Implementing PSI: From Elliptic Curves to Oblivious Transfer and Distance-Aware Extensions

Ni Trieu (Arizona State University)

Presented at NIST STPPA#8, 2025-Sep-18
Special Topics on Privacy and Public Auditability (Event 8)

This talk is based on joint work with:

- Mike Rosulek (OSU)
- Lucas Piske (ASU)
- Jaspal Singh, Vassilis Zikas, Vladimir Kolesnikov (Georgia Tech)

Slides partially based on Mike's CCS 2021 presentation and Jaspal's Simon Workshop talk.

Outline

- **Introduction to PSI**
- **Traditional PSI Protocol**
- **PSI for Small Sets**
- **PSI for Large Sets**
- **Fuzzy PSI**
- **Take-Home Messages**

PSI Functionality

Sender

Receiver

Sender's set

$Y = \{p, s, i\}$

f = intersection

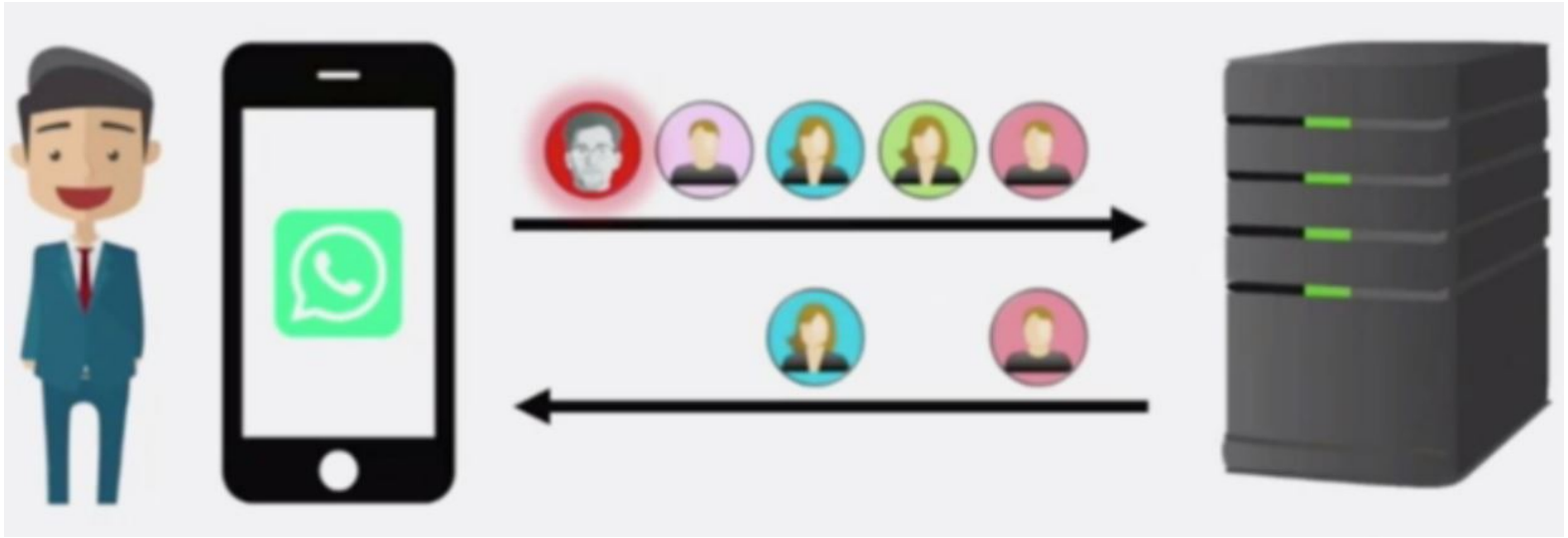
Receiver's set

$X = \{n, i, s, t\}$

$X \cap Y = \{s, i\}$

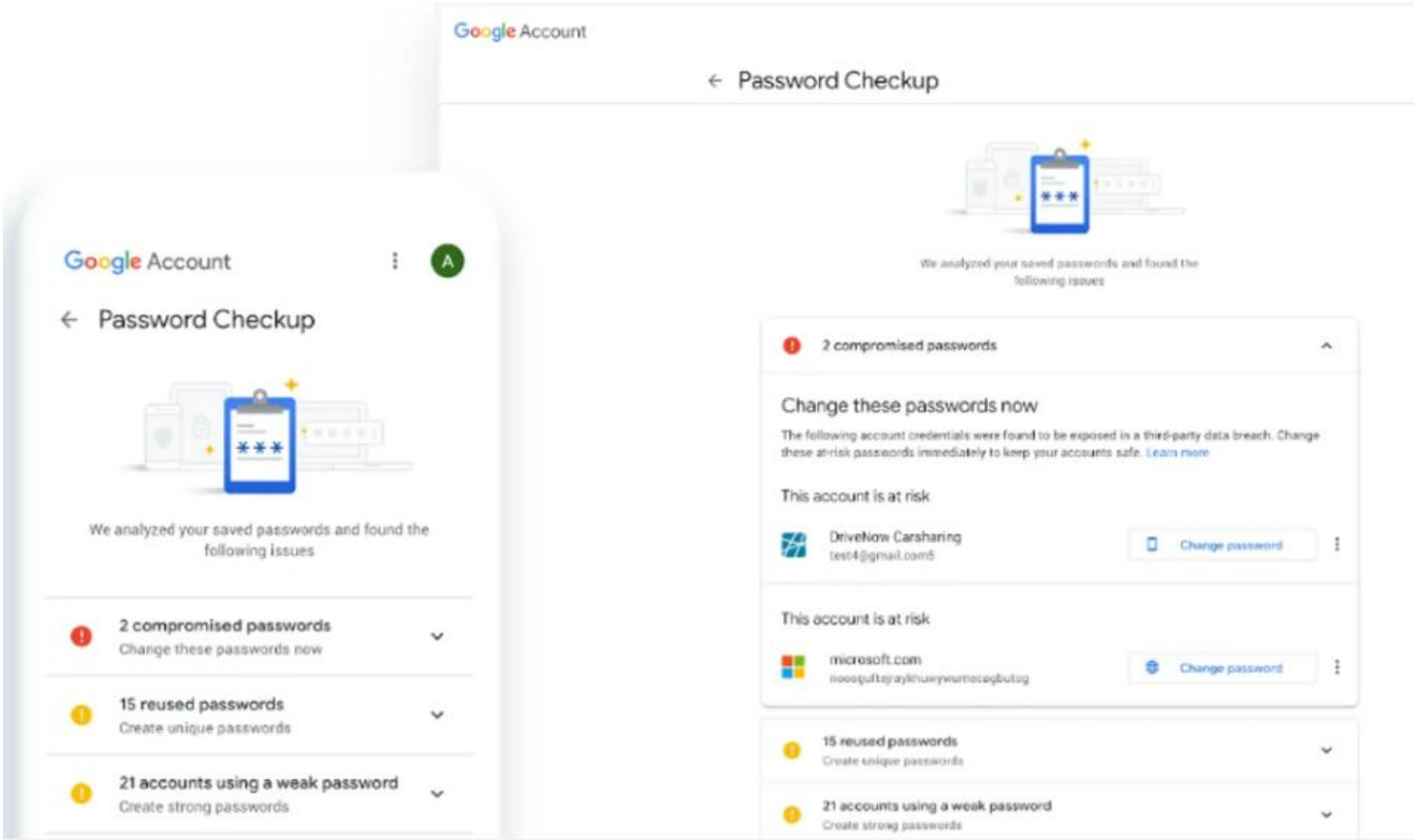
$\Rightarrow$ PSI outputs only the common items

PSI Applications



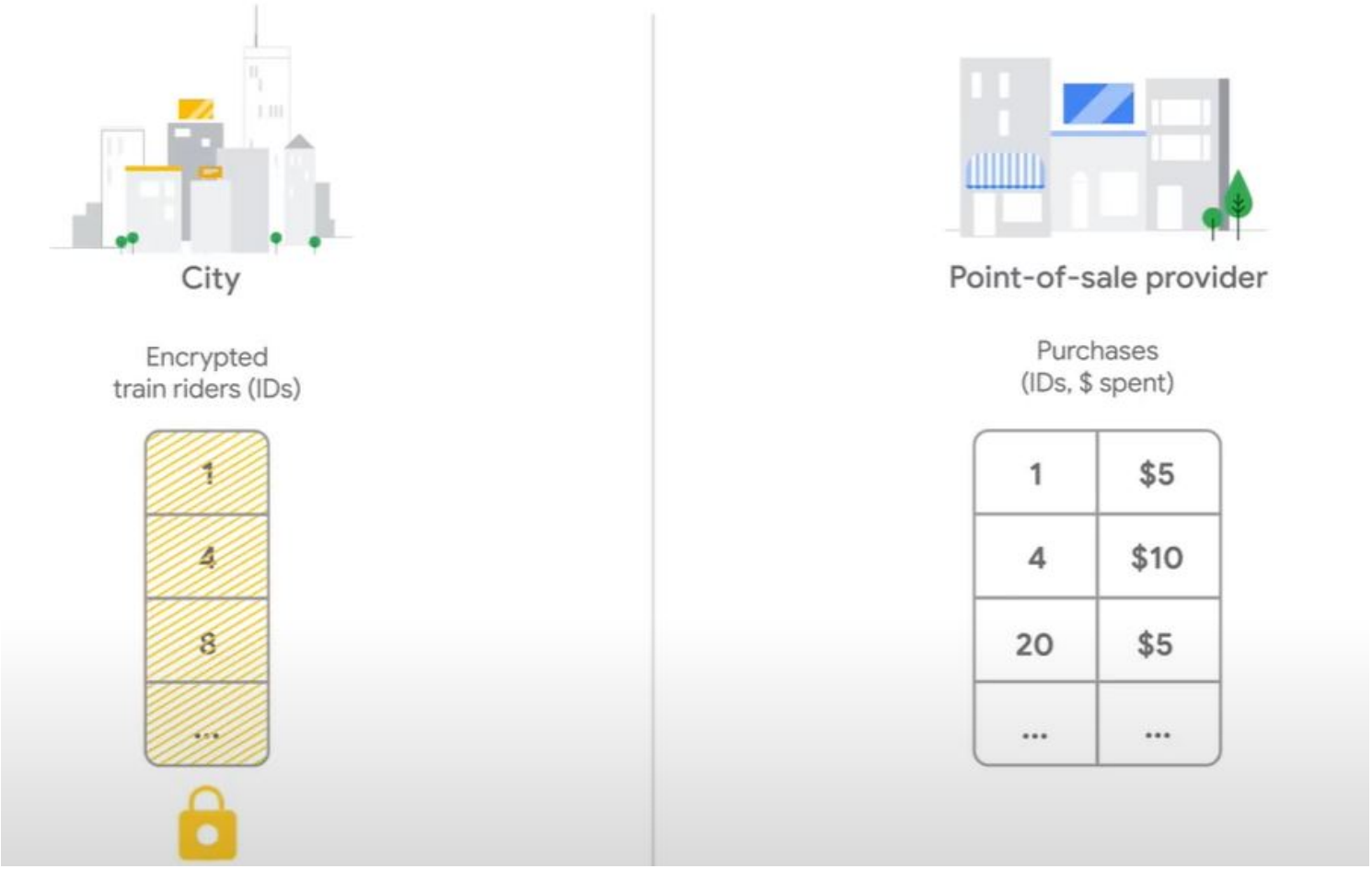
{my phone contacts} $\cap$ {users of your service}

PSI Applications



$$\{\text{my passwords}\} \cap \{\text{passwords found in breaches}\}$$

PSI Applications



$$\{\text{my purchases}\} \cap \{\text{your advertising}\}$$

A Naïve and Insecure PSI Protocol

- Bob sends $H(Y)$ to Alice.
- Alice compares the hash values and computes the intersection.
- **Pros:** Fast and low communication cost.
- **Cons:** Insecure against dictionary attacks — leaks Y.

A Naïve and Insecure PSI Protocol

- Bob sends $H(Y)$ to Alice.
- Alice compares the hash values and computes the intersection.
- **Pros:** Fast and low communication cost.
- **Cons:** Insecure against dictionary attacks — leaks Y.

A Naïve and Insecure PSI Protocol

- Bob sends $H(Y)$ to Alice.
- Alice compares the hash values and computes the intersection.
- **Pros:** Fast and low communication cost.
- **Cons:** Insecure against dictionary attacks — leaks Y.

MPC-based PSI Protocol

- Represent each element as a secret-shared value.
- Compare elements securely.
- Reconstruct only the elements that are in the intersection.

$$[z_1], \ldots, [z_{2n}] \leftarrow \text{Intersection}([x_1], \ldots, [x_n], [y_1], \ldots, [y_n])$$

Example: Sets $X = \{1, 2, 3\}$ and $Y = \{1, 3, 5\}$

- **Equality check:** $< 1, 1, 2, 3, 3, 5 > \rightarrow < 1, 3 >$
- **Note:** Expensive in MPC

MPC-based PSI Protocol

- Represent each element as a secret-shared value.
- Compare elements securely.
- Reconstruct only the elements that are in the intersection.

$$[z_1], \ldots, [z_{2n}] \leftarrow \text{Intersection}([x_1], \ldots, [x_n], [y_1], \ldots, [y_n])$$

Example: Sets $X = \{1, 2, 3\}$ and $Y = \{1, 3, 5\}$

- **Equality check:** $< 1, 1, 2, 3, 3, 5 > \rightarrow < 1, 3 >$
- **Note:** Expensive in MPC

Outline

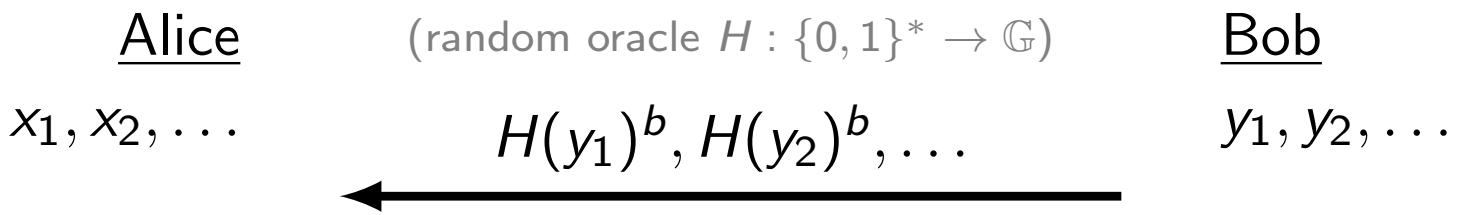
- ~~Introduction to PSI~~
- Traditional PSI Protocol
- PSI for Small Sets
- PSI for Large Sets
- Fuzzy PSI
- Take-Home Messages

DH-based PSI Protocol

Alice	(random oracle $H : \{0,1\}^* \to \mathbb{G}$)	Bob
$x_1, x_2, \ldots$		$y_1, y_2, \ldots$

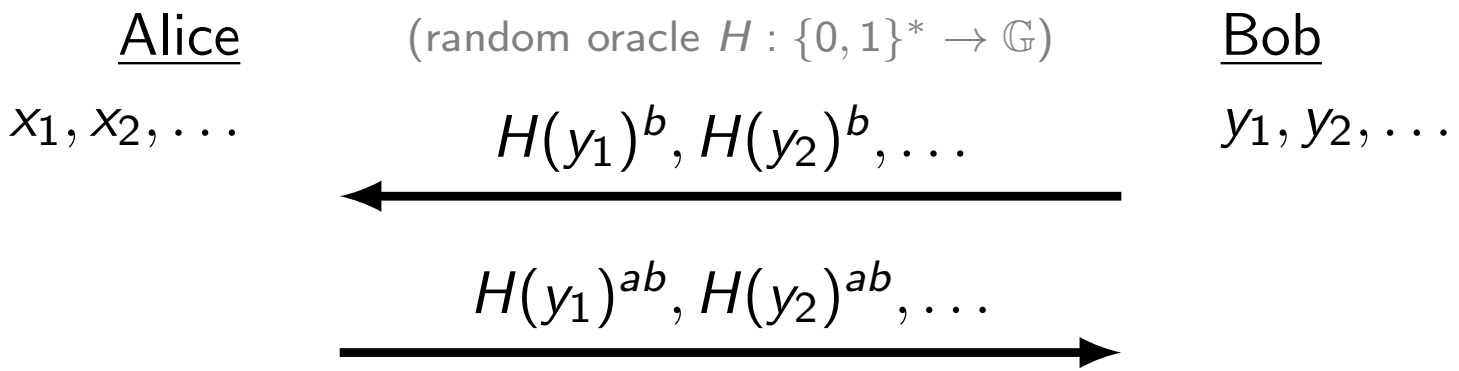
[HubermanFranklinHogg99]

DH-based PSI Protocol



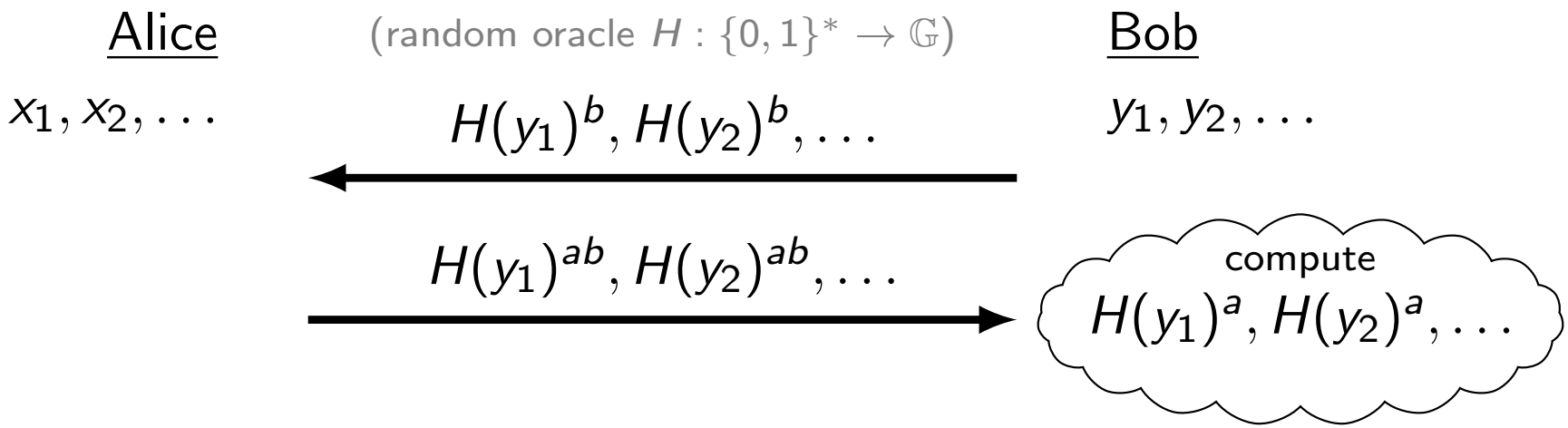
[HubermanFranklinHogg99]

DH-based PSI Protocol



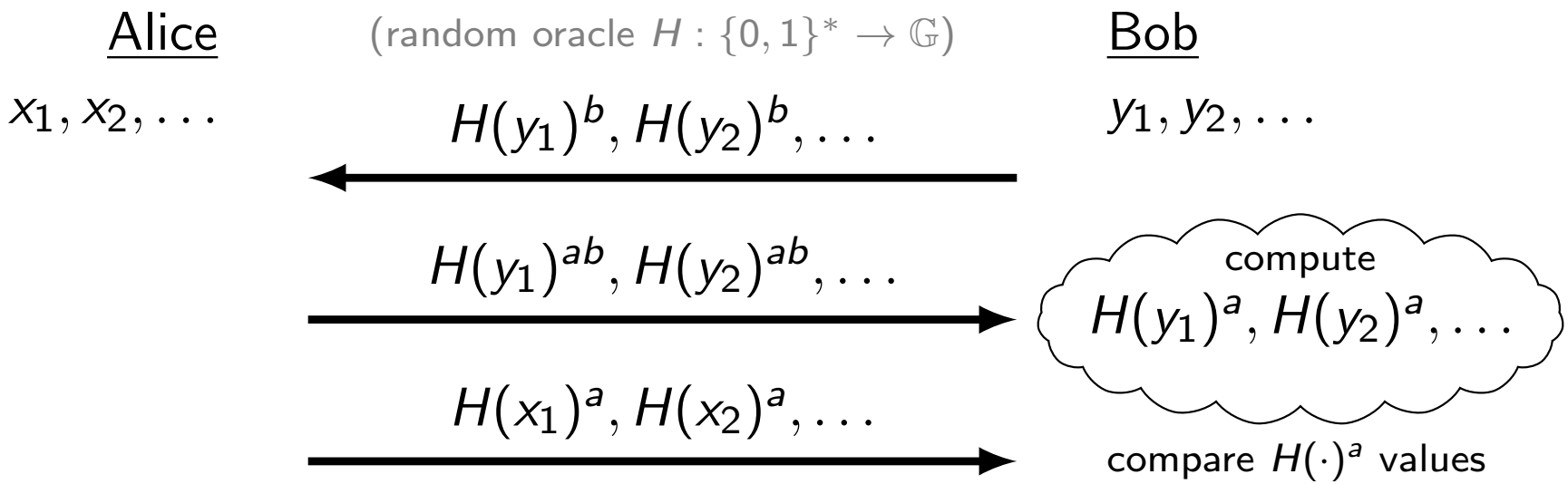
[HubermanFranklinHogg99]

DH-based PSI Protocol



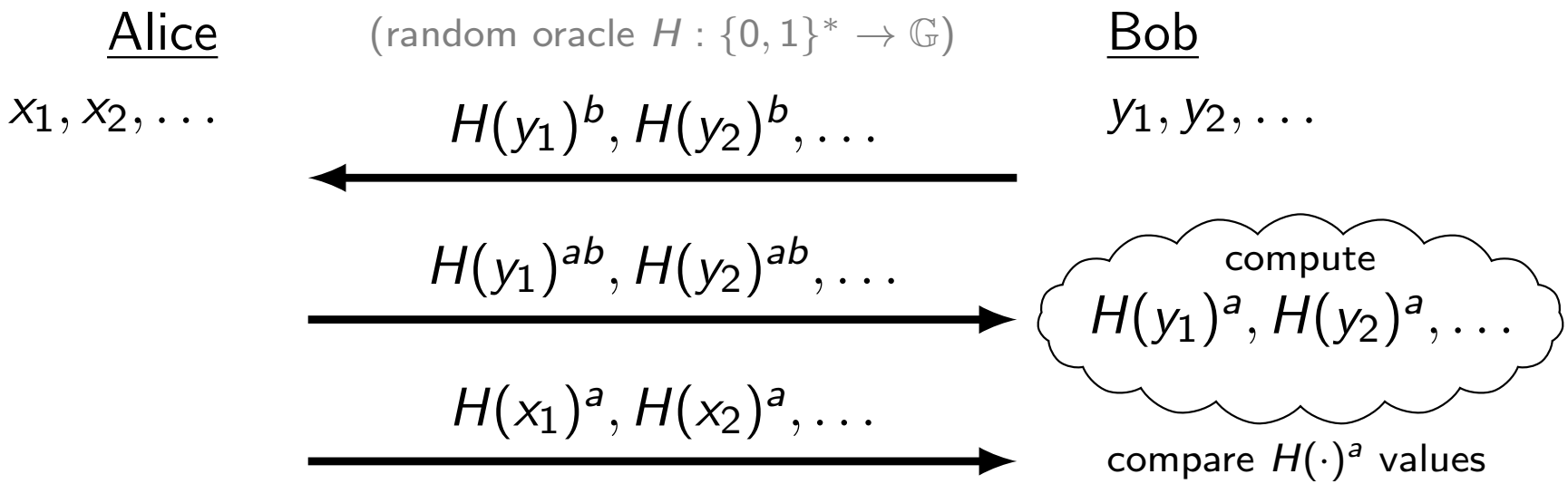
[HubermanFranklinHogg99]

DH-based PSI Protocol



[HubermanFranklinHogg99]

DH-based PSI Protocol

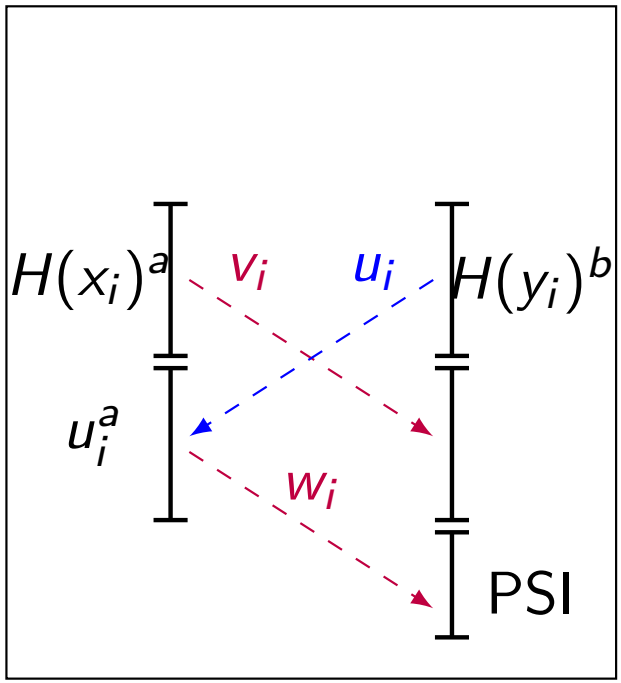


Semi-honest security:

- $x \mapsto H(x)^a$ is a PRF (DDH assumption + random oracle)

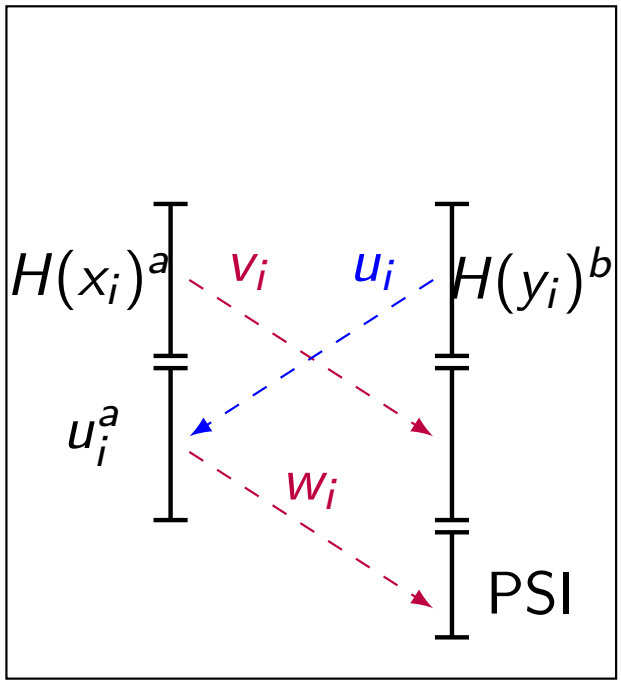
[HubermanFranklinHogg99]

Implementation of DH-based PSI Protocol



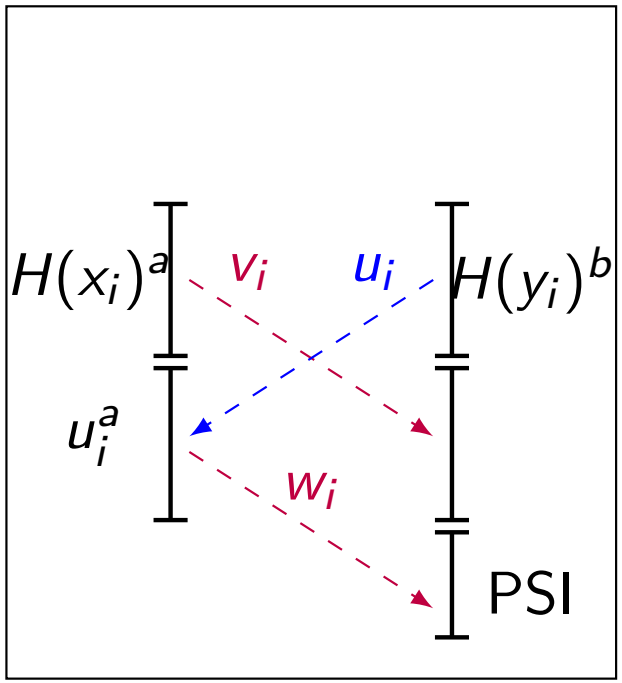
- Parallel computation of Step 1 and Step 3: improves runtime by $\sim 1.5\times$
- Sending messages such as $H(y_i)^{ab}$ using asynchronous communication: depends on network setting
- Alice's group elements can be truncated
- Using ECC to accelerate exponentiation: `libsodium` achieves $\sim$ 5–10$\times$ higher speed than `miracl`

Implementation of DH-based PSI Protocol



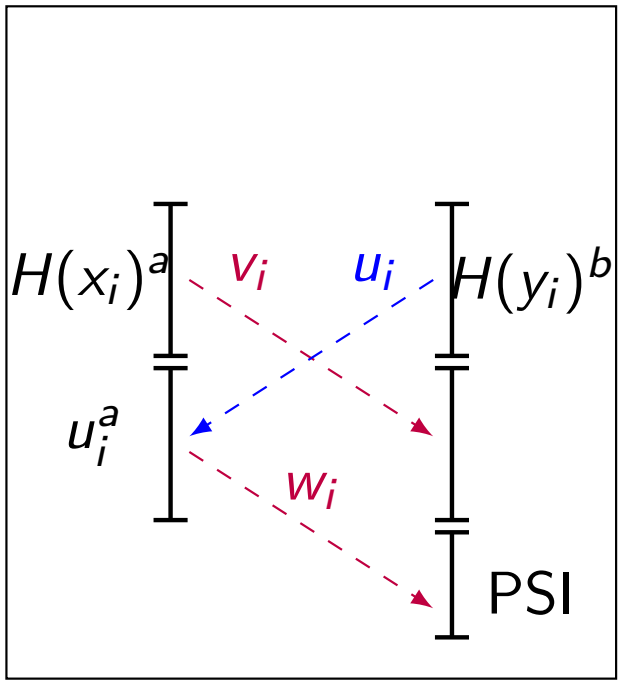
- Parallel computation of Step 1 and Step 3: improves runtime by $\sim 1.5\times$
- Sending messages such as $H(y_i)^{ab}$ using asynchronous communication: depends on network setting
- Alice's group elements can be truncated
- Using ECC to accelerate exponentiation: `libsodium` achieves $\sim$ 5–10$\times$ higher speed than `miracl`

Implementation of DH-based PSI Protocol



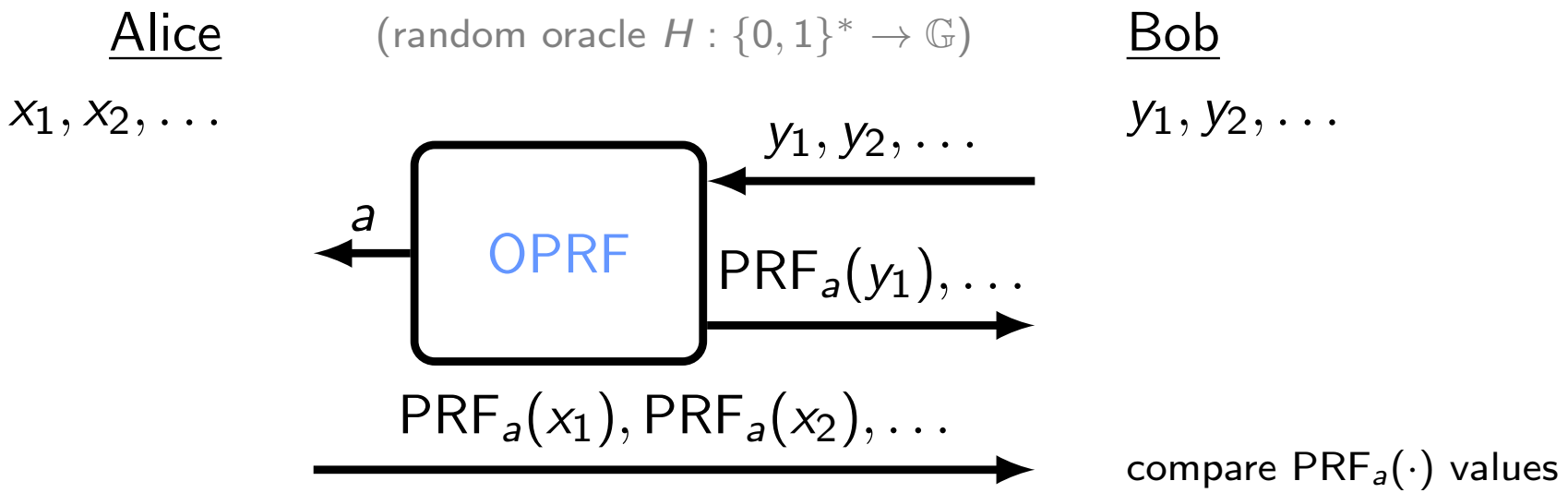
- Parallel computation of Step 1 and Step 3: improves runtime by $\sim 1.5\times$
- Sending messages such as $H(y_i)^{ab}$ using asynchronous communication: depends on network setting
- Alice's group elements can be truncated
- Using ECC to accelerate exponentiation: `libsodium` achieves $\sim$ 5–10$\times$ higher speed than `miracl`

Implementation of DH-based PSI Protocol



- Parallel computation of Step 1 and Step 3: improves runtime by $\sim 1.5\times$
- Sending messages such as $H(y_i)^{ab}$ using asynchronous communication: depends on network setting
- Alice's group elements can be truncated
- Using ECC to accelerate exponentiation: `libsodium` achieves $\sim$ 5–10$\times$ higher speed than `miracl`

DH-based PSI Protocol



Semi-honest security:

- $x \mapsto H(x)^a$ is a PRF (DDH assumption + random oracle)
- first two messages are an oblivious PRF protocol
- standard OPRF→PSI paradigm [FreedmanIshaiPinkasReingold05]

[HubermanFranklinHogg99]

OPRF for PSI

- Small-set PSI: built on public-key cryptography
- Large-set PSI: built on symmetric-key cryptography

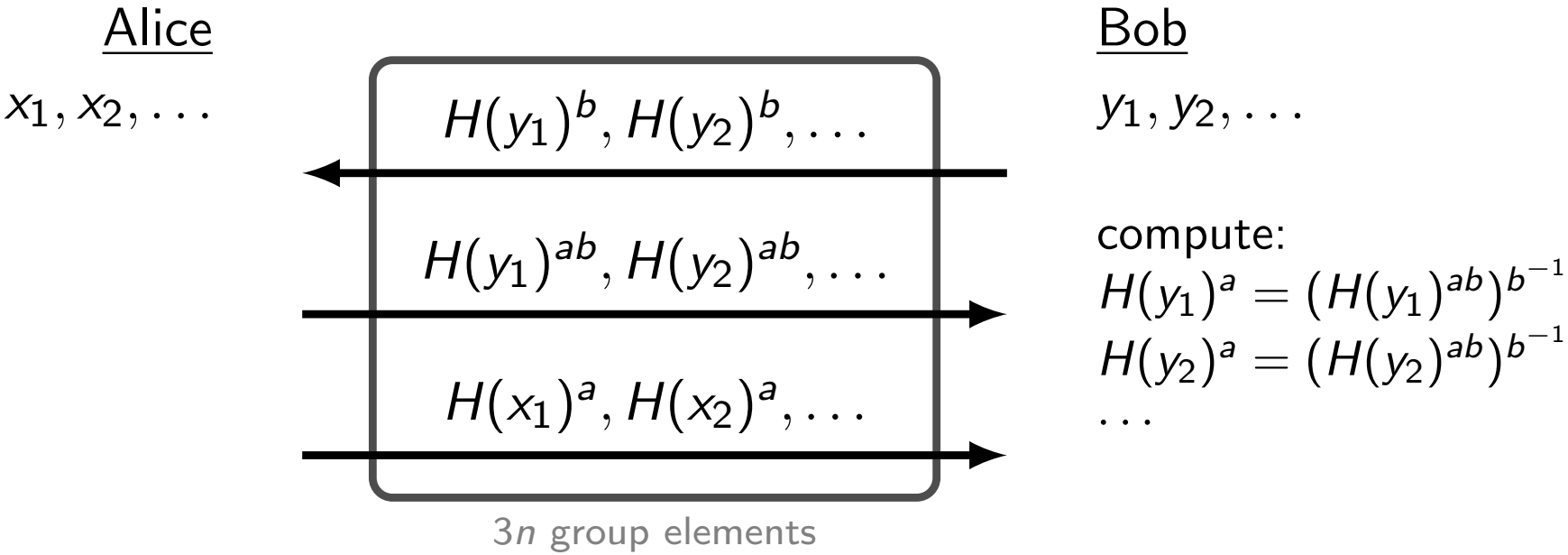
Outline

- ~~Introduction to PSI~~
- ~~Traditional PSI Protocol~~
- PSI for Small Sets
- PSI for Large Sets
- Fuzzy PSI
- Take-Home Messages

PSI for Small Sets: State of the Art

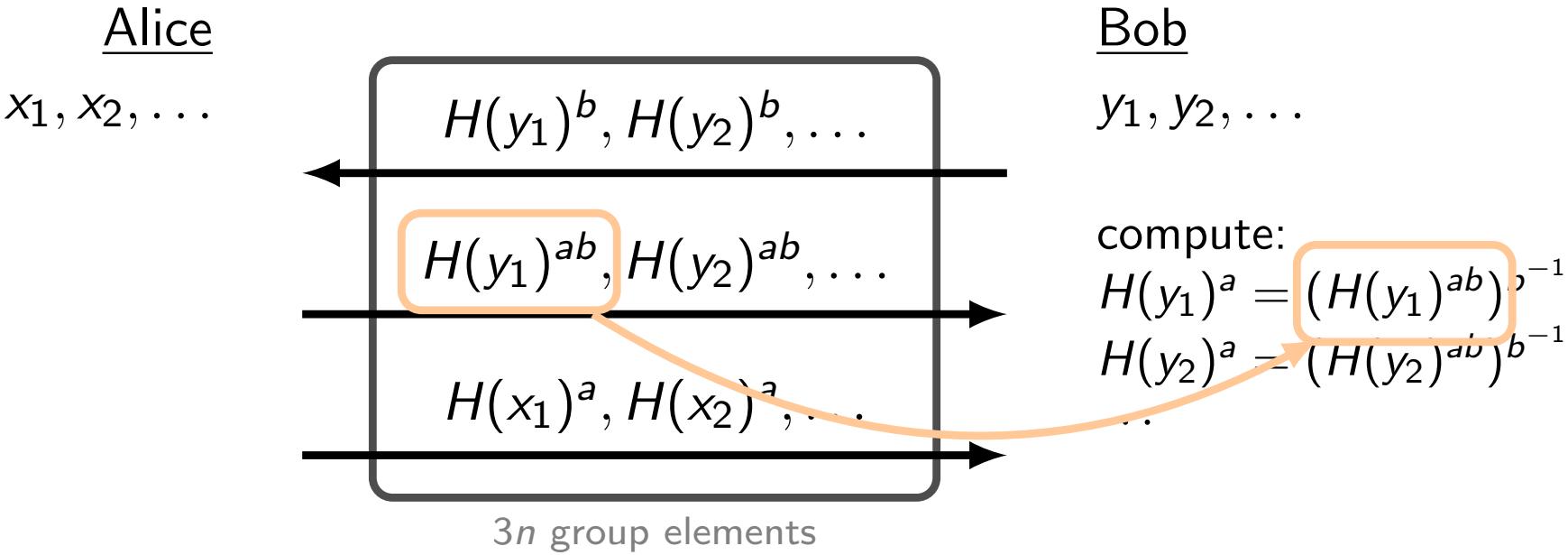
Compact and Malicious Private Set Intersection for Small Sets; (CCS'21).

DH-based PSI Protocol



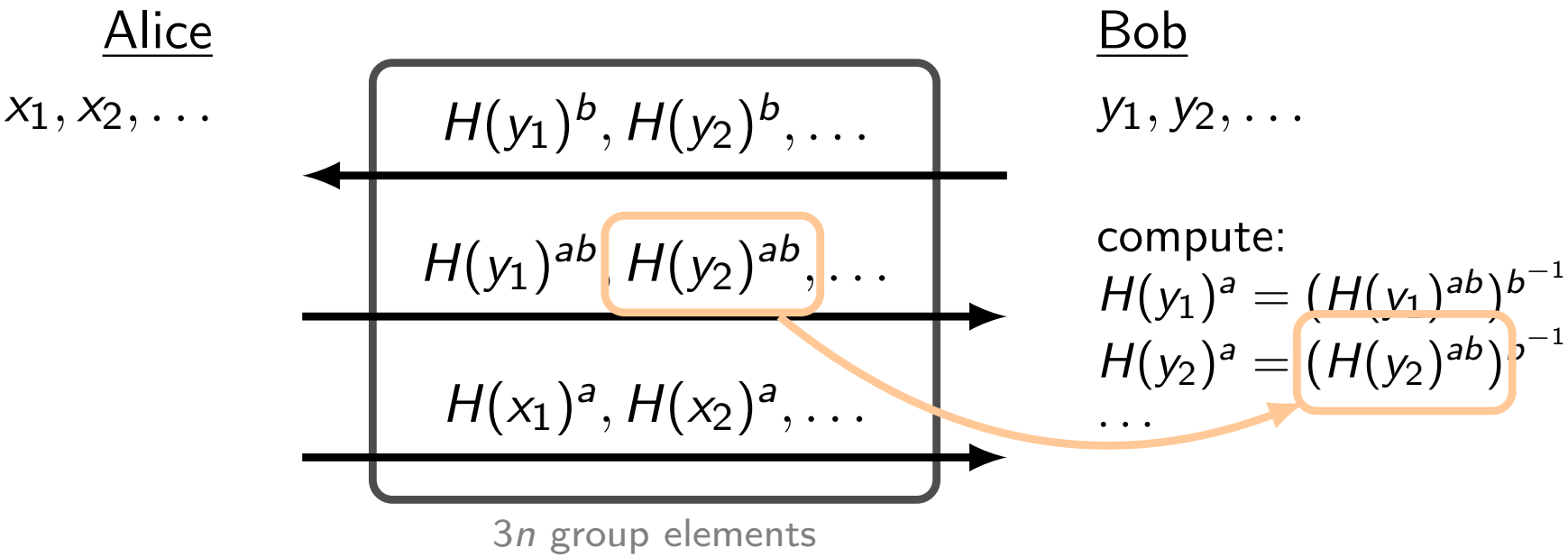
how could you possibly reduce communication?

DH-based PSI Protocol



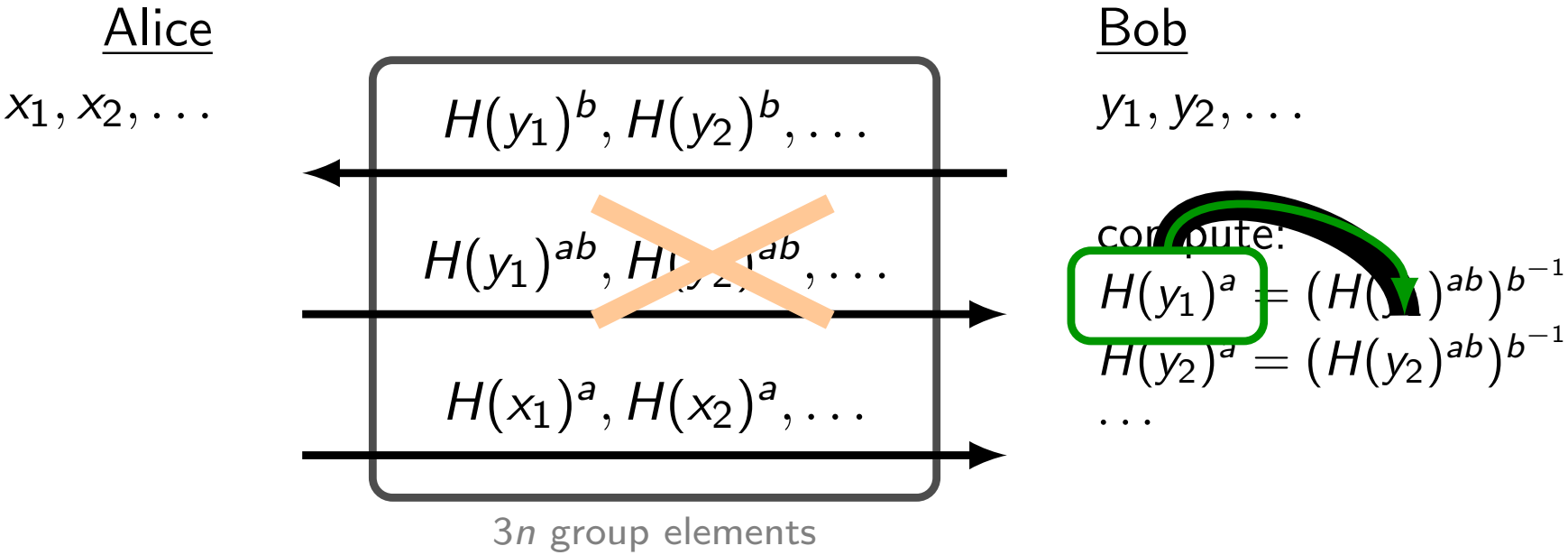
how could you possibly reduce communication?

DH-based PSI Protocol



how could you possibly reduce communication?

DH-based PSI Protocol

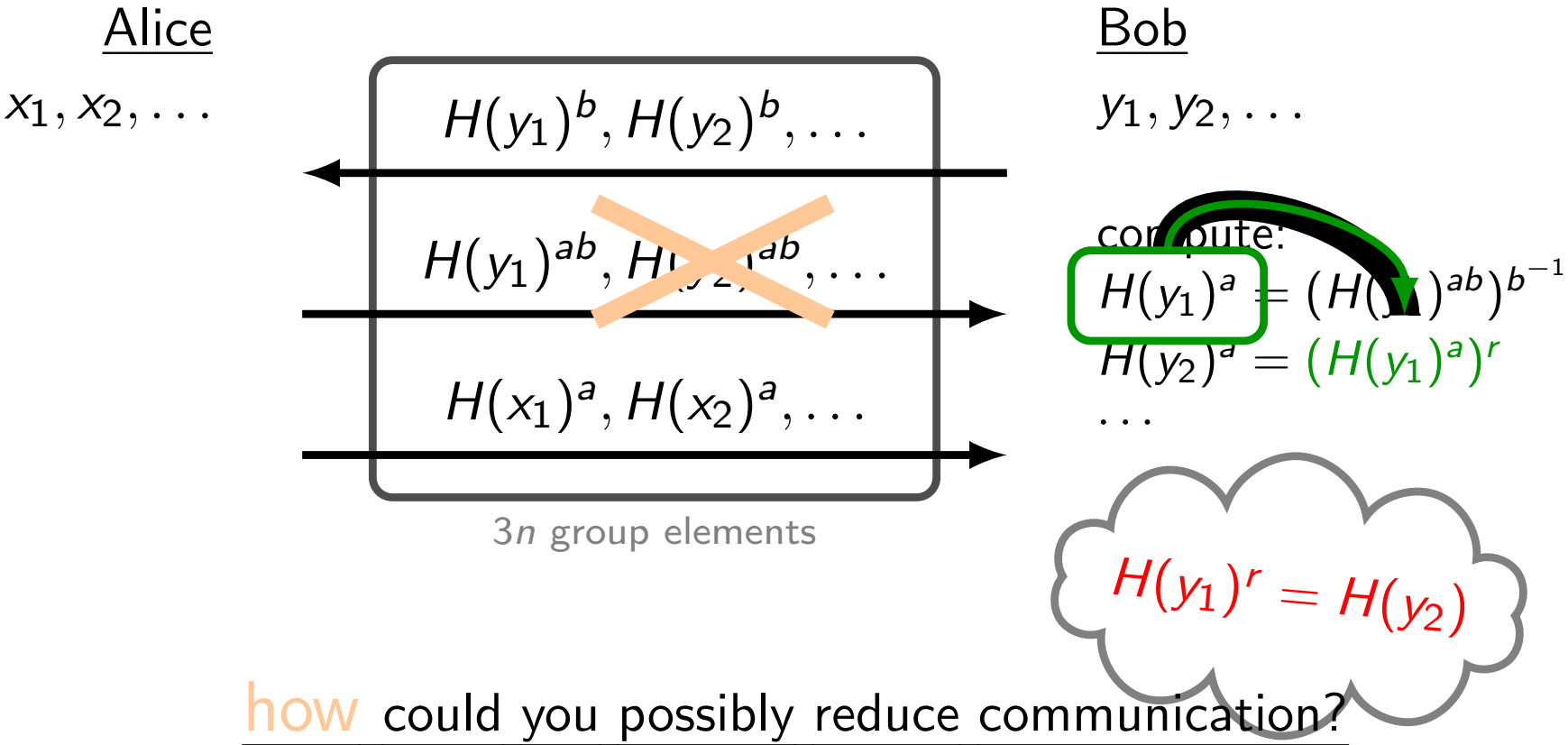


how could you possibly reduce communication?

replace random oracle with some "trapdoored" function

. . . where Bob knows dlog relationships between outputs

DH-based PSI Protocol

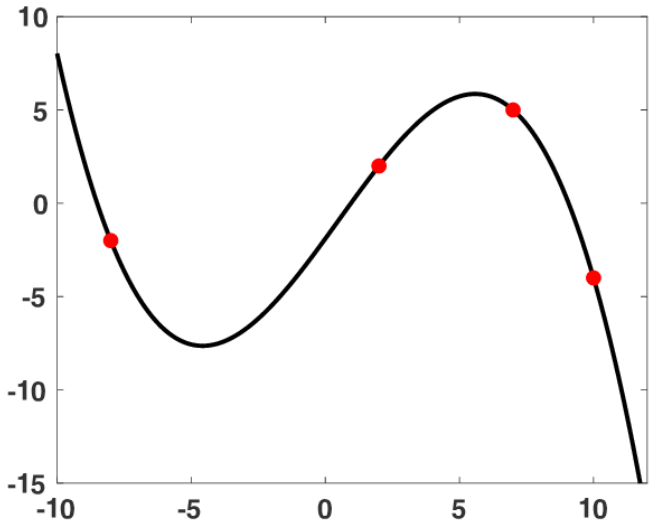


how could you possibly reduce communication?

replace random oracle with some "trapdoored" function

. . . where Bob knows dlog relationships between outputs

our approach:



polynomial interpolation!

Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

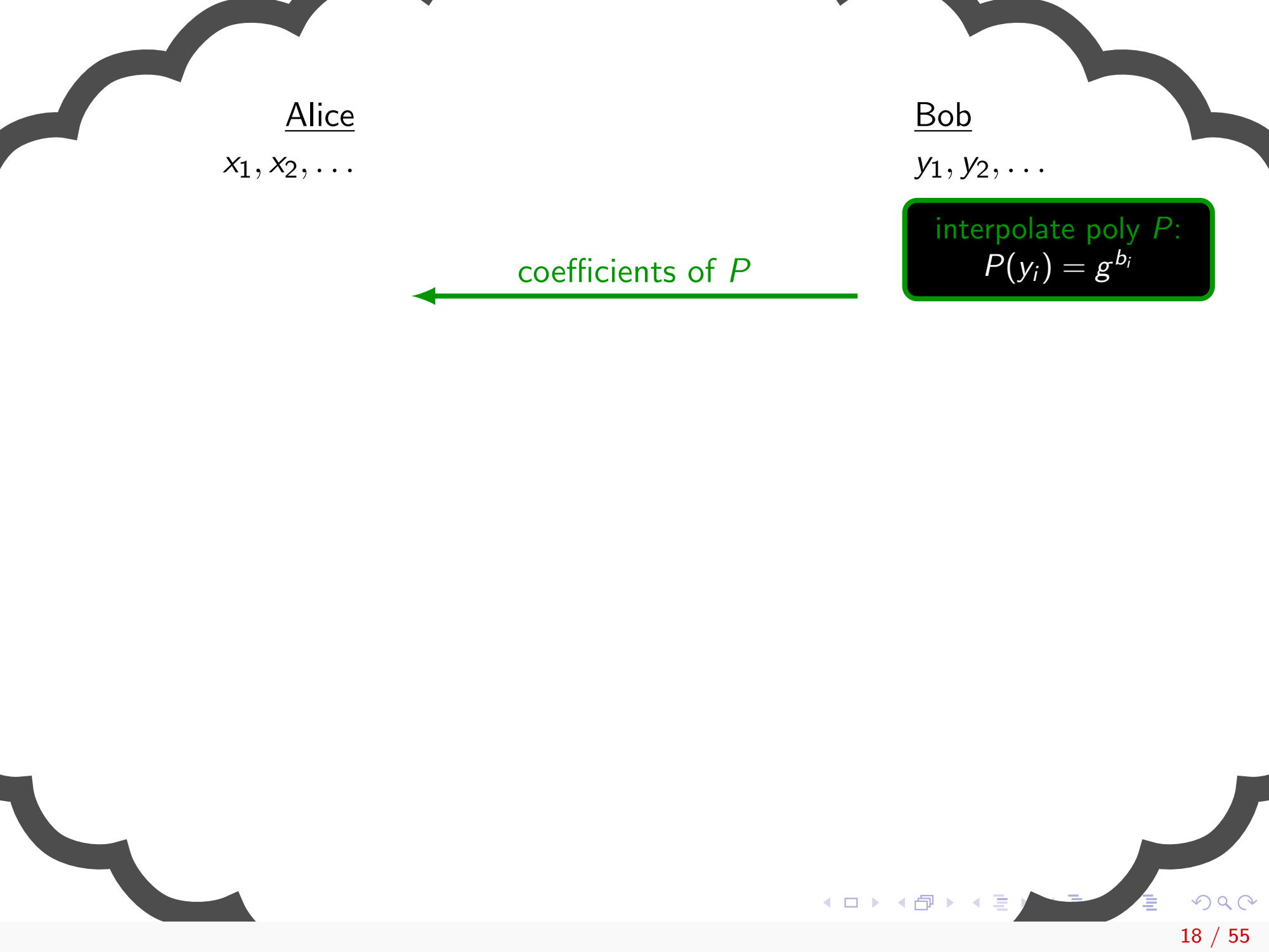
Alice

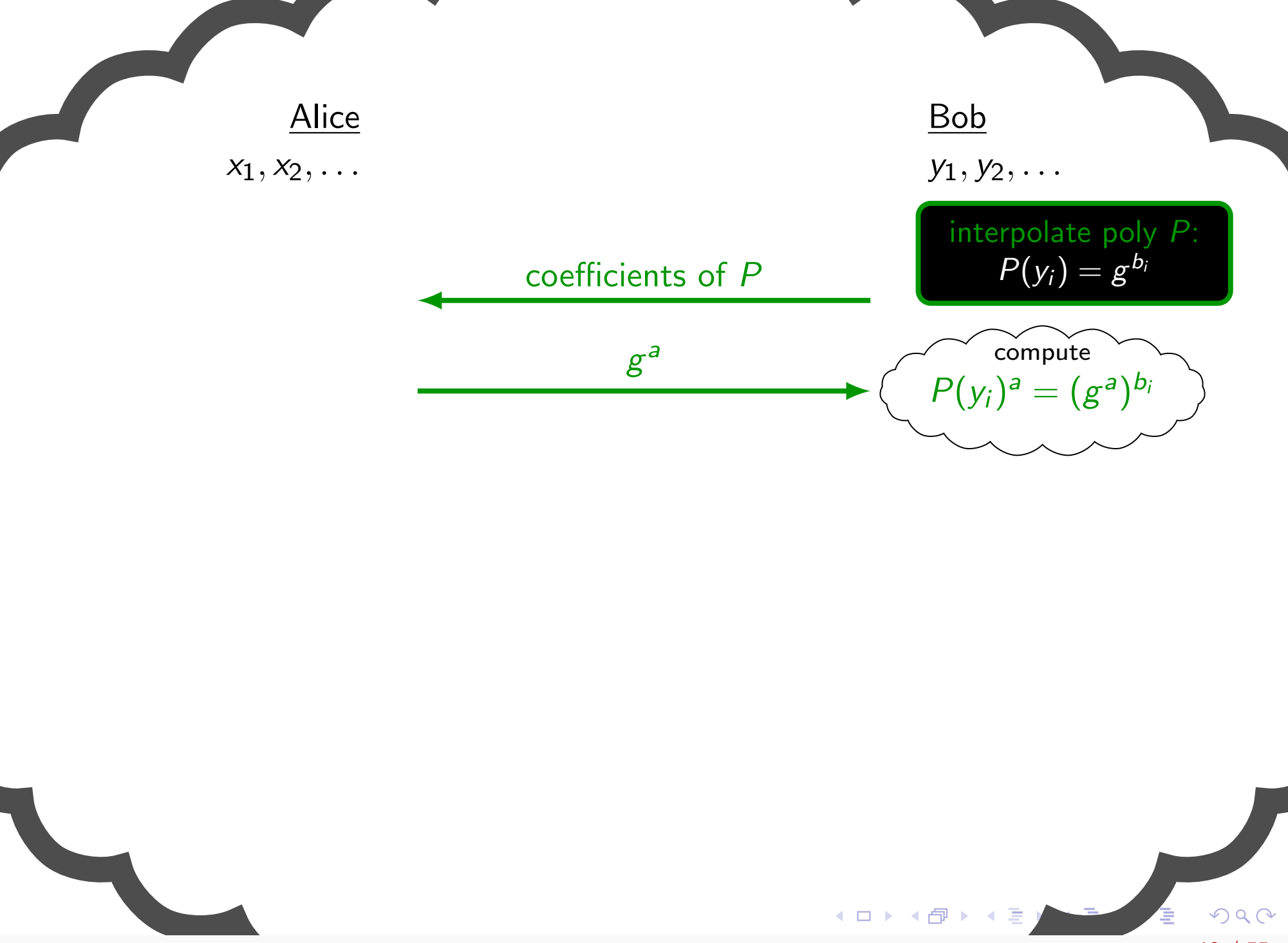
$x_1, x_2, \ldots$

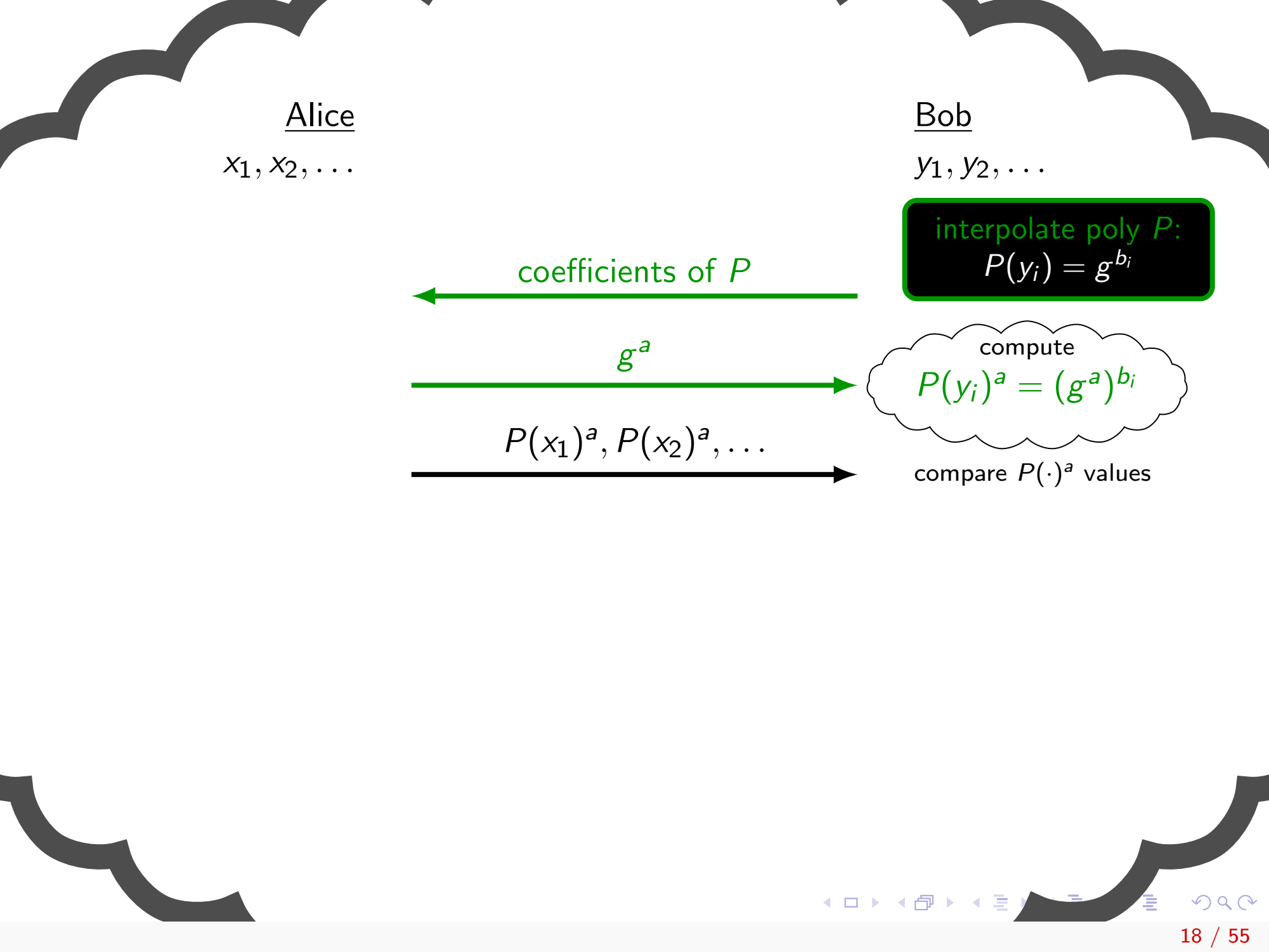
Bob

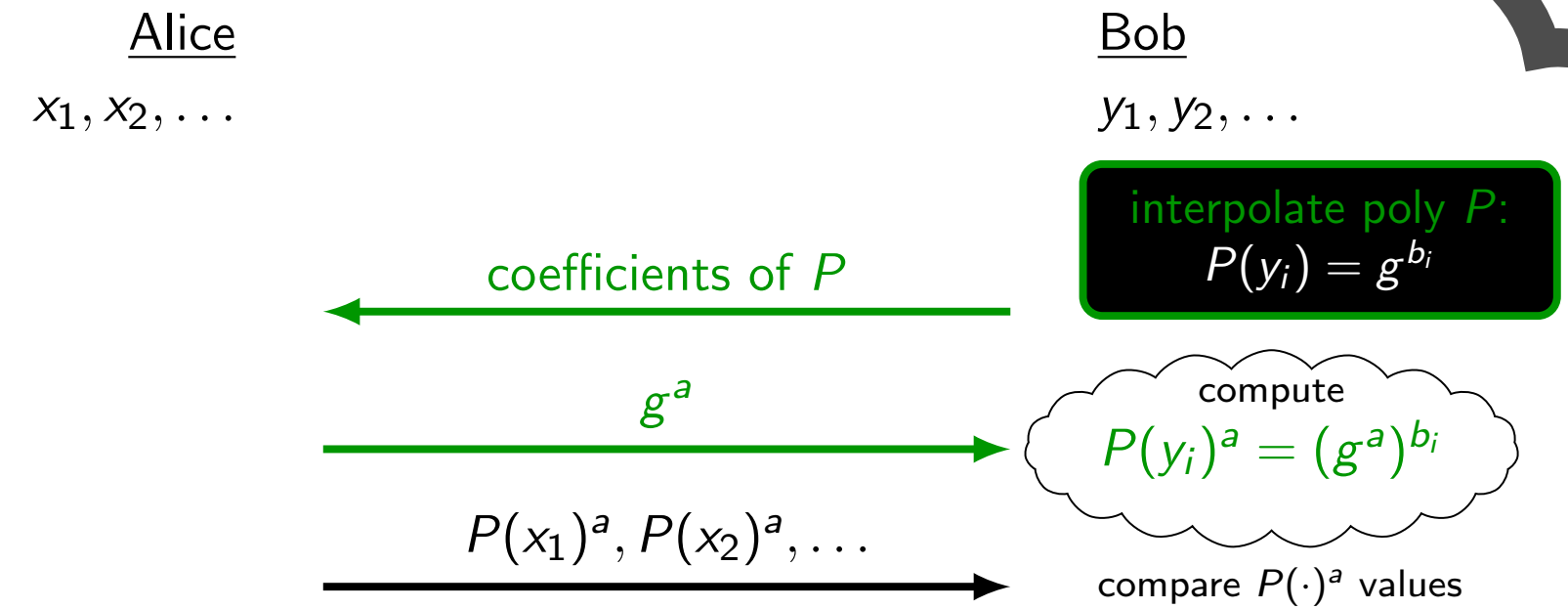
$y_1, y_2, \ldots$

interpolate poly P:
$P(y_i) = g^{b_i}$

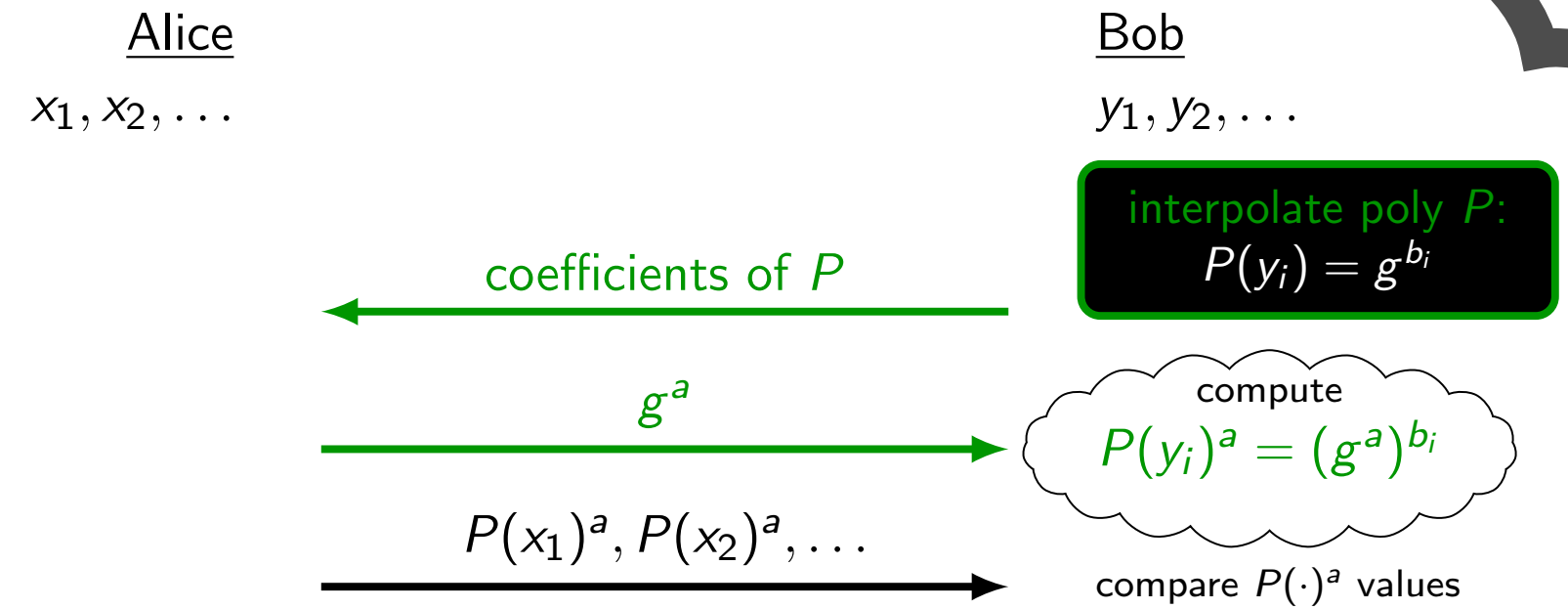

Alice
$x_1, x_2, \ldots$
Bob
$y_1, y_2, \ldots$
interpolate poly P:
$P(y_i) = g^{b_i}$
coefficients of P


Alice
$x_1, x_2, \ldots$
Bob
$y_1, y_2, \ldots$
interpolate poly P:
$P(y_i) = g^{b_i}$
coefficients of P
g^a
compute
$P(y_i)^a = (g^a)^{b_i}$


Alice
$x_1, x_2, \ldots$
Bob
$y_1, y_2, \ldots$
interpolate poly P:
$P(y_i) = g^{b_i}$
coefficients of P
g^a
compute
$P(y_i)^a = (g^a)^{b_i}$
$P(x_1)^a, P(x_2)^a, \ldots$
compare $P(\cdot)^a$ values

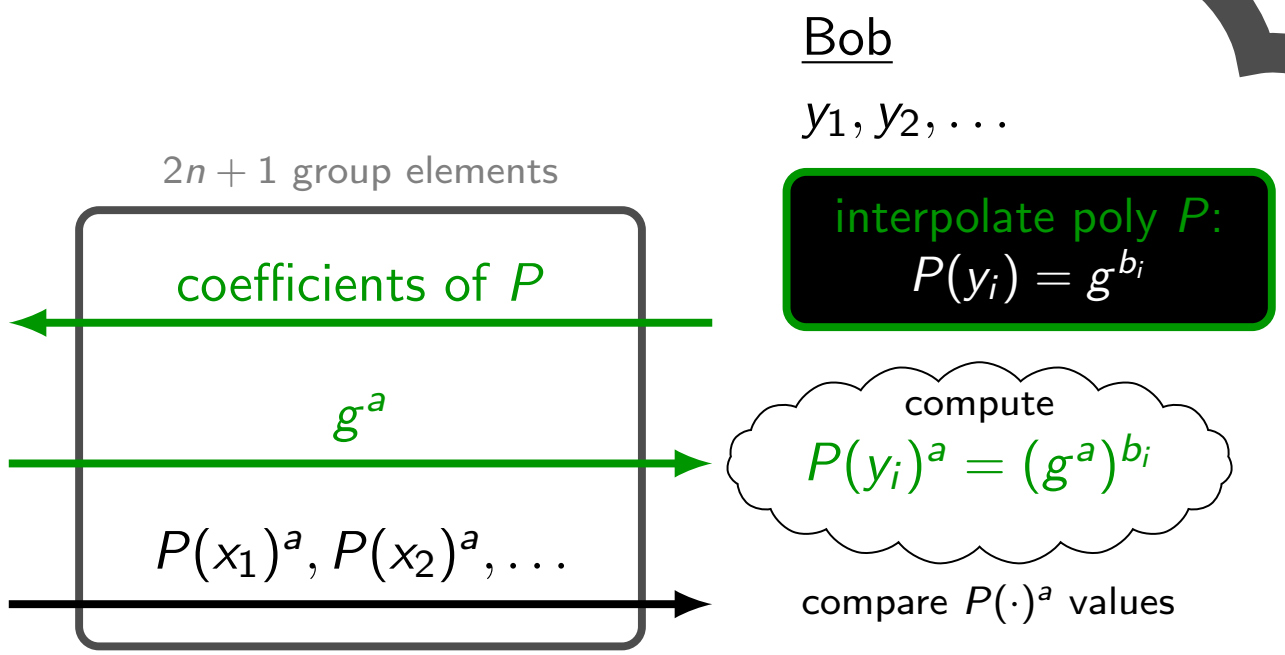


correctness: Bob knows dlog of $P(y)$ for programmed points ✓

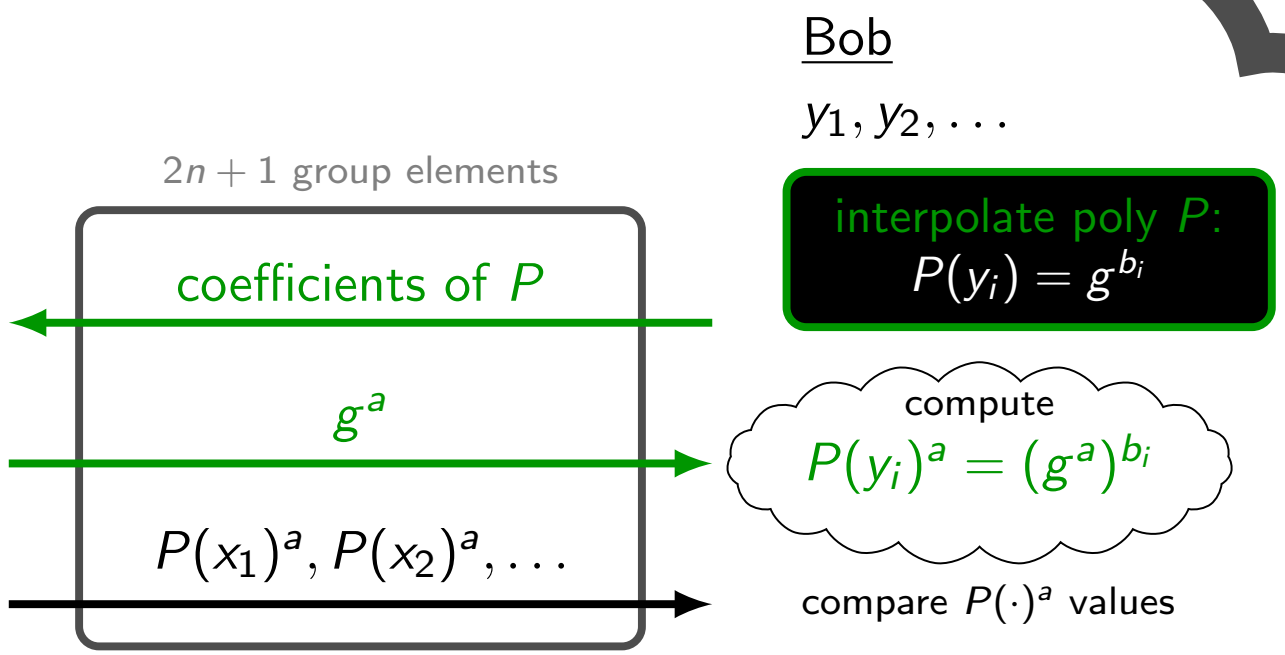


correctness: Bob knows dlog of $P(y)$ for programmed points ✓

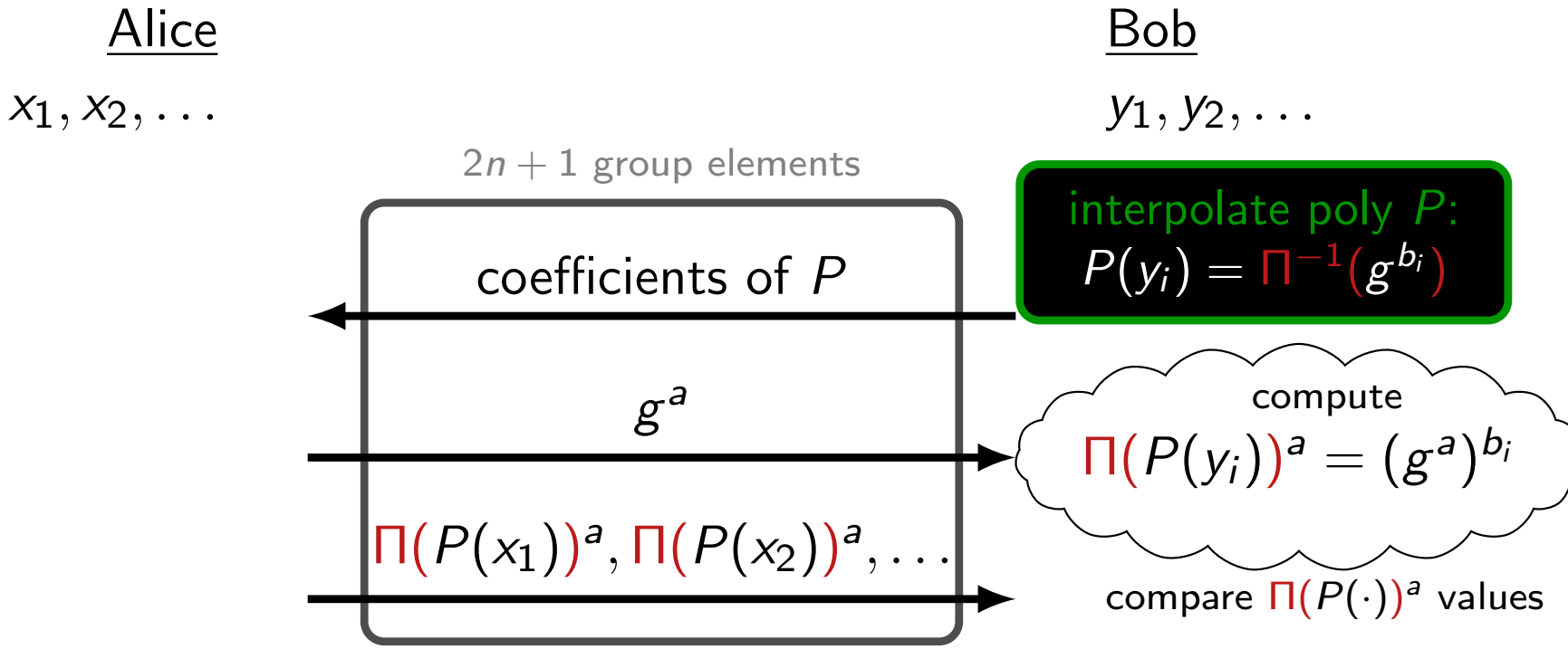
obliviousness: P reveals nothing about programmed points ✓



correctness:	Bob knows dlog of $P(y)$ for programmed points ✓
obliviousness:	P reveals nothing about programmed points ✓
efficiency:	$\lvert$description of $P\rvert = n$ group elements ✓

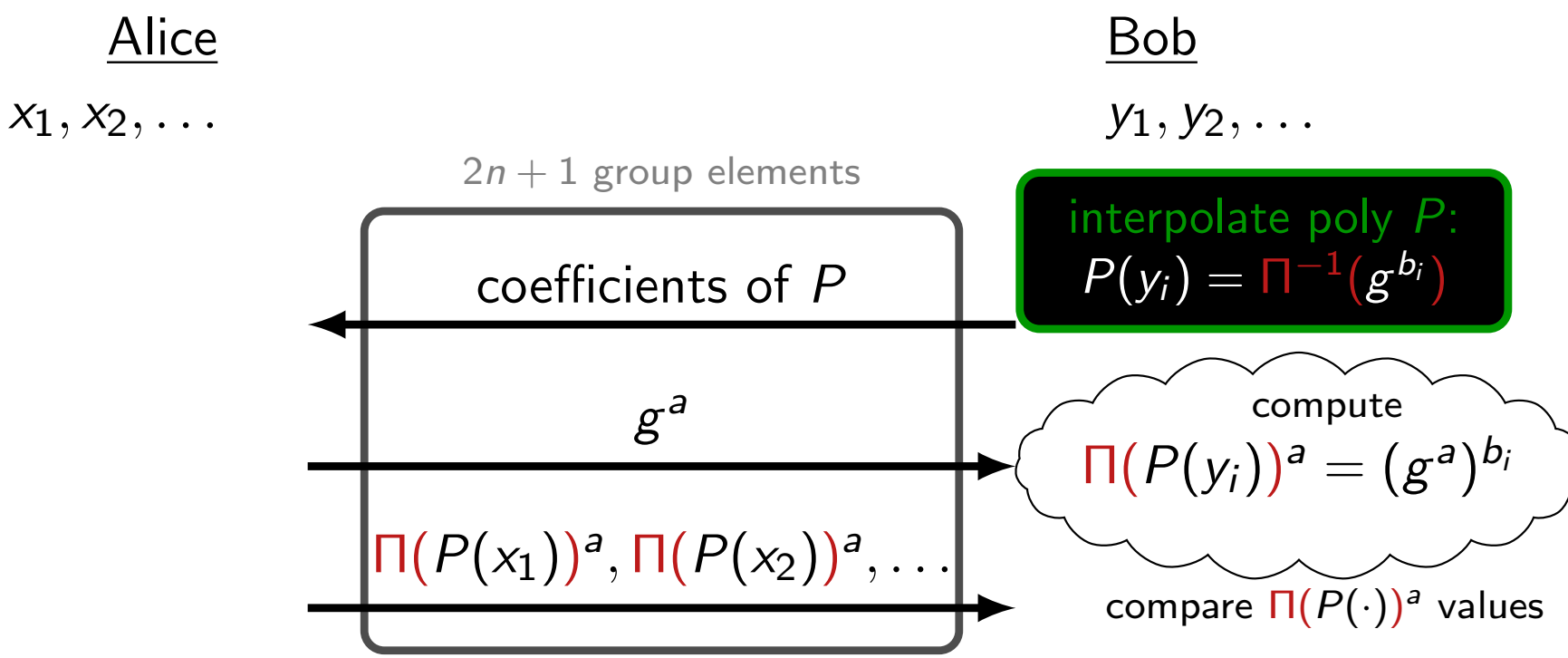


correctness:	Bob knows dlog of $P(y)$ for programmed points ✓
obliviousness:	P reveals nothing about programmed points ✓
efficiency:	$\lvert$description of $P\rvert = n$ group elements ✓
$P(\cdot)^a$ is PRF:	Bob **cannot know** dlog of any other $P(x)$
	$\Rightarrow$ **Ideal Permutation**



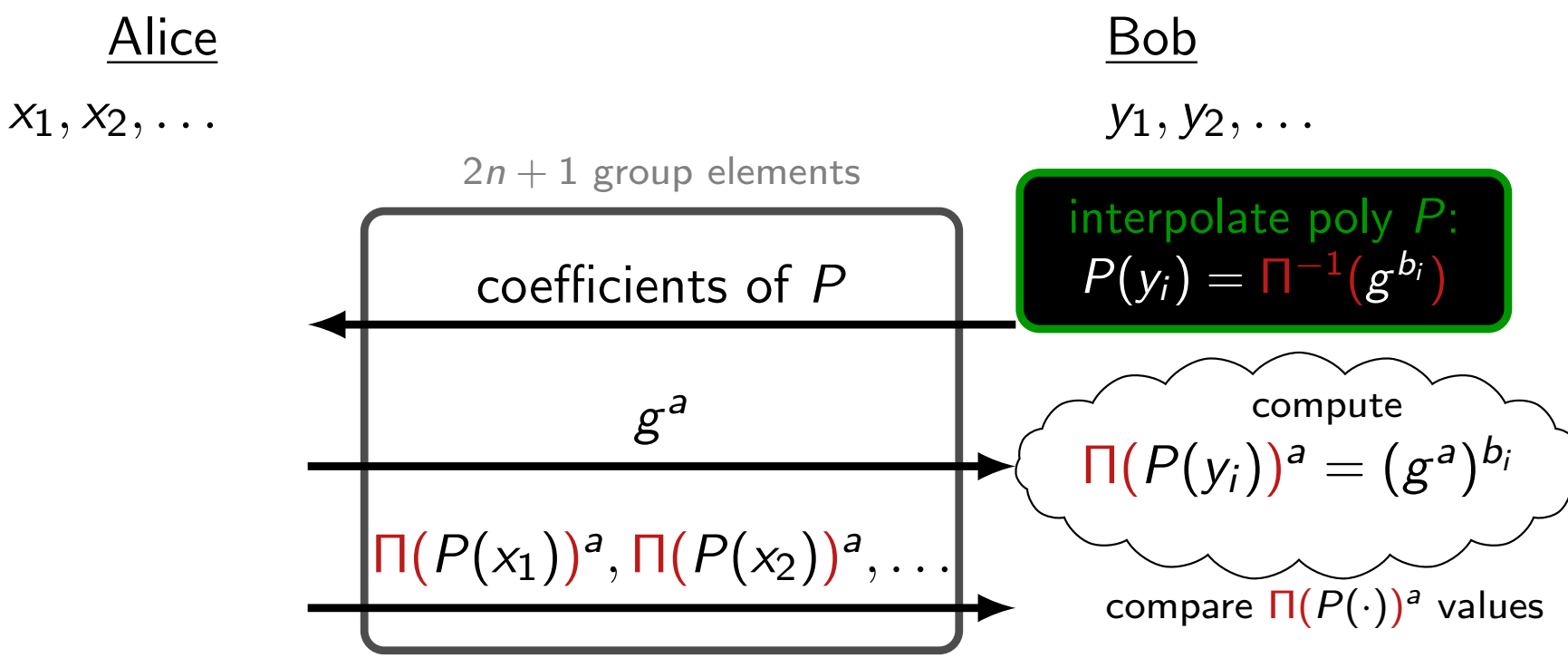
semi-honest: Alice's group elements can be truncated

malicious: a few more strategic RO calls



semi-honest: Alice's group elements can be truncated

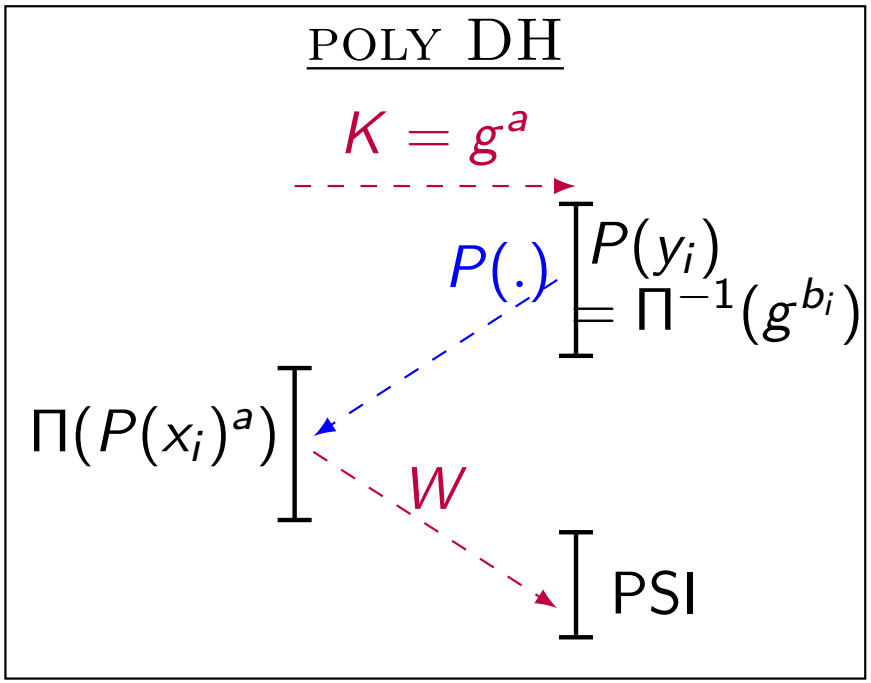
malicious: a few more strategic RO calls



semi-honest: Alice's group elements can be truncated

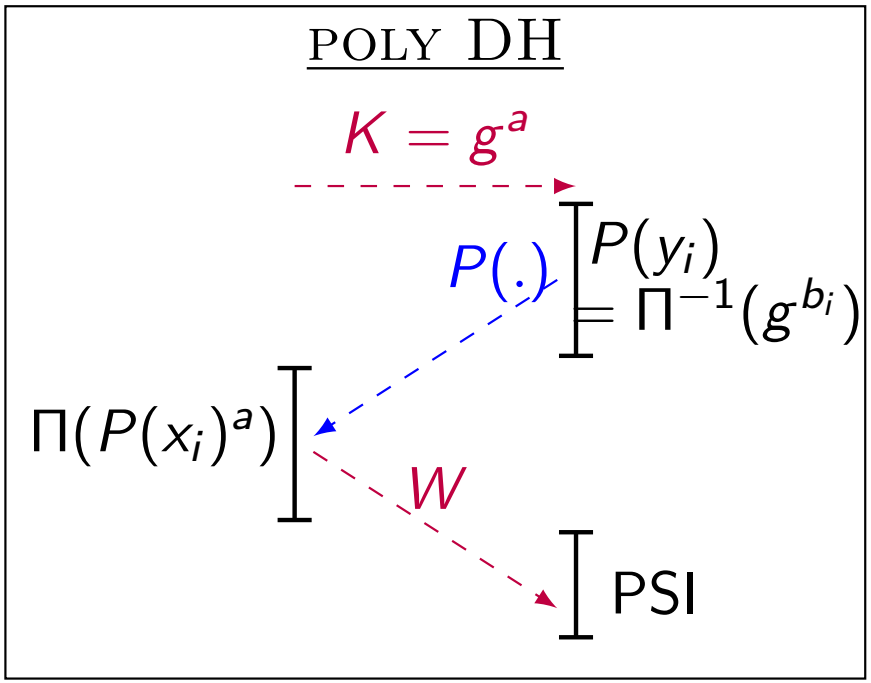
malicious: a few more strategic RO calls

Implementation of Poly-based PSI Protocol



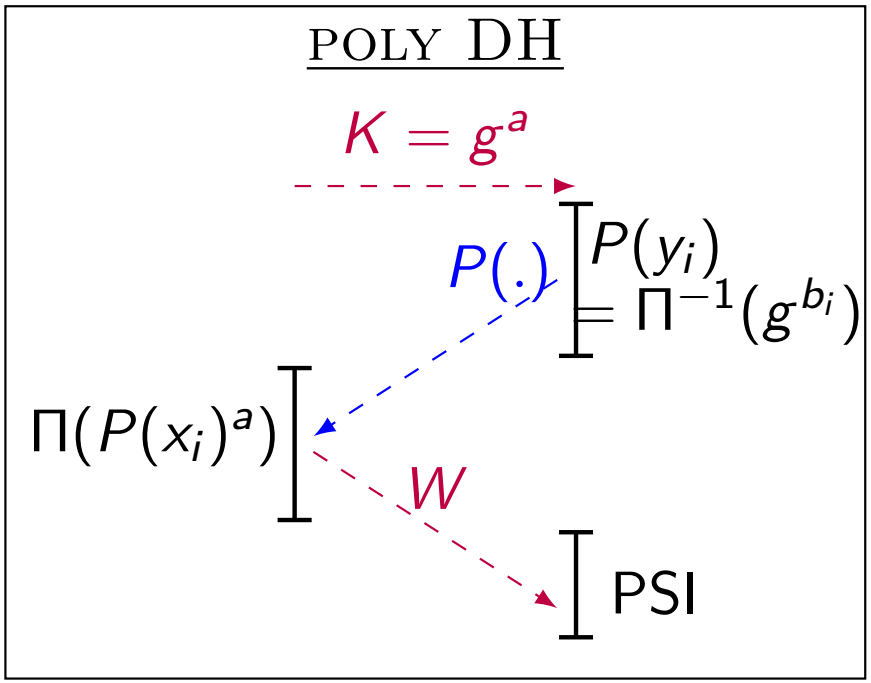
- Preprocessing: compute $(g^a)^{b_i}$
- Apply the Hidden Subset Sum (HSS) technique to derive g^{b_i}
- Use faster interpolation and multi-point evaluation with $O(n \log^2 n)$ computational complexity

Implementation of Poly-based PSI Protocol



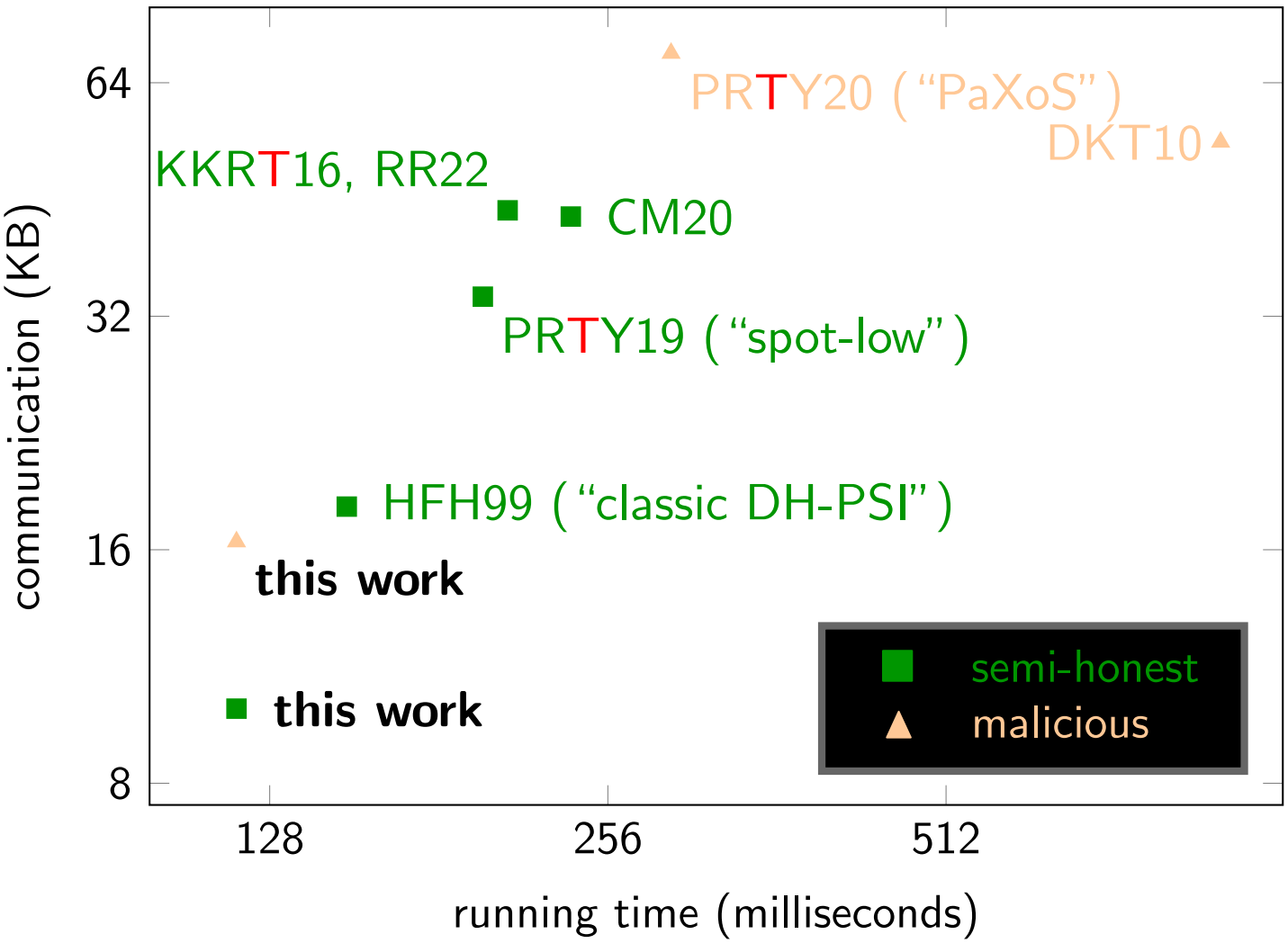
- Preprocessing: compute $(g^a)^{b_i}$
- Apply the Hidden Subset Sum (HSS) technique to derive g^{b_i}
- Use faster interpolation and multi-point evaluation with $O(n \log^2 n)$ computational complexity

Implementation of Poly-based PSI Protocol



- Preprocessing: compute $(g^a)^{b_i}$
- Apply the Hidden Subset Sum (HSS) technique to derive g^{b_i}
- Use faster interpolation and multi-point evaluation with $O(n \log^2 n)$ computational complexity

PSI cost: 256 items per party



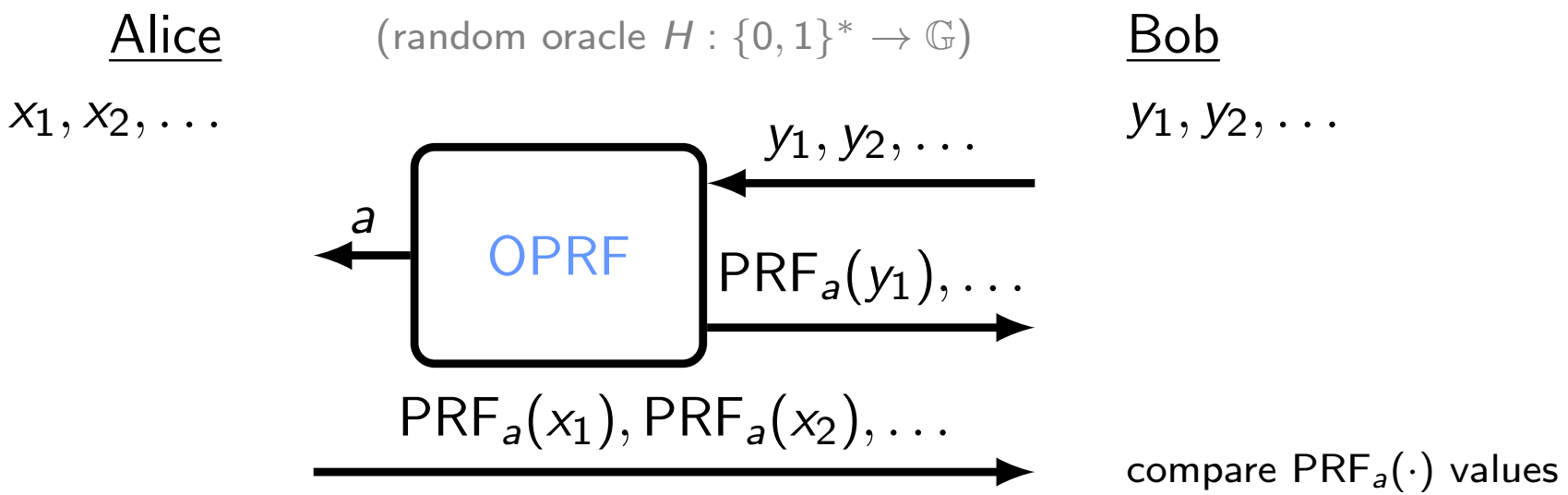
OPRF for PSI

- ~~Small-set PSI: built on public-key cryptography~~
- Large-set PSI: built on symmetric-key cryptography

PSI for Large Sets: State of the Art

- Scalable Private Set Intersection Based on OT Extension, ACM TOPS'18
- Oblivious Key-Value Stores and Amplification for Private Set Intersection, CRYPTO'21
- Blazing Fast PSI from Improved OKVS and Subfield VOLE, CCS'22
- ...

PSI Framework



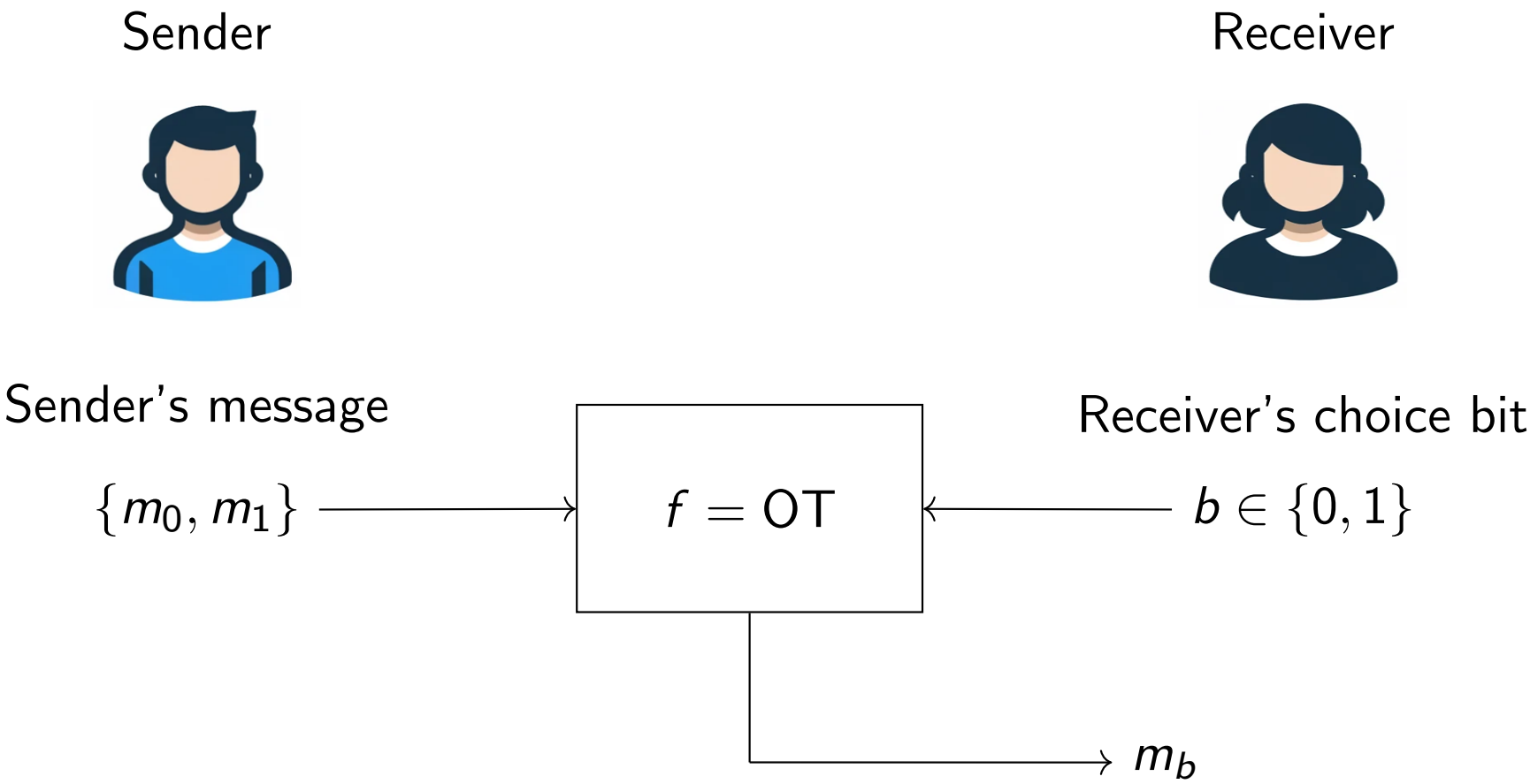
Semi-honest security:

- first two messages are an oblivious PRF protocol
- standard OPRF→PSI paradigm [FreedmanIshaiPinkasReingold05]

OPRF

How can we implement OPRF primarily with symmetric-key cryptography?

Building Block: 1-out-of-2 OT Functionality



$\Rightarrow$ OT ensures that the receiver learns only m_b, while m_{1-b} and b remain hidden from the receiver and sender, respectively.

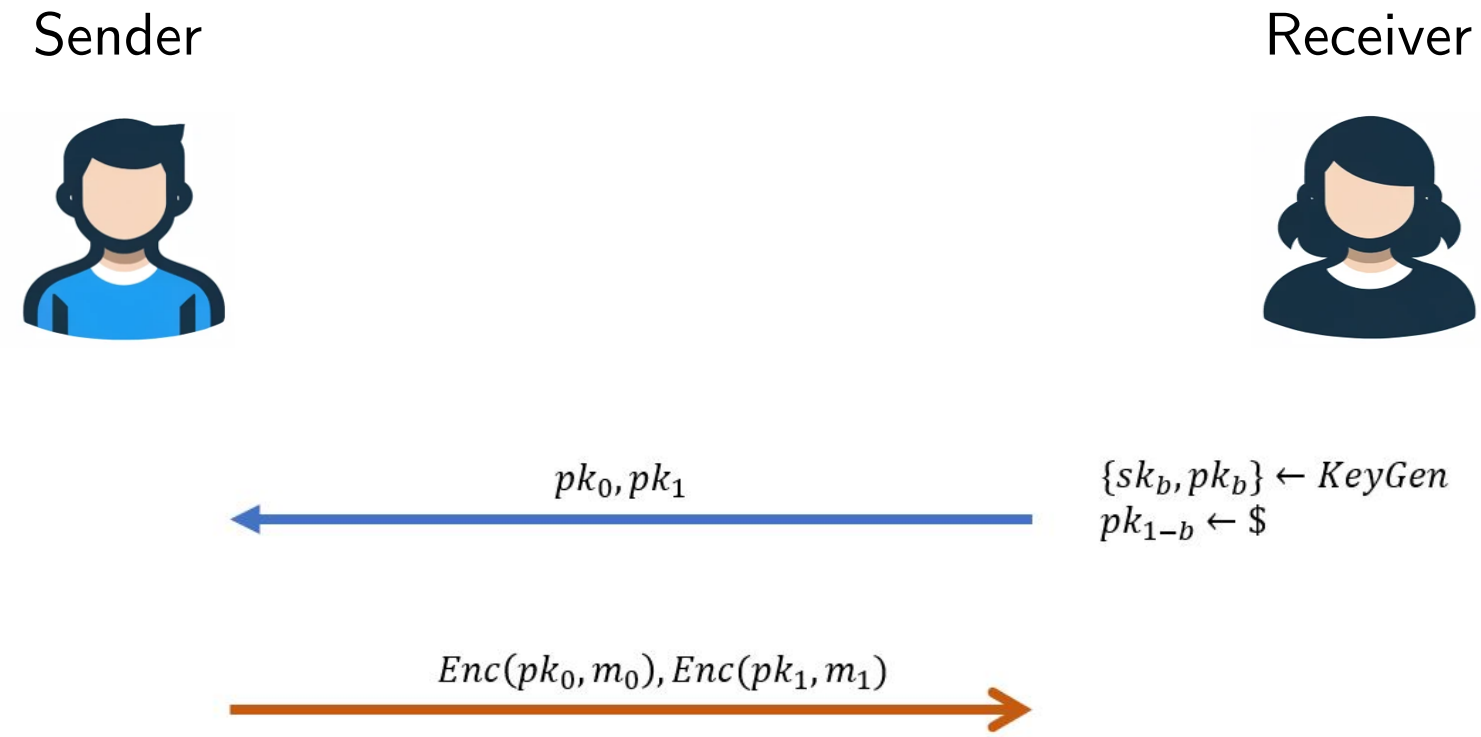
Paradigm Solution: OPRF vs 1-out-of-n OT

- Sender computes PRF values for all possible inputs: $\mathrm{PRF}_a(y)$
- Receiver obtains only the PRF outputs corresponding to their chosen inputs: $\mathrm{PRF}_a(y_i)$

1-out-of-"Infinite" OT

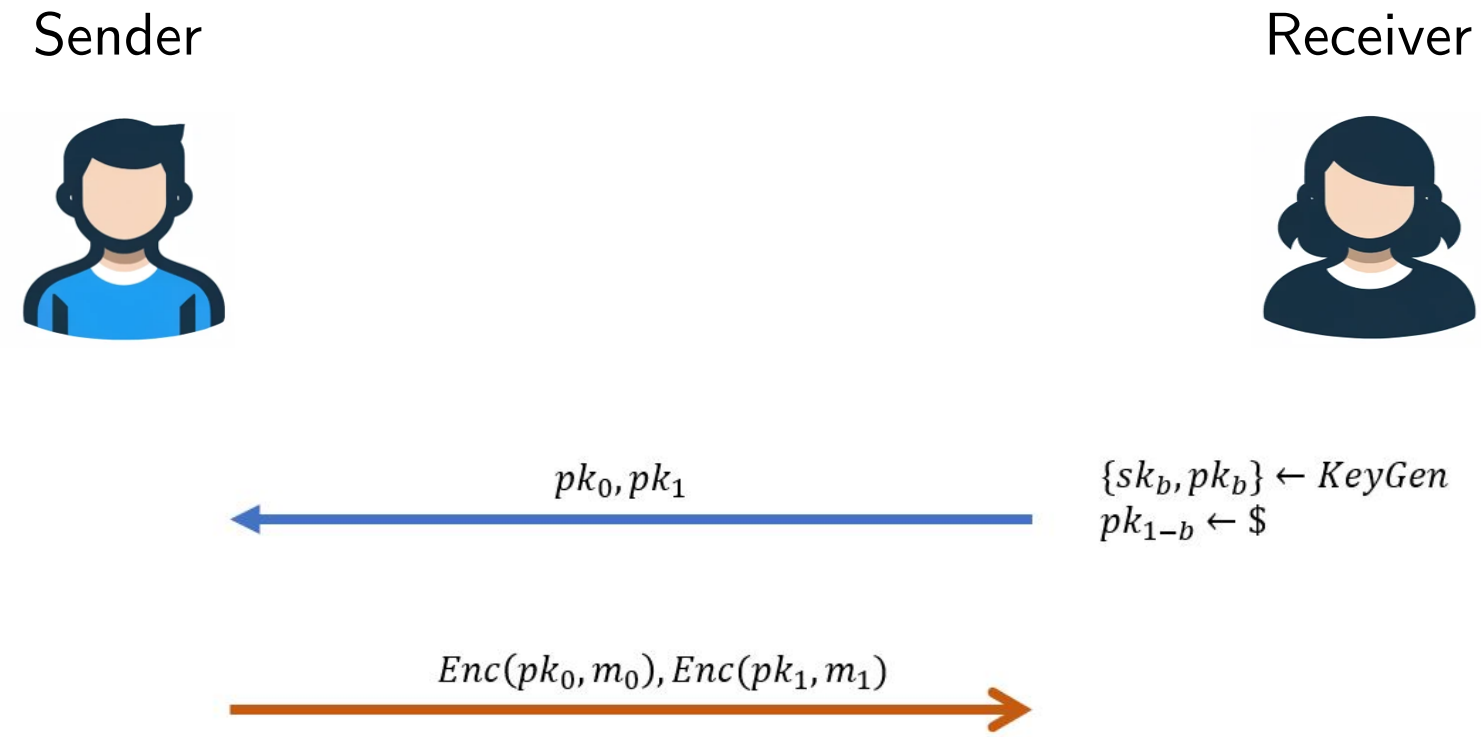
How can we implement OT primarily with symmetric-key cryptography?

A Simple 1-out-of-2 OT Protocol



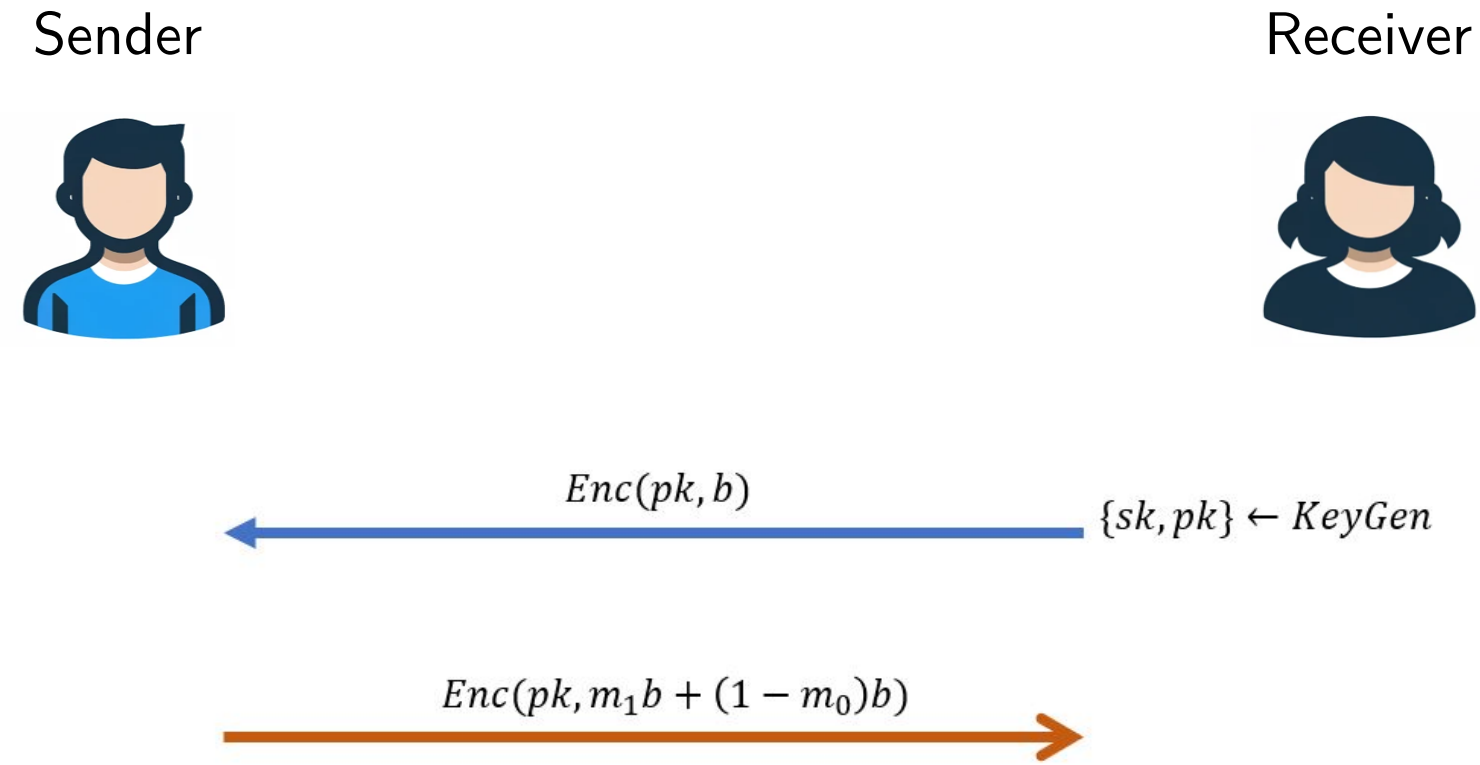
Generate a public key without knowing the corresponding secret key (e.g., in ElGamal, sample a group element without knowing its discrete logarithm). Any issue?

A Simple 1-out-of-2 OT Protocol



Generate a public key without knowing the corresponding secret key (e.g., in ElGamal, sample a group element without knowing its discrete logarithm). Any issue?

A 1-out-of-2 OT Protocol based on HE



Homomorphic Encryption (HE):

- $Enc(\mathrm{pk}, x) + Enc(\mathrm{pk}, y) = Enc(\mathrm{pk}, x + y)$.
- $Enc(\mathrm{pk}, x) \times Enc(\mathrm{pk}, y) = Enc(\mathrm{pk}, x \times y)$.

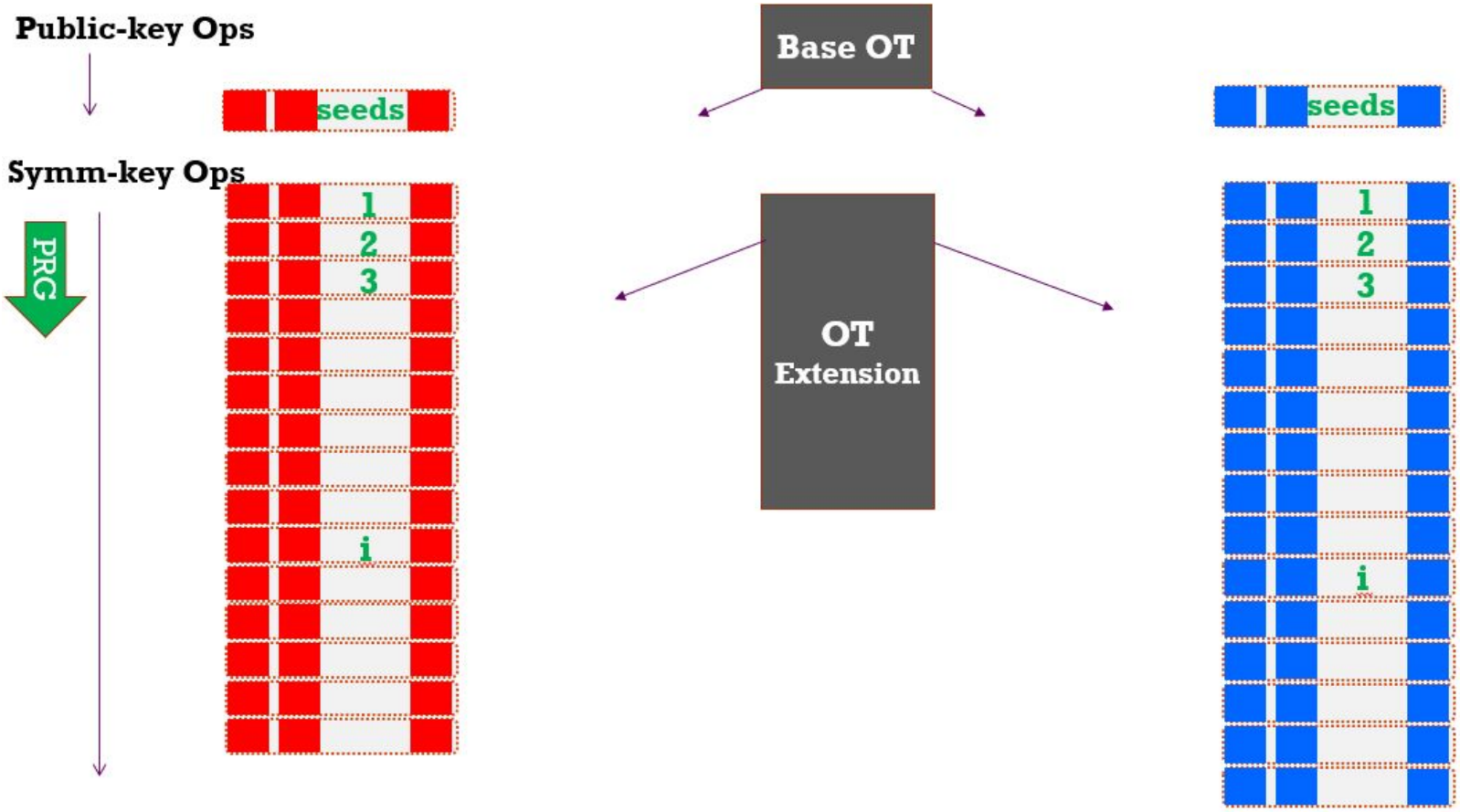
State of the Art in OT

- Challenges: Public-key based OT is expensive.
- Idea: use a few OTs with public-key crypto **(Base OT)** + symmetric-key techniques $\Rightarrow$ OT extension [IKNP03].

State of the Art in OT

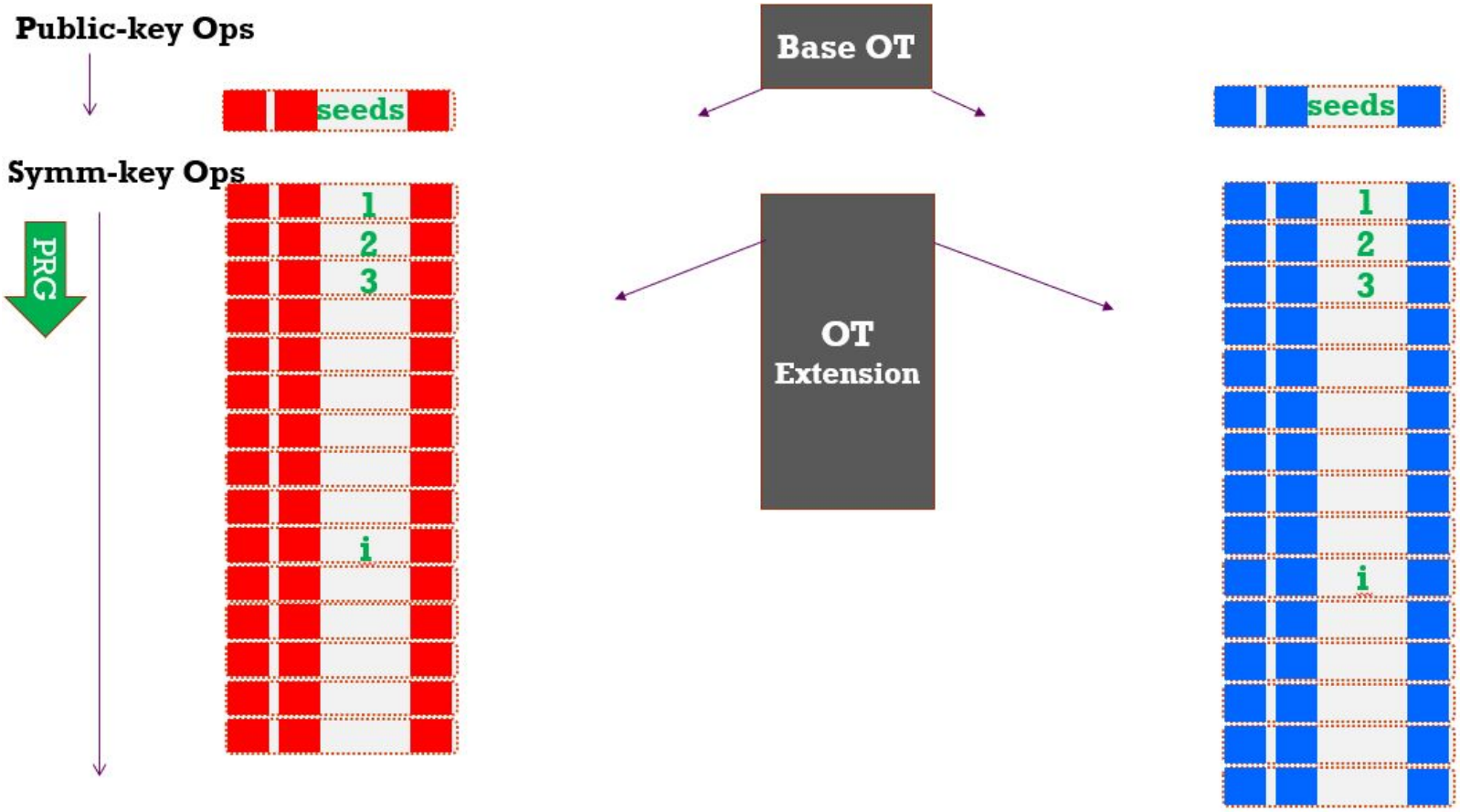
- Challenges: Public-key based OT is expensive.
- Idea: use a few OTs with public-key crypto **(Base OT)** + symmetric-key techniques $\Rightarrow$ OT extension [IKNP03].

Implementation: 1-out-of-n OT



- Preprocessing: Base OT
- Use AES with a fixed key for symmetric-key operations, making many PRG calls

Implementation: 1-out-of-n OT



- Preprocessing: Base OT
- Use AES with a fixed key for symmetric-key operations, making many PRG calls

State of the Art in OT

- **Traditional OT:**
  - Based on public-key cryptography (e.g., Elgamal)
  - Expensive when many OTs are needed
- **OT Extension:**
  - Extends a small number of base OTs to millions of OTs
  - Efficient, symmetric-key based after setup [IKNP03], [ALSZ13],[KK13], [KKRT16],...
- **Sparse OT:**
  - The selection vector is very sparse [PRTY19], ...
  - Reduces computation and communication
- **VOLE-based OT:**
  - Vector Oblivious Linear Evaluation (VOLE) [Roy22], [RR22]
  - Enables OT with low communication and fast amortization

State of the Art in OT

- **Traditional OT:**
  - Based on public-key cryptography (e.g., Elgamal)
  - Expensive when many OTs are needed
- **OT Extension:**
  - Extends a small number of base OTs to millions of OTs
  - Efficient, symmetric-key based after setup [IKNP03], [ALSZ13],[KK13], [KKRT16],...
- **Sparse OT:**
  - The selection vector is very sparse [PRTY19], ...
  - Reduces computation and communication
- **VOLE-based OT:**
  - Vector Oblivious Linear Evaluation (VOLE) [Roy22], [RR22]
  - Enables OT with low communication and fast amortization

State of the Art in OT

- **Traditional OT:**
  - Based on public-key cryptography (e.g., Elgamal)
  - Expensive when many OTs are needed
- **OT Extension:**
  - Extends a small number of base OTs to millions of OTs
  - Efficient, symmetric-key based after setup [IKNP03], [ALSZ13],[KK13], [KKRT16],...
- **Sparse OT:**
  - The selection vector is very sparse [PRTY19], ...
  - Reduces computation and communication
- **VOLE-based OT:**
  - Vector Oblivious Linear Evaluation (VOLE) [Roy22], [RR22]
  - Enables OT with low communication and fast amortization

State of the Art in OT

- **Traditional OT:**
  - Based on public-key cryptography (e.g., Elgamal)
  - Expensive when many OTs are needed
- **OT Extension:**
  - Extends a small number of base OTs to millions of OTs
  - Efficient, symmetric-key based after setup [IKNP03], [ALSZ13],[KK13], [KKRT16],...
- **Sparse OT:**
  - The selection vector is very sparse [PRTY19], ...
  - Reduces computation and communication
- **VOLE-based OT:**
  - Vector Oblivious Linear Evaluation (VOLE) [Roy22], [RR22]
  - Enables OT with low communication and fast amortization

PSI for Large Sets: State of the Art

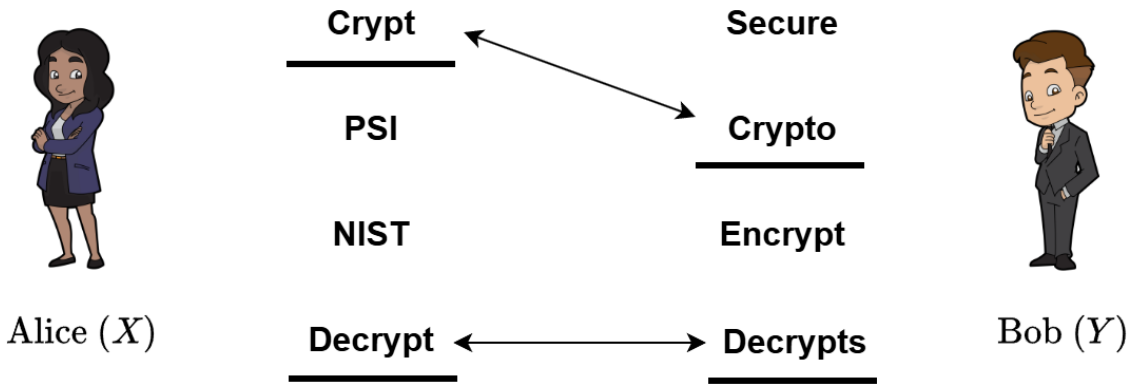
Protocol	Time (ms) 2^{16}	Time (ms) 2^{20}	Time (ms) 2^{24}	Comm. (bits/n) 2^{16}	Comm. (bits/n) 2^{20}	Comm. (bits/n) 2^{24}	Comm. asymptotic (bits) n_x,n_y
Semi-Honest PSI							
[KKRT16]	137	2,073	53,933	$984n$	$1008n$	$1032n$	$6\kappa n_x + 3(\lambda + \log(n_x n_y))n_y$
[PRTY20] ($w = 2$)	763	4,998	123,800	$1208n$	$1268n$	$1302n$	$9.3\kappa n_x + (\lambda + \log(n_x n_y))n_y$
[GPR+21] ($w = 3$, star)	180	2,268	–	$780n$	$788n$	$804n$	$5.6\kappa n_x + (\lambda + \log(n_x n_y))n_y$
[RS21] ($w = 2$)	499	4,580	113,994	$914n$	$426n$	$398n$	$2^{24} n_x^{0.05} + 307 n_x + 40 n_y + \log(n_x n_y) n_y$
Ours (fast)	**51**	**369**	**6,987**	$241n$	$251n$	$260n$	$1.3\kappa n_x + (\lambda + \log(n_x n_y))n_y + 2^{14.5}\kappa$
Ours (fast, MT)	**49**	**163**	**3,145**				
Ours (SD)	63	1,353	27,681	**206n**	**180n**	**196n**	$1.2\log(n_x n_y)n_x + (\lambda + \log(n_x n_y))n_y + 2^{14.5}\kappa$
Ours (SD, MT)	61	1,107	25,325				
Malicious PSI							
[PRTY20] ($w = 2$)	769	5,196	126,294	$1766n$			$11.8\kappa n_x + 2\kappa n_y$
[GPR+21] ($w = 3$, star)	184	2,291	–	$1357n$			$8.6\kappa n_x + 2\kappa n_y$
[RS21] ($w = 2$)	556	5,228	132,951	$960n$	$474n$	$438n$	$2.4\kappa n_x + \kappa n_y + 2^{17}\kappa n_x^{0.05}$
Ours	**62**	**439**	**8,055**	**343n**	**302n**	**300n**	$1.3\kappa n_x + \kappa n_y + 2^{14.5}\kappa$
Ours (MT)	65	**222**	**3,984**				
Semi-Honest Circuit PSI							
[RS21] ($w = 2$, IKNP)	1,810	25,300	-	$21,888n$	$14,640n$	-	$O(n\kappa\ell)$
[RS21] ($w = 2$, SilentOT)	5,021	112,421	-	$2,701n$	$2,216n$	-	$O(n\ell)$
[CGS22] (IKNP+)	2,851	28,723	-	$8,371n$	$8,856n$	-	$O(n\kappa\ell)$
[CGS22] (Silver) –**Estimated**–	2,337	23,840	-	$8,371n$	$8,856n$	-	$O(n\ell)$
Ours ($w = 3$, Silver)	**1,112**	**15,557**	-	$931n$	$921n$	-	$O(n\ell)$

Taken from [RR22]

Outline

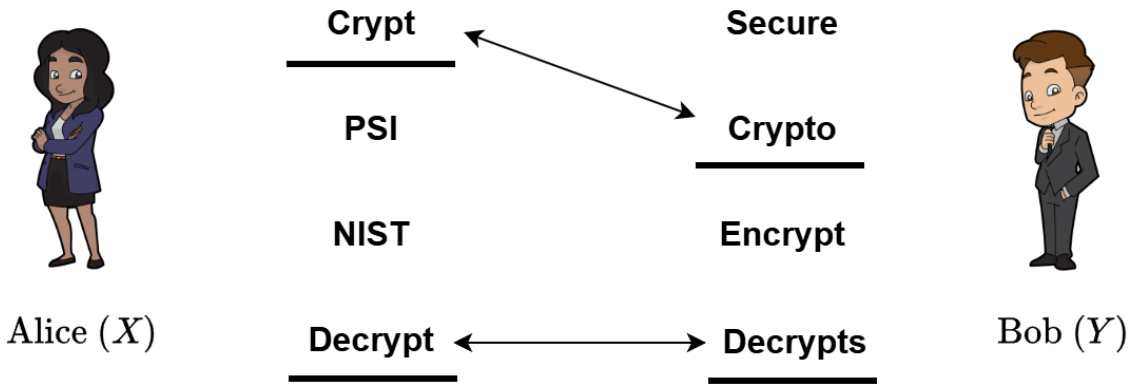
- ~~Introduction to PSI~~
- ~~Traditional PSI Protocol~~
- ~~PSI for Small Sets~~
- ~~PSI for Large Sets~~
- Fuzzy PSI
- Take-Home Messages

Fuzzy Private Set Intersection (Fuzzy PSI)



- Datasets may be noisy or contain inaccuracies.
- Alice learns the subset of Bob's elements that are "close enough" to her own elements.
- Formally: Alice learns $\{y \in Y \mid \text{dist}(x, y) \leq \delta \text{ for some } x \in X\}$, according to a chosen distance metric dist.

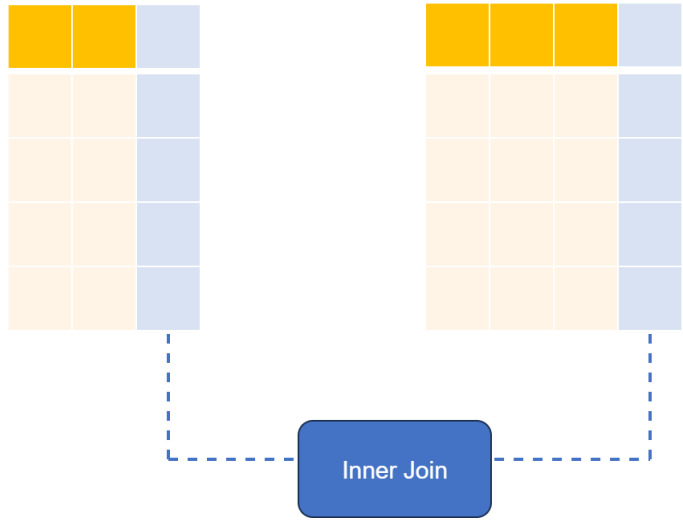
Fuzzy Private Set Intersection (Fuzzy PSI)



- Datasets may be noisy or contain inaccuracies.
- Alice learns the subset of Bob's elements that are "close enough" to her own elements.
- Formally: Alice learns $\{y \in Y \mid \text{dist}(x, y) \leq \delta \text{ for some } x \in X\}$, according to a chosen distance metric dist.

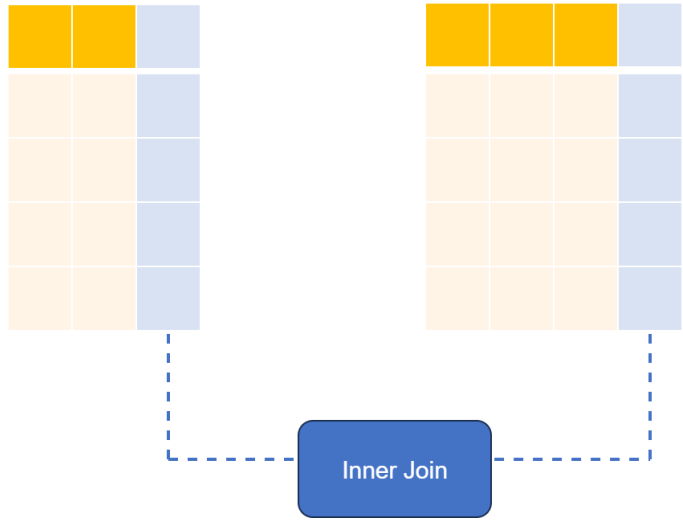
Some Privacy-Preserving Applications

- Checking for compromised credentials
- Matching biometric data
- Database fuzzy joins
- ...
- Any PSI scenario with noisy or measurement-based data



Some Privacy-Preserving Applications

- Checking for compromised credentials
- Matching biometric data
- Database fuzzy joins
- ...
- Any PSI scenario with noisy or measurement-based data



Fuzzy PSI: Previous Works

Function Secret Sharing (FSS)

- High computational cost
- Supports only limited distance metrics (L_1 and L_∞)
- References: [GRS22], [GRS23], [GGM24]

Garbled Circuits

- High concrete complexity
- Optimized for larger δ values
- Conceptually similar to our approach
- Reference: [RRX24]

Public-Key Based Approaches

- Poor scalability for large symmetric-sized input sets
- Can be efficiently extended to other PSI-like functionalities
- References: [BaaPu24], [GQLLW24]

$n = 2^{16}, \delta = 10, d = 2$ dimensions

Fuzzy PSI: Previous Works

Function Secret Sharing (FSS)

- High computational cost
- Supports only limited distance metrics (L_1 and L_∞)
- References: [GRS22], [GRS23], [GGM24]

Garbled Circuits

- High concrete complexity
- Optimized for larger δ values
- Conceptually similar to our approach
- Reference: [RRX24]

Public-Key Based Approaches

- Poor scalability for large symmetric-sized input sets
- Can be efficiently extended to other PSI-like functionalities
- References: [BaaPu24], [GQLLW24]

$n = 2^{16}, \delta = 10, d = 2$ dimensions

Fuzzy PSI: Previous Works

Function Secret Sharing (FSS)

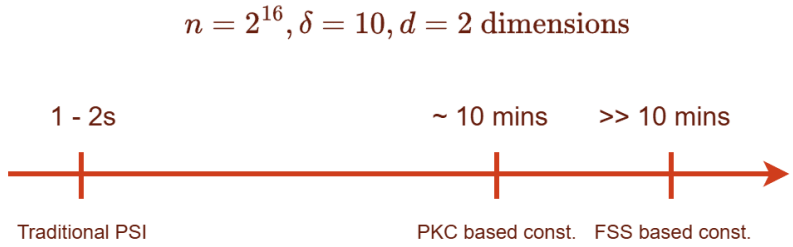
- High computational cost
- Supports only limited distance metrics (L_1 and L_∞)
- References: [GRS22], [GRS23], [GGM24]

Garbled Circuits

- High concrete complexity
- Optimized for larger δ values
- Conceptually similar to our approach
- Reference: [RRX24]

Public-Key Based Approaches

- Poor scalability for large symmetric-sized input sets
- Can be efficiently extended to other PSI-like functionalities
- References: [BaaPu24], [GQLLW24]

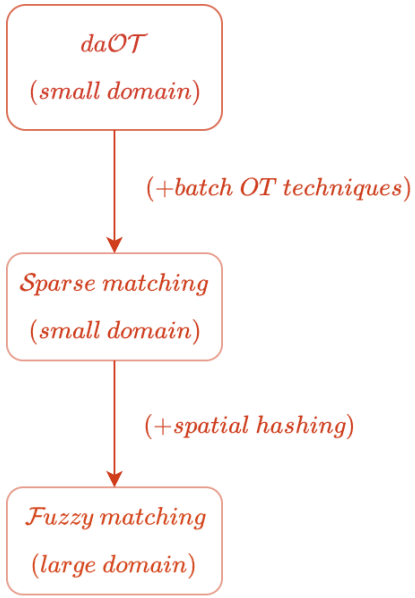


Fuzzy PSI: State of the Art

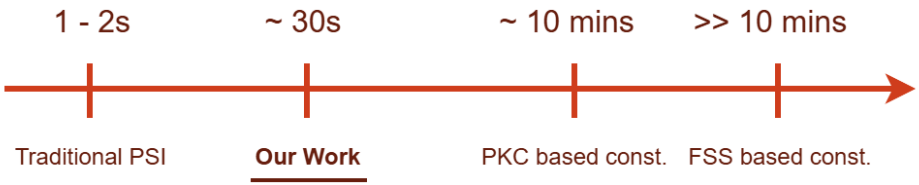
Distance-Aware OT with Application to Fuzzy PSI; (CCS'25).

Our Work

- **Goal**: close the efficiency gap between standard PSI and fuzzy PSI
- Propose a symmetric-key-based framework for fuzzy PSI
  - Modular design based on OT
  - Building blocks: distance-aware OT, sparse matching
  - Supports multiple distance metrics (e.g., L_1, L_2, L_∞)
  - Semi-honest security

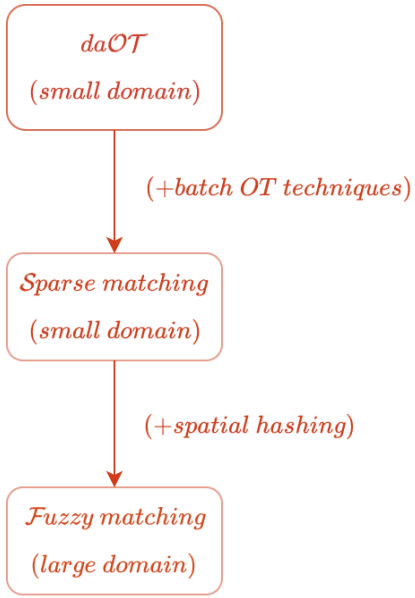


$n = 2^{16}, \delta = 10, d = 2$ dimensions

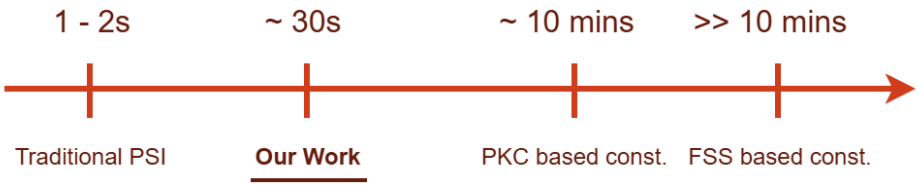


Our Work

- **Goal**: close the efficiency gap between standard PSI and fuzzy PSI
- Propose a symmetric-key-based framework for fuzzy PSI
  - Modular design based on OT
  - Building blocks: distance-aware OT, sparse matching
  - Supports multiple distance metrics (e.g., L_1, L_2, L_∞)
  - Semi-honest security

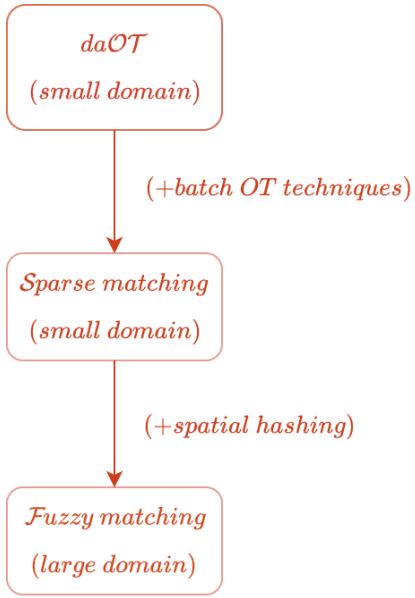


$n = 2^{16}, \delta = 10, d = 2$ dimensions

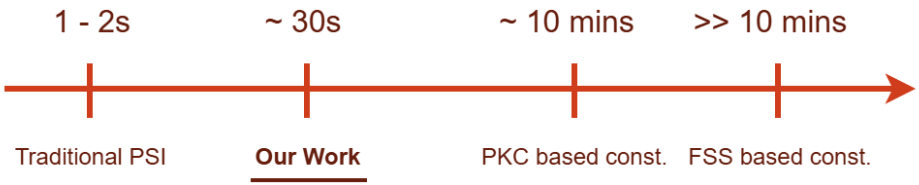


Our Work

- **Goal**: close the efficiency gap between standard PSI and fuzzy PSI
- Propose a symmetric-key-based framework for fuzzy PSI
  - Modular design based on OT
  - Building blocks: distance-aware OT, sparse matching
  - Supports multiple distance metrics (e.g., L_1, L_2, L_∞)
  - Semi-honest security

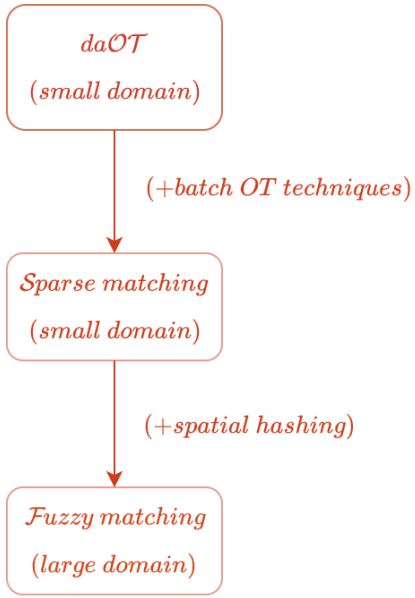


$n = 2^{16}, \delta = 10, d = 2$ dimensions

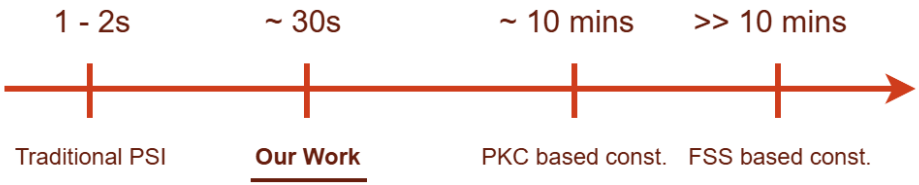


Our Work

- **Goal**: close the efficiency gap between standard PSI and fuzzy PSI
- Propose a symmetric-key-based framework for fuzzy PSI
  - Modular design based on OT
  - Building blocks: distance-aware OT, sparse matching
  - Supports multiple distance metrics (e.g., L_1, L_2, L_∞)
  - Semi-honest security

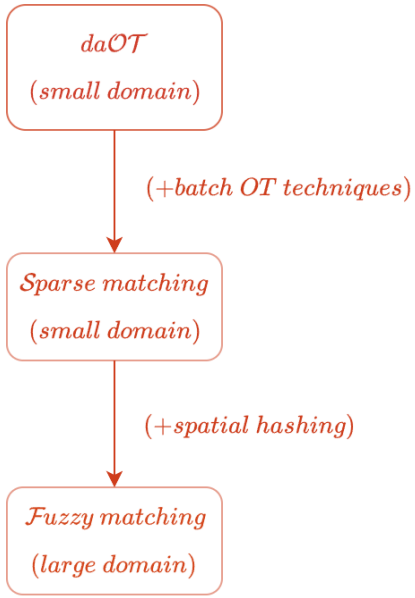


$n = 2^{16}, \delta = 10, d = 2$ dimensions

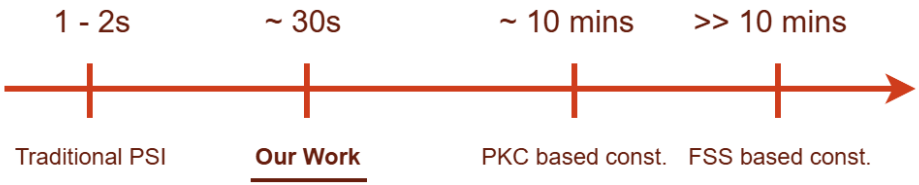


Our Work

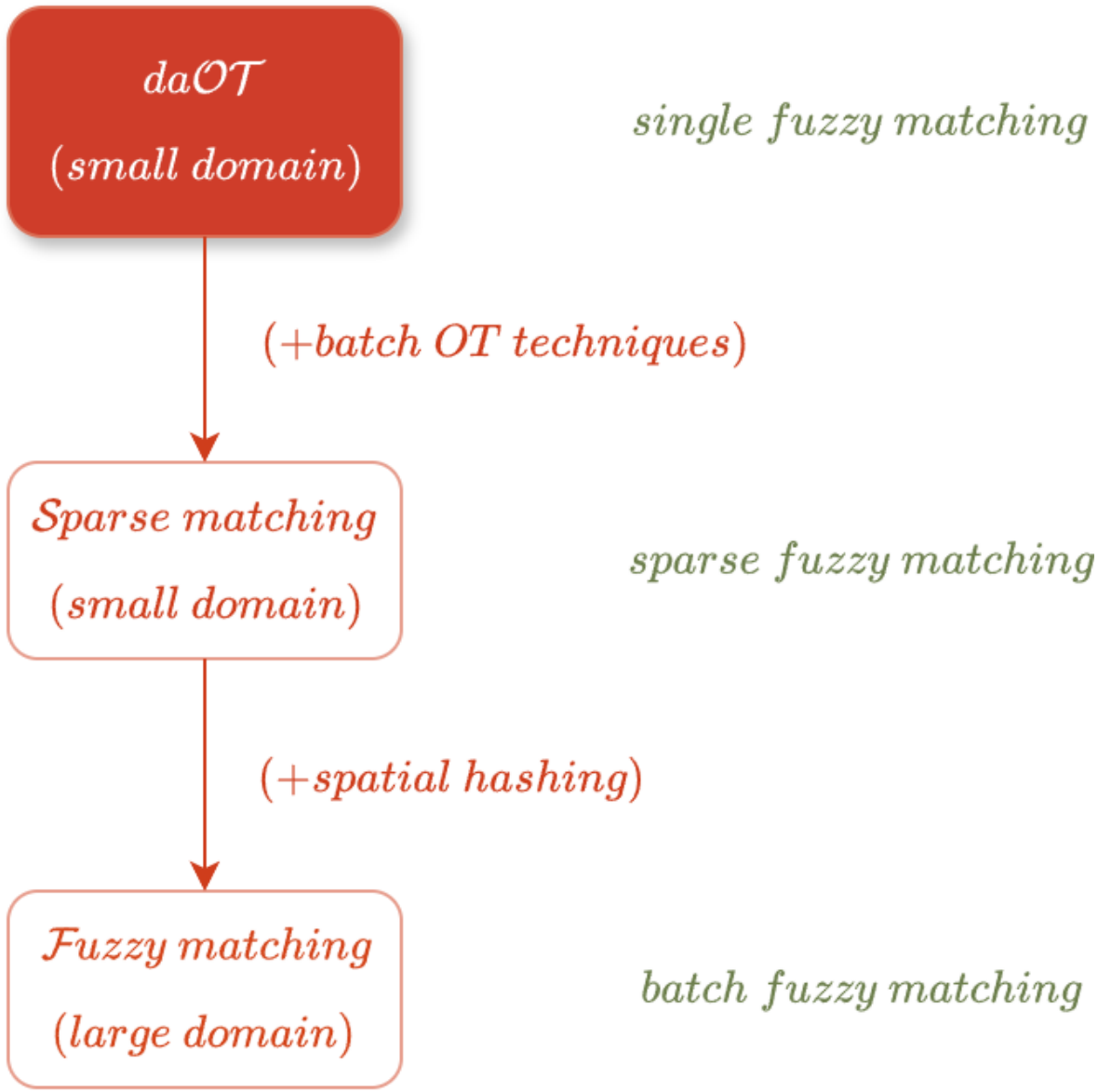
- **Goal**: close the efficiency gap between standard PSI and fuzzy PSI
- Propose a symmetric-key-based framework for fuzzy PSI
  - Modular design based on OT
  - Building blocks: distance-aware OT, sparse matching
  - Supports multiple distance metrics (e.g., L_1, L_2, L_∞)
  - Semi-honest security



$n = 2^{16}, \delta = 10, d = 2$ dimensions



Distance-Aware OT (daOT)



Distance-Aware OT (daOT)

$$a, b \in \mathcal{D}$$

$$r_0, r_1 \in \{0,1\}^\ell$$

$$c = \begin{cases} 0 \text{ if } \mathsf{dist}(a,b) \leq \delta \\ 1 \text{ otherwise} \end{cases}$$

- Similar to conditional OT [DOR99].
- Our construction supports distance metrics $\{L_\infty, L_1, L_2\}$.
- Based on Shared OT [PGNT25], where all OT inputs are in secret-shared form.

Distance-Aware OT (daOT)

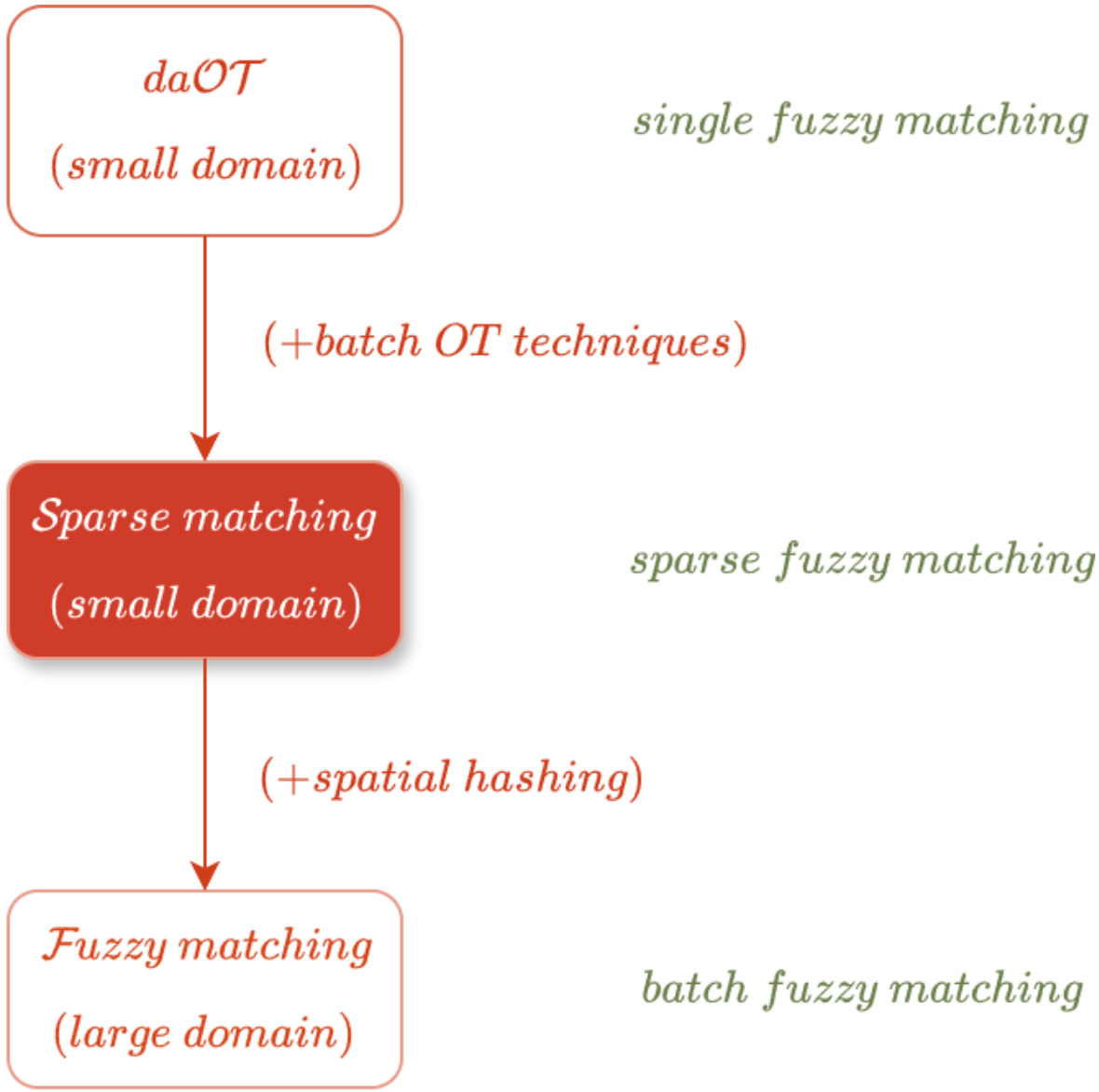
$$a, b \in \mathcal{D}$$

$$r_0, r_1 \in \{0,1\}^\ell$$

$$c = \begin{cases} 0 \text{ if } \mathsf{dist}(a,b) \leq \delta \\ 1 \text{ otherwise} \end{cases}$$

- Similar to conditional OT [DOR99].
- Our construction supports distance metrics $\{L_\infty, L_1, L_2\}$.
- Based on Shared OT [PGNT25], where all OT inputs are in secret-shared form.

Sparse Matching



Sparse Matching

r_1	$(1, x_1)$
r_3	$(3, x_3)$
r_6	$(6, x_6)$

Sparse Matching

$(1, y_1)$	(r_{10}, r_{11})
$(3, y_3)$	(r_{30}, r_{31})
$(5, y_5)$	(r_{50}, r_{51})
$(8, y_8)$	(r_{80}, r_{81})

For any common index i :

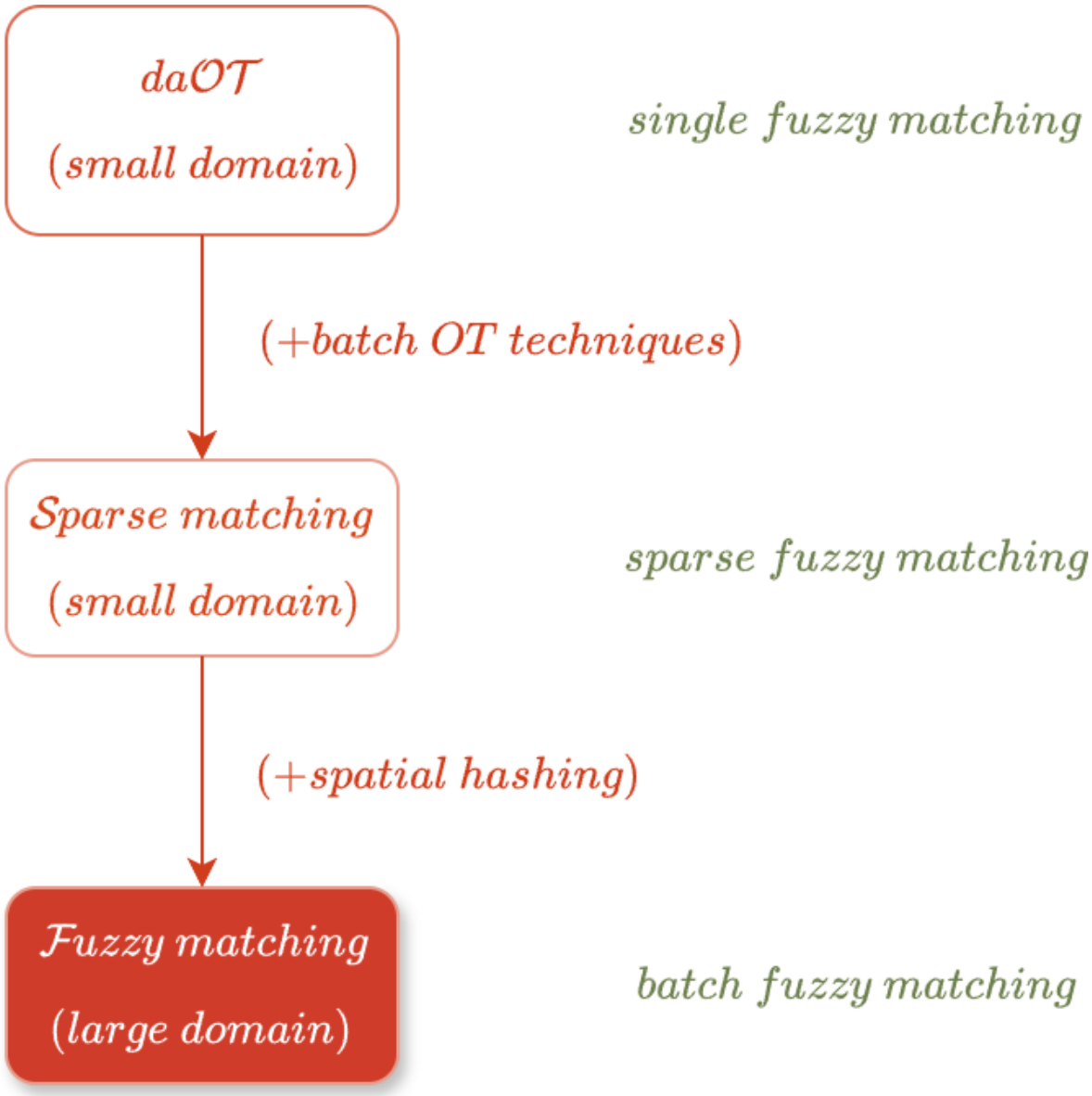
x_i → daOT ← y_i

r_i ← daOT → (r_{i0}, r_{i1})

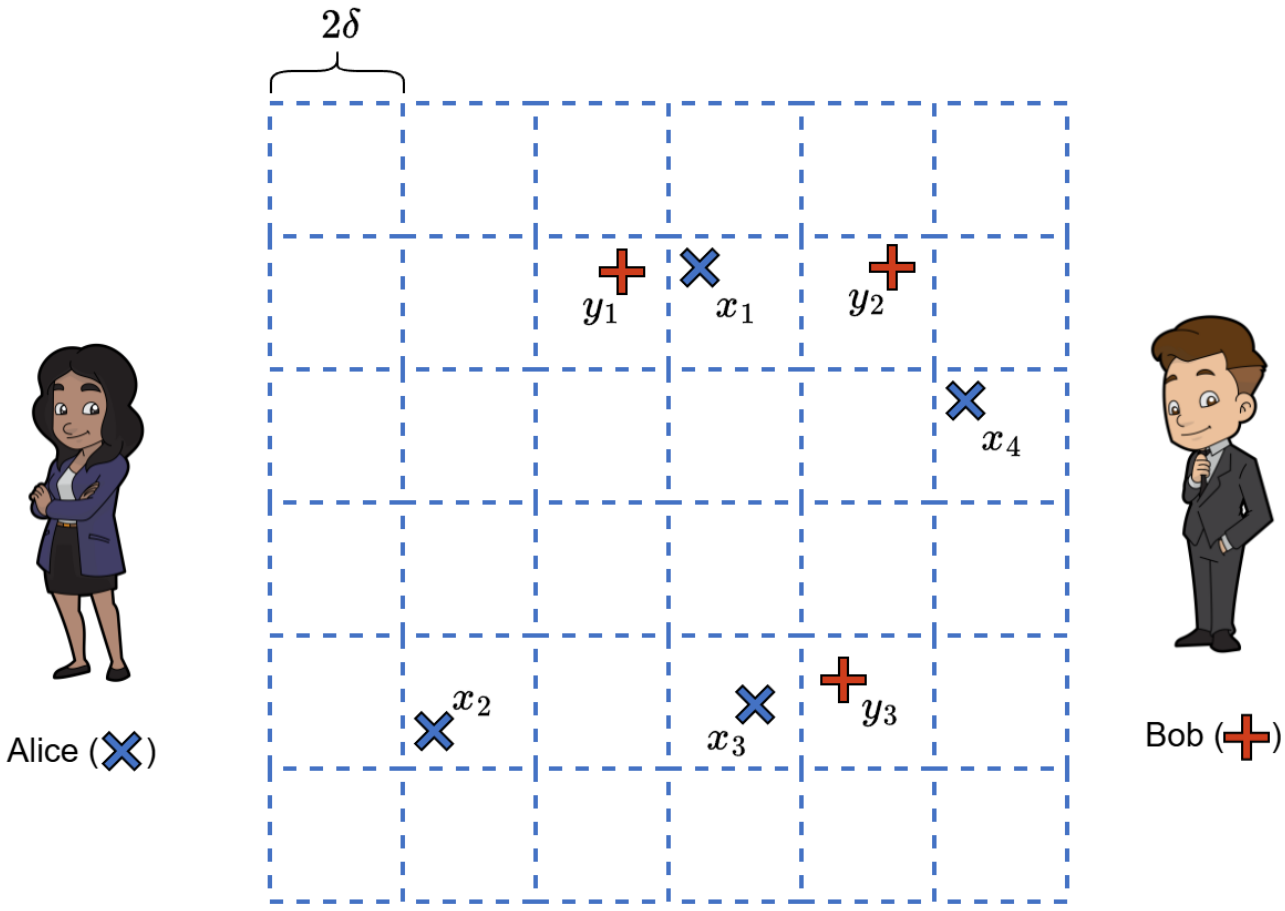
Otherwise: Both parties receive random outputs

- Similar to the "sparse OT" extension in [PRTY19]
- Introduce Sparse Shared OT construction based on OKVS

Fuzzy Matching



Fuzzy Matching: Spatial Hashing ([GRS22], [GRS23])



- Each of Alice's points is compared with the neighboring points of Bob.

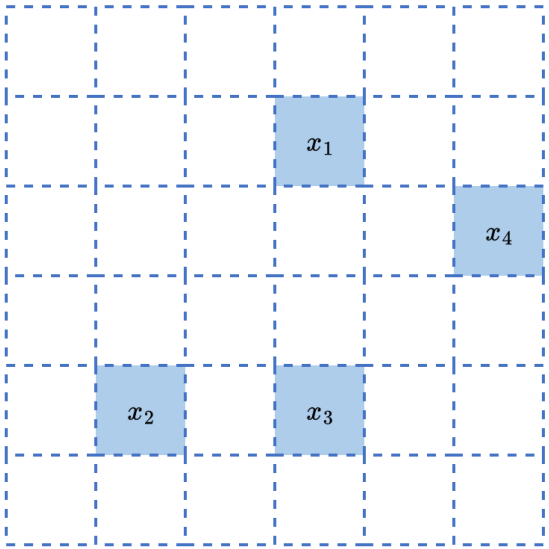
Fuzzy Matching

Each cell of Bob contains at most ρ hashed points

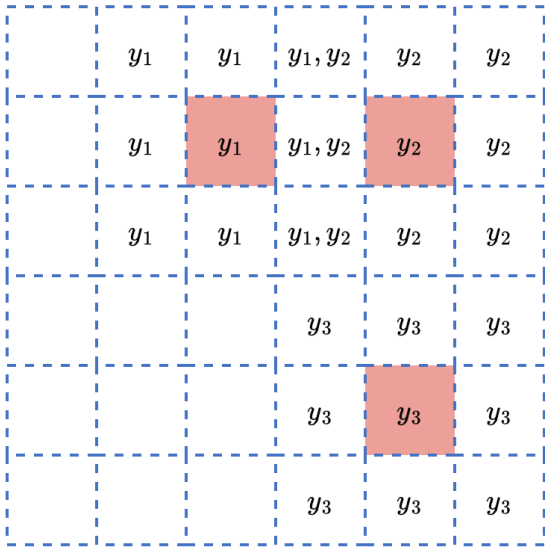
$\mathsf{dist}(x_i, y_i) \leq \delta \iff$

1. x_i and y_i are neighbors.
2. $\mathsf{dist}(\langle x_i \rangle, \langle y_i \rangle) \leq \delta$

Where truncated value* $\langle z \rangle = z \pmod{4\delta}$

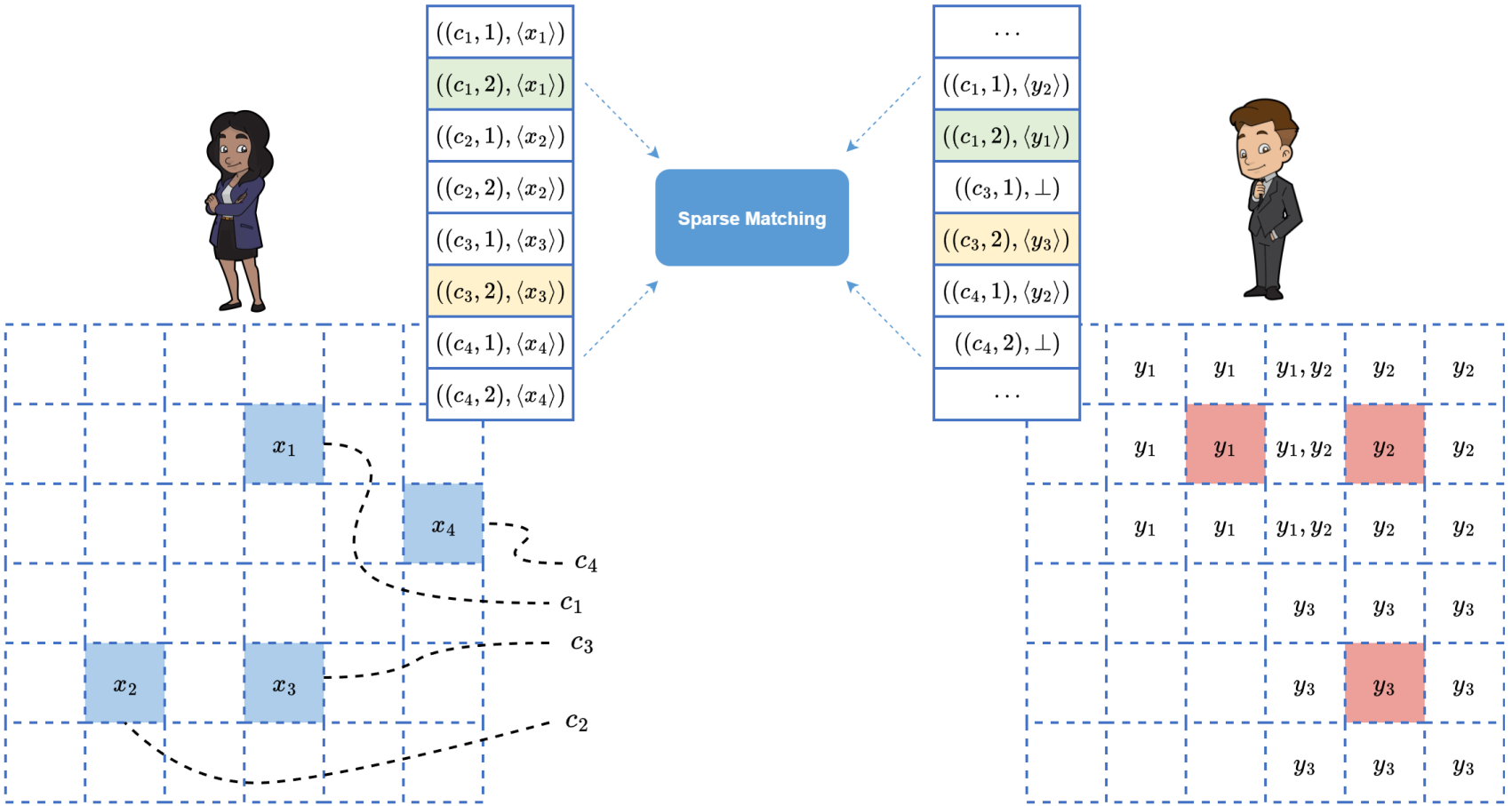


Reducing fuzzy matching over abritrary domain to a smaller domain of size ~ 4δ

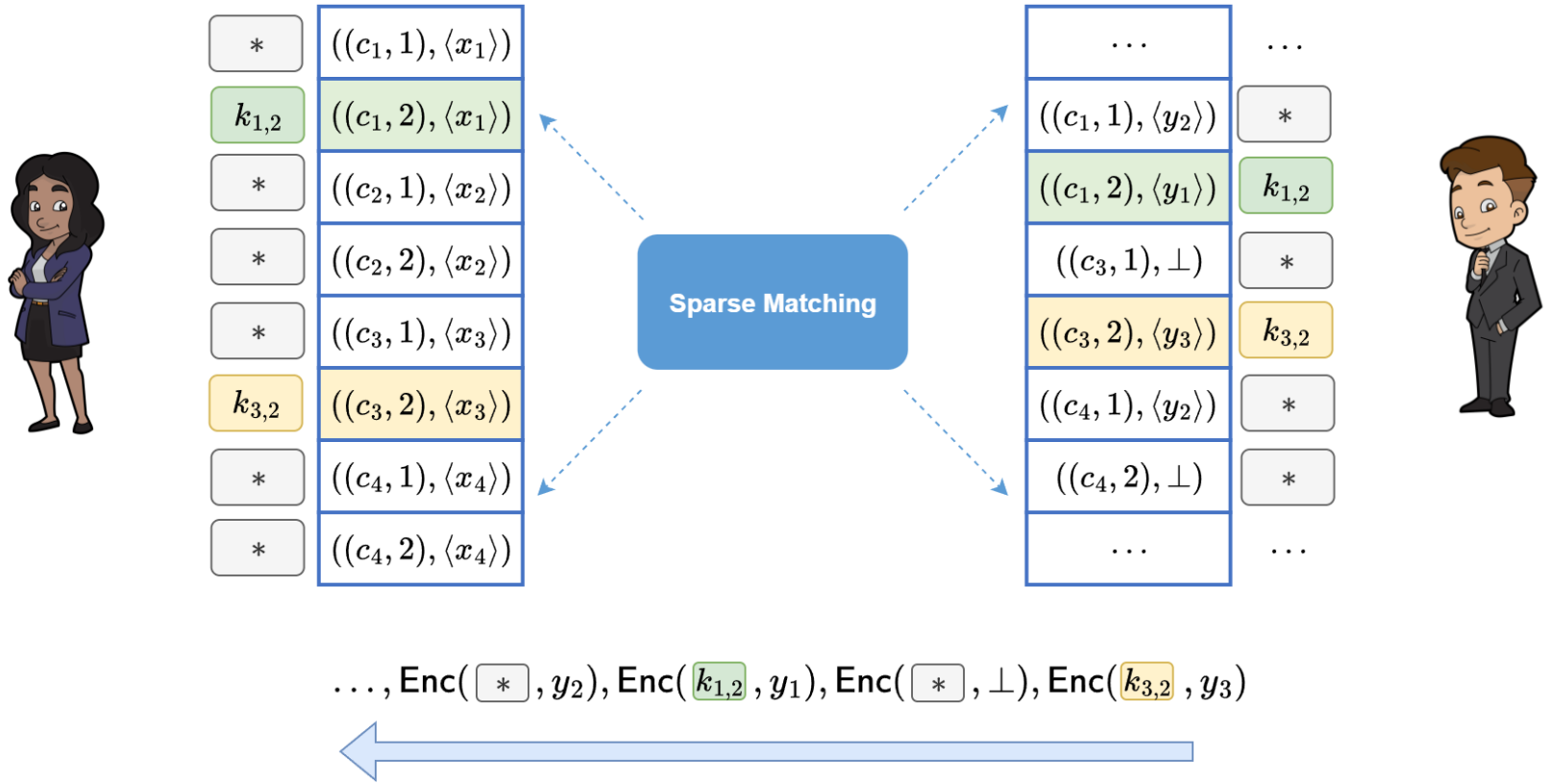


- In this case, $\rho = 2$

Fuzzy Matching

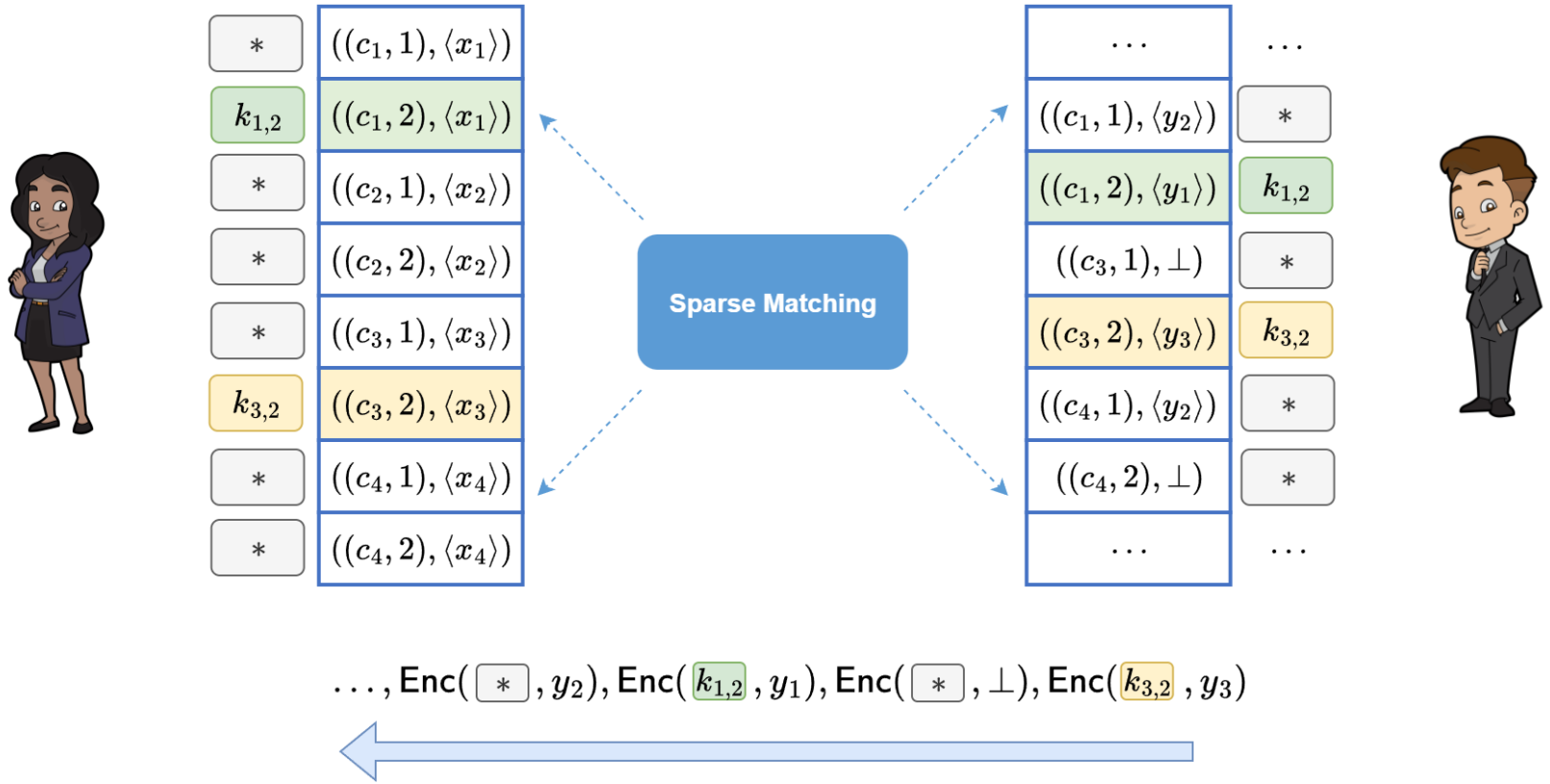


Fuzzy Matching



- Alice uses each key to decrypt received ciphertexts.
- A successful decryption means a fuzzy match.
- This last step was simplified for this presentation.
- Ciphertexts need to be sent inside an OKVS [GPR+21].

Fuzzy Matching



- Alice uses each key to decrypt received ciphertexts.
- A successful decryption means a fuzzy match.
- This last step was simplified for this presentation.
- Ciphertexts need to be sent inside an OKVS [GPR+21].

Implementation of OT-based Fuzzy PSI Protocol

- Our implementation relies on the following dependencies:
  - The `libOTe` library [PR] for OT extension.
  - The `volePSI` library [Res23] to instantiate the required OKVS scheme.
  - The `cryptoTools` library [R+23] for symmetric cryptographic primitives (e.g., PRNG).
- We instantiate the OPRF primitive using the construction from [KKRT16], but replace the polynomial interpolation technique with the state-of-the-art OKVS scheme from the `volePSI` library.

Implementation of OT-based Fuzzy PSI Protocol

- Our implementation relies on the following dependencies:
  - The `libOTe` library [PR] for OT extension.
  - The `volePSI` library [Res23] to instantiate the required OKVS scheme.
  - The `cryptoTools` library [R+23] for symmetric cryptographic primitives (e.g., PRNG).
- We instantiate the OPRF primitive using the construction from [KKRT16], but replace the polynomial interpolation technique with the state-of-the-art OKVS scheme from the `volePSI` library.

Summary: Fuzzy PSI

- Symmetric-key-based framework for fuzzy PSI.
  - Key building block: distance-aware OT (daOT)
- Extension to other PSI-like fuzzy primitives: cardinality, inner join.
- Open Problems:
  - Extending the framework to handle malicious adversaries.
  - Designing efficient daOT for various distance metrics.
  - Exploring other applications/protocols for "sparse" secure computation.

Summary: Fuzzy PSI

- Symmetric-key-based framework for fuzzy PSI.
  - Key building block: distance-aware OT (daOT)
- Extension to other PSI-like fuzzy primitives: cardinality, inner join.
- Open Problems:
  - Extending the framework to handle malicious adversaries.
  - Designing efficient daOT for various distance metrics.
  - Exploring other applications/protocols for "sparse" secure computation.

Summary: Fuzzy PSI

- Symmetric-key-based framework for fuzzy PSI.
  - Key building block: distance-aware OT (daOT)
- Extension to other PSI-like fuzzy primitives: cardinality, inner join.
- Open Problems:
  - Extending the framework to handle malicious adversaries.
  - Designing efficient daOT for various distance metrics.
  - Exploring other applications/protocols for "sparse" secure computation.

Outline

- ~~Introduction to PSI~~
- ~~Traditional PSI Protocol~~
- ~~PSI for Small Sets~~
- ~~PSI for Large Sets~~
- ~~Fuzzy PSI~~
- Take-Home Messages

Take-Home Messages

- **PSI for Small/Large Sets:** Fast and efficient; scales well with set size.
- **Fuzzy PSI:** Provides approximate matching using distance-aware OT (daOT). Still room for performance improvements.
- **Implementation Insights:**
  - Exploit parallelism and asynchronous sending/receiving for better runtime.
  - Use optimized libraries for core operations, e.g., AES-fixed key or ECC, to speed up computation.

Thank You

References

Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner.
More efficient oblivious transfer and extensions for faster secure computation.
In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, ACM CCS 2013, pages 535–548. ACM Press, November 2013.

Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan.
Conditional oblivious transfer and timed-release encryption.
In Jacques Stern, editor, EUROCRYPT'99, volume 1592 of LNCS, pages 74–89. Springer, Heidelberg, May 1999.

Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai.
Oblivious key-value stores and amplification for private set intersection.
Cryptology ePrint Archive, Report 2021/883, 2021.
https://eprint.iacr.org/2021/883.

Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank.
Extending oblivious transfers efficiently.
In Dan Boneh, editor, CRYPTO 2003, volume 2729 of LNCS, pages