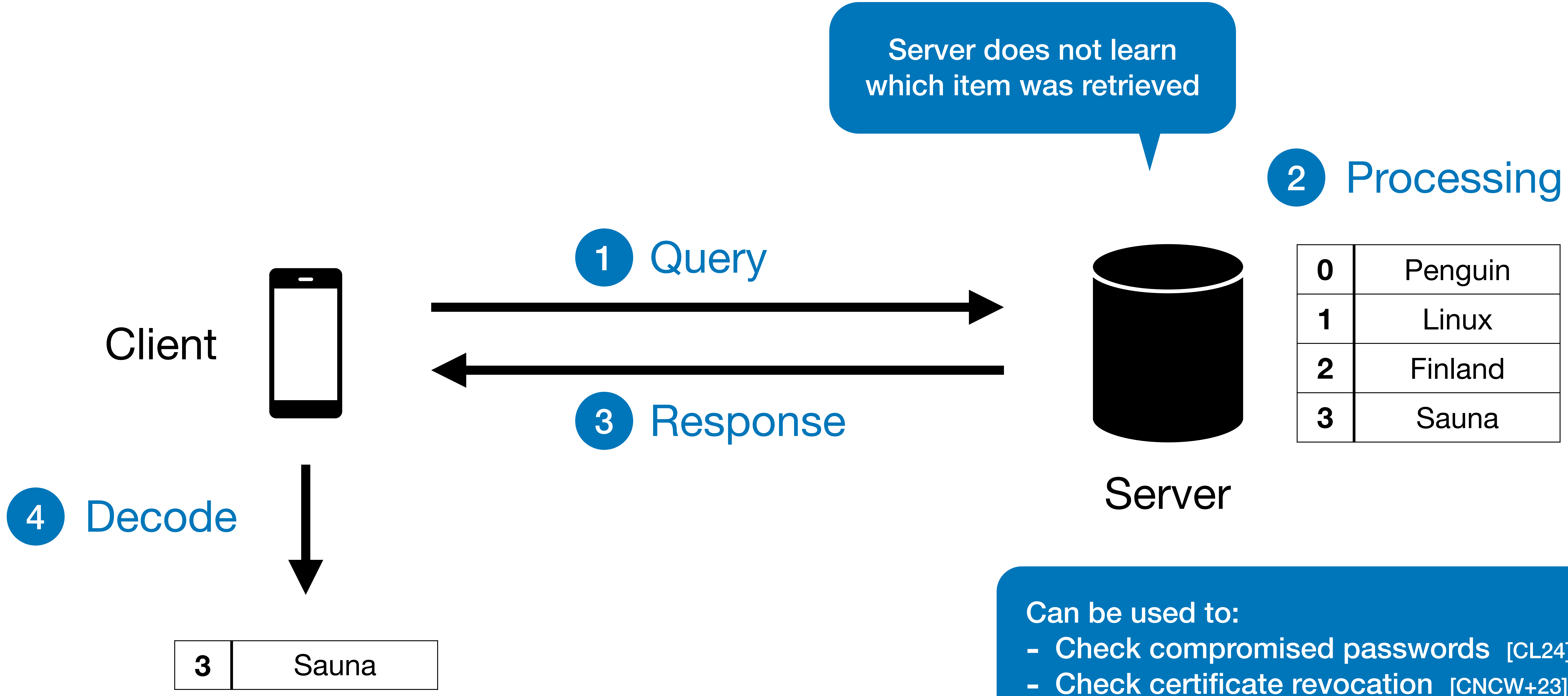
YPIR:
High-Throughput Single-Server PIR with Silent Preprocessing

Samir Jordan Menon (Blyss) and David J. Wu (UT Austin)

USENIX '24

[CGKS95]

Single-Server Private Information Retrieval (PIR)

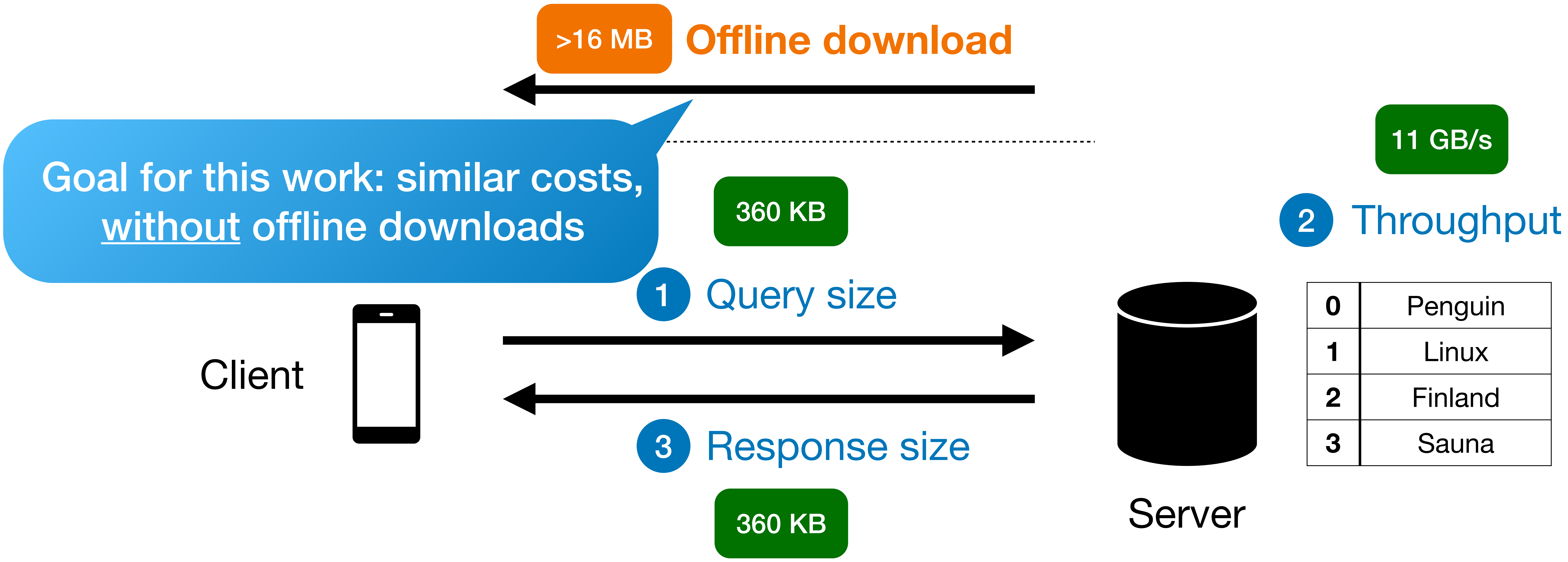


Can be used to:

- Check compromised passwords [CL24]
- Check certificate revocation [CNCW+23]
- Perform SCT auditing [HHCM+23]

[HHCM+23]

Costs of SimplePIR



(1) SimplePIR/DoublePIR [HHCM+23]

PIR based on *hints* that clients download offline

(2) HintlessPIR/Tiptoe = SimplePIR + hint packing [LMRS23/HDCZ23]

PIR *without* offline communication, but ~10× larger responses

(3) YPIR (this work) = SimplePIR/DoublePIR + better hint packing

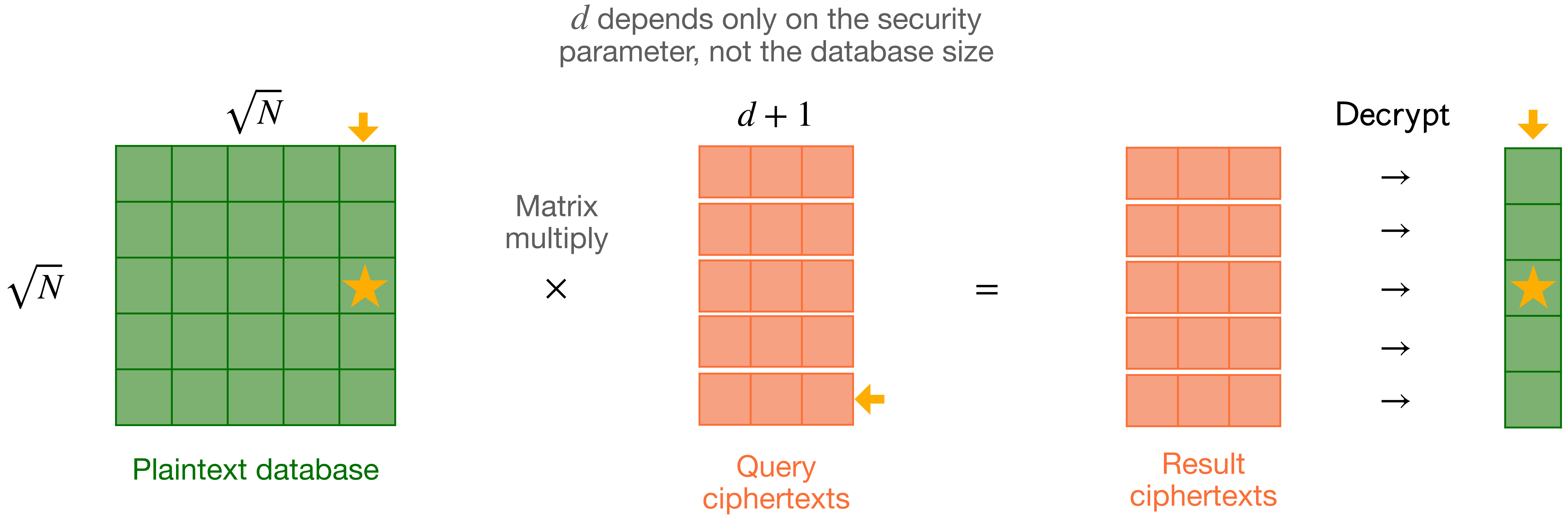
PIR *without* offline communication and *small responses*

For 1-bit retrieval: **similar costs to DoublePIR, with no hints!**
For large item retrieval: **8× smaller responses than HintlessPIR**

[HHCM+23]

DoublePIR: recurse one time on the result

(1) SimplePIR



Desired item = ★

Each row is an *additively homomorphic* LWE ciphertext

Last ciphertext encrypts “1”, the rest “0”

Query is $O(\sqrt{N})$

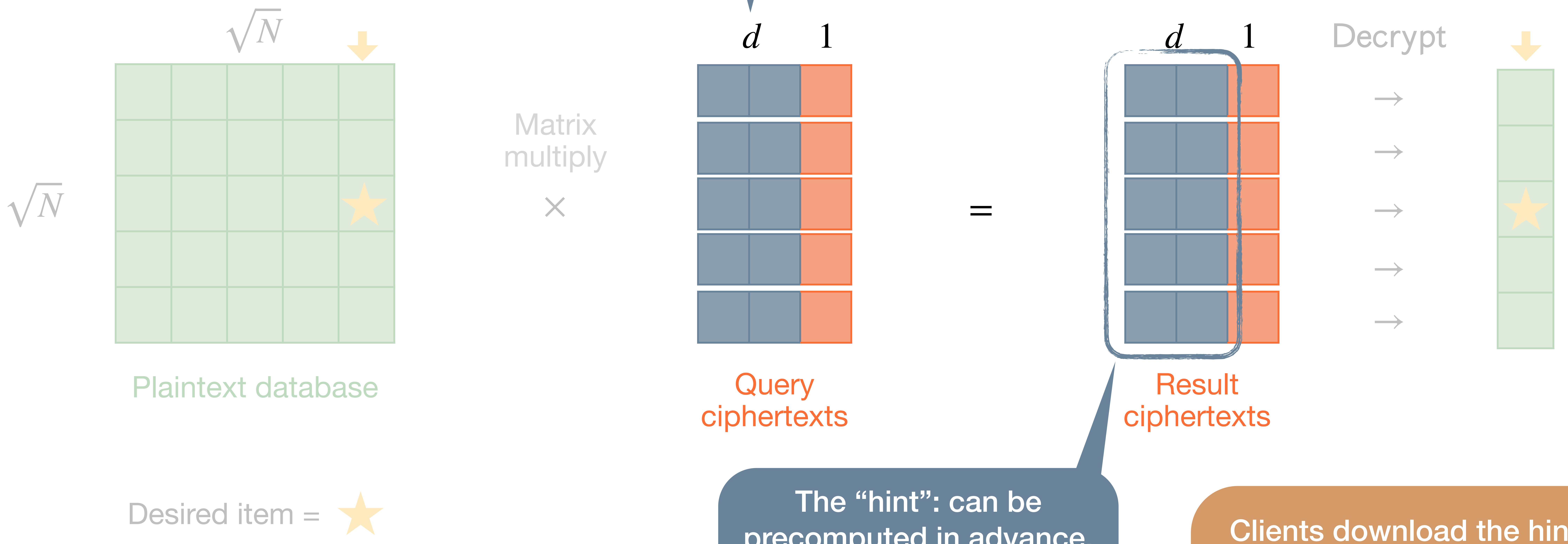
Result is an encryption of the target *column* of the database, containing the item of interest

Response is $O(\sqrt{N})$

[HHCM+23]

(1) SimplePIR

Key observation: *most* of query is pseudorandom, and can be *fixed* for all clients ($d = 1024$)



The “hint”: can be precomputed in advance, and depends only on the database contents

Clients download the hint offline. Clients *re-download* when database updates!

[HHCM+23]

(1) SimplePIR

The “hint”: can be precomputed in advance, and depends only on the database contents

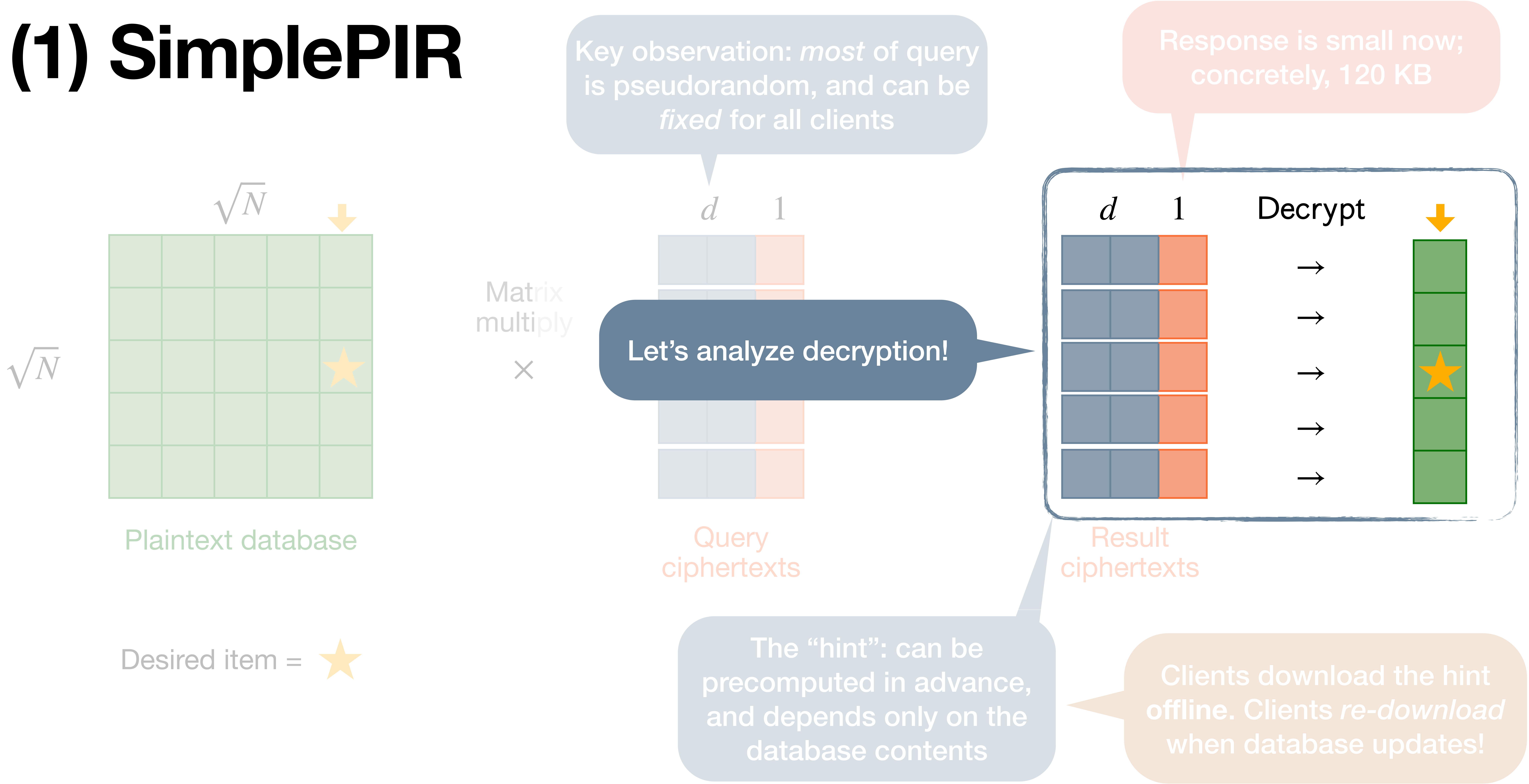
Clients download the hint offline. Clients *re-download* when database updates!

Megabytes of communication to every client on every update.

Goal for this work: similar costs, without offline downloads

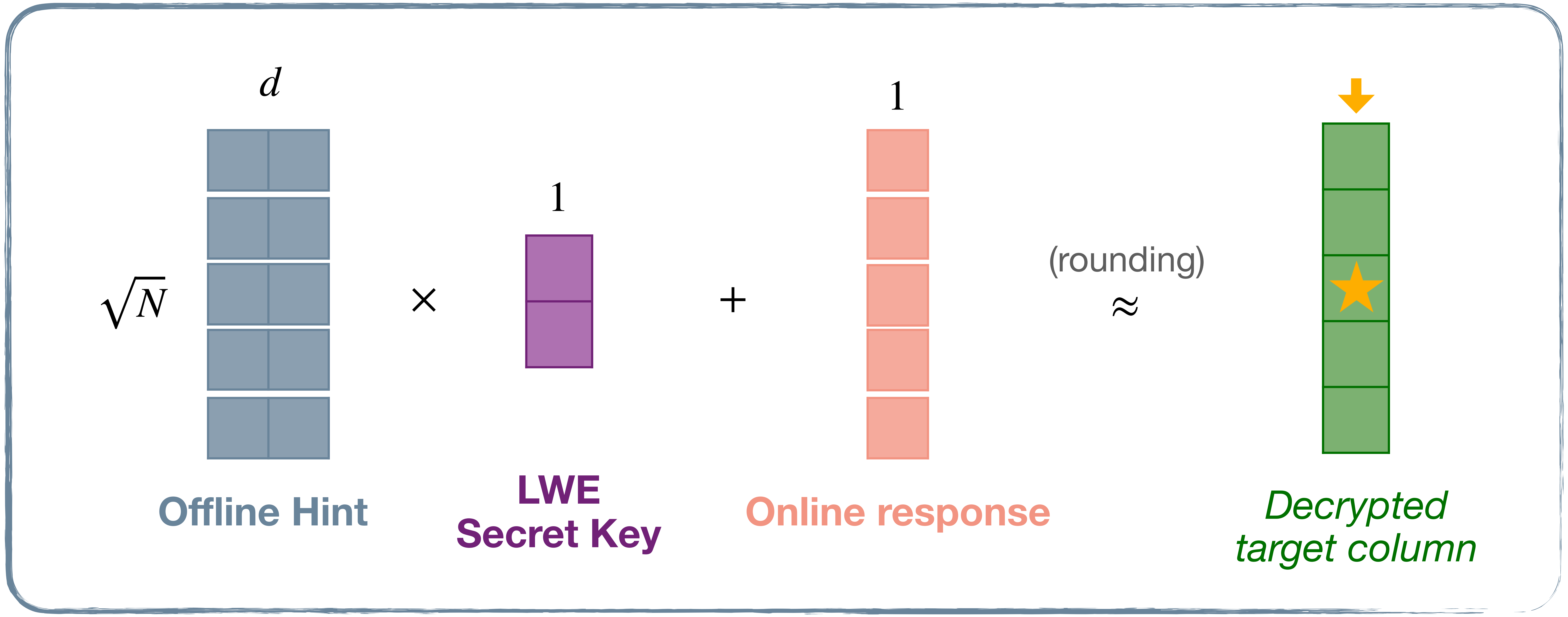
[HHCM+23]

(1) SimplePIR

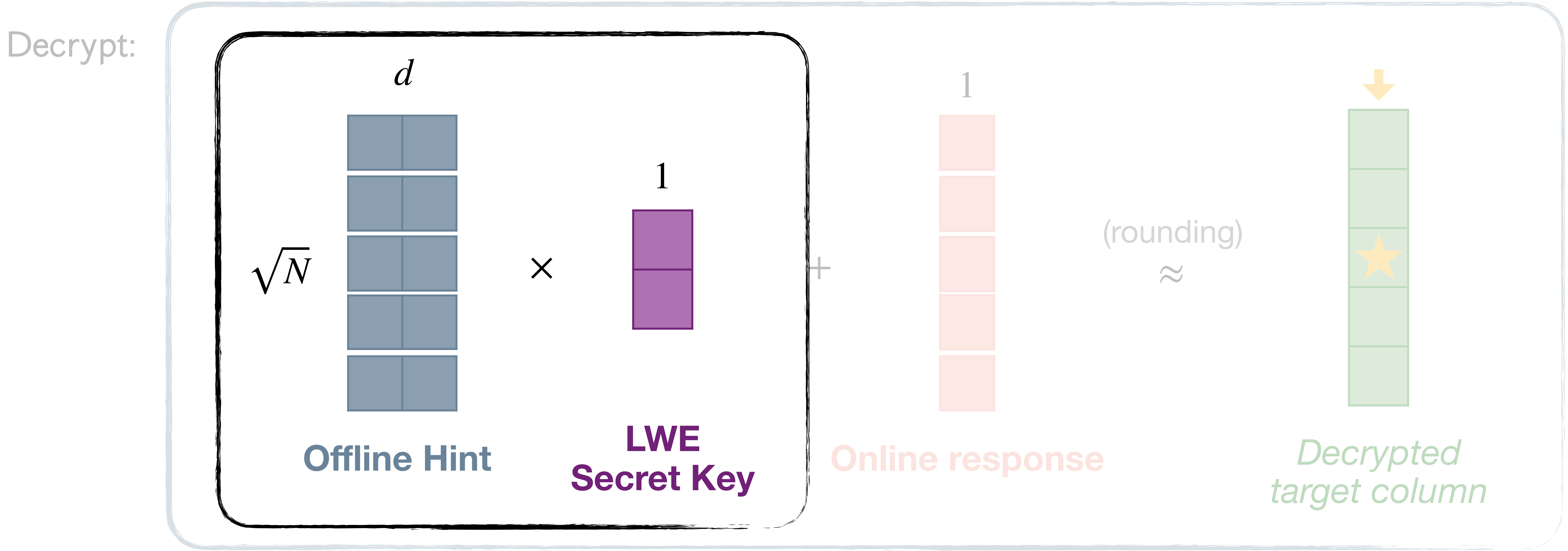


Analyzing decryption in SimplePIR

Decrypt:



Analyzing decryption in SimplePIR



Clients <u>don't need the whole hint</u> to decrypt!

They just need **offline hint** × **LWE secret key**.

How can clients get this inner product, without communicating the entire hint?

[LMRS23, HDCZ23]

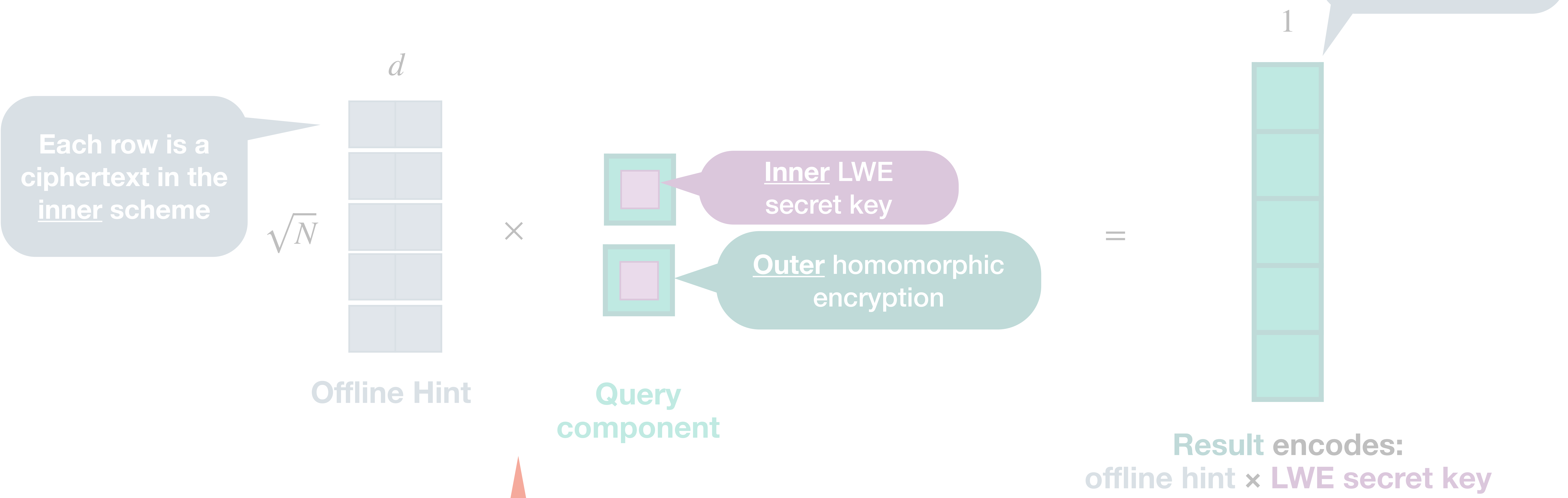
(2) HintlessPIR/Tiptoe: hint packing

Clients encrypt their <u>inner</u> LWE secret vector in another, **<u>outer</u> homomorphic encryption scheme based on Ring-LWE that is more compact**

Server treats the hint as a plaintext, and multiplies it by the encrypted secret vector in the **<u>outer</u> scheme**

[LMRS23, HDCZ23]

(2) HintlessPIR/Tiptoe: hint packing



Plaintext space of outer scheme must hold the *ciphertext* space of inner scheme (LWE)

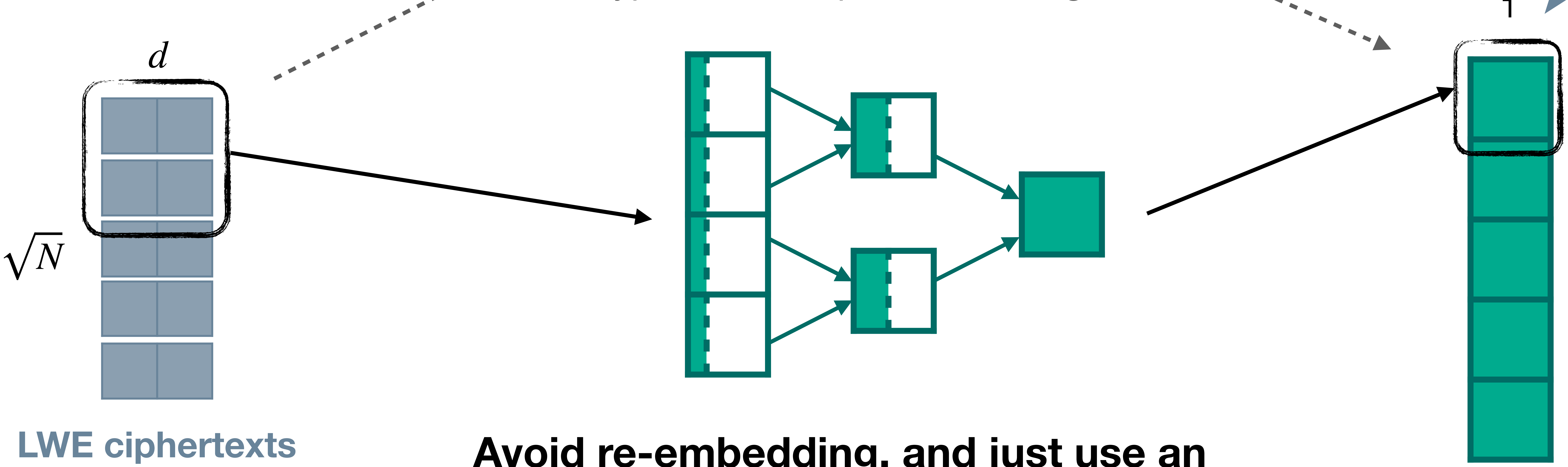
Drawback: the “double wrapping” increases the response size by ~10×!

(3) YPIR: better hint packing

“Double wrapping” makes ciphertexts ~10× larger in practice

Prior work:
Homomorphically compute LWE decryption’s inner product in Ring-LWE

$\approx d = 1024 \times$ **smaller!**



LWE ciphertexts (Offline Hint)

Ring-LWE ciphertext(s)

Avoid re-embedding, and just use an *algebraic* transform.

Yields much smaller responses

View LWE ciphertexts as ‘corrupted’ RLWE ciphertexts, and then perform <u>key switching</u> to uncorrupt them [CDKS21].

Also show: ~85% of work in this procedure can be moved to a one-time offline precomputation.

(3) YPIR: additional techniques

- **Small item retrieval:** we choose to use DoublePIR as the first phase PIR when database records are small, lowering response size from $O(\sqrt{N})$ to $O(1)$.
- **Cross-client batching:** process queries from multiple clients in batching to increase effective throughput beyond the memory bandwidth limit
- **Preprocessing:** speed up SimplePIR preprocessing using Ring-LWE
- **SCT Auditing:** application of PIR to verify the correctness of a signed certificate timestamp (SCT) using a frequently-updating data structure

See paper for details!

Performance

1-bit retrieval from an 8 GB database

Weekly cost to use YPIR to probabilistically check if a TLS certificate has appeared in a certificate transparency log containing 5 billion certificates is 16× lower than HintlessPIR.

		DoublePIR*	HintlessPIR	**YPIR (this work)**
Offline	Upload	-	-	
	Download	14 MB	-	
Online	Upload	1 MB	1.4 MB	1.5 MB
	Download	12 KB	1.5 MB	**12 KB**
	Throughput	13 GB/s	5 GB/s	**12 GB/s**

Similar costs, without offline downloads? Yes*!

Performance

32 KB retrieval from an 8 GB database

Check whether a password has appeared in a database of 250 million breached passwords

		SimplePIR	HintlessPIR	**YPIR+SP (this work)**
Offline	Upload	-	-	-
	Download	362 MB	-	-
Online	Upload	362 KB	1.4 MB	1.3 MB
	Download	362 KB	1.7 MB	**228 KB**
	Throughput	11 GB/s	5 GB/s	5 GB/s

Takeaways

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

- Offline costs matter - megabytes of communication per client, per database update
- For small items, YPIR removes all offline communication from DoublePIR at little cost
- For large items, YPIR has similar throughput and query size to HintlessPIR, with smaller responses
- Replacing a bootstrapping-like approach with an algebraic solution can yield better efficiency
- Paper at eprint.iacr.org/2024/270.pdf, code at github.com/menonsamir/ypir
- Open problems:
  - Smaller queries: queries less than $\sqrt{N}$ with high throughput
  - Silent preprocessing for PSI, ORAM, verifiable PIR