BATCH PIR AND LABELED PSI WITH OBLIVIOUS CIPHERTEXT COMPRESSION

ALEXANDER BIENSTOCK,[1] SARVAR PATEL,[2] JOON YOUNG SEO,[2] KEVIN YEO[3]

[1] JPMORGAN AI RESEARCH & ALGOCRYPT COE (WORK DONE WHILE AT NYU AND INTERNING AT GOOGLE)
[2] GOOGLE
[3] GOOGLE AND COLUMBIA UNIVERSITY

PRIVATE INFORMATION RETRIEVAL (PIR)

- Client has single item x, server has $Y = \{(y_1, v_1), \dots, (y_n, v_n)\}$

x

(y_1, v_1)
(y_2, v_2)
(y_3, v_3)
…
(y_n, v_n)

PRIVATE INFORMATION RETRIEVAL (PIR)

- Client has single item x, server has $Y = \{(y_1, v_1), \dots, (y_n, v_n)\}$
  - Client wants to privately check if $x = y_i$ in Y

PRIVATE INFORMATION RETRIEVAL (PIR)

- Client has single item x, server has $Y = \{(y_1, v_1), \dots, (y_n, v_n)\}$
  - Client wants to privately check if $x = y_i$ in Y

PRIVATE INFORMATION RETRIEVAL (PIR)

- Client has single item x, server has $Y = \{(y_1, v_1), \dots, (y_n, v_n)\}$
  - Client wants to privately check if $x = y_i$ in Y
  - If so retrieve v_i

v_i if $x = y_i$

(y_1, v_1)
(y_2, v_2)
(y_3, v_3)
…
(y_n, v_n)

PRIVATE INFORMATION RETRIEVAL (PIR)

- Client has single item x, server has $Y = \{(y_1, v_1), \dots, (y_n, v_n)\}$
  - Client wants to privately check if $x = y_i$ in Y
  - If so retrieve v_i
  - “Privately” – the server should not learn what x is

v_i if $x = y_i$

(y_1, v_1)
(y_2, v_2)
(y_3, v_3)
…
(y_n, v_n)

PRIVATE INFORMATION RETRIEVAL (PIR)

- Client has single item x, server has $Y = \{(y_1, v_1), \ldots, (y_n, v_n)\}$
  - Client wants to privately check if $x = y_i$ in Y
  - If so retrieve v_i
  - "Privately" – the server should not learn what x is
- This talk: no preprocessing

v_i if $x = y_i$

(y_1, v_1)
(y_2, v_2)
(y_3, v_3)
…
(y_n, v_n)

BATCH PRIVATE INFORMATION RETRIEVAL (PIR)

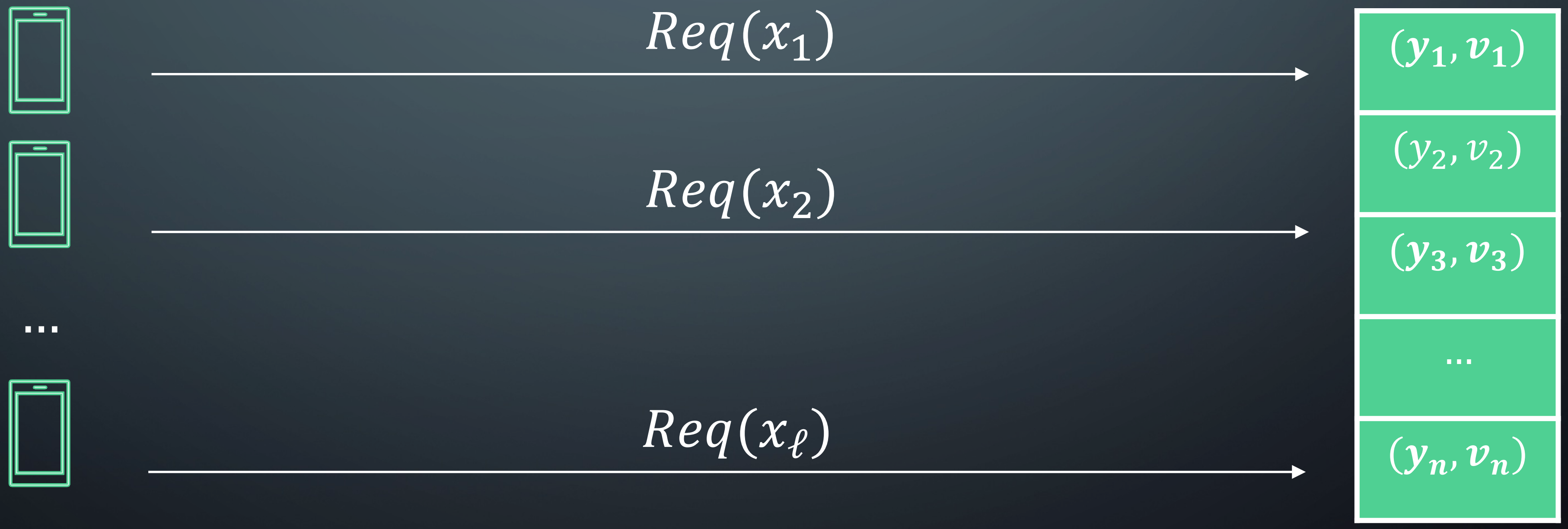
- *Batch* Private Information Retrieval (Batch PIR)
  - Client has $X = \{x_1, \dots, x_\ell\}$, server has $Y = \{(y_1, v_1), \dots, (y_n, v_n)\}$
  - Client wants to privately check if each $x_j = y_i$ in Y; if so retrieve v_i
  - "Privately" – the server should not learn what X is

$X = \{x_1, \dots, x_\ell\}$

(y_1, v_1)
(y_2, v_2)
(y_3, v_3)
…
(y_n, v_n)

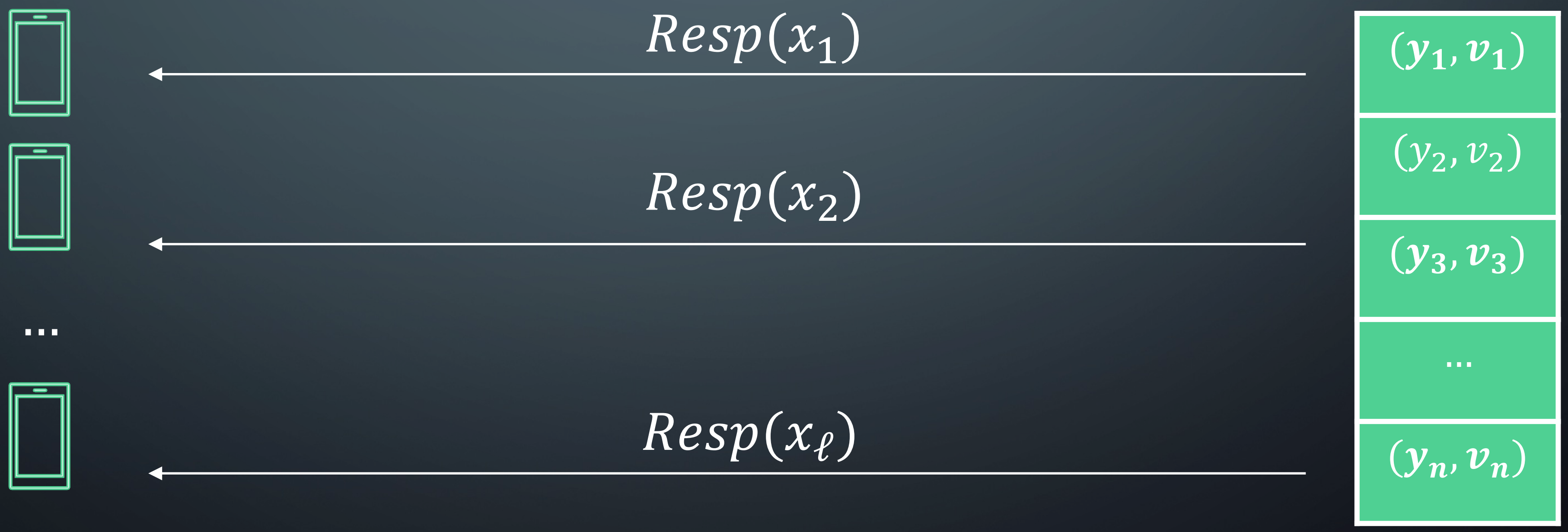
NAÏVE BATCH PIR FROM SINGLE REQUEST PIR

- Run single request PIR ℓ times



NAÏVE BATCH PIR FROM SINGLE REQUEST PIR

- Run single request PIR ℓ times



NAÏVE BATCH PIR FROM SINGLE REQUEST PIR

- Run single request PIR ℓ times

v_{i_1} if $x_1 = y_{i_1}$

v_{i_2} if $x_2 = y_{i_2}$

…

v_{i_ℓ} if $x_\ell = y_{i_\ell}$

(y_1, v_1)
(y_2, v_2)
(y_3, v_3)
…
(y_n, v_n)

NAÏVE BATCH PIR FROM SINGLE REQUEST PIR

- Run single request PIR ℓ times
  - Bad: factor ℓ blowup in overhead (server comp is now $O(\ell n)$)

v_{i_1} if $x_1 = y_{i_1}$

v_{i_2} if $x_2 = y_{i_2}$

…

v_{i_ℓ} if $x_\ell = y_{i_\ell}$

(y_1, v_1)
(y_2, v_2)
(y_3, v_3)
…
(y_n, v_n)

EFFICIENT BATCH PIR

- [ACLS18] use "Cuckoo Hashing" to remove factor ℓ server blowup
- But communication increases $1.5\times$
  - 1.5ℓ (encrypted) PIR requests/responses; 0.5ℓ of them "useless"
- Can we get rid of communication overhead?
  - This work: Yes – Oblivious Ciphertext (De)compression

OBLIVIOUS CIPHERTEXT COMPRESSION

PIR Responses

$$c = \boxed{Enc(20)}\boxed{Enc(0)}\boxed{Enc(0)}\boxed{Enc(14)}\boxed{Enc(3)}\boxed{Enc(0)}\boxed{Enc(35)}$$

ℓ-out-of-n non-zero

- Server compresses c to c' of size $\ell + \epsilon$, ϵ small

$$c' = \boxed{}\boxed{}\boxed{}\boxed{}\boxed{}$$

- Client decompresses and decrypts c' to $m = (m_1, \ldots, m_\ell)$ (nonzeros of c)

$$m = \boxed{20}\boxed{14}\boxed{3}\boxed{35}$$

Server must not learn non-zero m_i's or their locations

OUR MAIN TOOL: RANDOM-BAND MATRICES [DW19]

- Given $w \leq m$, sample $m \times \ell$ matrix M column-by-column as follows:
  - Pick random starting location from $1, \ldots, m - w$
  - Pick random w-bit string
  - Embed random w-bit string at starting location; everything else 0

OUR MAIN TOOL: RANDOM-BAND MATRICES [DW19]

- **Theorem** [DW19]: There exists constant $\epsilon > 0$ such that if $m = (1+\epsilon) \cdot \ell$, $w = O((\log \ell)/\epsilon)$, then randomly sampled $m \times \ell$ random band matrices have full column rank, time $O(\ell w)$ Gaussian elimination, and time $O(\ell w)$ vector multiplication, with high probability
- **Our Theorem:** There exists constant $\epsilon > 0$ such that if $m = (1+\epsilon) \cdot \ell$, $w = O(\log \ell + \lambda/\epsilon)$, then randomly sampled $m \times \ell$ random band matrices have full column rank, time $O(\ell w)$ Gaussian elimination, and time $O(\ell w)$ vector multiplication, with probability $1 - 2^{-\lambda}$

OUR MAIN TOOL: RANDOM-BAND MATRICES [DW19]

- Intuition for full rank:
  - Each band short, so intersects with few other bands
  - But still long enough to find a pivot

1				
0	1	1		
0	1	0		
1	1	0		
1	0	1	1	
0	1	1	1	
	0	0	0	…
			1	
			0	
			1	

OUR MAIN TOOL: RANDOM-BAND MATRICES [DW19]

- Intuition for fast Gaussian elimination: first sort columns by first non-zero position
  - Each band intersects with few other bands, so each row also consists of a single short band (Chernoff: length $O(w)$)
  - Each row band (by def) intersect with few other bands
  - Thus, getting to echelon form fast
  - Also, back substitution only on $\sim w$ entries for each row

$$
\begin{array}{ccccc}
1 & & & & \\
0 & 1 & 1 & & \\
0 & 1 & 0 & & \\
1 & 1 & 0 & & \\
1 & 0 & 1 & 1 & \\
0 & 1 & 1 & 1 & \\
 & 0 & 0 & 0 & \cdots \\
 & & & 1 & \\
 & & & 0 & \\
 & & & 1 &
\end{array}
$$

OBLIVIOUS CIPHERTEXT COMPRESSION

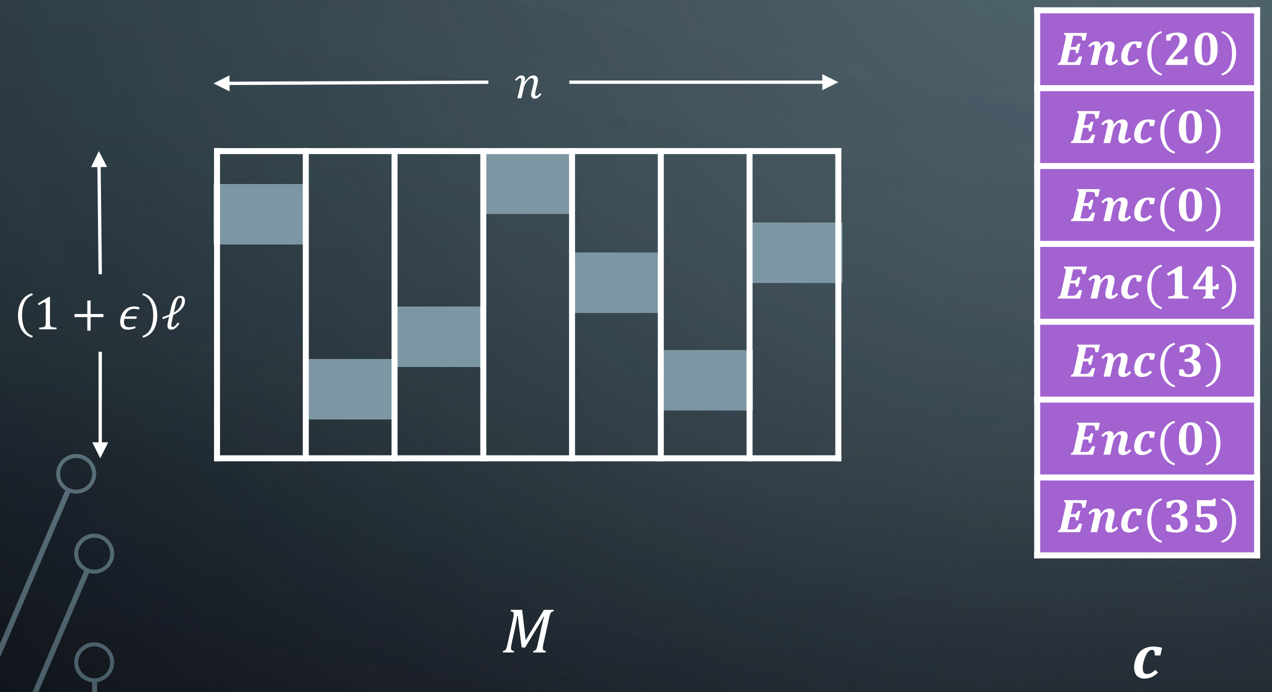
Compress: (assume Enc is additively homomorphic)

$Enc(20)$
$Enc(0)$
$Enc(0)$
$Enc(14)$
$Enc(3)$
$Enc(0)$
$Enc(35)$

$\boldsymbol{c}$

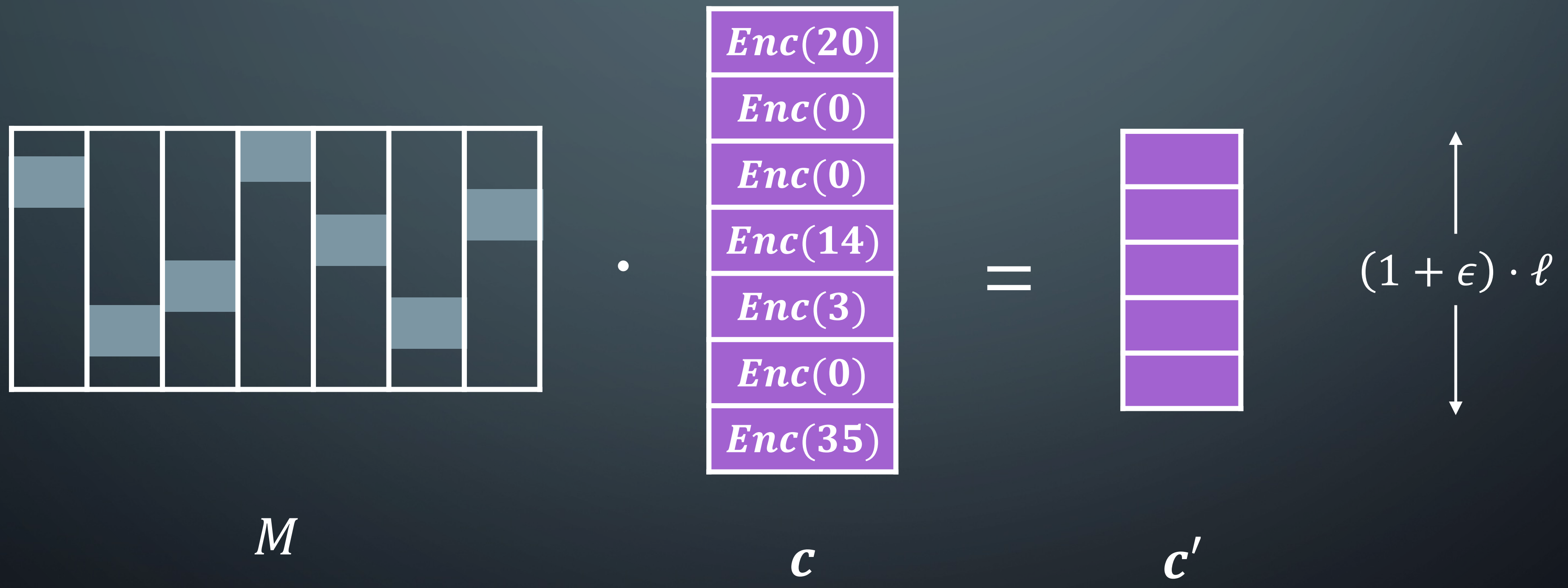
OBLIVIOUS CIPHERTEXT COMPRESSION

Compress: (assume Enc is additively homomorphic)



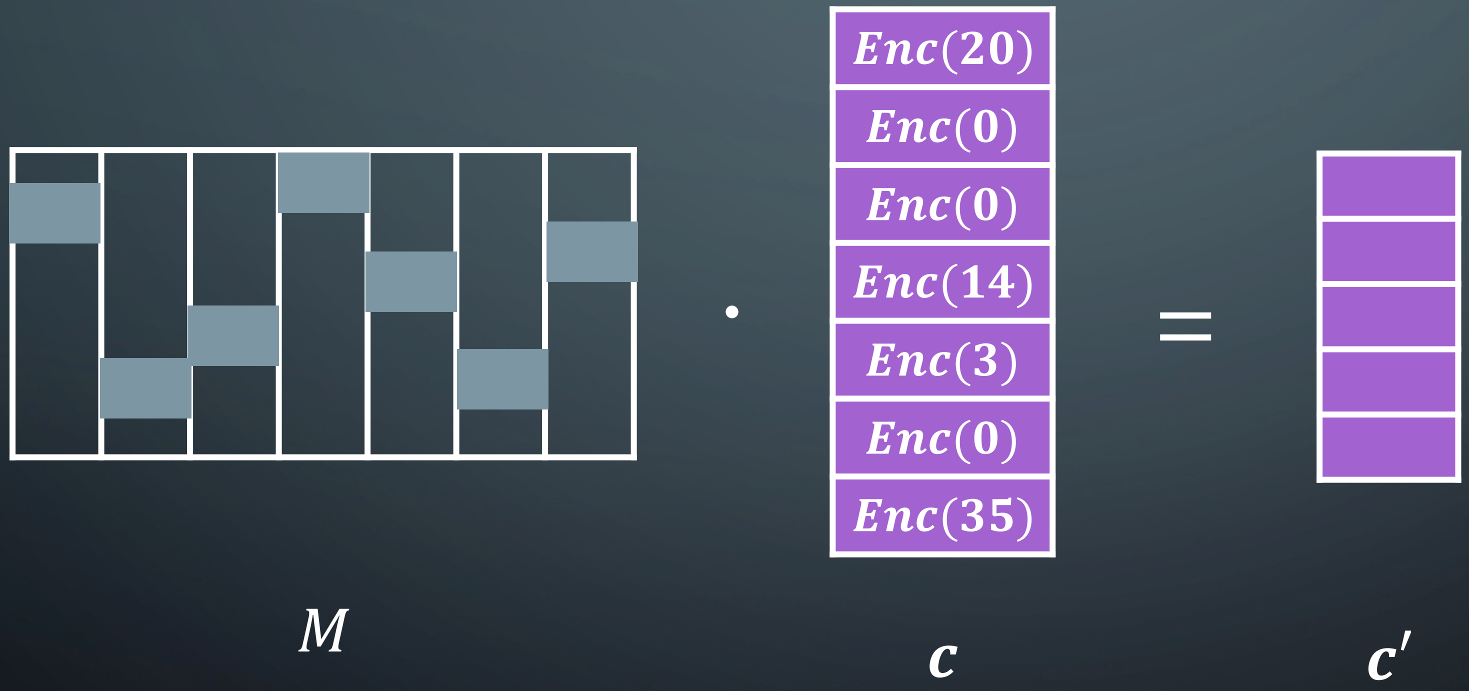
OBLIVIOUS CIPHERTEXT COMPRESSION

Compress: (assume Enc is additively homomorphic)



OBLIVIOUS CIPHERTEXT COMPRESSION

Compress: (assume Enc is additively homomorphic)

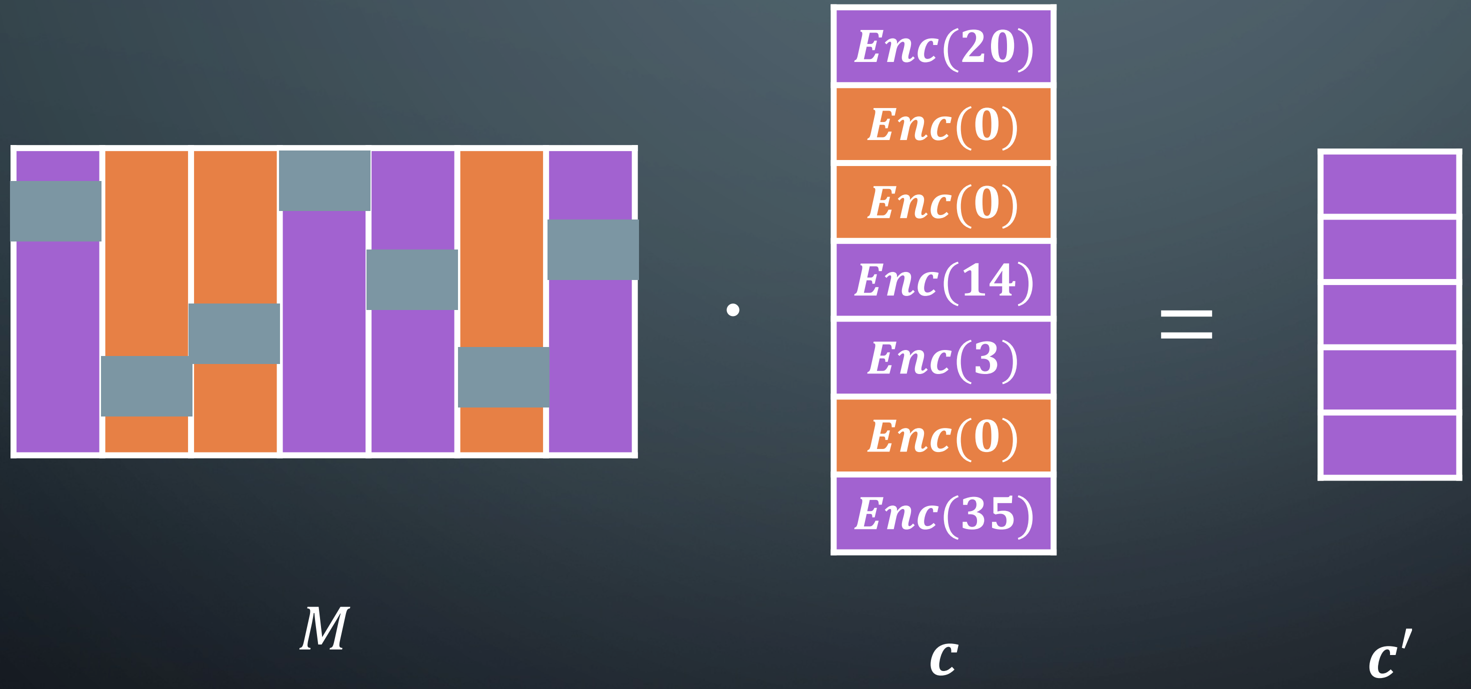


columns

$$= \sum_{i=1}^{n} \boldsymbol{c}_i \cdot M^i$$

OBLIVIOUS CIPHERTEXT COMPRESSION

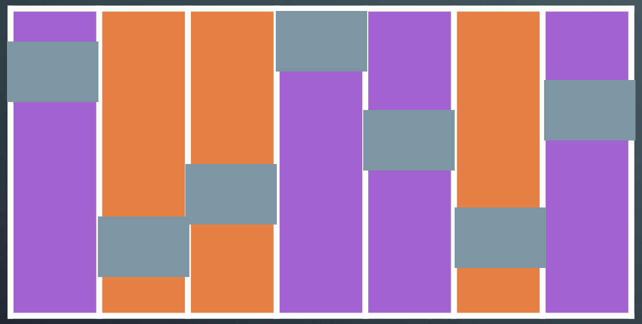
Compress: (assume Enc is additively homomorphic)

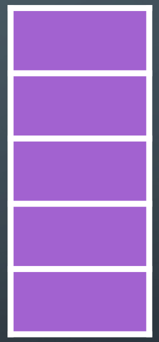


$$= \sum_{i=1}^{n} \boldsymbol{c}_i \cdot M^i$$

OBLIVIOUS CIPHERTEXT COMPRESSION

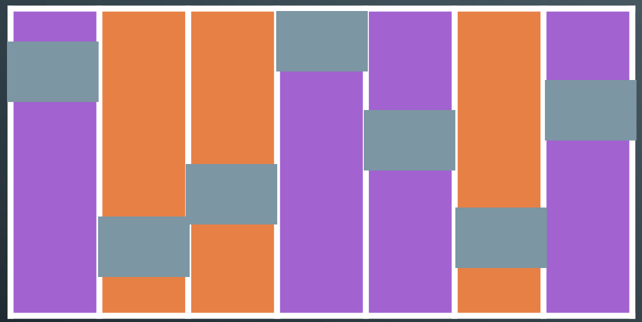
Decompress: (client knows nonzero indices)



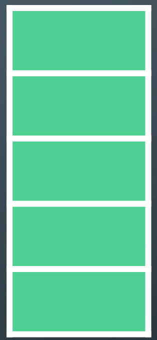


OBLIVIOUS CIPHERTEXT COMPRESSION

Decompress: (client knows nonzero indices)



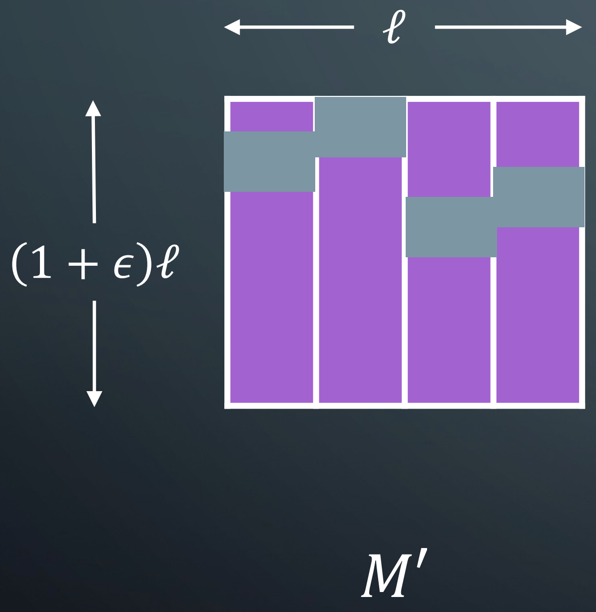
M

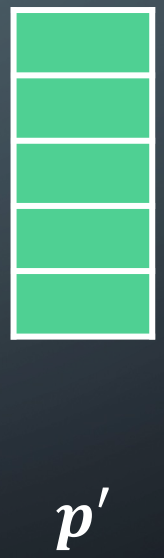


$\boldsymbol{p}'$

OBLIVIOUS CIPHERTEXT COMPRESSION

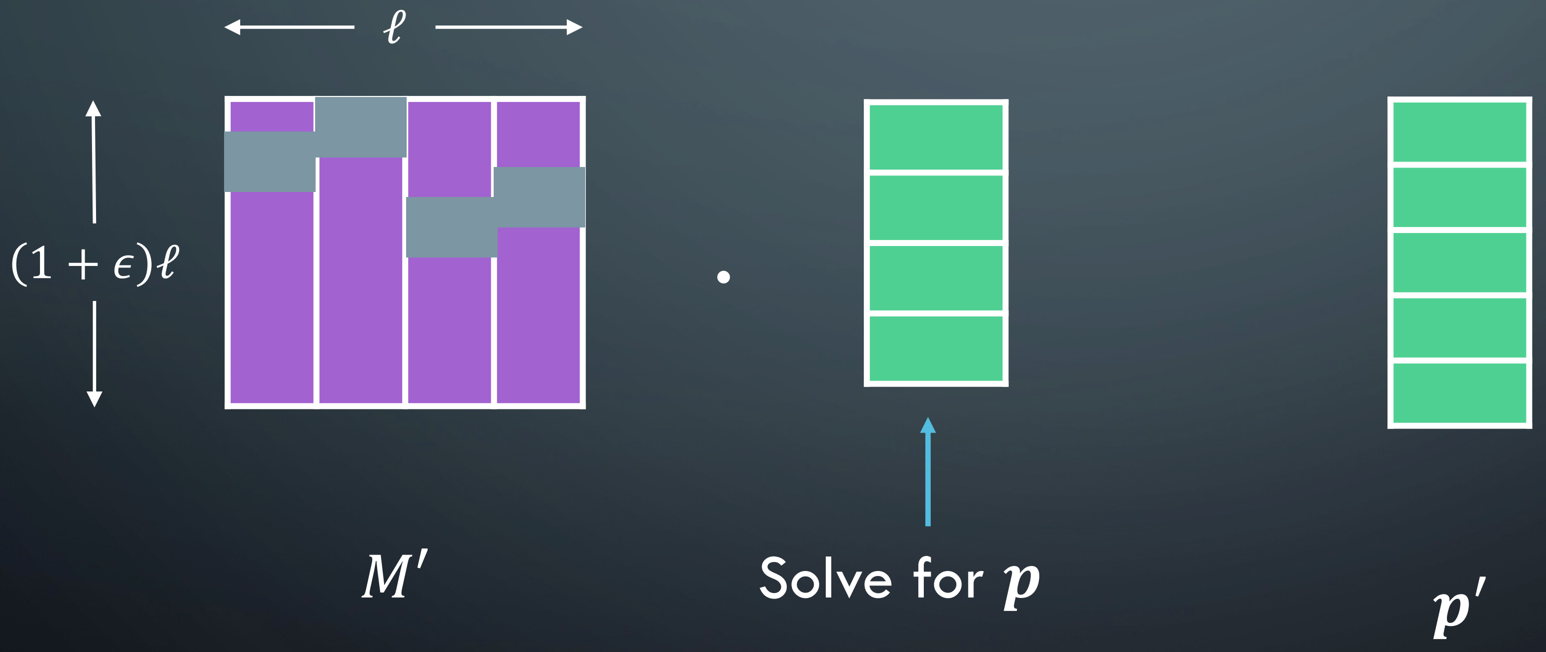
Decompress: (client knows nonzero indices)





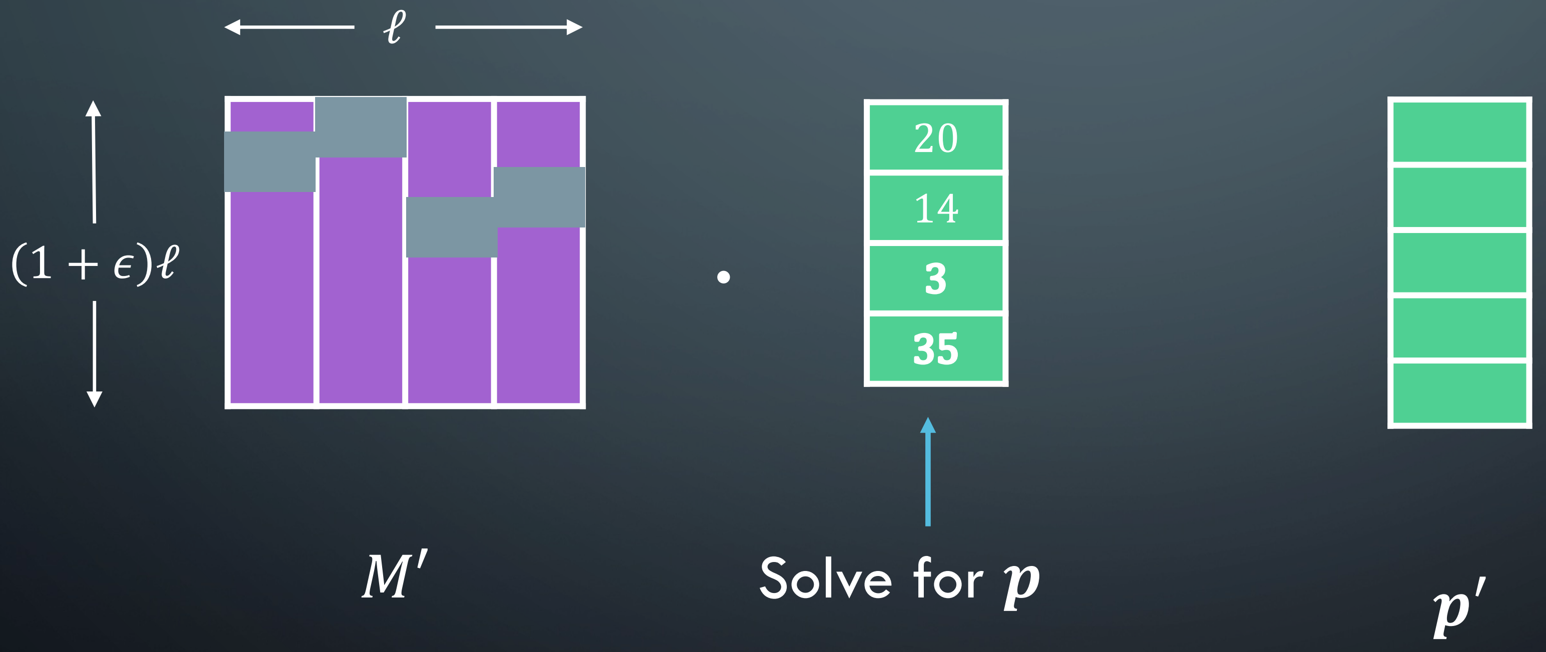
OBLIVIOUS CIPHERTEXT COMPRESSION

Decompress: (client knows nonzero indices)



OBLIVIOUS CIPHERTEXT COMPRESSION

Decompress: (client knows nonzero indices)



OBLIVIOUS CIPHERTEXT DECOMPRESSION

PIR Requests

$\boldsymbol{m} =$

20	5	0	0	14	3	0

ℓ-out-of-n non-zero

- Client compresses and encrypts $\boldsymbol{m} = (m_1, \dots, m_n)$ to $\boldsymbol{c}$ of size $\ell + \epsilon$, ϵ small

$\boldsymbol{c} =$

- Server decompresses $\boldsymbol{c}$ to $\mathbf{c}' = (c_1, \dots, c_n)$ such that if $m_i \neq 0$, c_i encrypts m_i

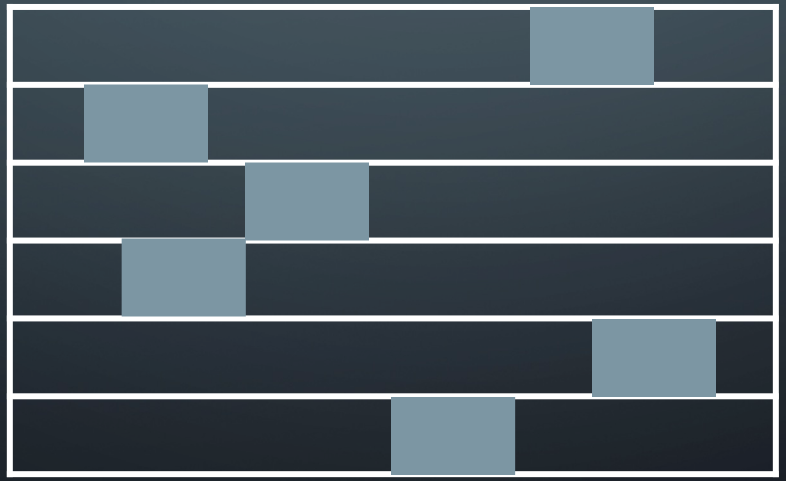
$\mathbf{c}' =$

$\boldsymbol{Enc(20)}$	$\boldsymbol{Enc(5)}$	$\boldsymbol{Enc(g_1)}$	$\boldsymbol{Enc(g_2)}$	$\boldsymbol{Enc(14)}$	$\boldsymbol{Enc(3)}$	$\boldsymbol{Enc(g_3)}$

Server must not learn non-zero m_i's or their locations

OUR MAIN TOOL: RANDOM-BAND MATRICES [DW19]

- Given $w \leq m$, sample $\ell \times m$ matrix M row-by-row as follows:
  - Pick random starting location from $1, \ldots, m - w$
  - Pick random w-bit string
  - Embed random w-bit string at starting location; everything else 0



OUR MAIN TOOL: RANDOM-BAND MATRICES [DW19]

- **Theorem:** There exists some constant $\epsilon > 0$ such that if $m = (1 + \epsilon) \cdot \ell$ and $w = O(\log \ell + \lambda/\epsilon)$, then randomly sampled $\ell \times m$ random band matrices have full row rank, time $O(\ell w)$ Gaussian elimination, and time $O(\ell w)$ vector multiplication, with probability $1 - 2^{-\lambda}$

OBLIVIOUS CIPHERTEXT DECOMPRESSION

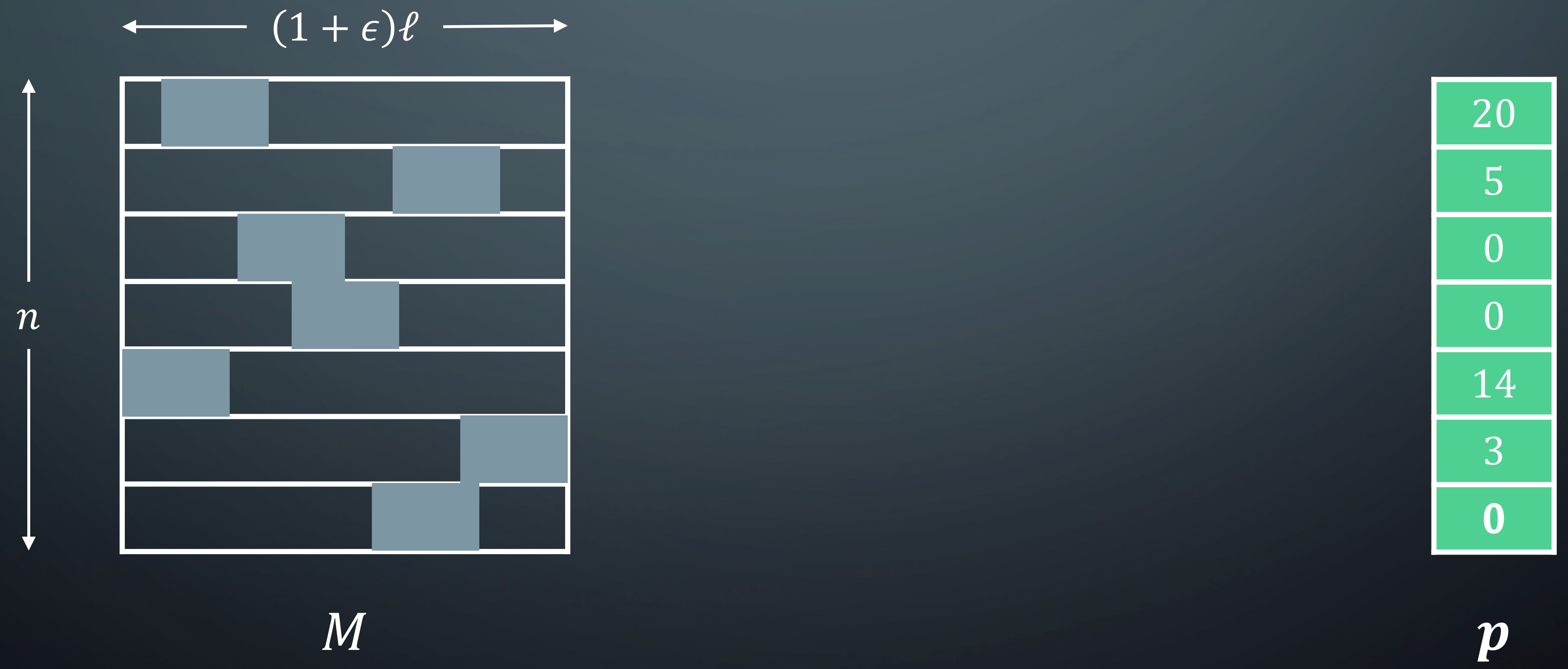
Compress (client knows nonzero indices):

20
5
0
0
14
3
0

$\boldsymbol{p}$

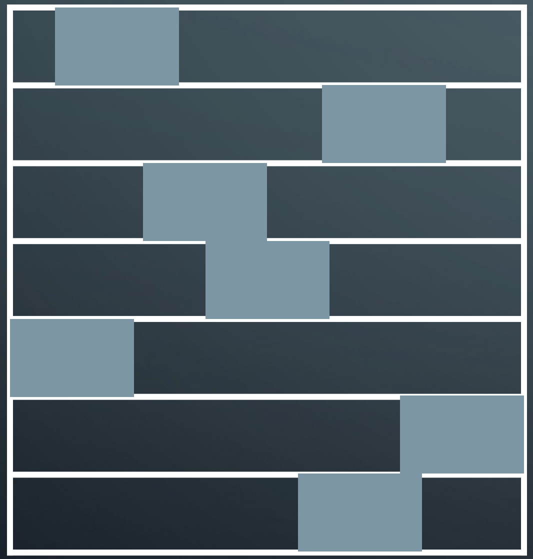
OBLIVIOUS CIPHERTEXT DECOMPRESSION

Compress (client knows nonzero indices):

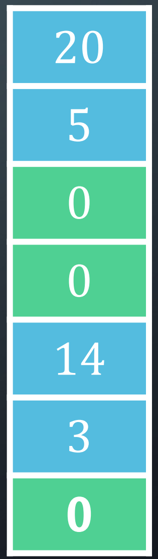


OBLIVIOUS CIPHERTEXT DECOMPRESSION

Compress (client knows nonzero indices):



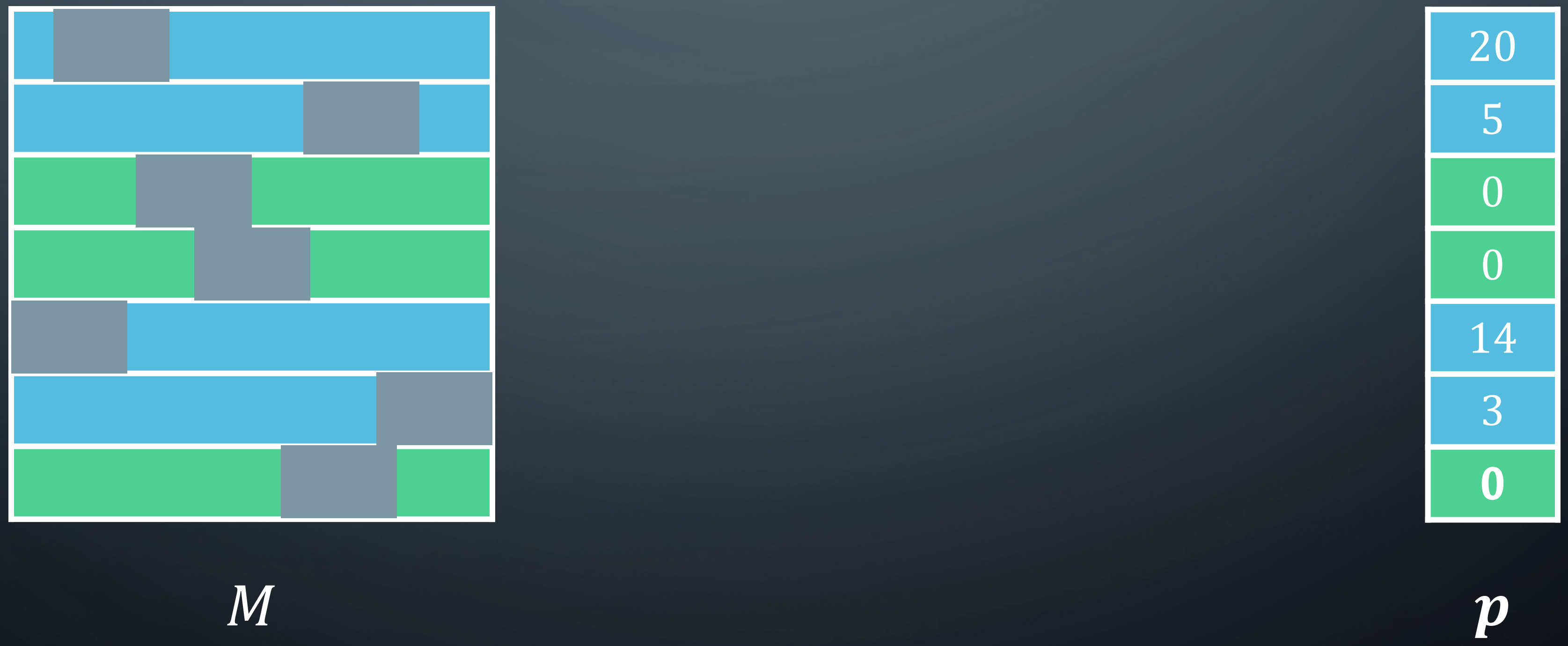
M



$\boldsymbol{p}$

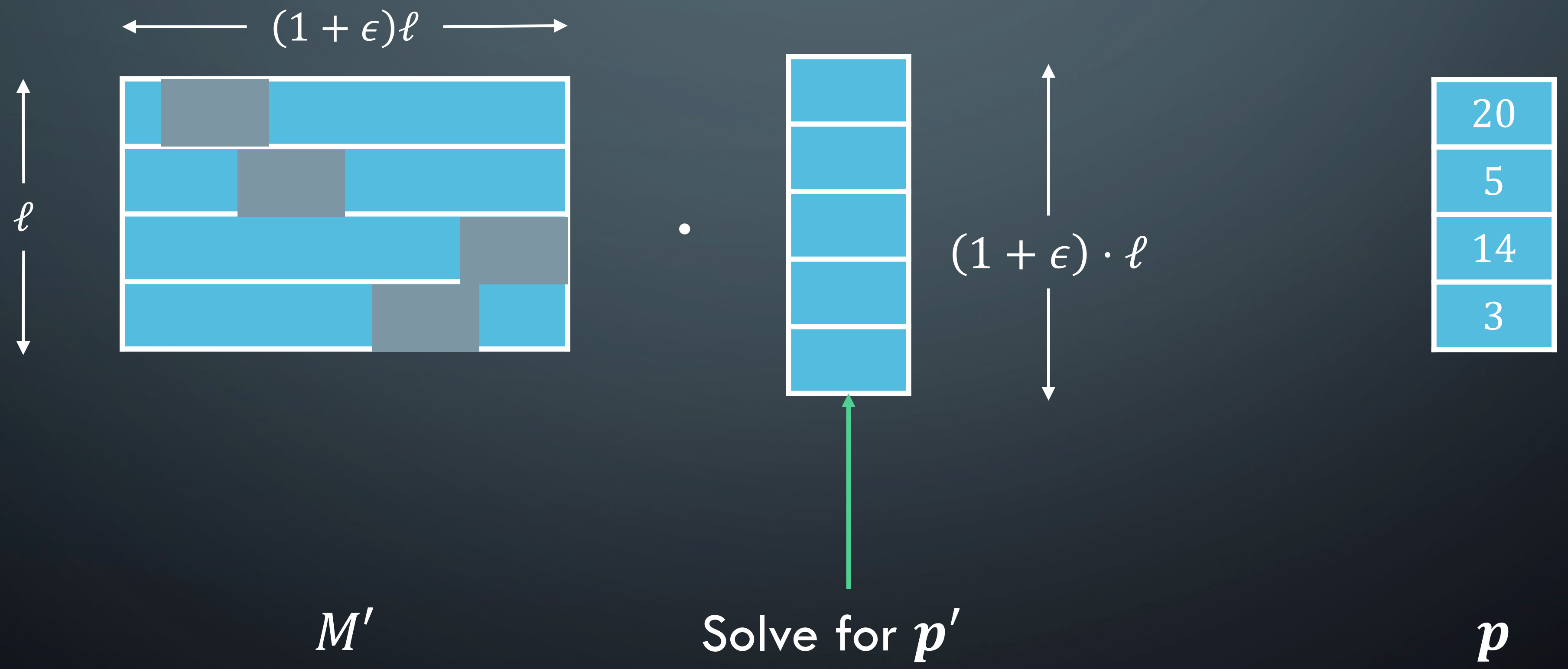
OBLIVIOUS CIPHERTEXT DECOMPRESSION

Compress (client knows nonzero indices):



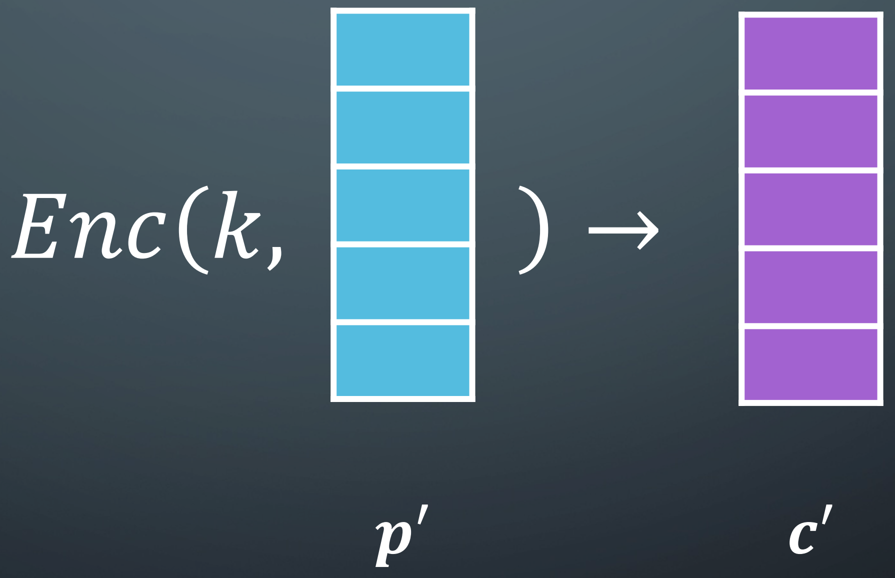
OBLIVIOUS CIPHERTEXT DECOMPRESSION

Compress (client knows nonzero indices):



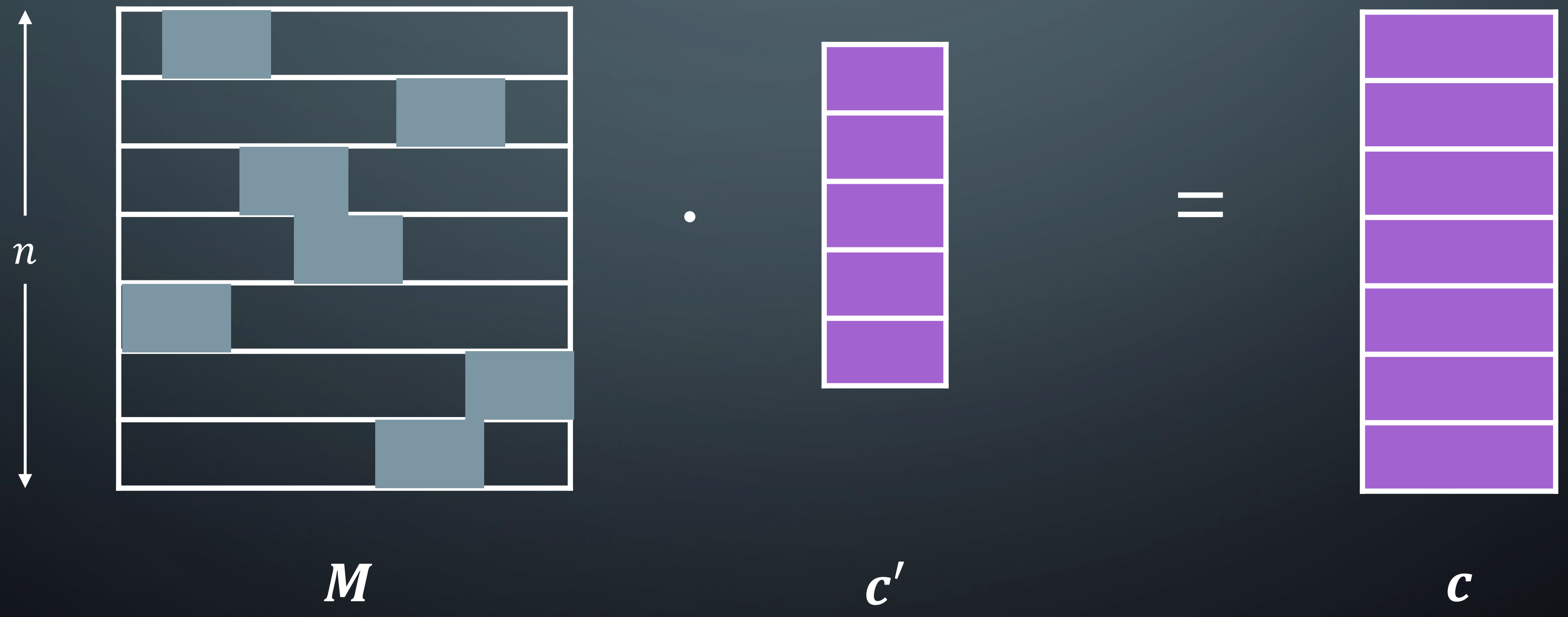
OBLIVIOUS CIPHERTEXT DECOMPRESSION

Compress (client knows nonzero indices):



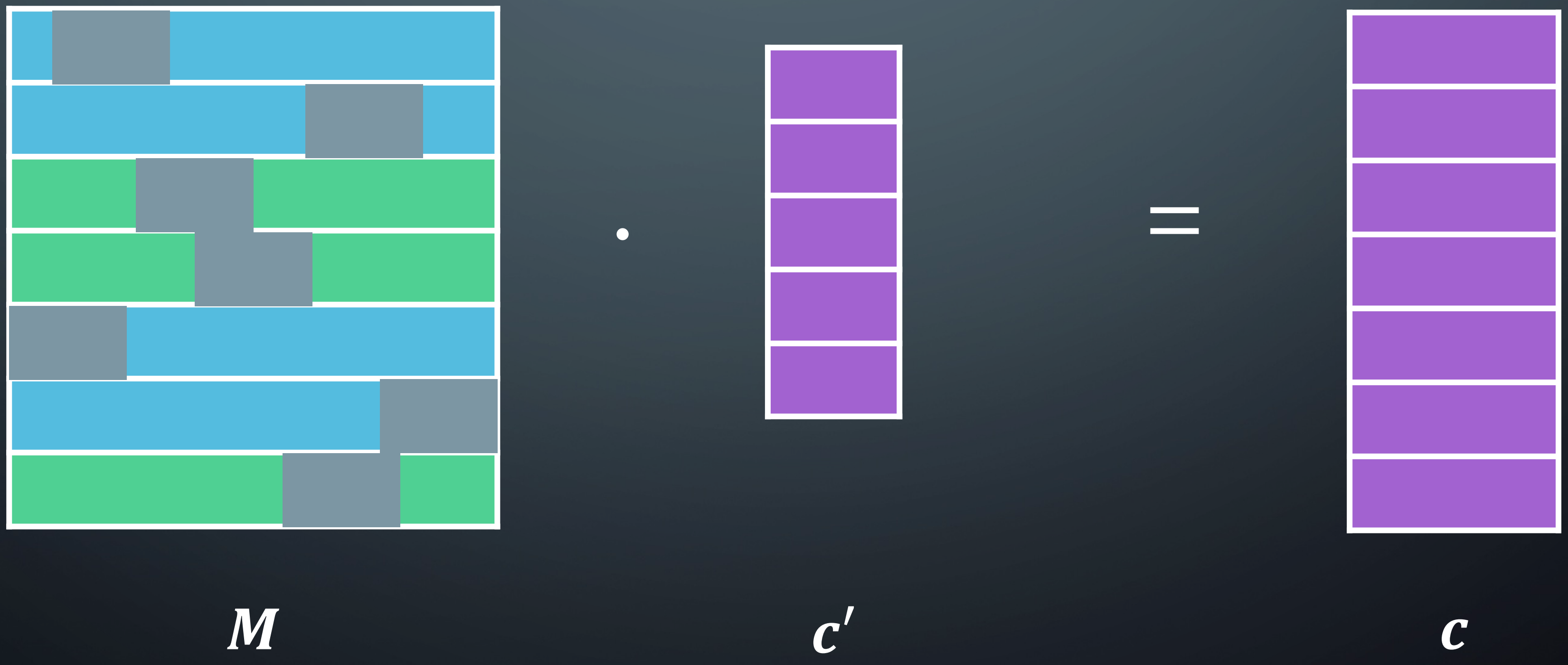
OBLIVIOUS CIPHERTEXT DECOMPRESSION

Decompress: (assume Enc is additively homomorphic)



OBLIVIOUS CIPHERTEXT DECOMPRESSION

Decompress: (assume Enc is additively homomorphic)



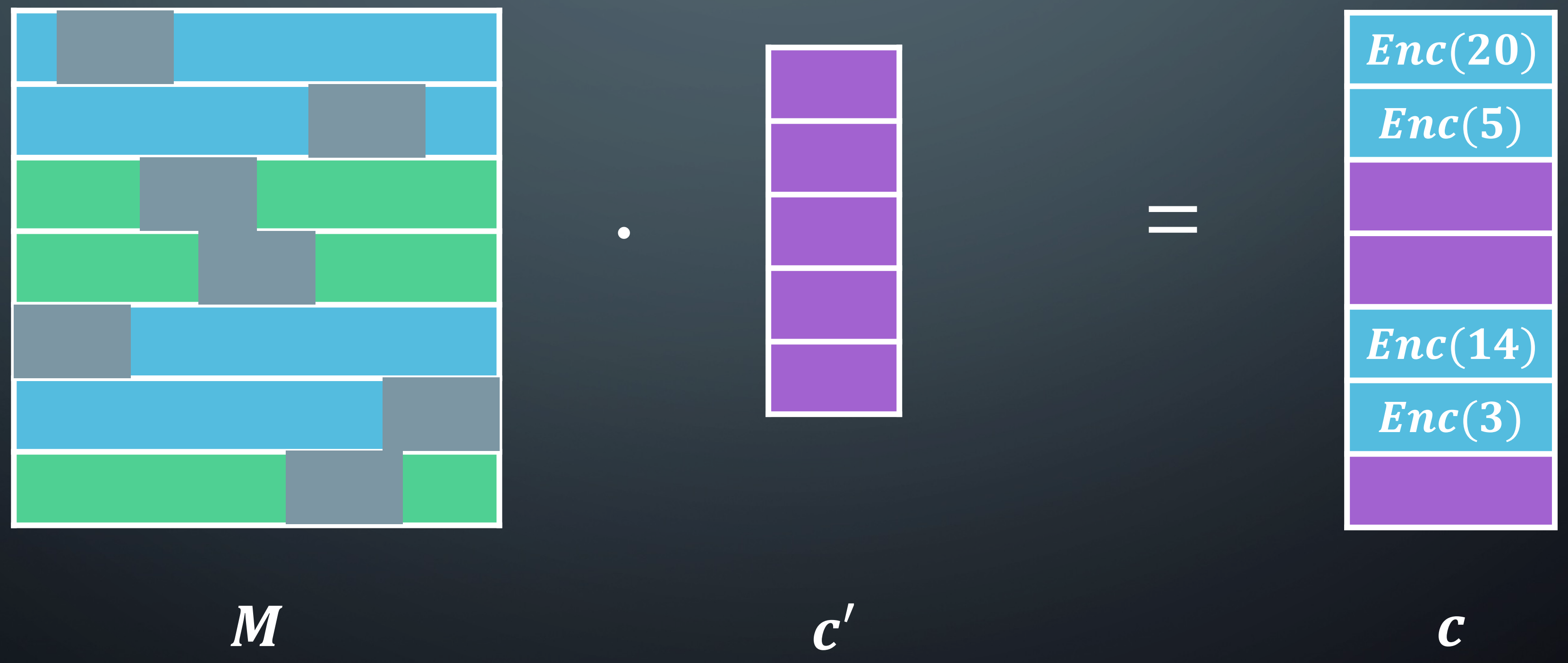
OBLIVIOUS CIPHERTEXT DECOMPRESSION

Decompress: (assume Enc is additively homomorphic)

$$\boldsymbol{M} \cdot \boldsymbol{c}' = \boldsymbol{c}$$

OBLIVIOUS CIPHERTEXT DECOMPRESSION

Decompress: (assume Enc is additively homomorphic)



TECHNICALITIES/ADDITIONAL RESULTS

- Cannot simultaneously use Oblivious Ciphertext Decompression and Compression in Batch PIR
  - State of the art Batch PIR uses FHE---accounting for added noise from both techniques makes ciphertexts too big
- Our techniques are compatible with “vectorization” techinques of [MR23]
  - Can fit d requests/responses into a single ciphertext
- (Only) Oblivious Ciphertext Compression also applies to 2-server batch PIR
- From standard Batch PIR + OPRF => Labeled PSI framework, get better Labeled PSI

EXPERIMENTS

DB Entry Size	Batch Size & Schemes	Public Param Size	Request Size	Response Size	Total Server Time	Amortized Server Time	Total Client Time	Server Monetary Cost
8 KB	$\ell = 512$							
	Baseline	20.87 MB	1.81 MB	15.59 MB	**840 s**	**1.64 s**	**6.1 s**	$0.00646
	LSObvCompress	22.62 MB	1.81 MB	**10.92 MB**	890 s	1.74 s	6.7 s	**$0.00633**
	LSObvDecompress	20.87 MB	**1.40 MB**	15.59 MB	863 s	1.69 s	6.4 s	$0.00656
	$\ell = 1024$							
	Baseline	20.87 MB	3.35 MB	31.18 MB	**1,256 s**	**1.23 s**	**6.8 s**	$0.01043
	LSObvCompress	23.11 MB	3.35 MB	**21.84 MB**	1,369 s	1.34 s	8.2 s	**$0.01025**
	LSObvDecompress	20.87 MB	**2.55 MB**	31.18 MB	1,323 s	1.29 s	8.0 s	$0.01075
	$\ell = 2048$							
	Baseline	20.87 MB	3.96 MB	62.37 MB	**1,750 s**	**0.85 s**	**7.0s**	$0.01617
	LSObvCompress	23.43 MB	3.96 MB	**43.67 MB**	1,871 s	0.91 s	9.7 s	**$0.01520**
	LSObvDecompress	20.87 MB	**3.15 MB**	62.37 MB	1,812 s	0.89 s	9.4 s	$0.01646
16 KB	$\ell = 512$							
	Baseline	20.87 MB	1.81 MB	31.18 MB	**1,286 s**	**2.51 s**	**6.3 s**	$0.01047
	LSObvCompress	22.62 MB	1.81 MB	**21.84 MB**	1,348 s	2.63 s	8.4 s	**$0.00999**
	LSObvDecompress	20.87 MB	**1.40 MB**	31.18 MB	1,308 s	2.55 s	8.3 s	$0.01056
	$\ell = 1024$							
	Baseline	20.87 MB	3.35 MB	62.37 MB	**1,775 s**	**1.73 s**	**7.9 s**	$0.01626
	LSObvCompress	23.11 MB	3.35 MB	**43.69 MB**	1,929 s	1.88 s	9.6 s	**$0.01548**
	LSObvDecompress	20.87 MB	**2.55 MB**	62.37 MB	1,881 s	1.83 s	9.4 s	$0.01681
	$\ell = 2048$							
	Baseline	20.87 MB	3.96 MB	124.74 MB	**2,634 s**	**1.29 s**	**8.5 s**	$0.02694
	LSObvCompress	23.43 MB	3.96 MB	**87.34 MB**	2,773 s	1.35 s	12.4 s	**$0.02439**
	LSObvDecompress	20.87 MB	**3.15 MB**	124.74 MB	2,746 s	1.34 s	12.4 s	$0.02752

Figure 5: Evaluations of Spiral Batch PIR [14, 58] with and without our compression techniques, LSObvCompress and LSObvDecompress with $\epsilon = 0.05$. We fix the number of entries to $n = 1$ million for all our results.

Batch Size & Schemes	Response Size	Server Time	Client Time	Server Monetary Cost
$\ell = \mathbf{512}$				
Baseline	221 KB	**9.63** s	**0.01 s**	\$0.000076
LSObvCompress	**155 KB**	9.69 s	0.08 s	**\$0.000070**
$\ell = \mathbf{1024}$				
Baseline	442 KB	**9.76 s**	**0.01 s**	\$0.000096
LSObvCompress	**310 KB**	9.92 s	0.16 s	**\$0.000085**
$\ell = \mathbf{2048}$				
Baseline	885 KB	**9.79 s**	**0.01 s**	\$0.000136
LSObvCompress	**619 KB**	10.09 s	0.33 s	**\$0.000114**
$\ell = \mathbf{4096}$				
Baseline	1,769 KB	**9.81 s**	**0.01 s**	\$0.000216
LSObvCompress	**1,238 KB**	10.53 s	0.78 s	**\$0.000172**
$\ell = \mathbf{8192}$				
Baseline	3,539 KB	**9.82 s**	**0.01 s**	\$0.000375
LSObvCompress	**2,477 KB**	11.13 s	1.80 s	**\$0.000287**

Figure 7: Comparison of DPF based two server batch PIR protocol [2] with and without LSObvCompress ($\epsilon = 0.05$). We fix the number of database entries to $n = 1$ million and each entry size to 288 B for all our results.

Label Size & Schemes	Total Online Comm.	Total Online Time	Server Monetary Cost
512 B			
Cong *et al.* [28]	33.2 MB	**169 s**	$0.00397
LSObvCompress	**11.4 MB**	304 s	**$0.00279**
1024 B			
Cong *et al.* [28]	66.1 MB	**331 s**	$0.00787
LSObvCompress	**11.6 MB**	355 s	**$0.00311**
1536 B			
Cong *et al.* [28]	103.6 MB	535 s	$0.01244
LSObvCompress	**11.9 MB**	**446 s**	**$0.00367**

Figure 8: Comparisons of Cong *et al.* [28]'s labeled PSI and our LSObvCompress-based PSI with $\epsilon = 0.05$. We fix the size of the sender's set to 1 million and the receiver's set to 512.

THANKS!