Communication-Computation Trade-offs in PIR

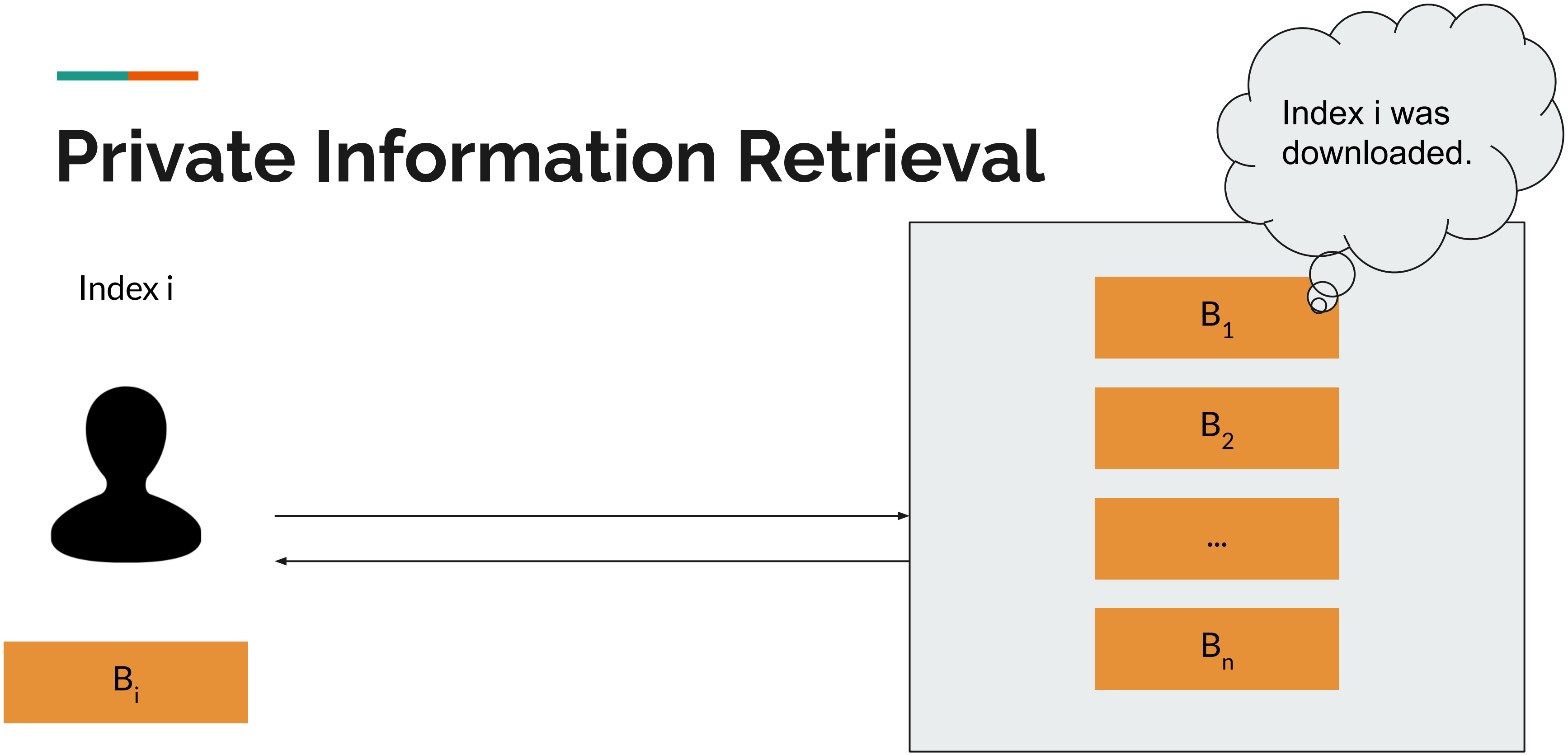
Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth and **Kevin Yeo**

Google
Humboldt-Universität zu Berlin

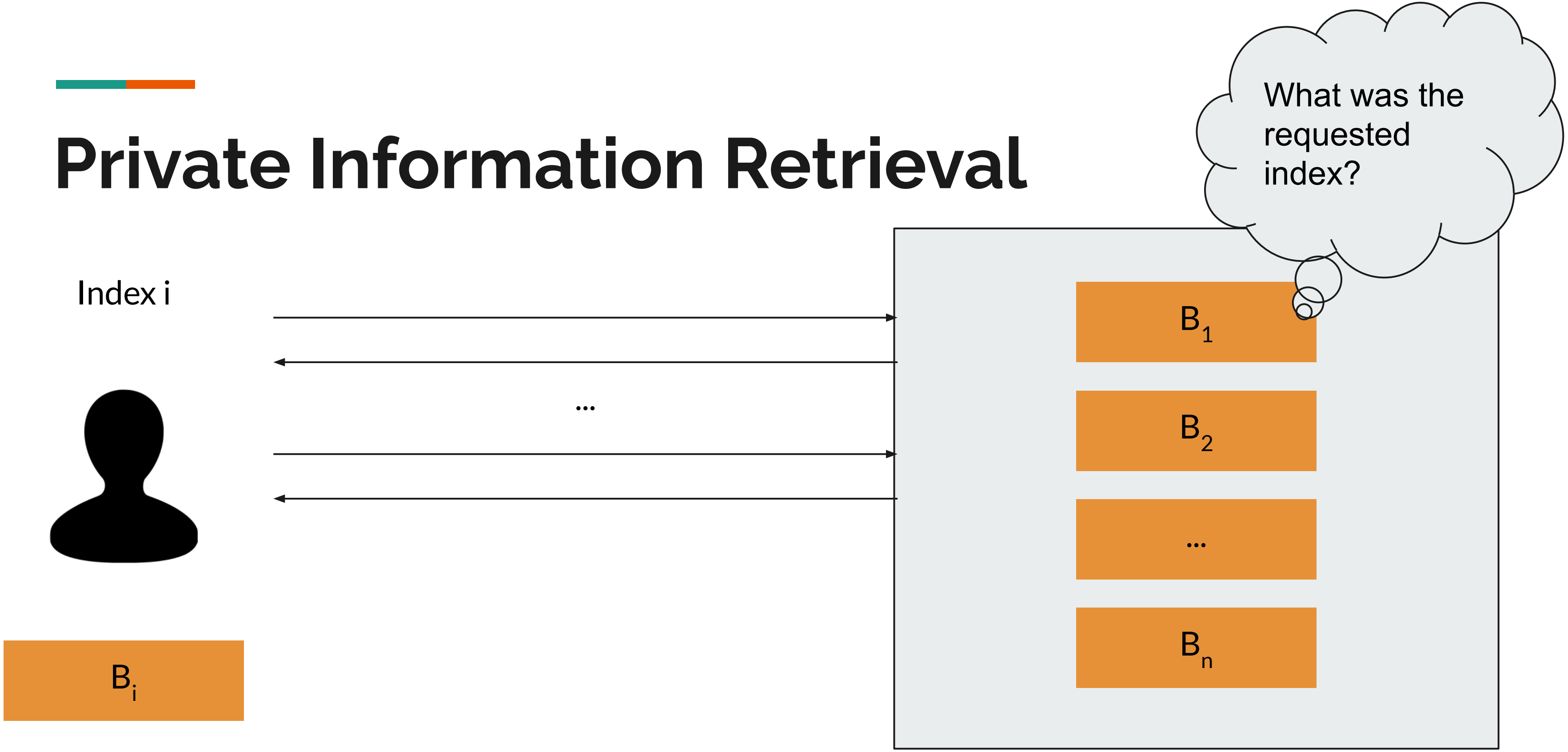
Outline

- What is Private Information Retrieval (PIR)?
- Our Contributions
  - MulPIR: Improved Communication
  - Gentry-Ramzan Improvements
  - Sparse PIR

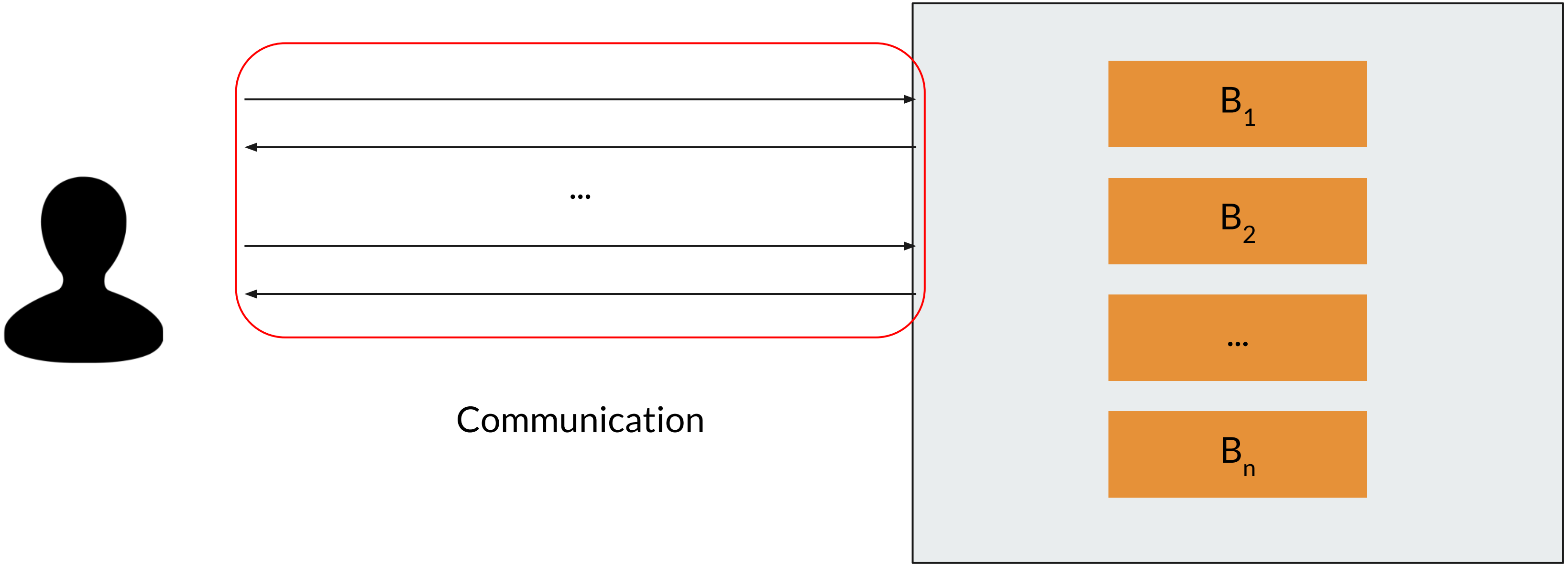
Private Information Retrieval



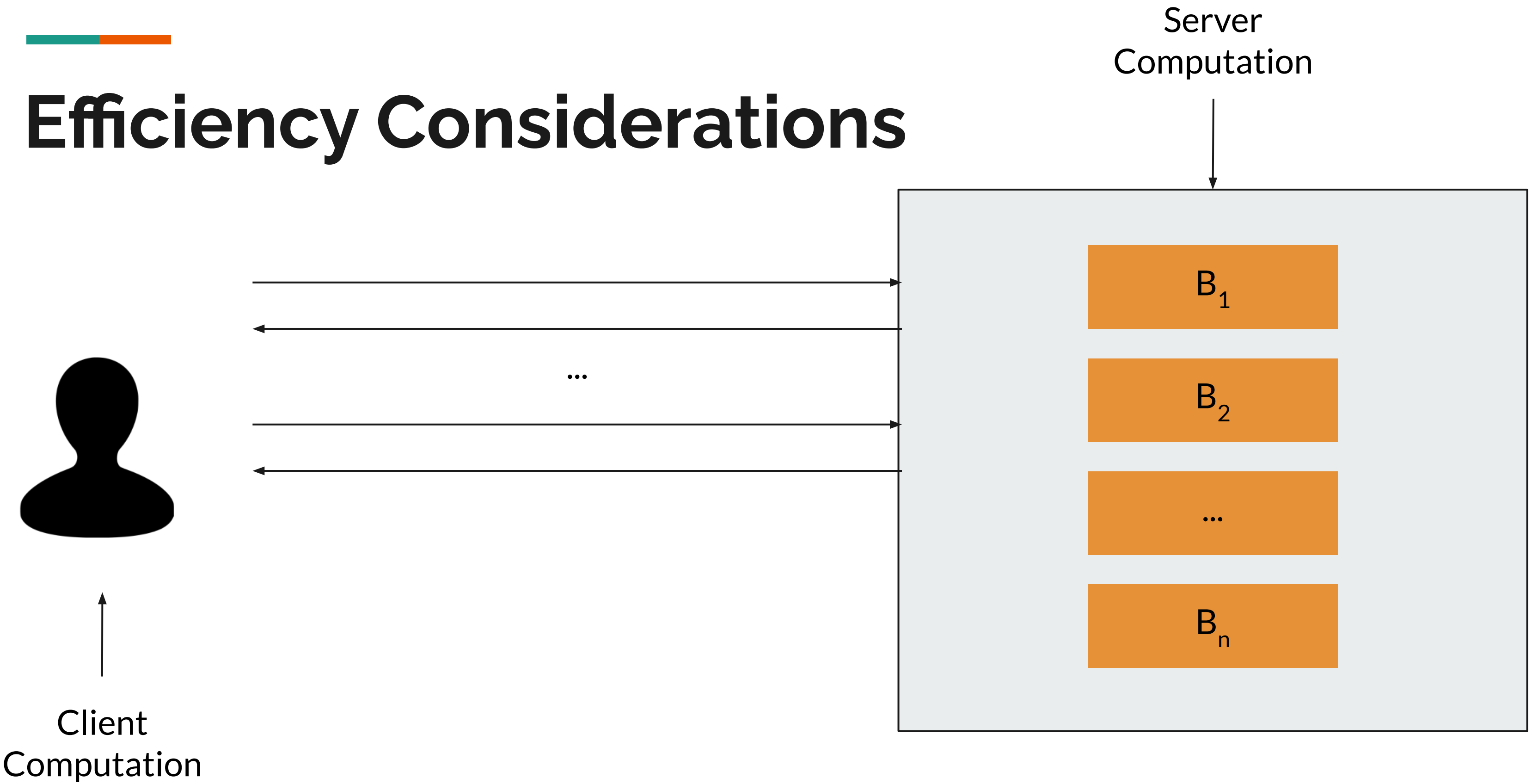
Private Information Retrieval



Efficiency Considerations



Efficiency Considerations



Communication-Computation Trade-offs

$\uparrow$ Communication

$\downarrow$ Computation

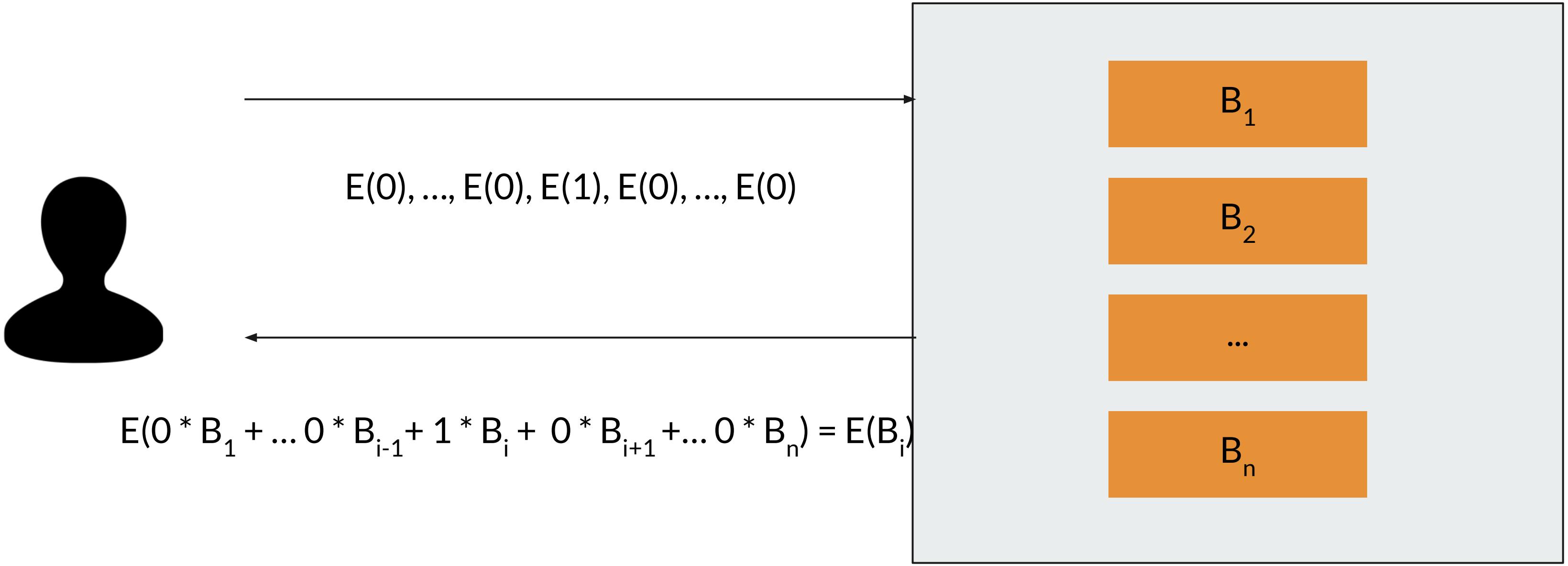
$\downarrow$ Communication

$\uparrow$ Computation

What is the best trade-off?

- Completely depends on context
- Typically, client computation must be small as querier is a user device
  - Example: Mobile phones querying a cloud storage provider
- Can estimate best trade-off using monetary costs
  - Example: Cloud computing prices

Homomorphic Encryption-based PIR



Recursion for PIR

	Enc(0)	Enc(0)	Enc(1)	Enc(0)
Enc(0)	B_1	B_2	B_3	B_4
Enc(0)	B_5	B_6	B_7	B_8
Enc(1)	B_9	B_{10}	$\boxed{B_{11}}$	B_{12}
Enc(0)	B_{13}	B_{14}	B_{15}	B_{16}

MulPIR: Improved Communication

Improve upload by up to 75% and download up to 80% with minimal computational increases over prior SealPIR implementation of Angel, Chen, Laine, Setty ‘18.

MulPIR: Improved Communication

Improve upload by up to 75% and download up to 80% with minimal computational increases over prior SealPIR implementation of Angel, Chen, Laine, Setty ‘18.

1. Use secret key encryption on client-side.
2. **Replace long randomness with a short PRG seed.**
3. Compress downloaded ciphertext using modulus switching.
4. **Improved oblivious expansion.**
5. Leverage multiplicative homomorphism.

Expandable Randomness using PRG

- Private key encryption is of the form ($\mathbf{c}_0$, $\mathbf{c}_1$) where each element in R/qR.
- $\mathbf{c}_0$ is a uniformly random element independent of public and private keys
- Replace $\mathbf{c}_0$ with a PRG seed S.
- **Reduces upload by half already!**

Improved Oblivious Expansion

SealPIR introduced the notion of oblivious expansion. Instead of a single ciphertext per bit, encrypt multiple bits per ciphertext.

Without Oblivious Expansion:

E(0), …, E(0), E(1), E(0), …, E(0) with N ciphertexts

With Oblivious Expansion:

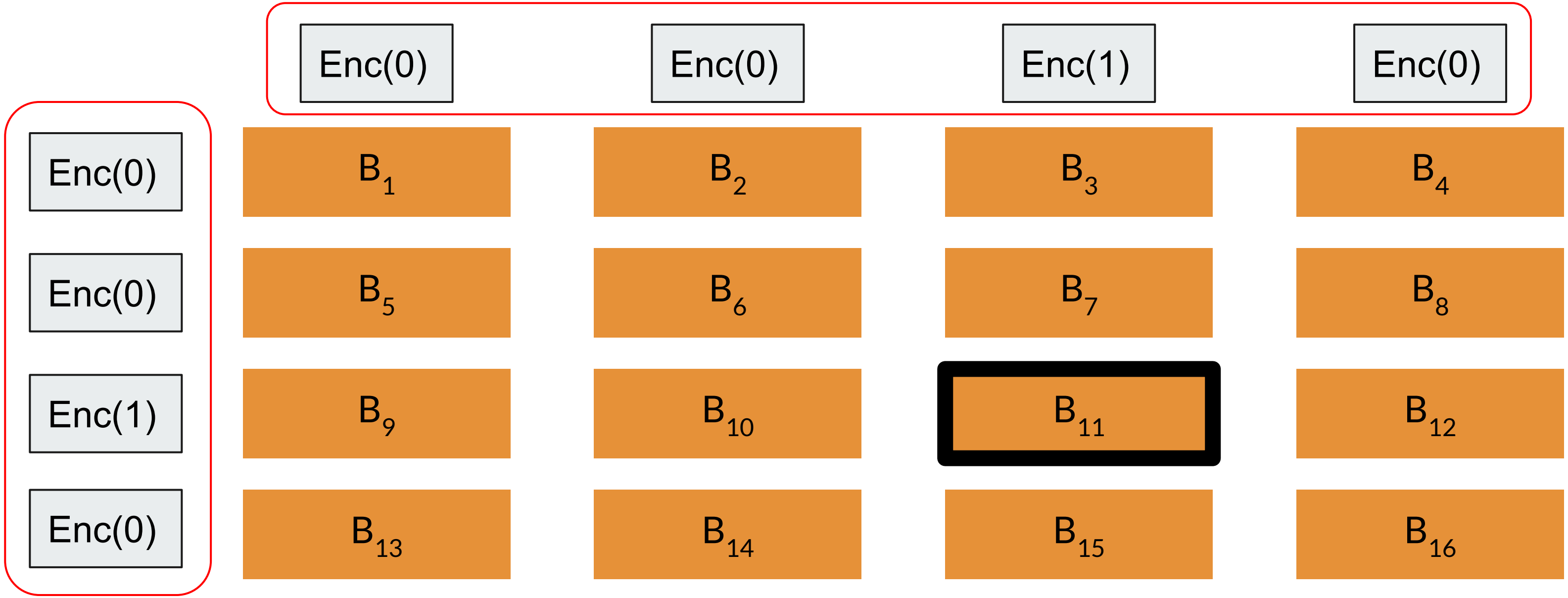
E(0, …, 0), E(0, …, 0, 1, 0, …, 0), E(0, …, 0) with < N ciphertexts depending on parameters.

Server will obliviously expand compressed vector.

Limits of SealPIR's Oblivious Expansion

Limitation: To-be-compressed bit vector must have Hamming weight <= 1.

Limits of SealPIR’s Oblivious Expansion



Limits of SealPIR's Oblivious Expansion

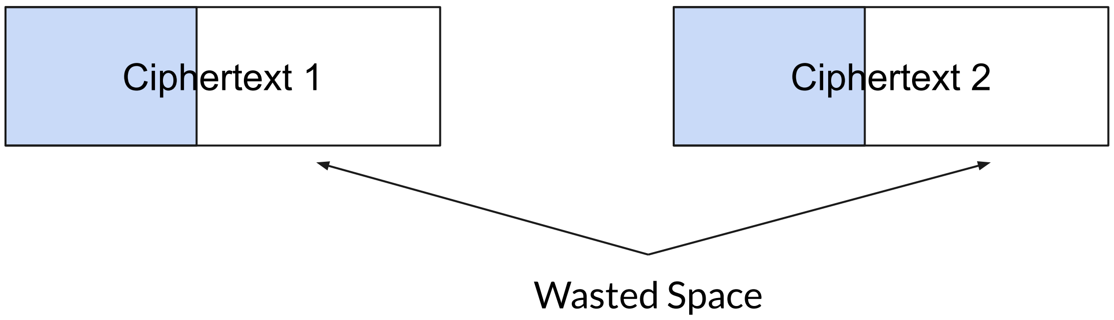
Limitation: To-be-compressed bit vector must have Hamming weight <= 1.

Limitation with Recursion: Must compress two sets of vectors separately. In the worst case, lots of wasted space.

Limits of SealPIR's Oblivious Expansion

Limitation: To-be-compressed bit vector must have Hamming weight <= 1.

Limitation with Recursion: Must compress two sets of vectors separately. In the worst case, lots of wasted space.



Improved Oblivious Expansion

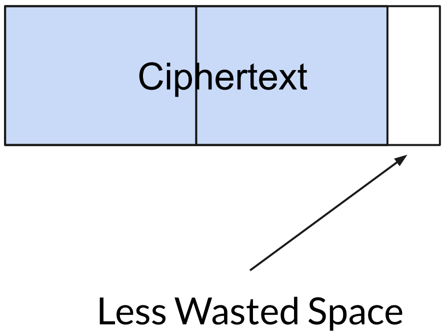
Improvement: To-be-compressed bit vector can have arbitrary Hamming weight.

Improvement with Recursion: Compress all uploaded bit vectors into one ciphertext.

Improved Oblivious Expansion

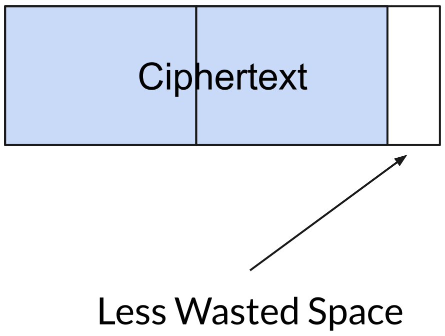
Improvement: To-be-compressed bit vector can have arbitrary Hamming weight.

Improvement with Recursion: Compress all uploaded bit vectors into one ciphertext.



Improved Oblivious Expansion

Observation: Server oblivious expansion is linear in the plaintext space. Compression and expansion work for arbitrary vectors (not just bit vectors!).



Experimental Evaluation

Table 3: Communication and CPU costs (in ms) of SealPIR and MulPIR (recursion $d = 2$) for a database of n elements of 288B.

	SealPIR [3] ($d = 2$)			SealPIR [3] ($d = 3$)			MulPIR ($d = 2$)			MulPIR ($d = 3$)		
Database size n	262144	1048576	4194304	262144	1048576	4194304	262144	1048576	4194304	262144	1048576	4194304
Actual number of rows after packing	*26215*	*104858*	*419431*	18725	74899	299594	*3693*	*14769*	*59075*	*4682*	*18725*	*74899*
Client Query	19	19	19	19	19	19	172	192	213	126	128	161
Server Expand	145	294	590	33	55	90	391	783	1610	396	395	841
Server Respond	1020	3520	12891	1136	3519	11554	1919	5213	16307	3268	11677	30501
Upload (kB)	61.4	61.4	61.4	92.2	92.2	92.2	122	122	122	130	130	130
Download (kB)	307	307	307	1966	1966	1966	119	119	119	130	130	130
Server Cost (US cents)	0.0033	0.0040	**0.0067**	0.017	0.017	0.020	**0.0026**	**0.0036**	0.0069	0.0031	0.0054	0.011

Gentry-Ramzan PIR Improvements

Present improvements to Gentry-Ramzan PIR to enable tunable communication-computation trade-offs.

Reduces server computation by up to 85% for larger communication sizes.

Experimental Evaluation

Table 5: Communication and computation costs for PIR protocols for two databases, without recursion.

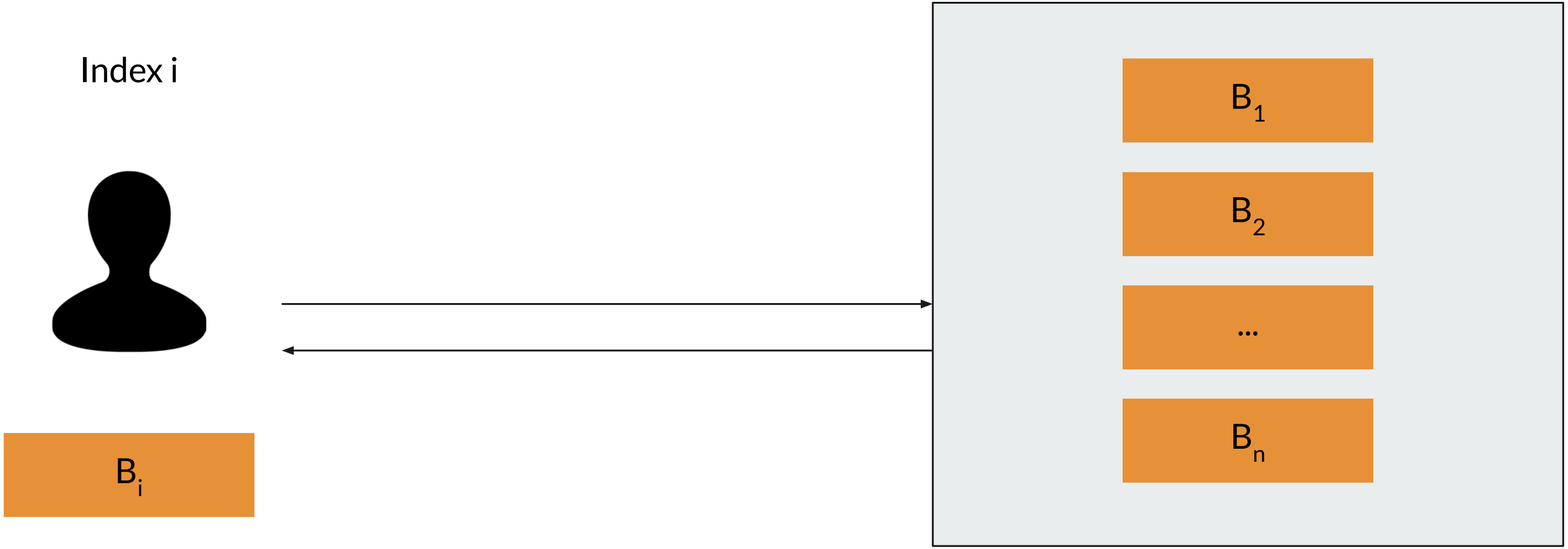
	# chunks	Communication (kB)		Computation (ms)					Server Cost (US cents)
		upload	download	C.Setup	S.Setup	C.Create	S.Respond	C.Process	
1MB database: 5000 elements of 288B.									
MulPIR	1	14	21	0	39	154	3,910	0	0.0019
Gentry–Ramzan (1 generator)	5	0.5	1.3	0	1,532	3,294	51,803	377	0.0145
Client-Aided Gentry–Ramzan (15 generators)	5	4.1	1.3	0	1,540	2,688	5,495	381	0.0016
Client-Aided Gentry–Ramzan (50 generators)	5	13.1	1.3	0	1,594	3,966	2,988	393	**0.0011**
Client-Aided Gentry–Ramzan (100 generators)	5	25.8	1.3	0	1,796	7,980	2,904	417	0.0014
Damgård–Jurik ($s = 1$)	1	1,480	0.6	40,636	2	14,334	20,710	6	0.0382
ElGamal	72	280	8	283	29	893	10,105	26,544	0.0091
Private File Download – 3GB database: 10,000 elements of 307kB.									
MulPIR	100	79.4	1,385	0	88,815	198	34,388	23	**0.0417**
Client-Aided Gentry–Ramzan (50 generators)	4,955	13.1	1,259	6	1,347,036	28,684	5,221,052	355,940	1.4782
Damgård–Jurik ($s = 1$)	1,060	2,960	614	$\approx$ 80,000	$\approx$ 3,200	$\approx$ 28600	$\approx$ 42,000,000	$\approx$ 2,500	11.7451
ElGamal	76,800	280	4,300	$\approx$ 300	$\approx$ 88,800	$\approx$ 2250	$\approx$ 4,800,000	$\approx$ 30,715,200	1.4338

Median over 10 computations. The timings indicated with $\approx$ have been estimated on a smaller number of chunks to finish in a reasonable amount of time.

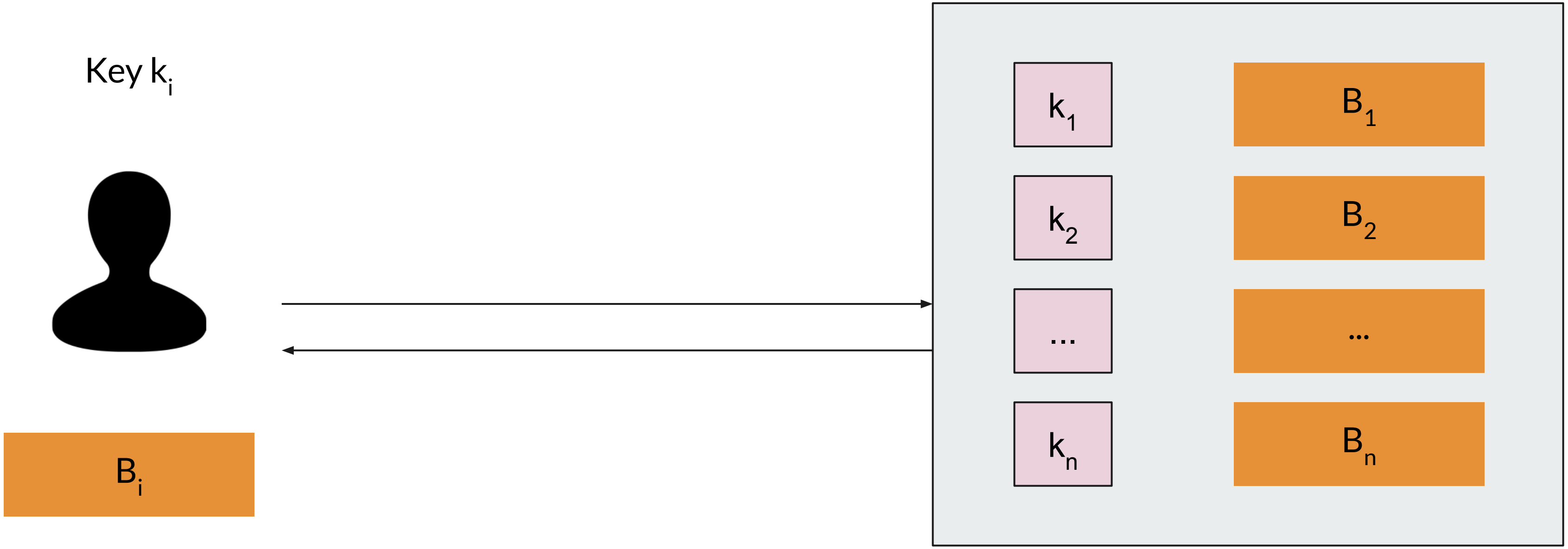
Sparse PIR to (Dense) PIR Transformation

Generic transformation from Sparse PIR to (Dense) PIR.

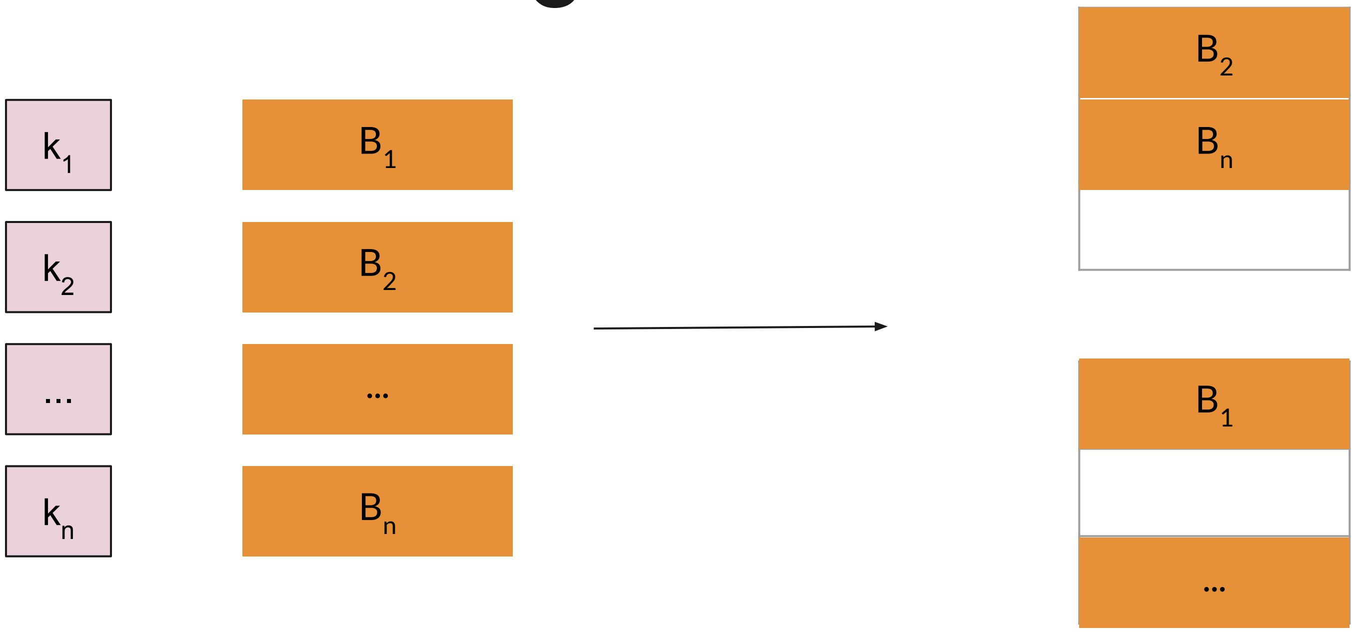
What is Sparse PIR?



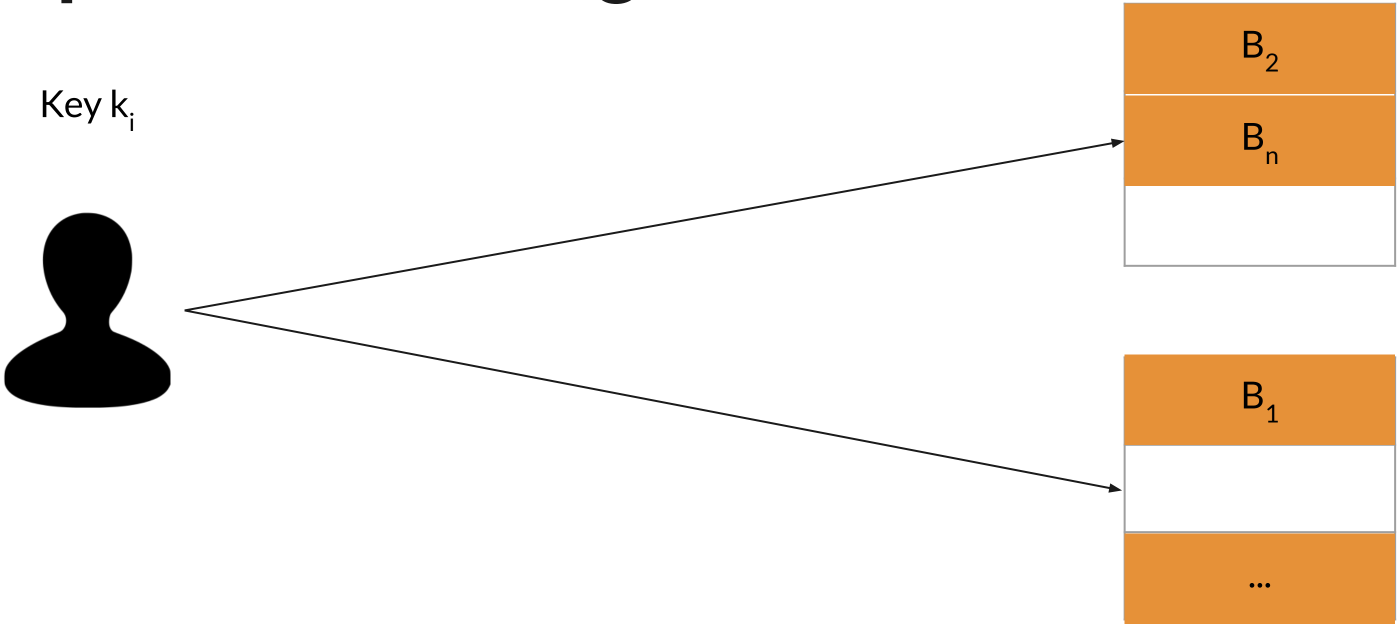
What is Sparse PIR?



Cuckoo Hashing



Sparse PIR using Dense PIR



Questions?