cwPSU: Efficient Unbalanced Private Set Union via Constant-weight Codes

Qingwen Li[1], Song Bian[2], Hui Li[1]

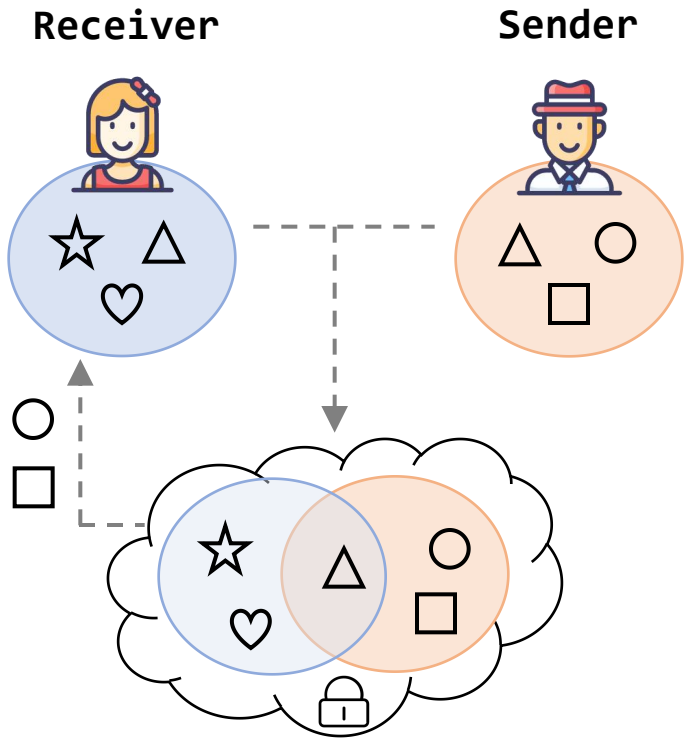
[1] Xidian University

[2] Beihang University

Private Set Union

What is Private Set Union?

- A cryptographic protocol for computing **the union of private sets**.
- Ensures no party reveals extra information beyond the union result.
- Enables privacy-preserving data collaboration.



Applications

VISA MasterCard

Alipay

Fraud-Prevention Collaboration

Uber ebay

amazon

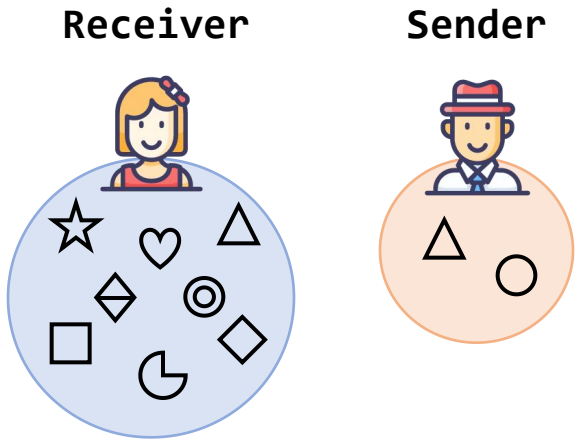
Shared Risk Blacklists

Unbalanced Private Set Union

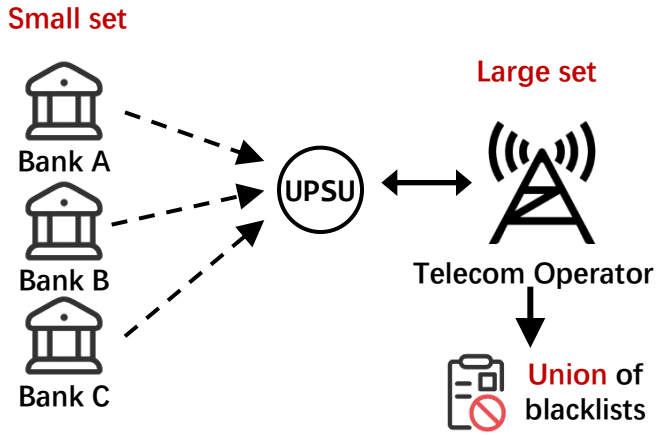
Unbalanced PSU matters in practice

- **Unbalanced Setting**: One input set is **significantly larger** than the other
- **Challenge:** Existing PSU protocols incur communication linear in the larger set
- **Motivation:** Such unbalanced scenarios are common in practice

e.g. Unbalanced PSU in blacklist aggregation: small bank inputs and a large telecom database.



Unbalanced Setting



Blacklist Aggregation

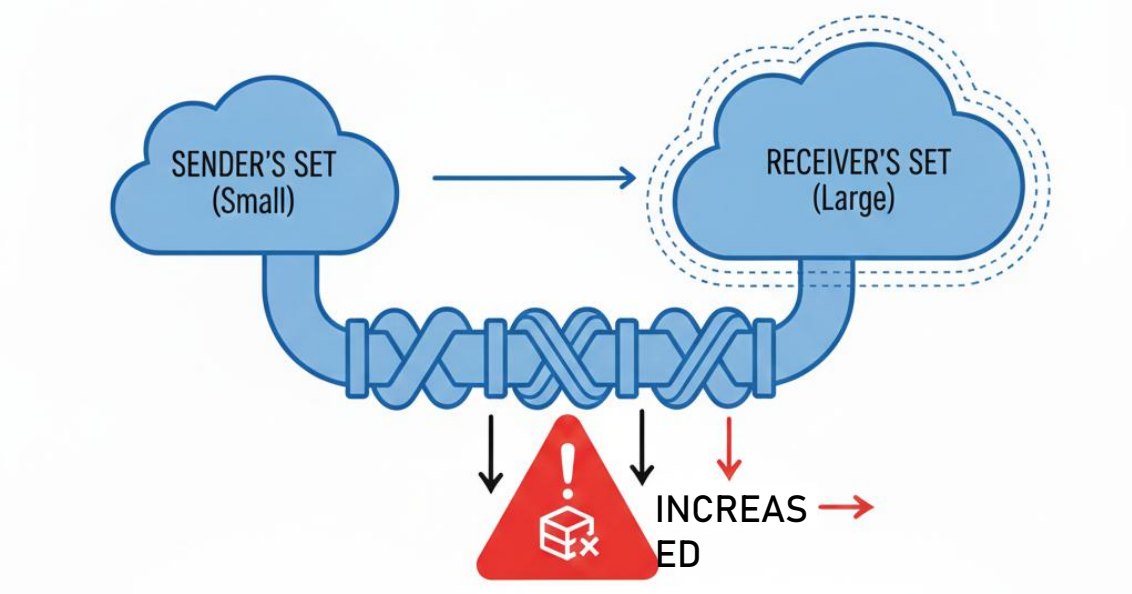
Motivation

Prior works on unbalanced private set union

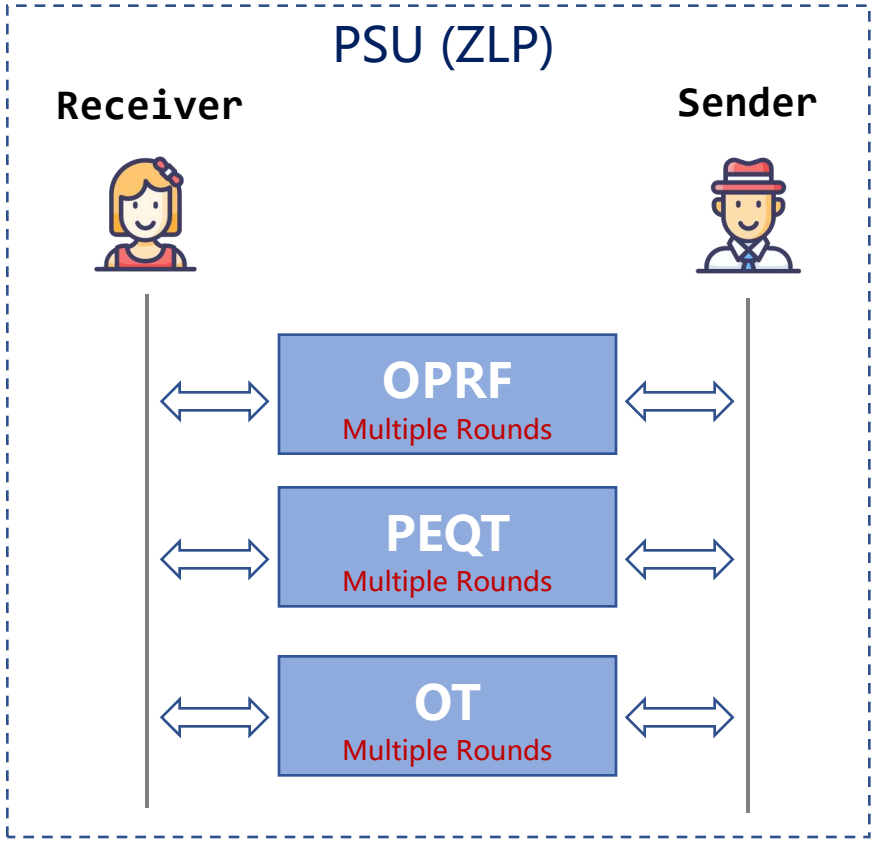
- **TCLZ (CCS 2023)**
  - Proposes Permuted Matrix Private Equality Test (pmPEQT)
  - First LFHE-based Unbalanced PSU protocol
  - Communication linear in small set size, logarithmic in large set size
- **ZLP (CCS 2024)**
  - Introduces Oblivious Key-Value Store (OKVS)
  - Shifts computation and communication to offline phase
  - Improved online efficiency over TCLZ

Motivation

Limitations of existing UPSU protocols



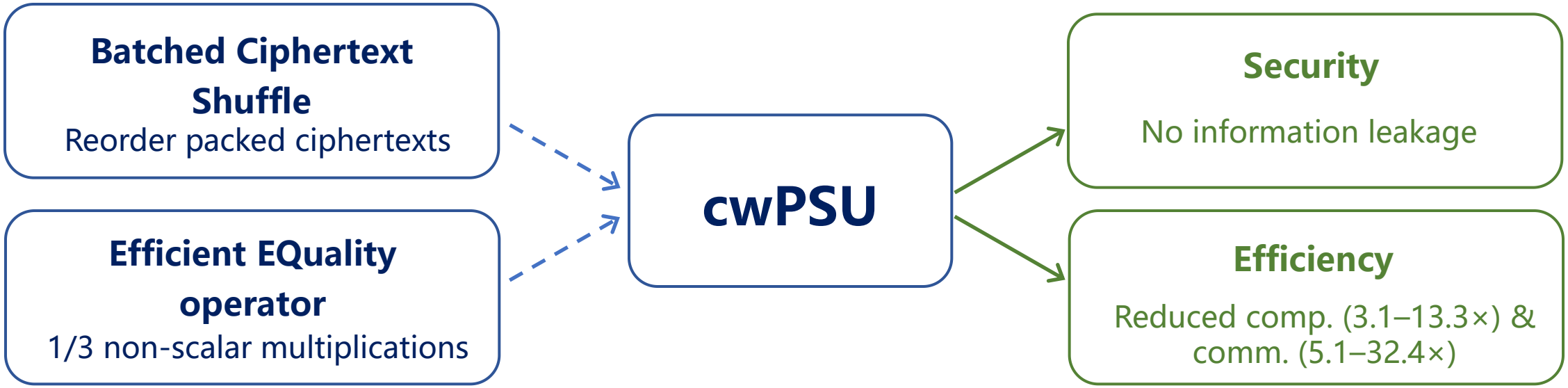
Communication overhead increases with large set size



Multiple rounds of communication

Our Work

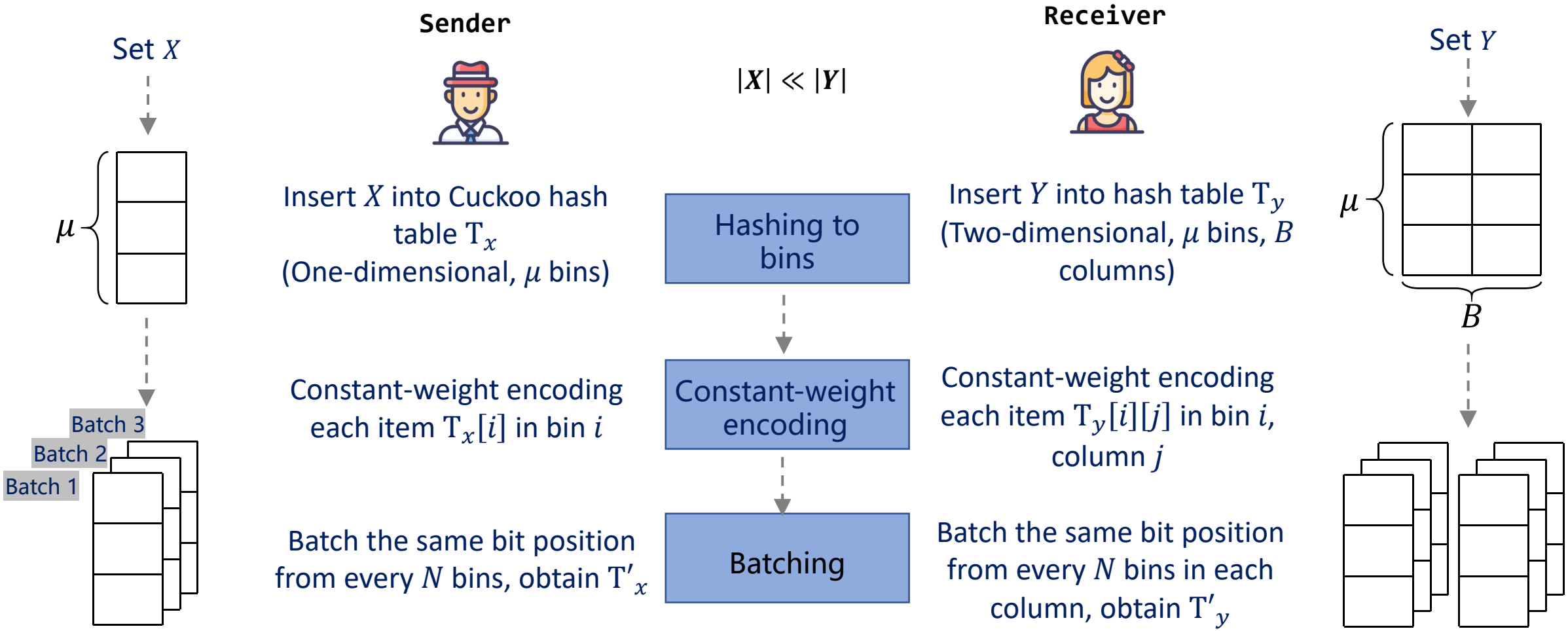
Key contributions of cwPSU



- A novel unbalanced PSU protocol based on constant-weight encoding
- Communication remains independent of the large set
- Require just one round of online communication

Basic Protocol

Data process



Basic Protocol

Private set union based on constant-weight codes

Sender

Encrypt each batch using its public key

$[\![\mathrm{T'}_x]\!]_{pk_s} = Enc_{pk_s}(\mathrm{T'}_x)$

$[\![\mathrm{T'}_x]\!]_{pk_s}$ →

Receiver

Invoke the arithmetic constant-weight equality operator to homomorphically compute

$[\![\mathrm{E}]\!]_{pk_s} = EQ([\![\mathrm{T'}_x]\!]_{pk_s}, \mathrm{T'}_y)$

For bin i: if $\mathrm{T}_x[i]$ is in $\mathrm{T}_y[i]$, $\mathrm{E}[i] = 1$, else $\mathrm{E}[i] = 0$

$[\![\mathrm{B}]\!]_{pk_s} = 1 - [\![\mathrm{E}]\!]_{pk_s}$ $\quad [\![\widehat{\mathrm{B}}]\!]_{pk_s}$ $\quad [\![R]\!]_{pk_r}$

e.g.

$[\![\mathrm{B}]\!]_{pk_s}$
0
1
1

$[\![\widehat{\mathrm{B}}]\!]_{pk_s}$
$r_1 + 0$
$r_2 + 1$
$r_3 + 1$

$[\![R]\!]_{pk_r}$
r_1
r_2
r_3

$[\![\widehat{\mathrm{B}}]\!]_{pk_s}$, $[\![R]\!]_{pk_r}$ ←

$\widehat{\mathrm{B}}$ $\quad \widehat{\mathrm{B}} - [\![R]\!]_{pk_r}$ $\quad [\![\boldsymbol{X/Y}]\!]_{pk_r}$

e.g.

$\widehat{\mathrm{B}}$
$r_1 + 0$
$r_2 + 1$
$r_3 + 1$

$\widehat{\mathrm{B}} - [\![R]\!]_{pk_r}$
0
1
1

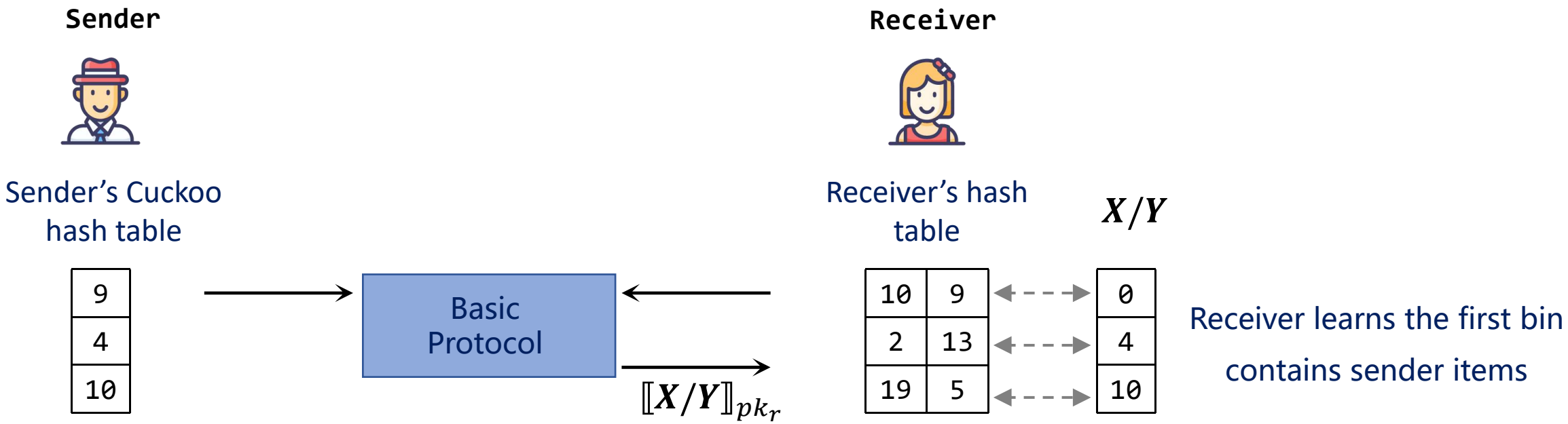
⊗

$\boldsymbol{X}$
9
4
10

$[\![\boldsymbol{X/Y}]\!]_{pk_r}$
0
4
10

$[\![\boldsymbol{X/Y}]\!]_{pk_r}$ →

Information leakage — An example

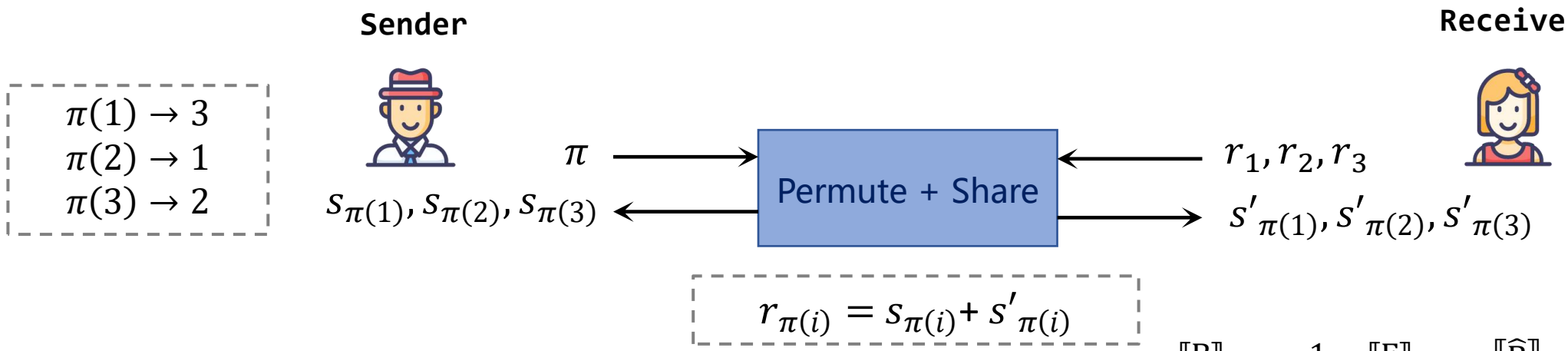


How can sender shuffle the batched ciphertext $[\![X/Y]\!]_{pk_r}$ without decrypt ?

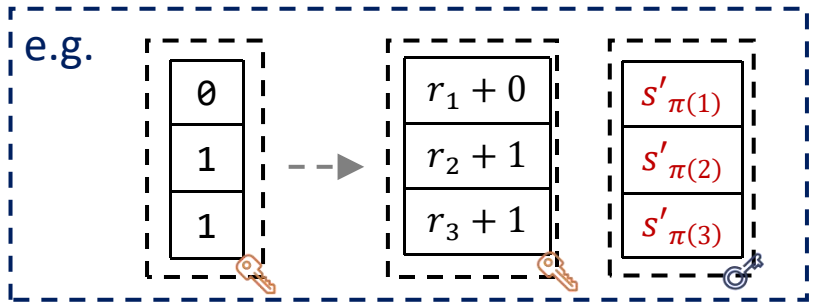
→ **Batched Ciphertext Shuffle**

Batched Ciphertext Shuffle

How batched ciphertext shuffle works?

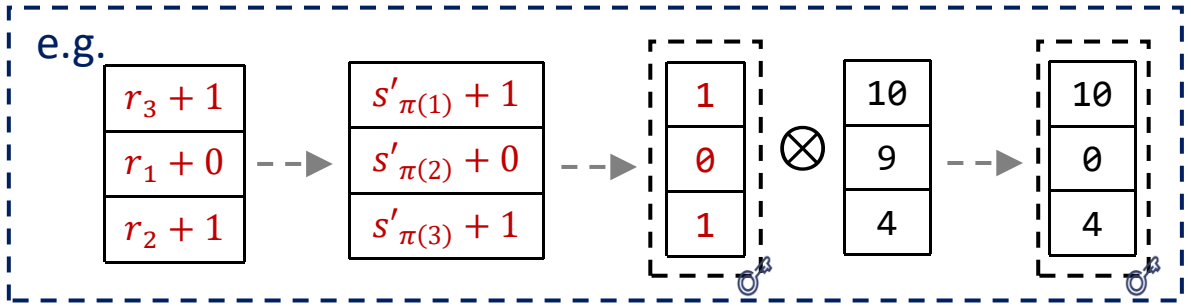


$[\![\mathrm{B}]\!]_{pk_s} = 1 - [\![\mathrm{E}]\!]_{pk_s}$ $[\![\widehat{\mathrm{B}}]\!]_{pk_s}$ $[\![s'_{\pi(i)}]\!]_{pk_r}$



$[\![\widehat{\mathrm{B}}]\!]_{pk_s}, [\![s'_{\pi(i)}]\!]_{pk_r}$

$\widehat{\mathrm{B}}^*$ $\widehat{\mathrm{B}}^* - s_{\pi(i)}$ X^* $[\![X/Y]\!]^*_{pk_r}$



$[\![X/Y]\!]^*_{pk_r}$

$\widehat{\mathrm{B}}^* - s_{\pi(i)} - [\![s'_{\pi(i)}]\!]_{pk_r}$

* means it was shuffled by π

Efficient Equality Operator

Efficient Arithmetic Constant-weight Equality Operator

Algorithm 1 Arithmetic Constant-weight Equality Operator $\mathrm{cwEQ}(x, y)$.

Input: $x, y \in CW(l, h)$

1: $k = \sum_{i\in[l]} x[i] \cdot y[i]$

2: $e = \frac{1}{h!} \cdot \prod_{i\in[h]} (k - i)$

Output: $e \in \{0, 1\}$

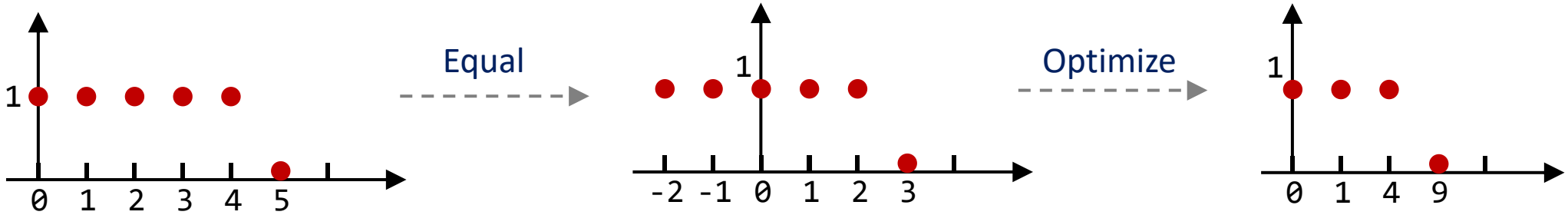
- $x[i] \cdot y[i]$ is plaintext-ciphertext multiplications, k is a ciphertext
- To evaluate e, $h - 1$ ciphertext-ciphertext multiplications is needed

$$e = \frac{1}{h!} \prod_{i\in[h]} (k - i), \iff f(k) = \begin{cases} 0, & 0 \le k < h, \\ 1, & k = h. \end{cases}$$

$$f(k) = \begin{cases} 0, & \lceil -(h-1)/2 \rceil \le k < \lfloor (h-1)/2 \rfloor, \\ 1, & k = \lfloor (h-1)/2 \rfloor, \end{cases}$$

$$z = \lfloor h/2 + 1 \rfloor$$

$$f(k) = \prod_{i=0}^{z} \frac{k^2 - i^2}{h^2 - i^2}.$$



Efficient Equality Operator

Further optimization

$$f(k) = \prod_{i=0}^{z} \frac{k^2 - i^2}{h^2 - i^2} = \sum_{i=0}^{z} a_i k^{2(i+1)}, \quad \xrightarrow{\text{Paterson-Stockmeyer algorithm}} \quad f(\llbracket k \rrbracket) = \sum_{i=0}^{H-1} \left(\llbracket k \rrbracket^{2L}\right)^i \left(\sum_{j=0}^{L-1} a_j \left(\llbracket k \rrbracket^2\right)^{j+1} \right).$$

Optimal choices of *L* and *H* and required ciphertext-ciphertext multiplications for different *h*

h	7	8–11	12–15	16–19	20–23
L	2	3	4	5	6
H	2	2	2	2	2
Mult.	3	4	5	6	7

Algorithm 2 Efficient Arithmetic Constant-weight Equality Operator $\mathrm{EEQ}(x, \llbracket y \rrbracket)$.

Input: $x, \llbracket y \rrbracket \in CW(l, h), L, H, \{a_i\}_{i \in [h]}$

1: $\llbracket k \rrbracket = \sum_{i \in [l]} x[i] \cdot \llbracket y[i] \rrbracket$
2: $z = \lfloor h/2 + 1 \rfloor$
3: $\llbracket k' \rrbracket = \llbracket k \rrbracket - z$
4: Get $\llbracket k' \rrbracket^2, \cdots, \llbracket k' \rrbracket^{2L}$ and $\llbracket k' \rrbracket^{4L}, \cdots, \llbracket k' \rrbracket^{(H-1)\cdot 2L}$
5: $e = \sum_{i=0}^{H-1} \left(\llbracket k \rrbracket^{2L}\right)^i \left(\sum_{j=0}^{L-1} a_j \left(\llbracket k \rrbracket^2\right)^{j+1} \right)$

Output: $e \in \{0, 1\}$

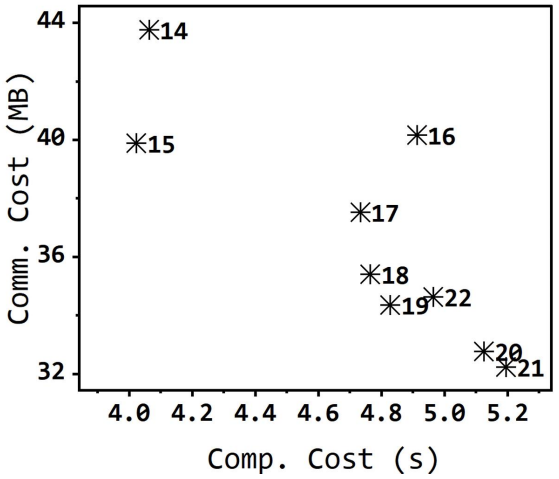
Implementation Detail

Setup phase

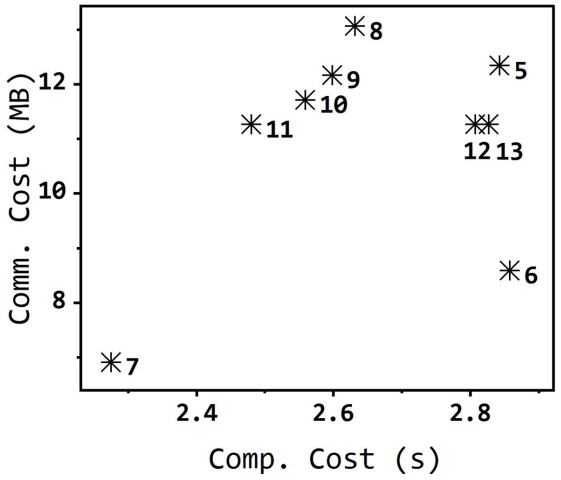
- Assumption: Sender input is available at setup
- Observation: BCS permute + share is input-independent

Optimal Hamming weight

(a) 128-bit item



(b) 32-bit item



Result in cwPSU

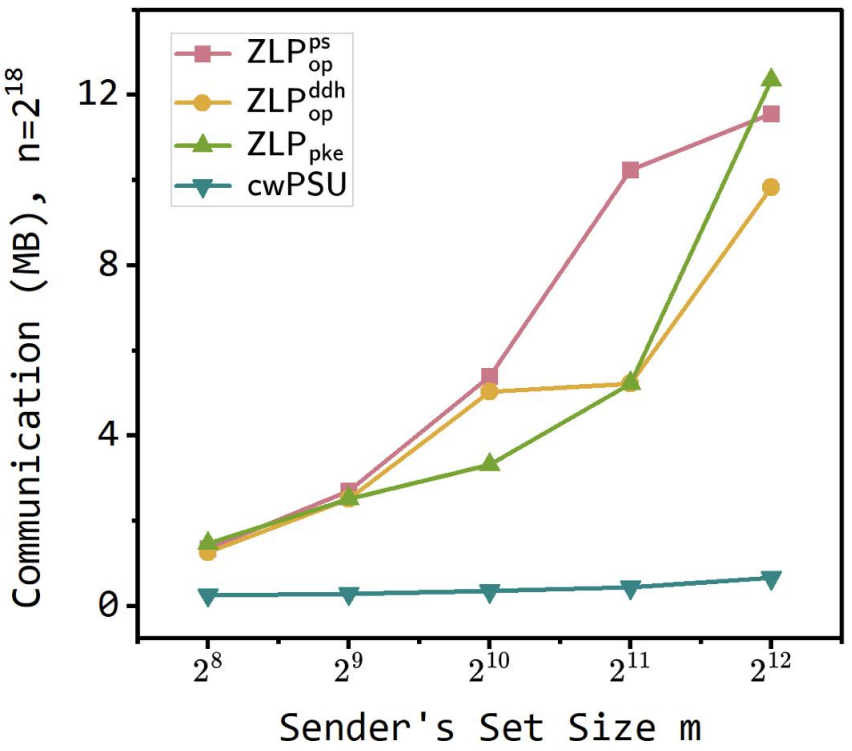
- Online communication of cwPSU remains constant for a fixed small set size m

Param.		Online Comm. (MB)		
m	n	cwPSU	ZLP_{op}	ZLP_{pke}
2^8	2^{18}	0.26	1.35	1.47
	2^{20}	0.26	2.60	2.63
	2^{22}	0.26	5.20	3.21
2^{10}	2^{18}	0.35	5.39	3.32
	2^{20}	0.35	5.47	5.88
	2^{22}	0.35	9.88	9.92

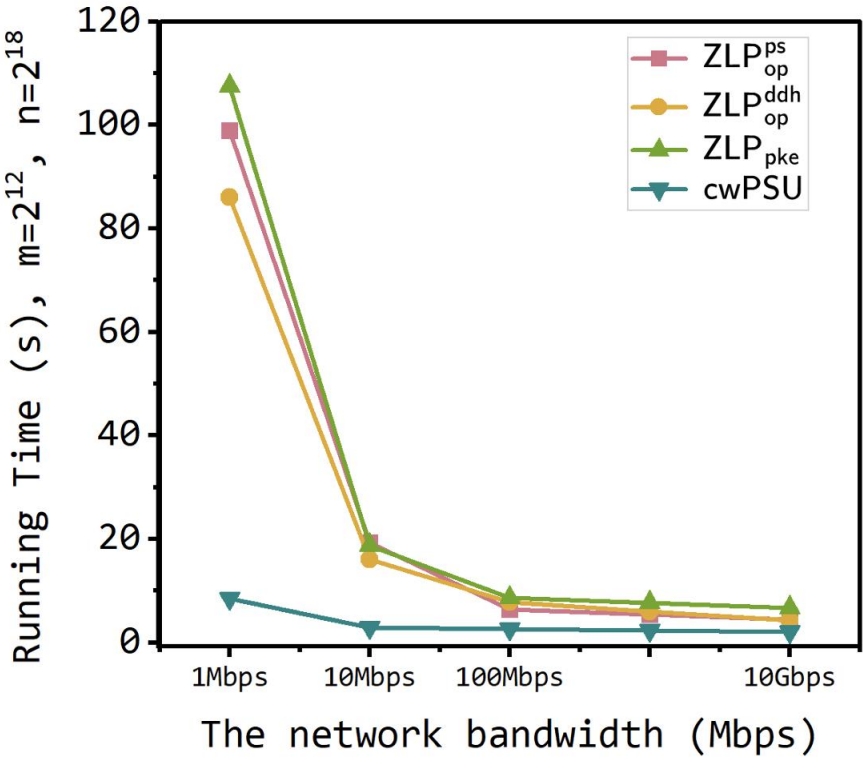
Evaluation

Result in cwPSU

Communication Overhead **vs.** Small Set Size

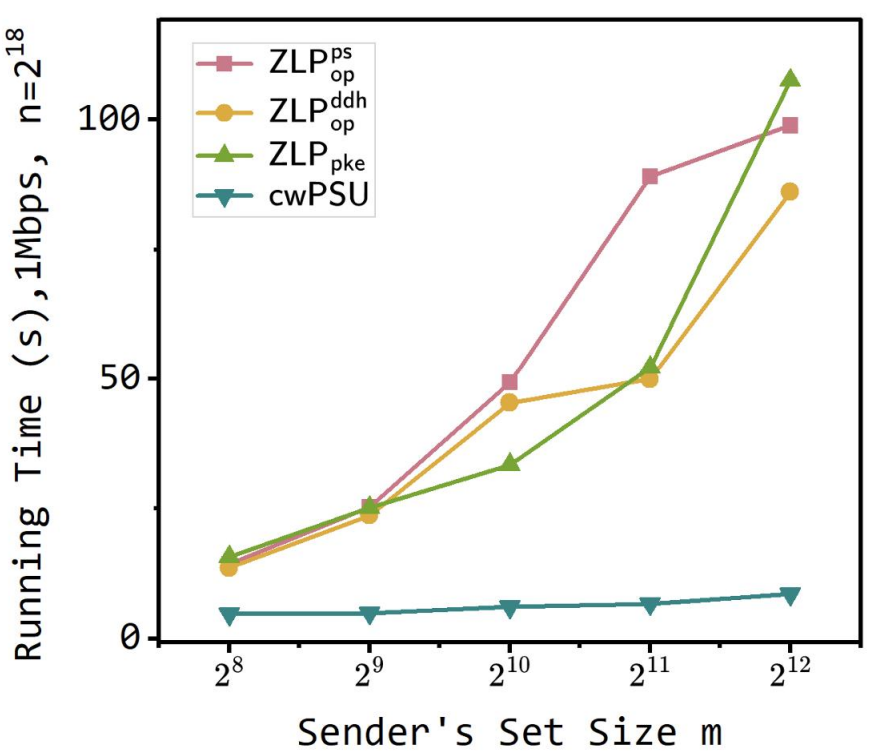


Running Time Performance under Different Network Bandwidths

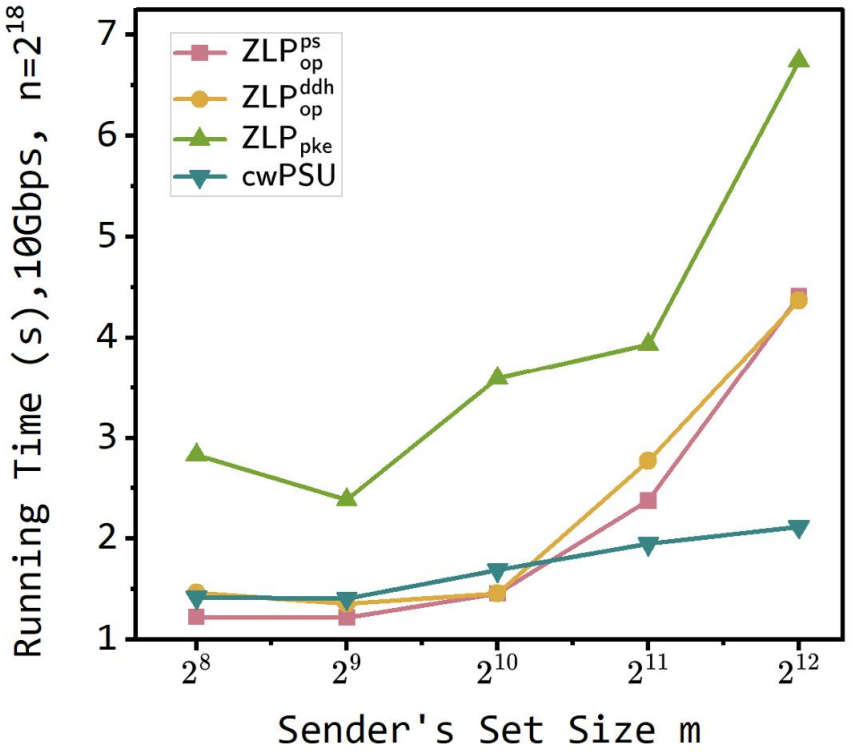


Result in cwPSU

Running Time **vs.** Small Set Size
(Low Bandwidth)



Running Time **vs.** Small Set Size
(High Bandwidth)



Result in EEQ

- ~2× runtime improvement
- Modest ciphertext size increase

Item bit length	Param.		Time (seconds)		Ciphertext size (MB)	
	n	m	cwEQ	EEQ	cwEQ	EEQ
32 bit	2^{18}	2^{10}	0.88	0.48	4.46	4.82
		2^{12}	0.86	0.49	4.46	4.84
	2^{20}	2^{10}	2.20	1.29	4.47	4.84
		2^{12}	2.24	1.31	4.47	4.84
128 bit	2^{18}	2^{10}	2.44	1.35	26.13	27.63
		2^{12}	2.35	1.34	26.13	27.63
	2^{20}	2^{10}	6.84	4.03	28.38	29.76
		2^{12}	6.80	3.86	28.38	29.76

Thank you

cwPSU is an unbalanced private set union protocol

- Efficiency: Achieves communication complexity linear in the smaller set size and independent of the larger set
- Round complexity: Requires only one round of online communication
- Performance: Significantly outperforms state-of-the-art PSU protocols in both communication and runtime, especially under low-bandwidth settings