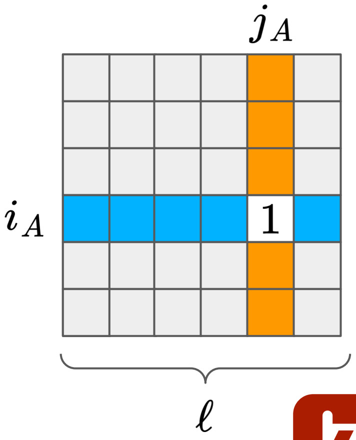
Bob

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

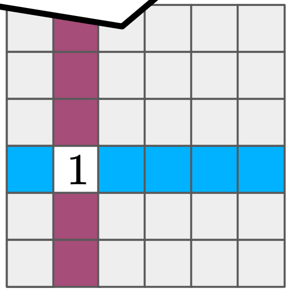
$$t_B \Rightarrow t_B = i_B \cdot \ell + 3$$

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

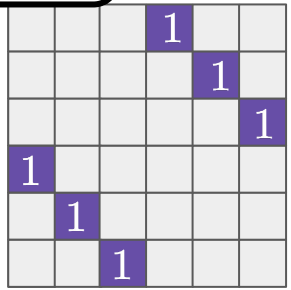
$\mathbf{A} =$



$\mathbf{AB} =$



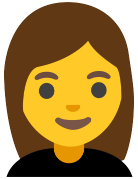
$\mathbf{B} =$



Problem: Matrix multiplication just shifts the columns

Stepping Back

Non-Interactive Distributed "Point Function"

t_A 

Alice

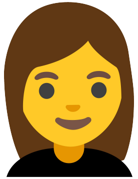
 t_B

Bob

Non-Interactive Distributed "Point Function"

$t_A \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$

$t_B \in \{0, 1, 2\}$

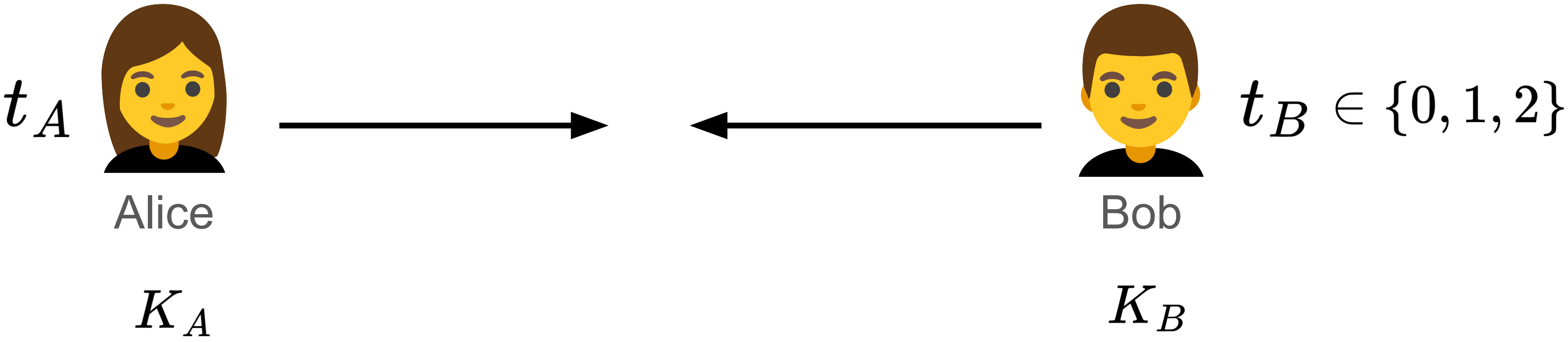
t_A 

Alice

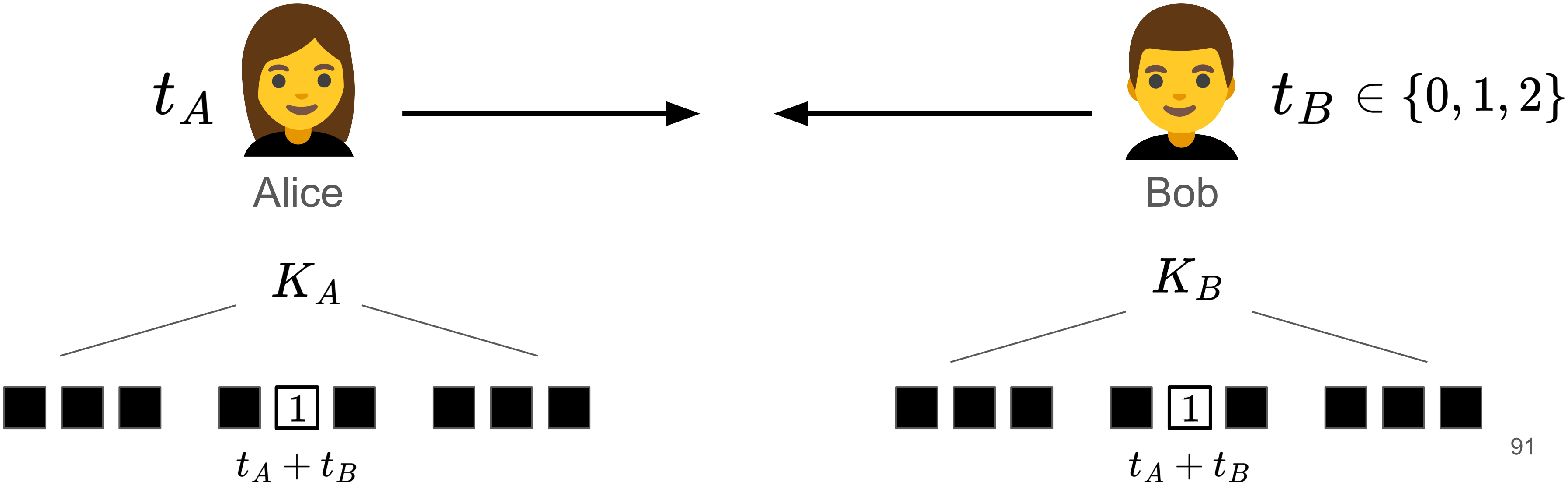
 t_B

Bob

Non-Interactive Distributed “Point Function”

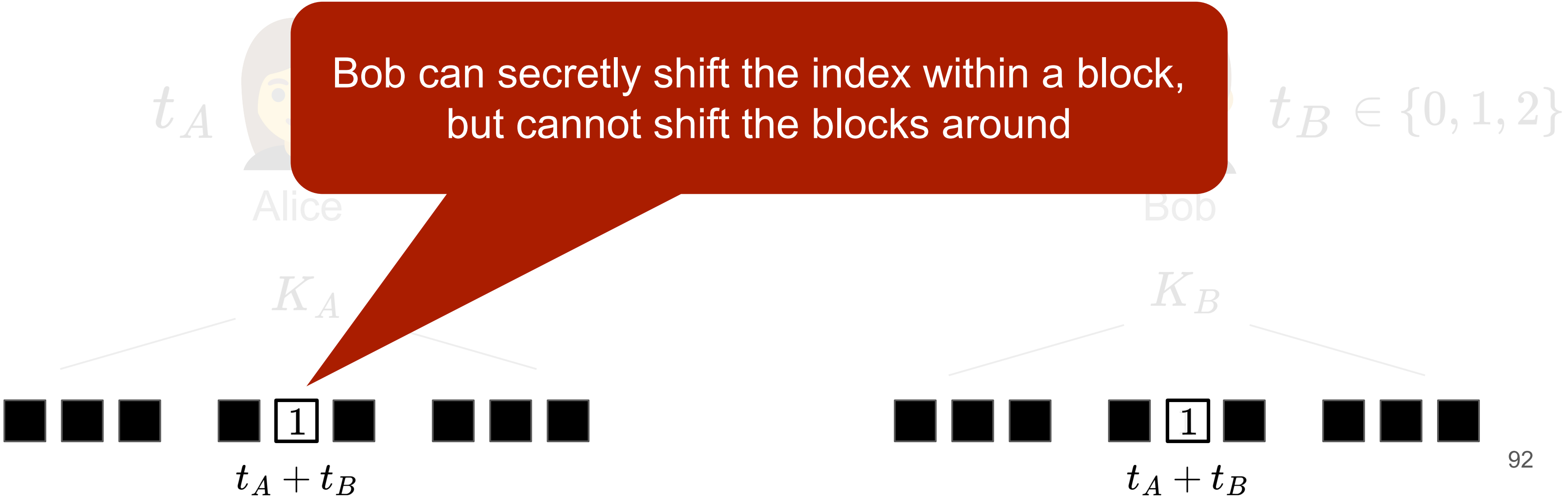


Non-Interactive Distributed "Point Function"

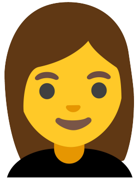


Non-Interactive Distributed "Point Function"

Bob can secretly shift the index within a block, but cannot shift the blocks around



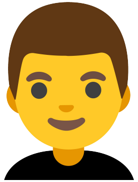
Examining the problem via an example:

t_A 

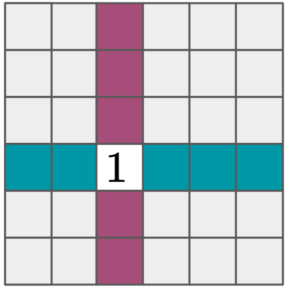
Alice

$$t_A \Rightarrow t_A = 3 \cdot \ell + 5$$

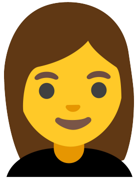
$$t_B \Rightarrow t_B = 0 \cdot \ell + 3$$

 t_B

Bob



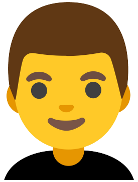
Examining the problem via an example:

t_A 

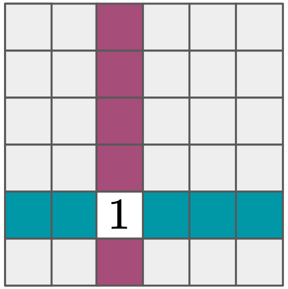
Alice

$$t_A \Rightarrow t_A = 3 \cdot \ell + 5$$

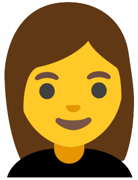
$$t_B \Rightarrow t_B = 1 \cdot \ell + 3$$

 t_B

Bob



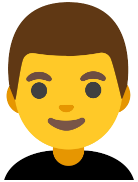
Examining the problem via an example:

t_A 

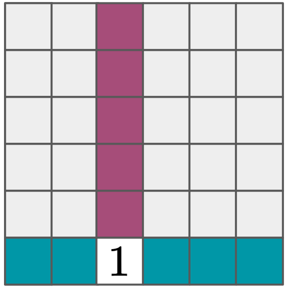
Alice

$$t_A \rightarrow t_A = 3 \cdot \ell + 5$$

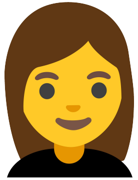
$$t_B \rightarrow t_B = 2 \cdot \ell + 3$$

 t_B

Bob

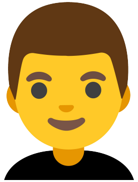


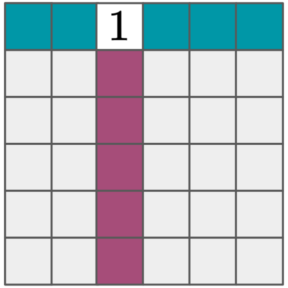
Examining the problem via an example:

t_A 
Alice

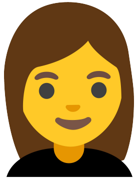
$$t_A \Rightarrow t_A = 3 \cdot \ell + 5$$

$$t_B \Rightarrow t_B = 3 \cdot \ell + 3$$

 t_B
Bob



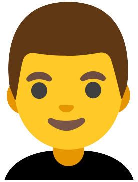
Examining the problem via an example:

t_A 

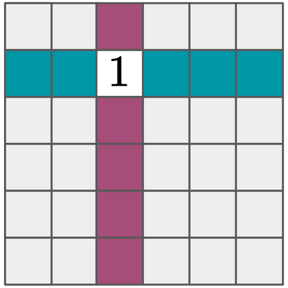
Alice

$$t_A \rightarrow t_A = 3 \cdot \ell + 5$$

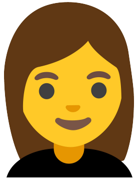
$$t_B \rightarrow t_B = 4 \cdot \ell + 3$$

 t_B

Bob



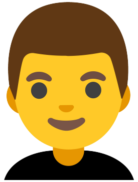
Examining the problem via an example:

t_A 

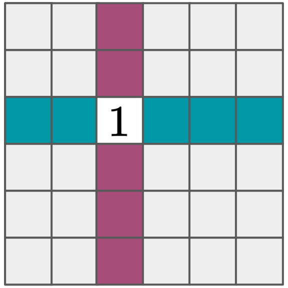
Alice

$$t_A \rightarrow t_A = 3 \cdot \ell + 5$$

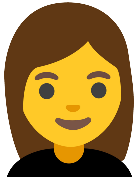
$$t_B \rightarrow t_B = 5 \cdot \ell + 3$$

 t_B

Bob



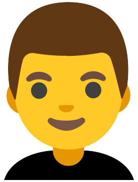
Examining the problem via an example:

t_A 

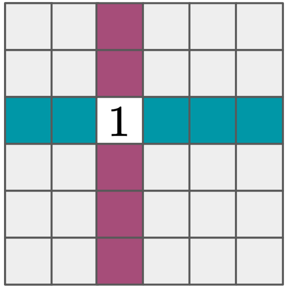
Alice

$$t_A \Rightarrow t_A = 3 \cdot \ell + 5$$

$$t_B \Rightarrow t_B = 5 \cdot \ell + 3$$

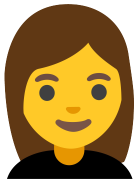
 t_B

Bob



Problem: Need to multiply again with another shift matrix

Examining the problem via an example:

t_A 

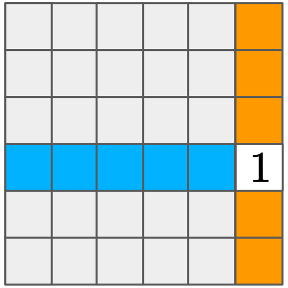
Alice

$$t_A \Rightarrow t_A = 3 \cdot \ell + 5$$

$$t_B \Rightarrow t_B = 5 \cdot \ell + 3$$

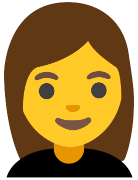
 t_B

Bob



Problem: Need to multiply again with another shift matrix

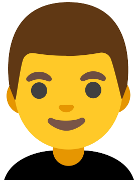
Examining the problem via an example:

t_A 

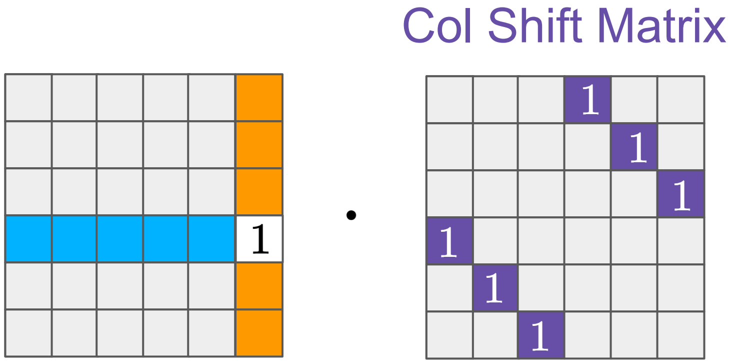
Alice

$$t_A \Rightarrow t_A = 3 \cdot \ell + 5$$

$$t_B \Rightarrow t_B = 5 \cdot \ell + 3$$

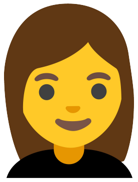
 t_B

Bob



Problem: Need to multiply again with another shift matrix

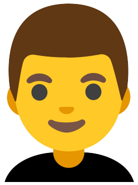
Examining the problem via an example:

t_A 

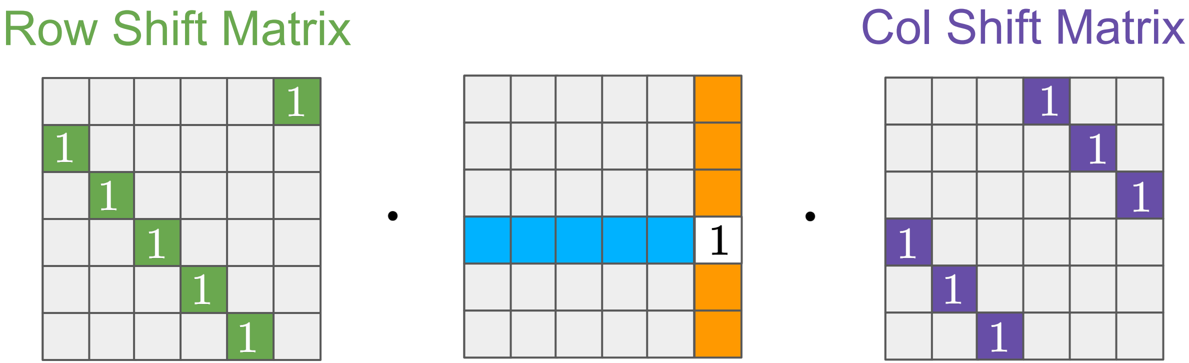
Alice

$$t_A \Rightarrow t_A = 3 \cdot \ell + 5$$

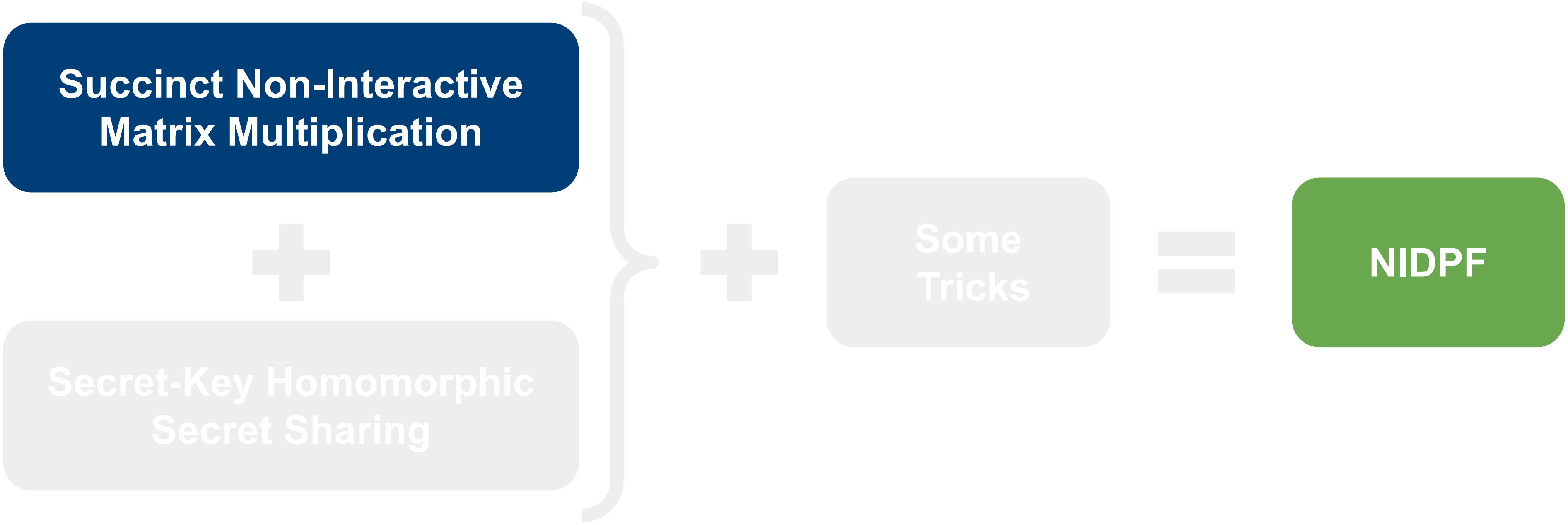
$$t_B \Rightarrow t_B = 5 \cdot \ell + 3$$

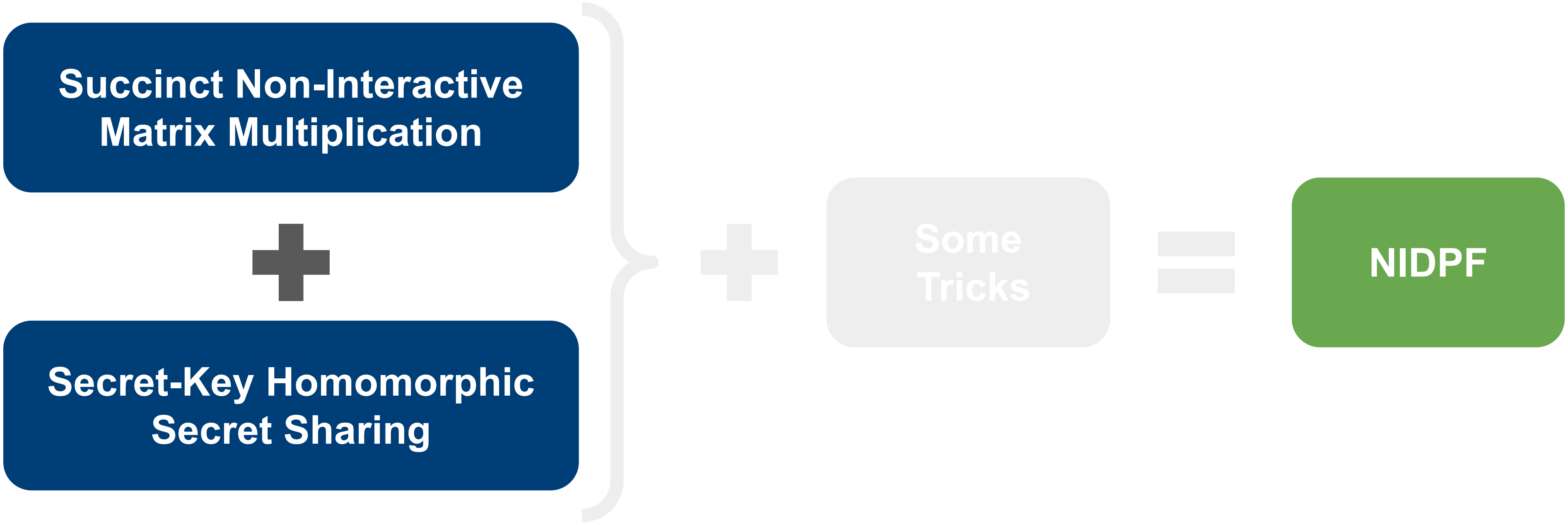
 t_B

Bob

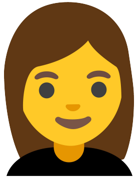


Problem: Need to multiply again with another shift matrix

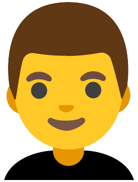

Succinct Non-Interactive Matrix Multiplication
Secret-Key Homomorphic Secret Sharing
Some Tricks
NIDPF


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

Tool: Homomorphic Secret Sharing [BGI'16]



Alice

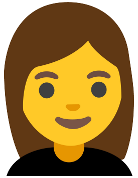


Bob

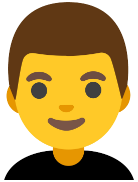
Tool: Homomorphic Secret Sharing [BGI'16]

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

Trusted Setup



Alice



Bob

Tool: Homomorphic Secret Sharing [BGI'16]

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

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

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

Tool: Homomorphic Secret Sharing [BGI'16]

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

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

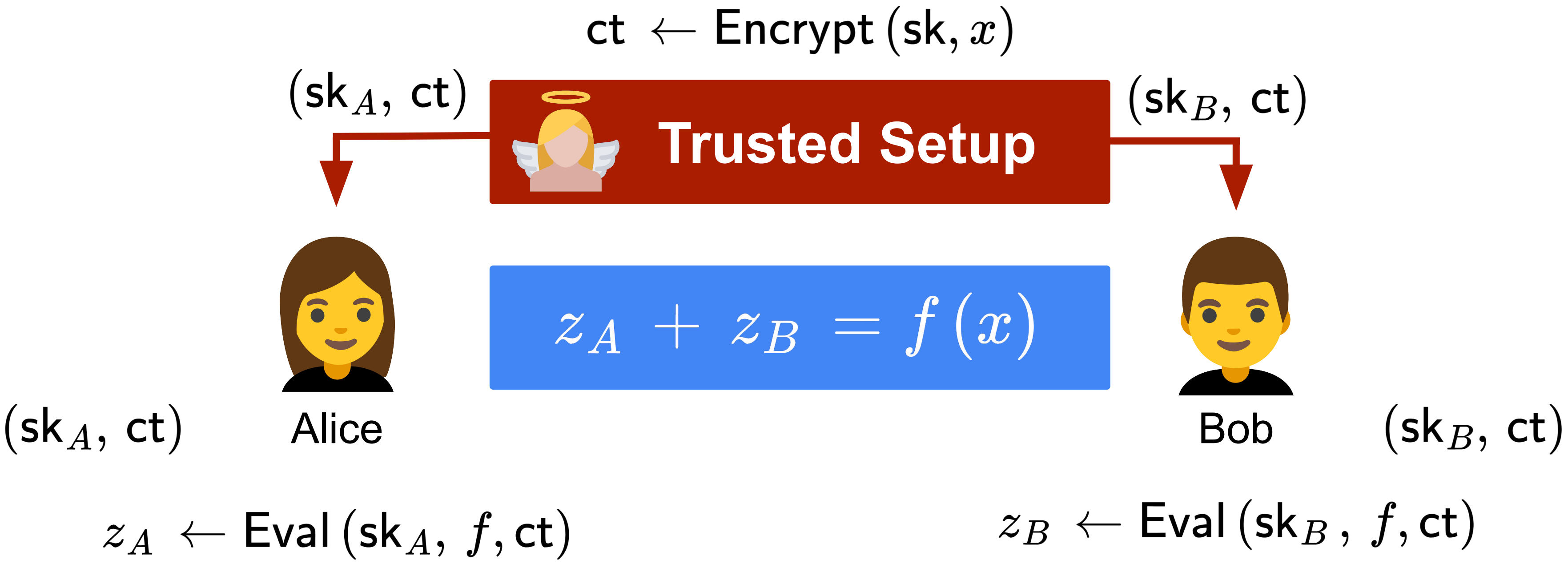
$(\mathsf{sk}_A, \mathsf{ct})$ Alice

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

$z_A \leftarrow \mathsf{Eval}(\mathsf{sk}_A, f, \mathsf{ct})$

$z_B \leftarrow \mathsf{Eval}(\mathsf{sk}_B, f, \mathsf{ct})$

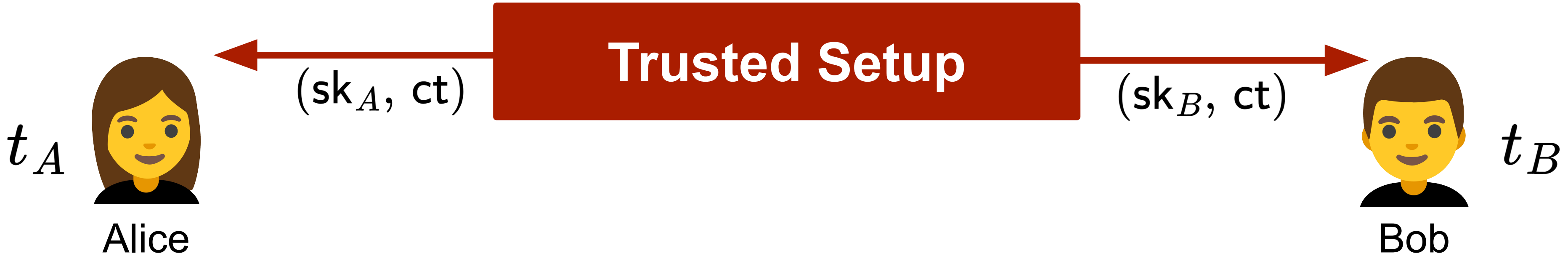
Tool: Homomorphic Secret Sharing [BGI'16]

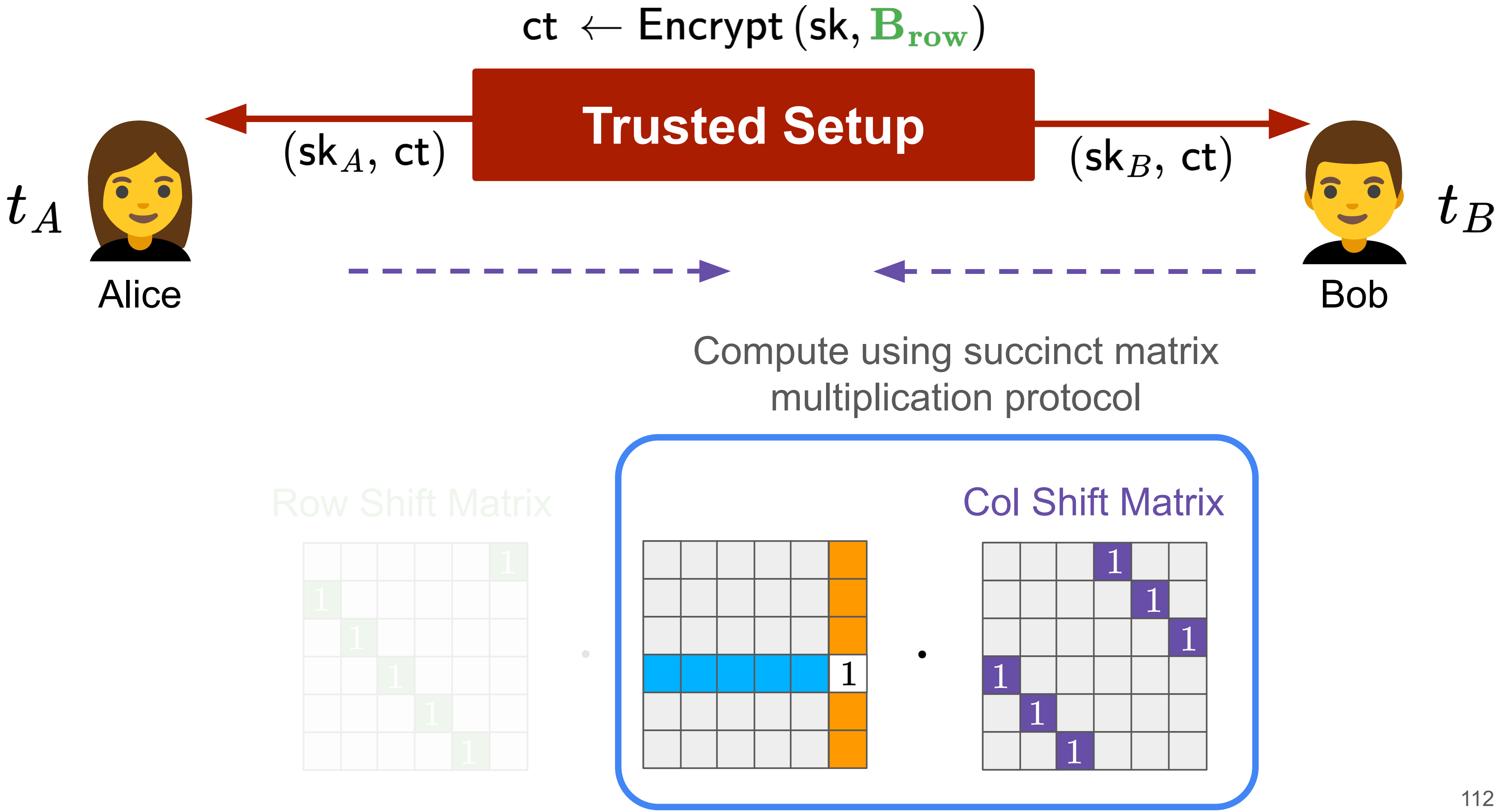


Informal Theorem ([BGI'16 + follow-up work])

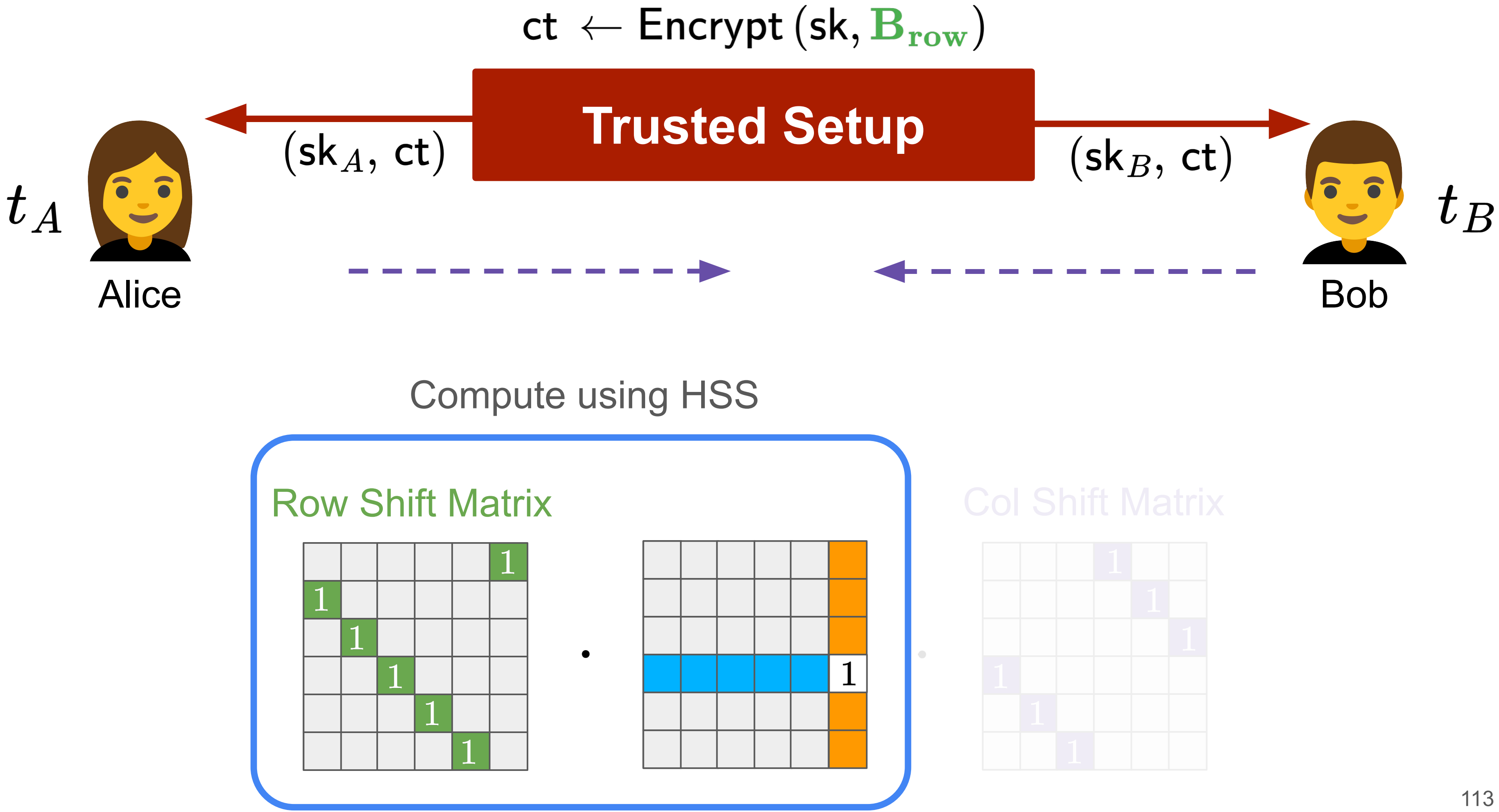
There exists a two-party, degree-2 **homomorphic secret sharing scheme** under any of the following assumptions: DDH, DCR, QR, or LWE.

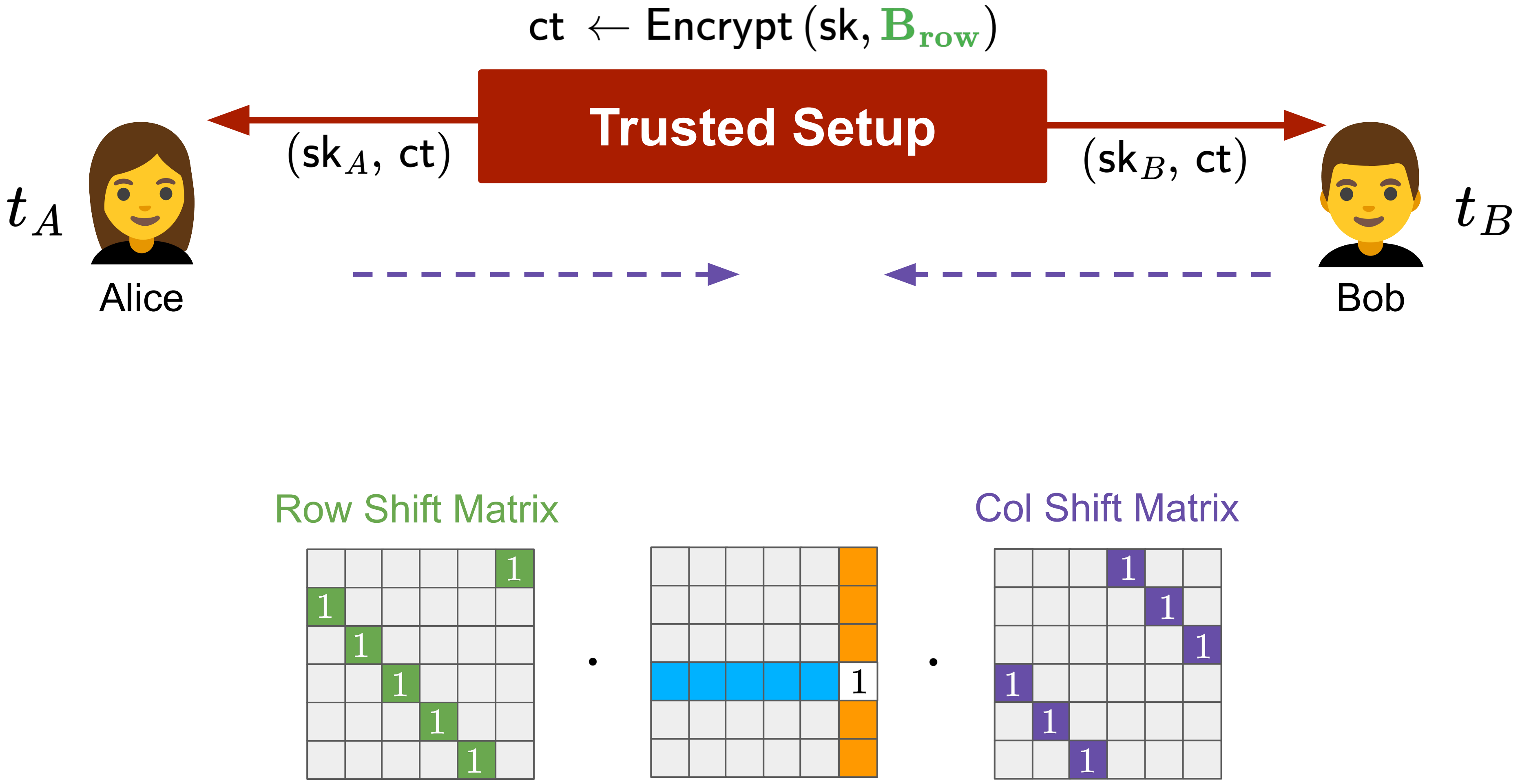
$$\mathsf{ct} \leftarrow \mathsf{Encrypt}\left(\mathsf{sk}, \mathbf{B}_{\mathbf{row}}\right)$$





ct ← Encrypt (sk, B_row)
Trusted Setup
(sk_A, ct)
(sk_B, ct)
t_A
t_B
Alice
Bob
Compute using succinct matrix multiplication protocol
Row Shift Matrix
Col Shift Matrix
1


ct ← Encrypt (sk, $\mathbf{B}_{\mathbf{row}}$)
Trusted Setup
(sk$_A$, ct)
(sk$_B$, ct)
t_A
t_B
Alice
Bob
Compute using HSS
Row Shift Matrix
Col Shift Matrix
1
1
1
1
1
1
1


$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{B}_{\mathbf{row}})$
Trusted Setup
$(\mathsf{sk}_A, \mathsf{ct})$
$(\mathsf{sk}_B, \mathsf{ct})$
t_A
Alice
t_B
Bob
Row Shift Matrix
Col Shift Matrix
1
1
1
1
1
1
1
1
1
1
1
1
1

$$\text{ct} \leftarrow \text{Encrypt}(\text{sk}, \mathbf{B}_{\text{row}})$$

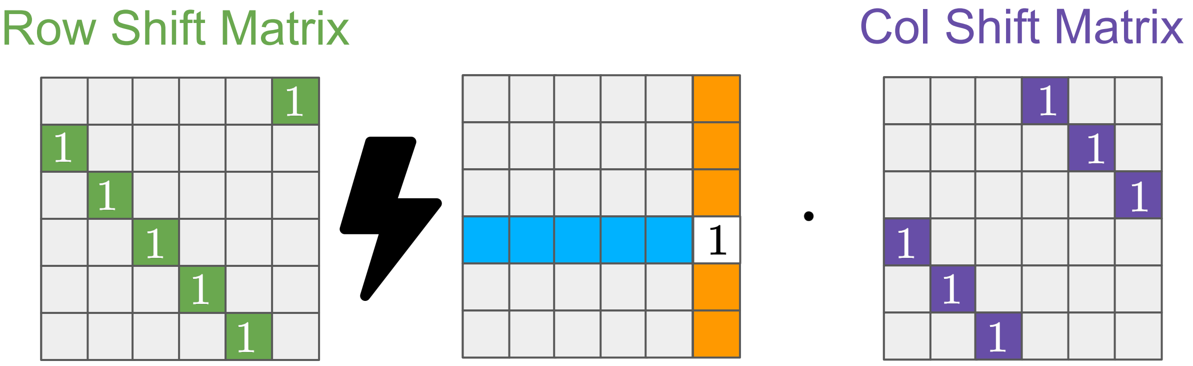
Trusted Setup

t_A Alice (sk_A, ct)

(sk_B, ct) Bob t_B

Problem 1: who runs the trusted setup?

Problem 2: cannot evaluate HSS over secret shares



Succinct Non-Interactive Matrix Multiplication

+

Secret-key Homomorphic Secret Sharing

+

Some Tricks

=

NIDPF

Succinct Non-Interactive Matrix Multiplication

+

Secret-key Homomorphic Secret Sharing

+

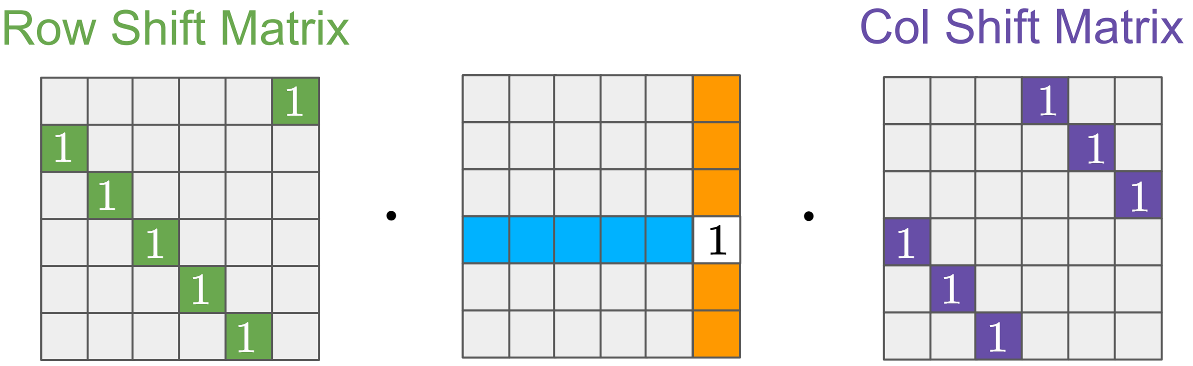
Some Tricks

=

NIDPF

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

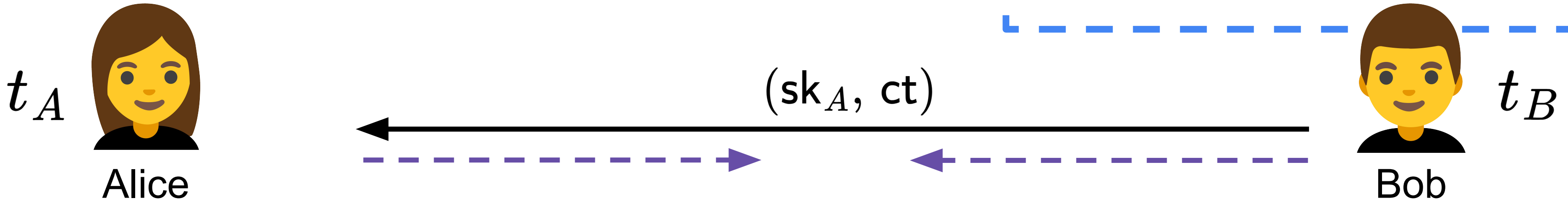
Problem 1: who runs the trusted setup?

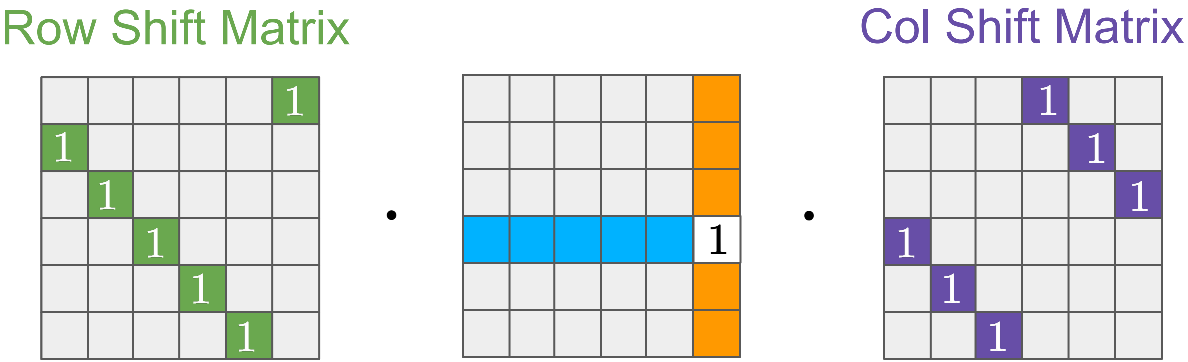


Trick 1: Bob runs the setup!

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

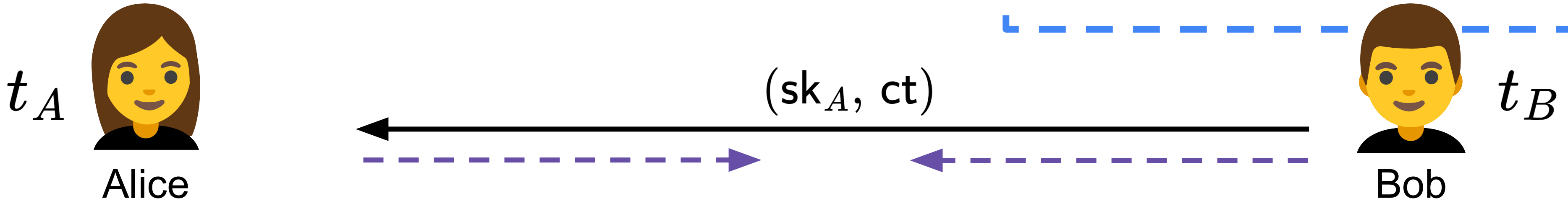
$(\mathsf{sk}_A, \mathsf{ct}) \quad (\mathsf{sk}_B, \mathsf{ct})$



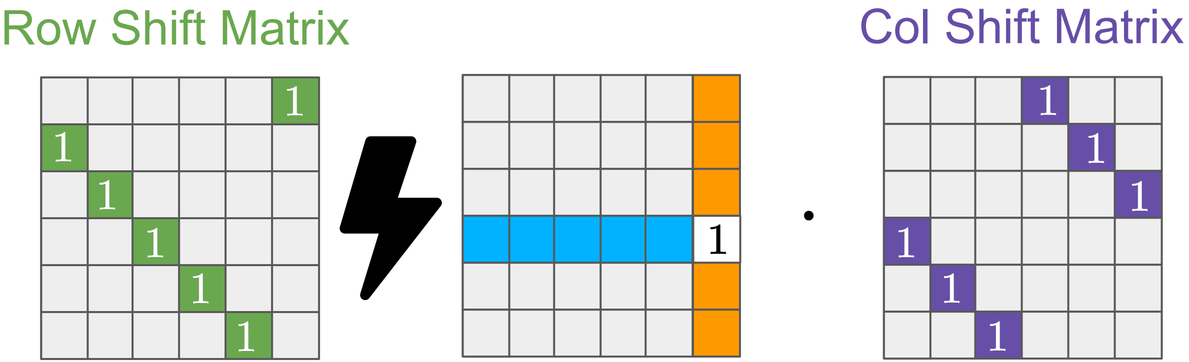


$\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mathbf{B}_{\mathbf{row}})$

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



Problem 2: cannot evaluate HSS over secret shares



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.

Informal Theorem ([CMPR'23])

Informal Theorem ([CMPR'23])

1. Input and memory shares: $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, x)$ $\quad \vec{y}_A + \vec{y}_B = (y, \mathsf{sk} \cdot y)$

Informal Theorem ([CMPR'23])

1. Input and memory shares: $\mathsf{ct} \leftarrow \mathsf{Encrypt}\left(\mathsf{sk}, x\right)$ $\quad \vec{y}_A + \vec{y}_B = (y,\ \mathsf{sk} \cdot y)$

2. Local evaluation: $z_A := \mathsf{Mult}\left(\mathsf{ct}, \vec{y}_A\right)$ $\quad z_B := \mathsf{Mult}\left(\mathsf{ct}, \vec{y}_B\right)$

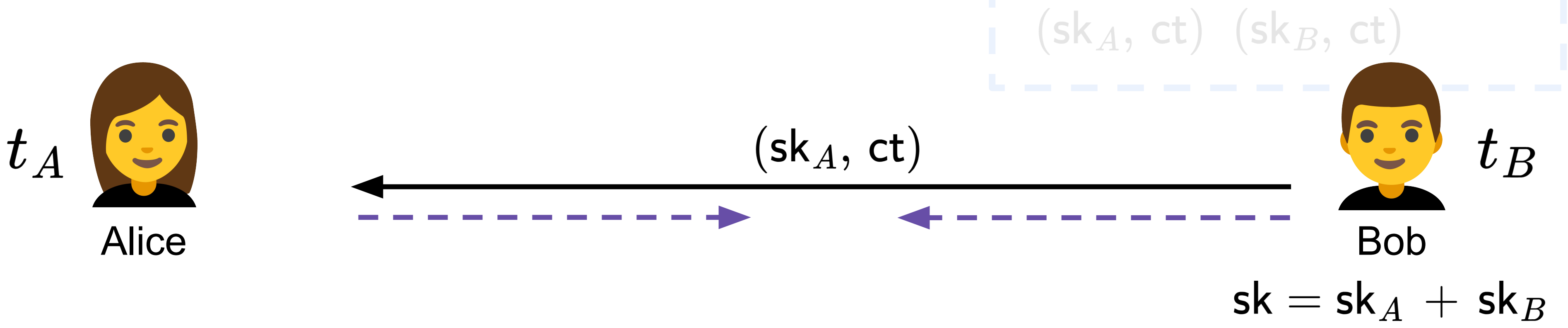
Informal Theorem ([CMPR'23])

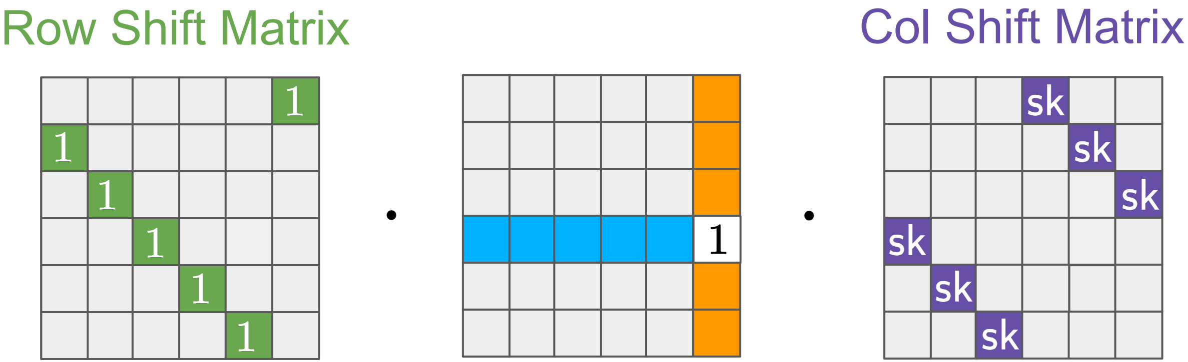
1. Input and memory shares: $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{sk}, x)$ $\quad \vec{y}_A + \vec{y}_B = (y, \mathsf{sk} \cdot y)$

2. Local evaluation: $z_A := \mathsf{Mult}(\mathsf{ct}, \vec{y}_A)$ $\quad z_B := \mathsf{Mult}(\mathsf{ct}, \vec{y}_B)$

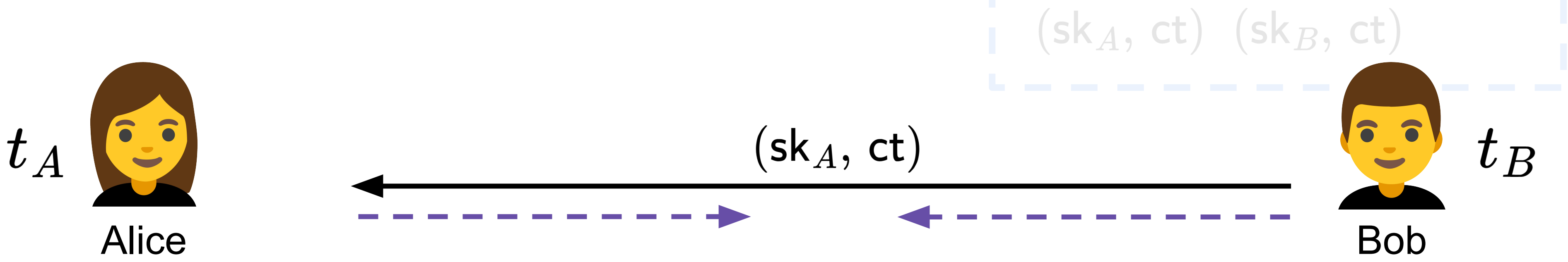
3. $z_A + z_B = xy$

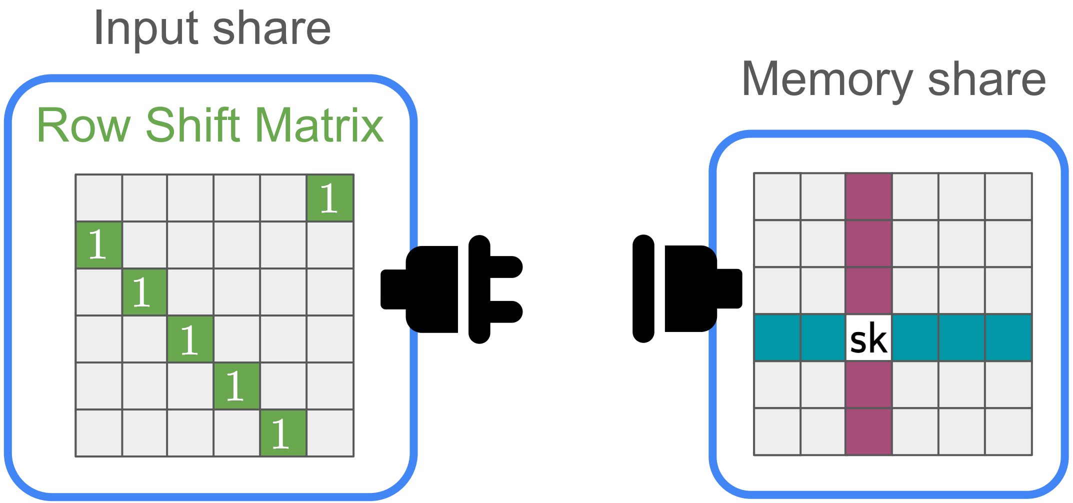
Trick 2: Convert additive shares into "memory shares"





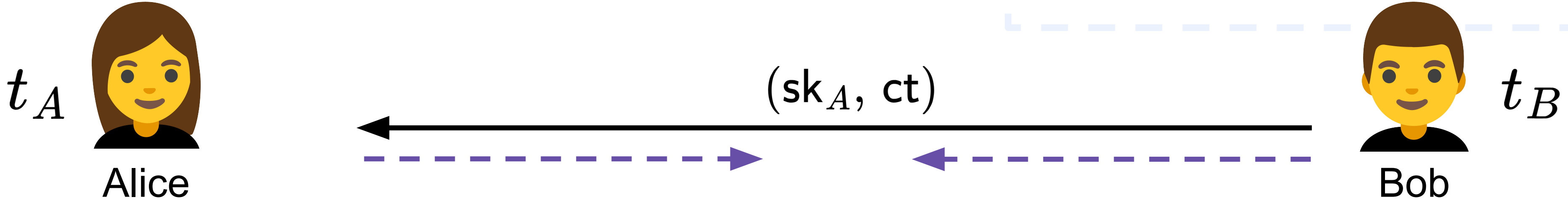
Trick 2: Convert additive shares into “memory shares”

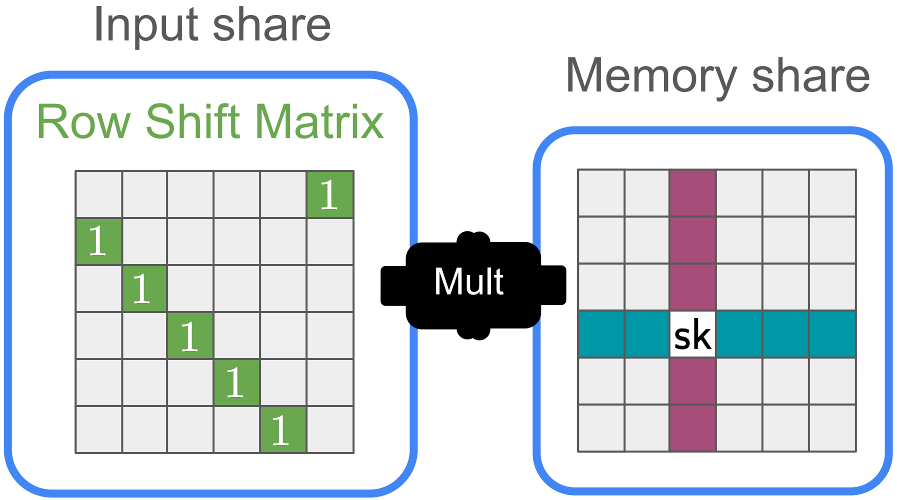




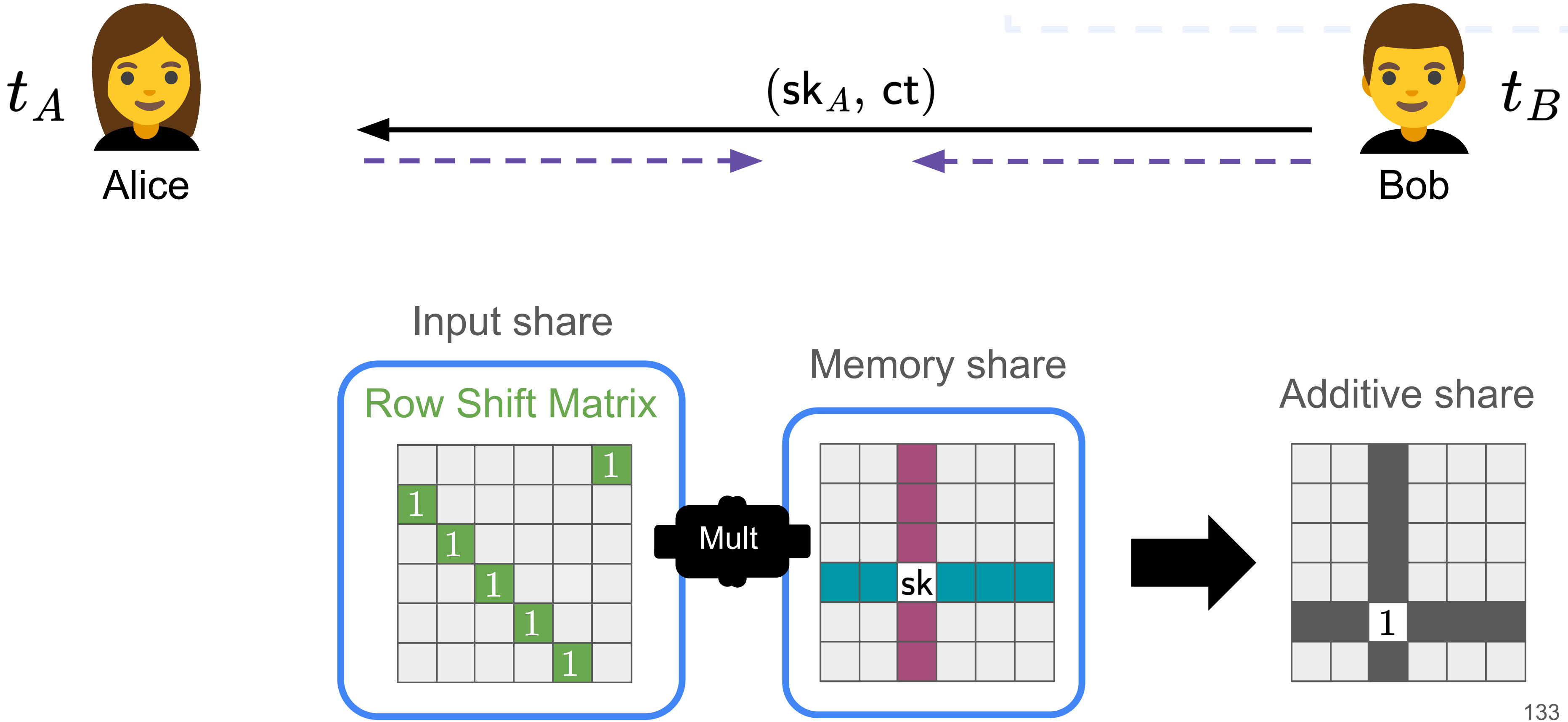
Trick 2: Convert additive shares into “memory shares”

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





Trick 2: Convert additive shares into “memory shares”

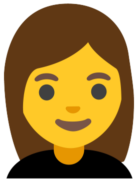


Putting things together

Putting things together

1

t_A



Alice

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

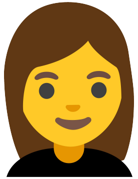
$$t_B \rightarrow t_B = i_B \cdot \ell + j_B$$

Bob

t_B

Putting things together

2

t_A   t_B

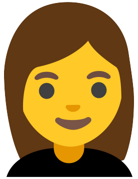
Alice Bob

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

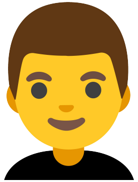
Putting things together

2

t_A



Alice

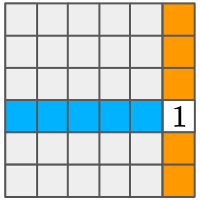


Bob

t_B

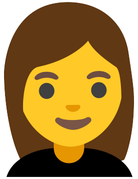
$\mathsf{pk}_A := \mathsf{pk}_A^{\mathsf{matmul}}$

$\mathbf{A} =$

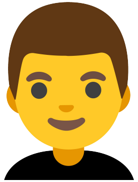


Putting things together

2

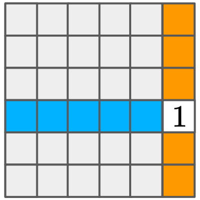
t_A 

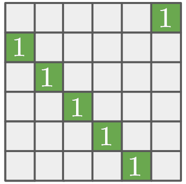
Alice

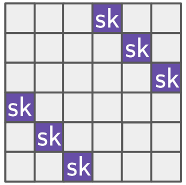
 t_B

Bob

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

$\mathbf{A} =$ 

$\mathbf{B}_{\mathbf{row}} =$ 

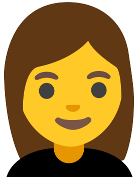
$\mathbf{B}_{\mathbf{col}} =$ 

Putting things together

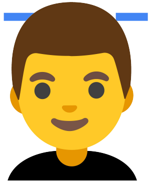
$\text{ct} \leftarrow \text{Encrypt}(\text{sk}, \mathbf{B}_{\mathbf{row}})$

$(\text{sk}_A, \text{ct}) \quad (\text{sk}_B, \text{ct})$

2

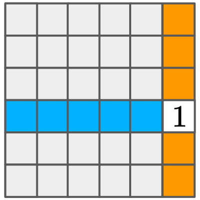
t_A 

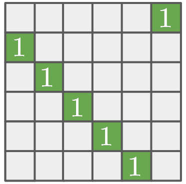
Alice

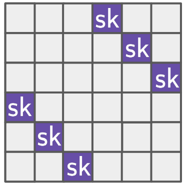
 t_B

Bob

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

$\mathbf{A} =$ 

$\mathbf{B}_{\mathbf{row}} =$ 

$\mathbf{B}_{\mathbf{col}} =$ 

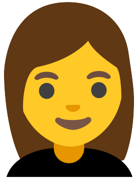
Putting things together

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

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

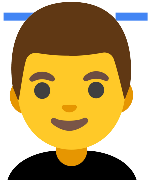
2

t_A



Alice

t_B

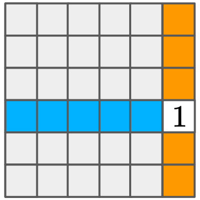


Bob

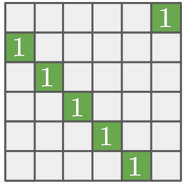
$\mathsf{pk}_A := \mathsf{pk}_A^{\mathsf{matmul}}$

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

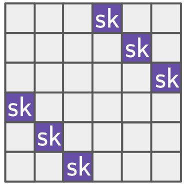
$\mathbf{A} =$



$\mathbf{B}_{\mathbf{row}} =$



$\mathbf{B}_{\mathbf{col}} =$

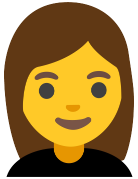


Putting things together

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

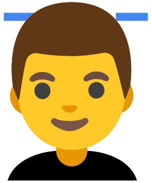
$(\mathsf{sk}_A, \mathsf{ct}) \quad (\mathsf{sk}_B, \mathsf{ct})$

2

t_A  Alice

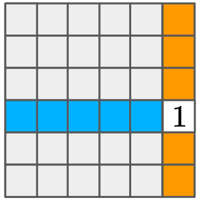
$\mathsf{pk}_A \rightarrow$

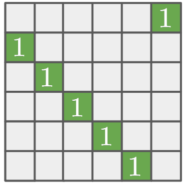
$\leftarrow \mathsf{pk}_B$

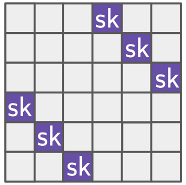
 t_B Bob

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

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

$\mathbf{A} =$ 

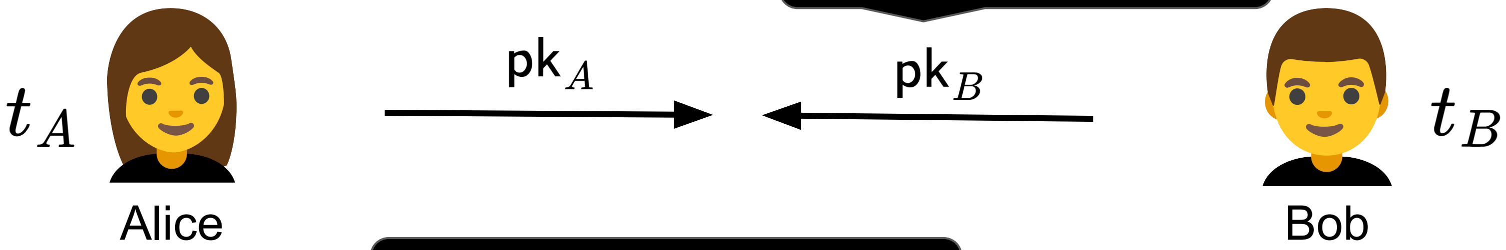
$\mathbf{B}_{\mathbf{row}} =$ 

$\mathbf{B}_{\mathbf{col}} =$ 

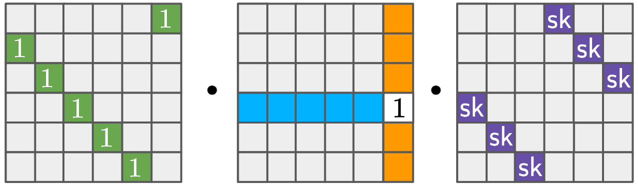
Putting things together

3

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

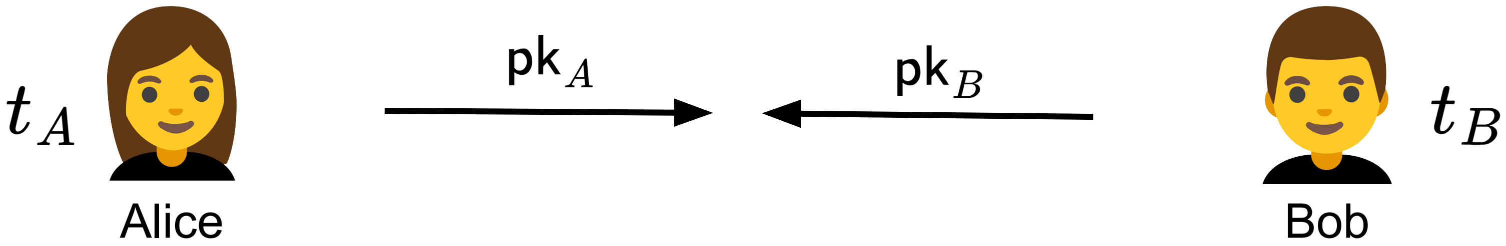


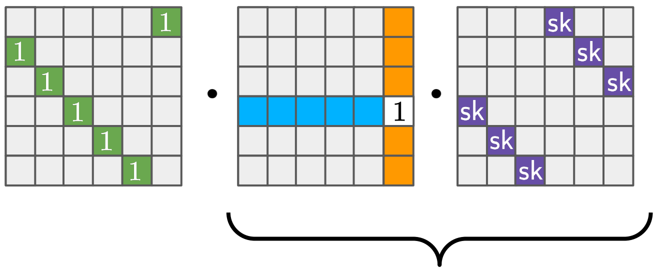
Sparse and compressible



Putting things together

3

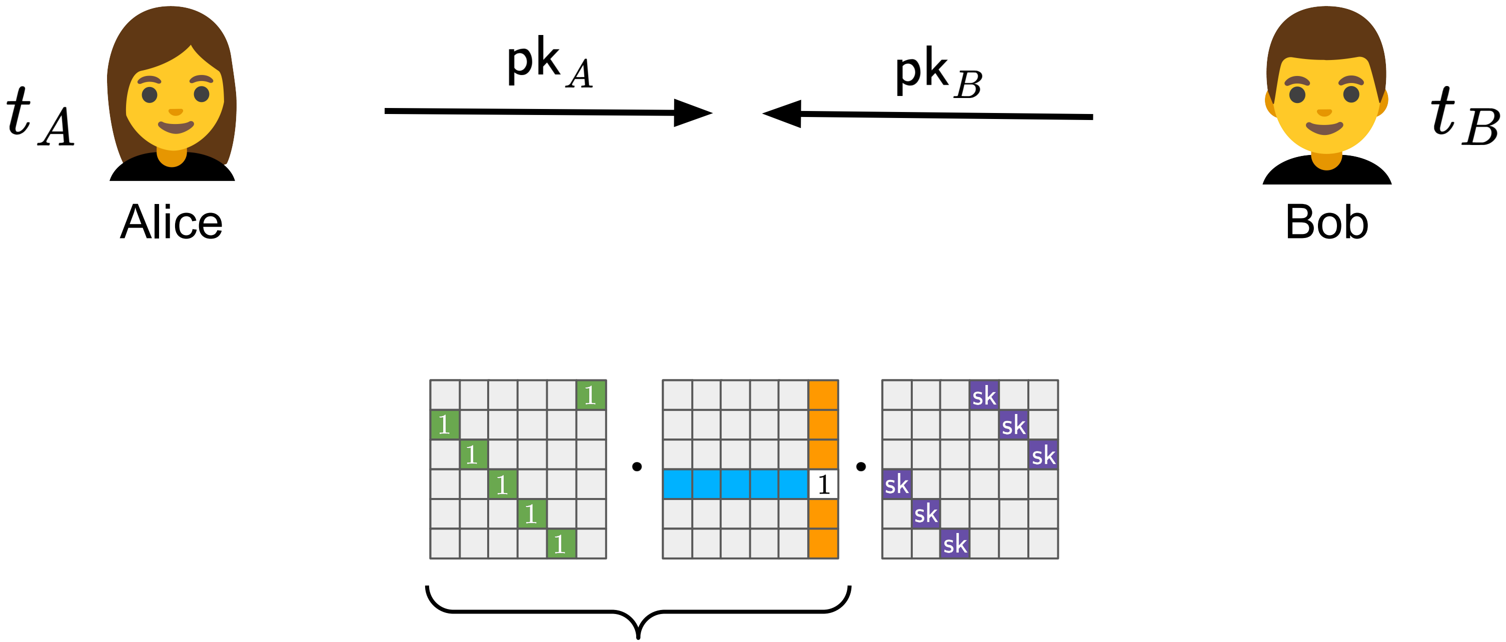




Multiply using Non-Interactive Matrix Multiplication

Putting things together

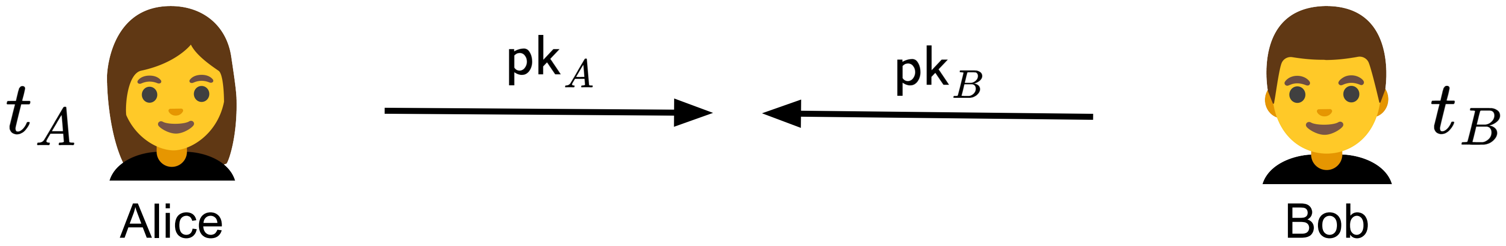
3

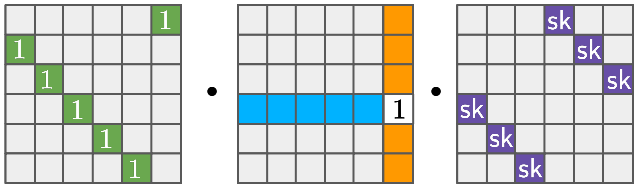


Multiply using Homomorphic Secret Sharing

Putting things together

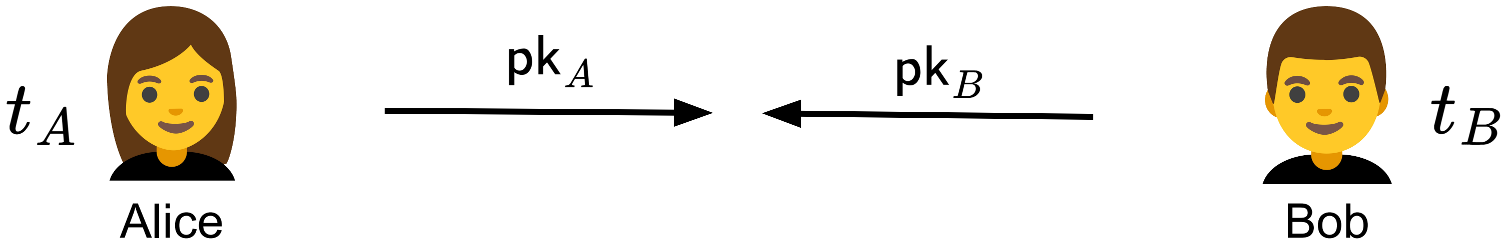
3

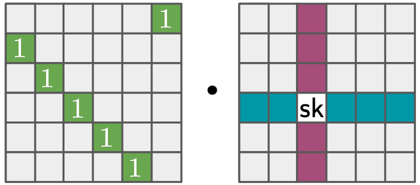




Putting things together

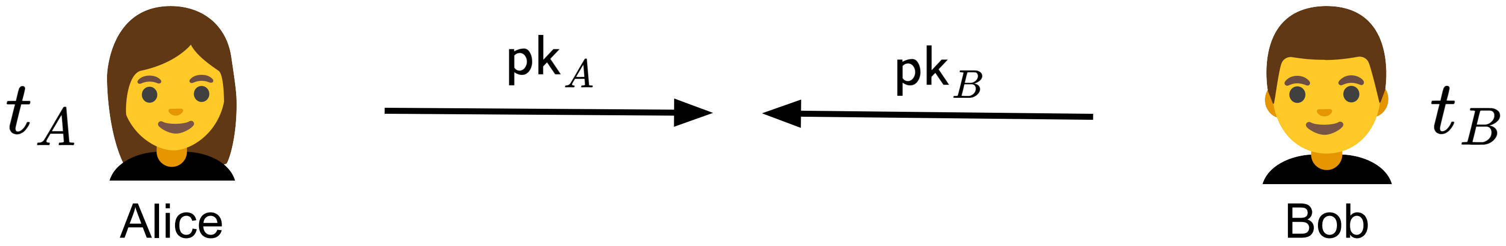
3

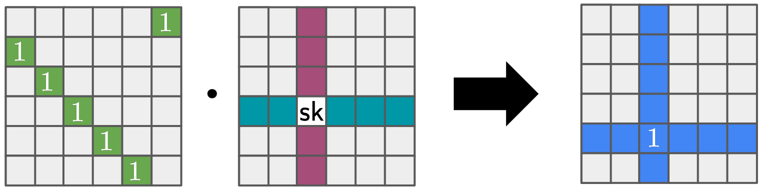




Putting things together

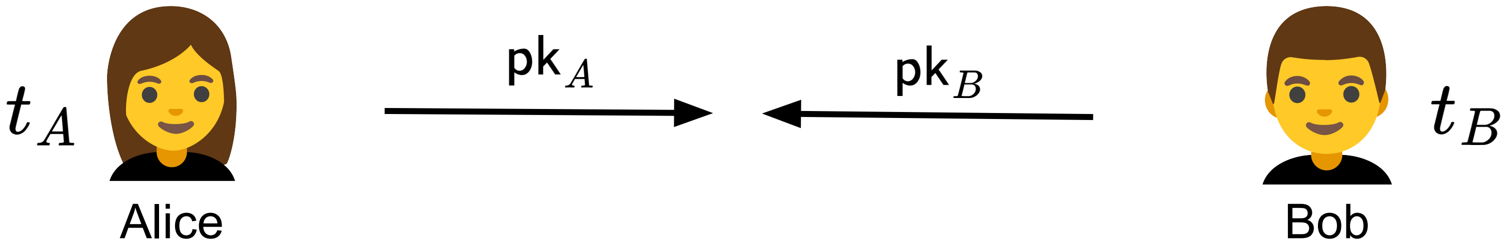
3

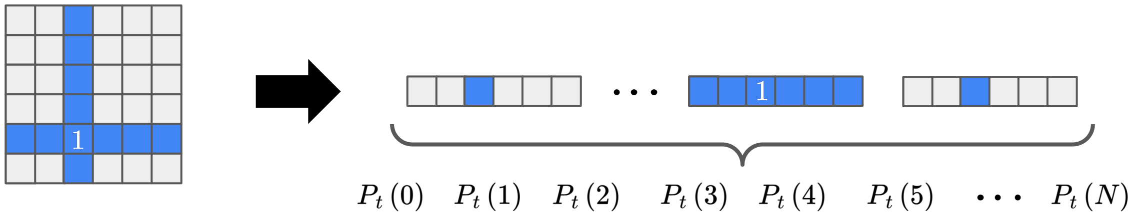




Putting things together

4





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**

More in the paper!

More in the paper!

Generalization to succinct "multi-key" homomorphic secret sharing

More in the paper!

Generalization to succinct “multi-key” homomorphic secret sharing

Additional tricks to construct NIDPFs from the SXDH assumption

More in the paper!

Generalization to succinct “multi-key” homomorphic secret sharing

Additional tricks to construct NIDPFs from the SXDH assumption

Open questions:

- Asymptotically optimal key sizes?
- Concretely efficient construction?

Thank you!

Email: 3s@mit.edu
ePrint: ia.cr/**2024/1079**

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."