Virginia Commonwealth University

Recent Progress on Private Information Retrieval

Wei-Kai Lin

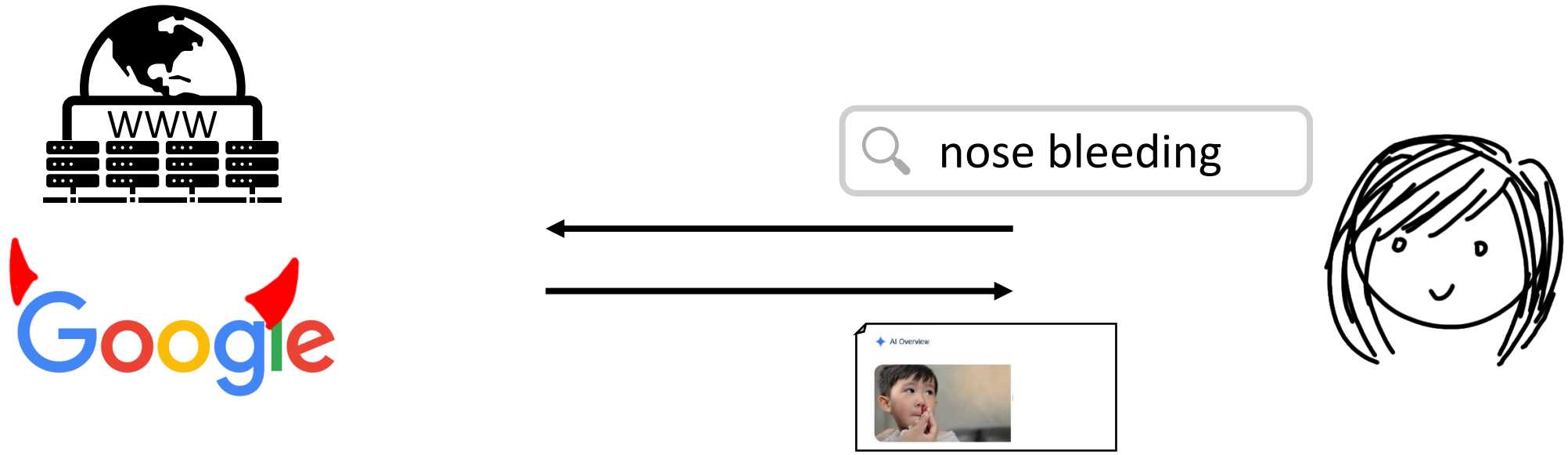
Feb 27, 2026

Ethan Mook

Daniel Wichs

Today’s web-search: non-private

Adversary: search engine



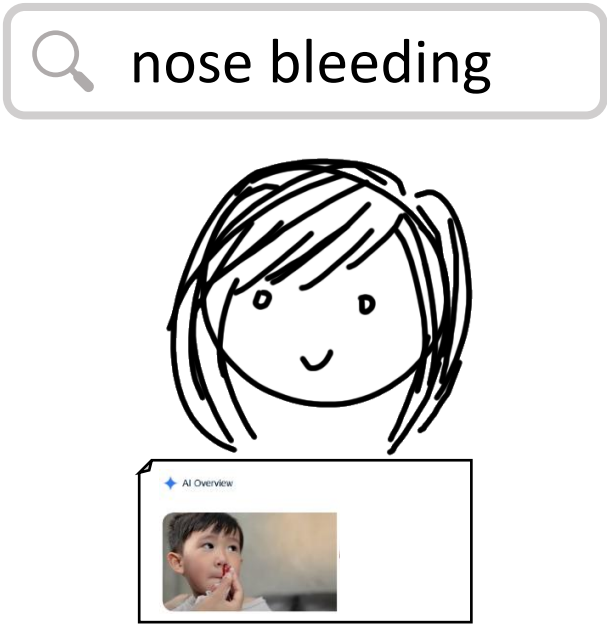
Private web search?

Naïve: download everything

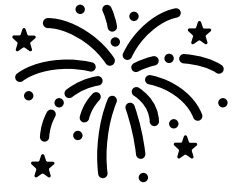
Huge bandwidth… impossible

Google

100+ Peta Bytes



This talk: **Efficient** and **Private** web search

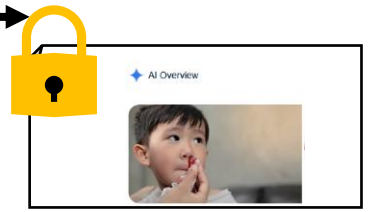


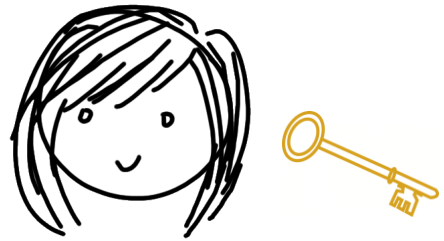
New: FHE for **RAM** programs [**L**-Mook-Wichs, STOC'23 Best Paper]

➔ Resolve query by looking up few locations

RAM-FHE search

nose bleeding



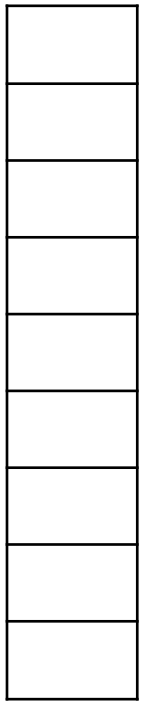


[Chor-Goldreich-Kushilevitz-Sudan'95]

This talk: Private Information Retrieval (PIR)

(RAM prog that reads 1 bit from memory)

Public DB = $\{0,1\}^N$



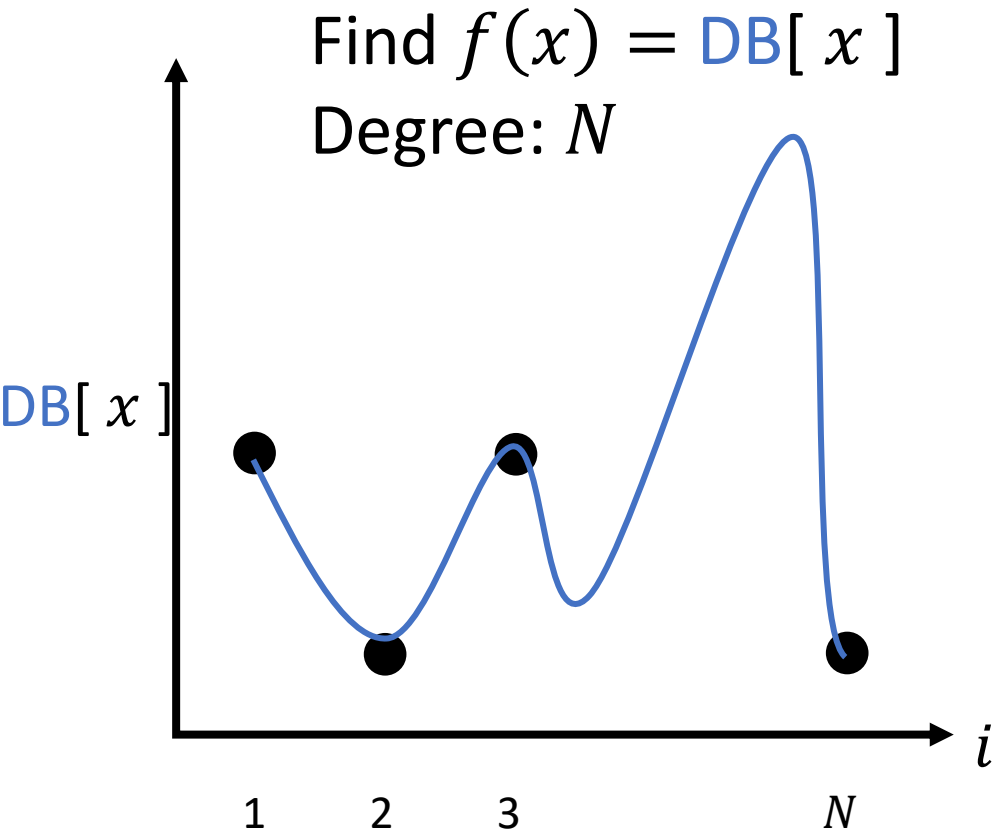
Private i $\in [N]$

Want: DB[i]

Challenge: Efficient commun $\ll N$

Common Technique: Database as a Polynomial

Public DB = $\{0,1\}^N$



Private i ∈ $[N]$

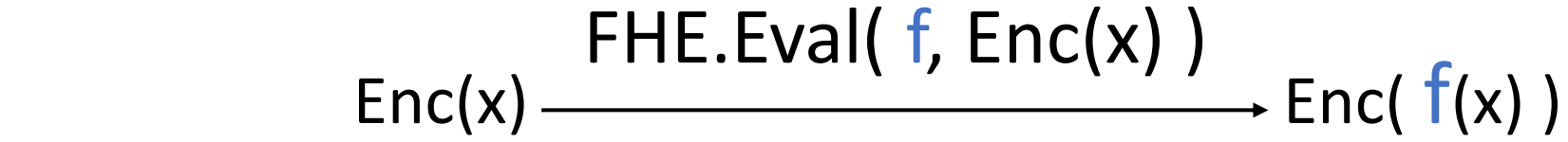
Want: DB[i]

Oversimplified diagram
for illustration

Crypto: Fully Homomorphic Encryption (FHE)

[Rivest-Adleman-Dertouzos'78] [Gentry'09]
[van Dijik-Gentry-Halevi-Vaikuntanathan'10] [Brakerski-Vaikuntanathan'11] …

Compute on encryption, without decrypt



Computation f is a polynomial

f(x) =
x^100 + x^98 + …
over a (finite) field

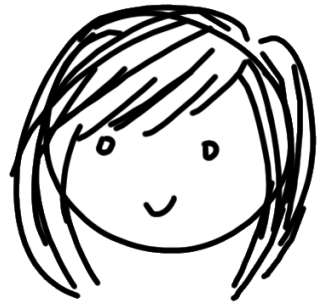
Construct PIR using FHE

Private i $\in [N]$

Public $f(x) =$ DB[x]

Secret key: 

Google



ct: FHE.Enc( , i)

ct[$f(i)$]:
FHE.Eval(f, ct)

ct[$f(i)$]

DB[i] = $f(i)$:
FHE.Dec( , ct[$f(i)$])

Communication efficient ☺
But server time?

Construct PIR using FHE

Private i $\in [N]$

Public $f(x) =$ DB[x]

Secret key:

ct: FHE.Enc(, i)

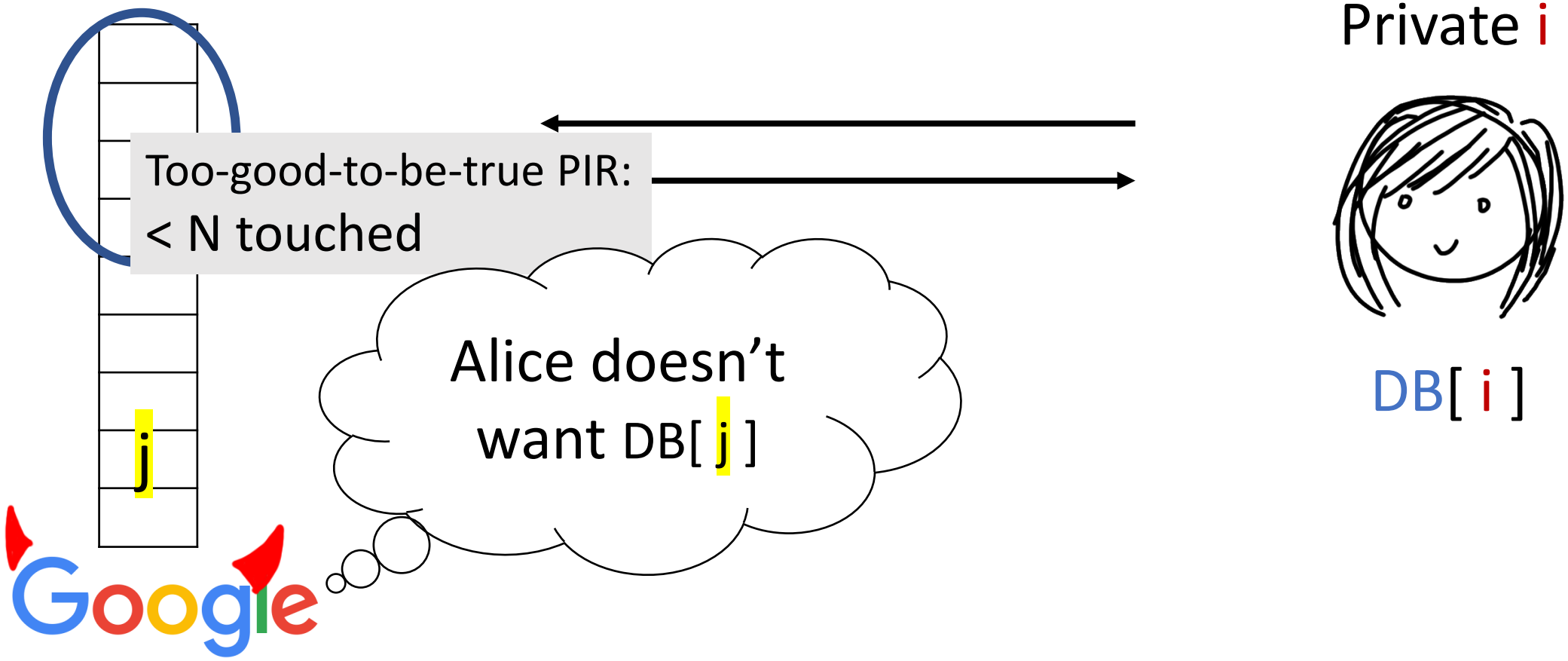
ct[$f(i)$]:
FHE.Eval(f, ct)

Google index of web: 100+ Peta Bytes,
f is more than 100 PB.
Server time: Process 100 PB per query, **infeasible!**
Is it because FHE is slow?

Lower Bound on Server Time

[Beimel-Ishai-Malkin'00]

DB = $\{0,1\}^N$



Construct PIR from FHE

Private i $\in [N]$

Public $f(x) = \text{DB}[\, x \,]$

Secret key:

Google

ct: FHE.Enc(, i)

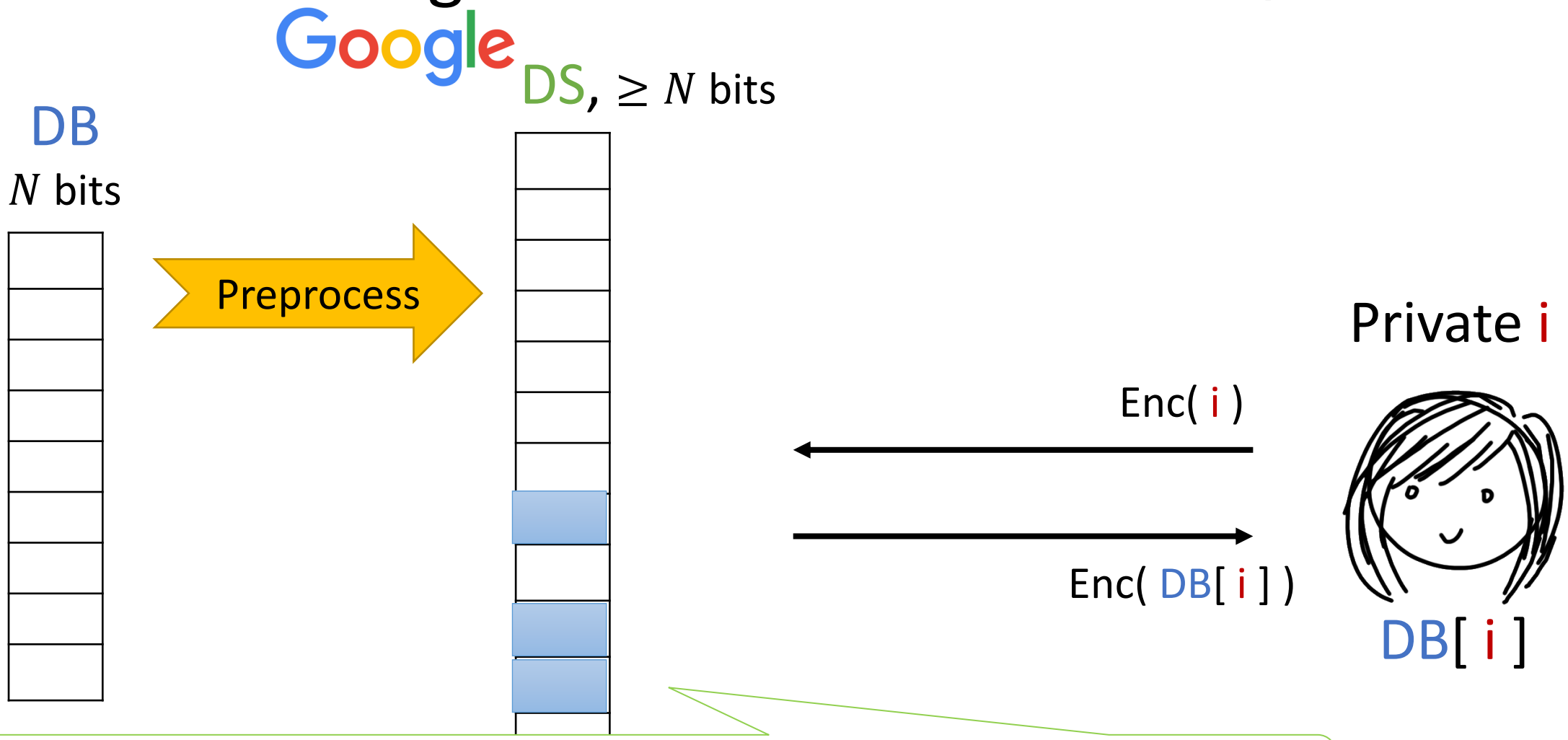
ct[$f(i)$]:
FHE.Eval(f, ct)

Google index of web: 100+ Peta Bytes,
f is more than 100 PB.
Server time: Process 100 PB per query, **infeasible!**
Is it because FHE is slow?

Not really.
Workaround?

Circumventing LB?

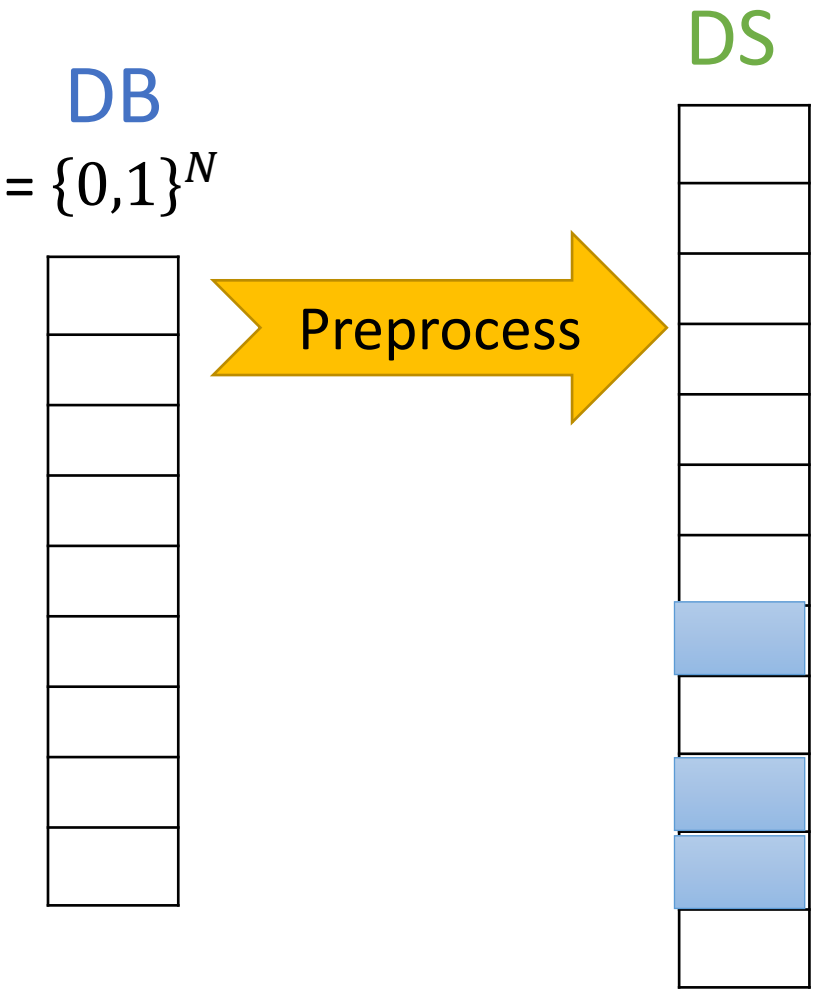
[Beimel-Ishai-Malkin'00]



Doubly Efficient: both time & commun $\ll N$

Later: Other preprocessing settings

Challenge: Preprocessing & Crypto Assump



Short time = small info

➔ Unbounded adversary can map info to DB

➔ Need crypto assump

Online Time $\ll N$

Earlier Attempts

Theorem: [**L**-Mook-Wichs, STOC'23 Best Paper]

We construct FHE scheme for RAM programs:

Homomorphic evaluation time

$\approx$ program running time $\cdot$ $poly \log N$

Assuming Ring Learning-With-Errors.

Resolved long-open questions:

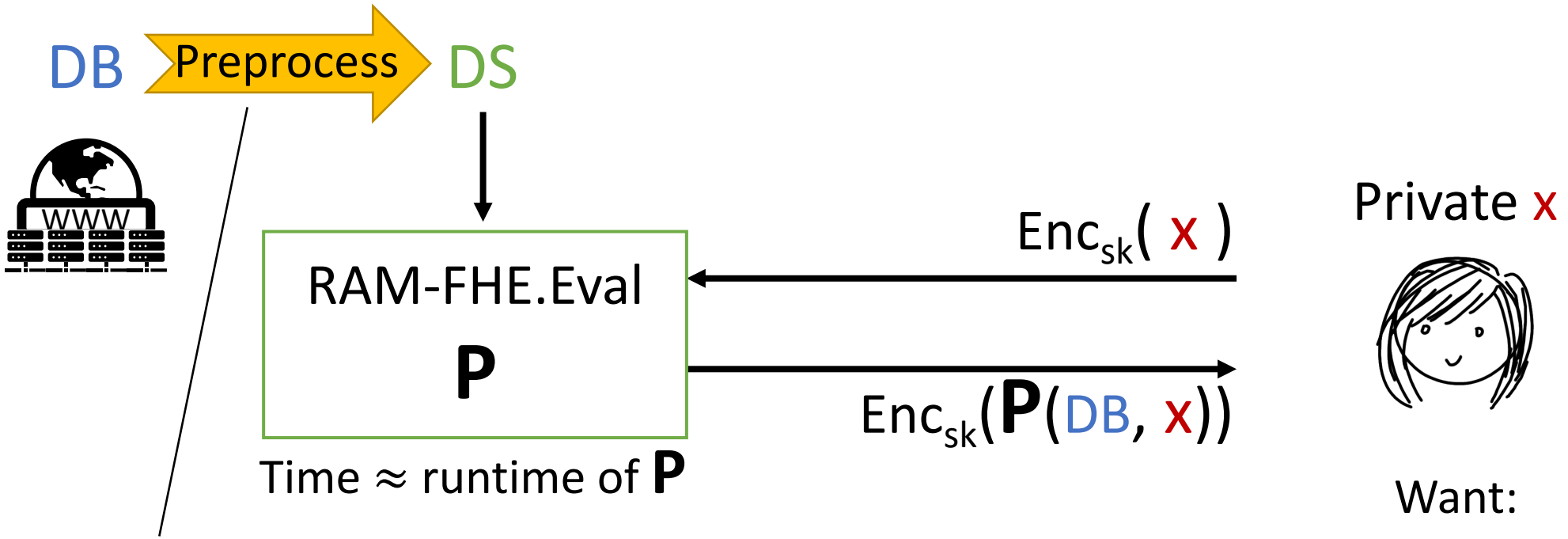
RAM-FHE, DEPIR

Our Assumption: Ring Learning-With-Errors (Ring-LWE)

(used in NIST post-quantum encryption)

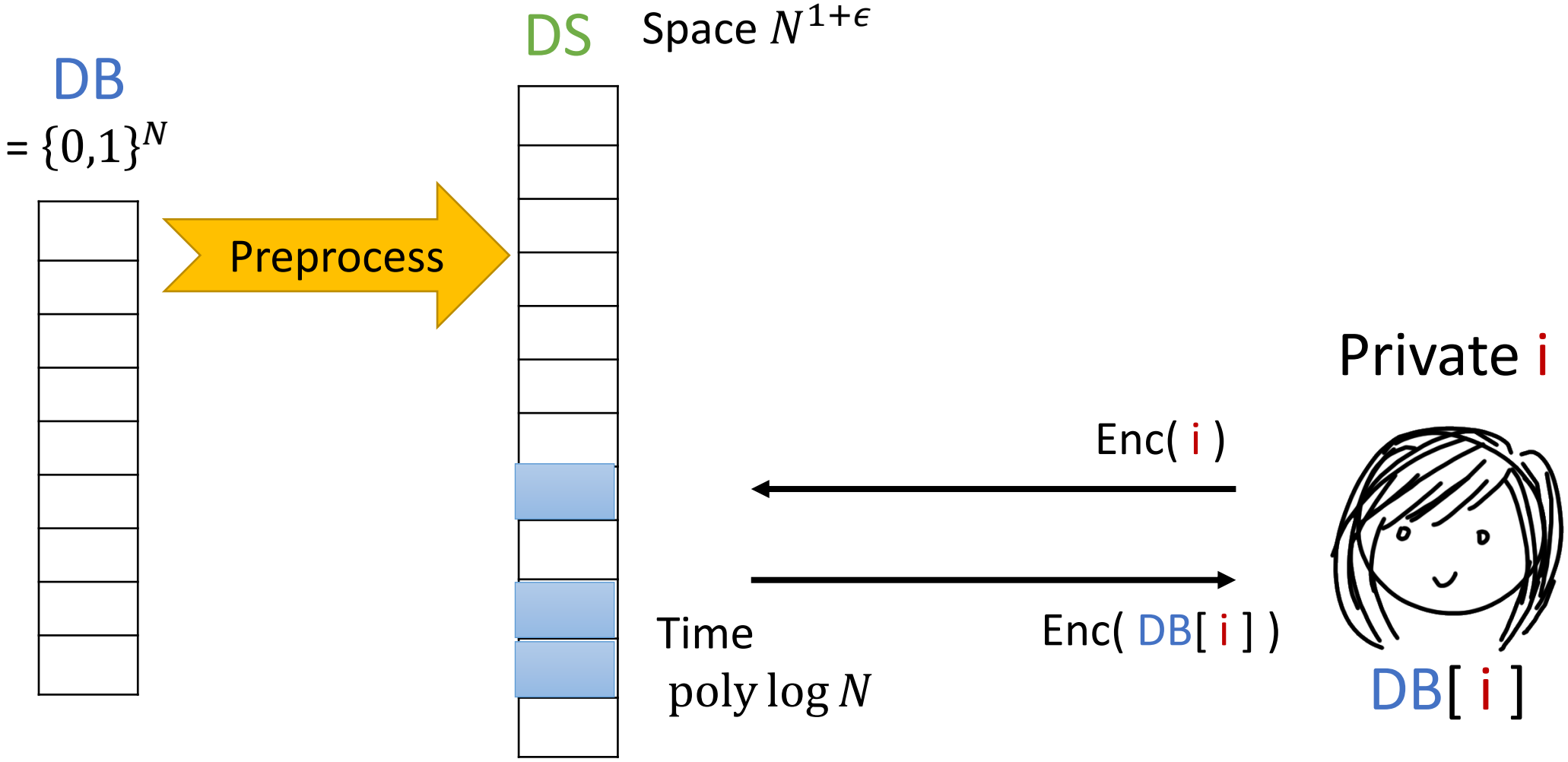
> “If the ring underlying the module has a sufficiently high degree (like 256), then these lattices inherit all the efficiency of the ones used in the Ring-LWE problem, and additionally have the following advantages, when used in our cryptographic algorithms”

Main Result 1: RAM-FHE with Preprocessing



Necessary
One-time, many users, many **P**

Main Result 2: Doubly Efficient PIR



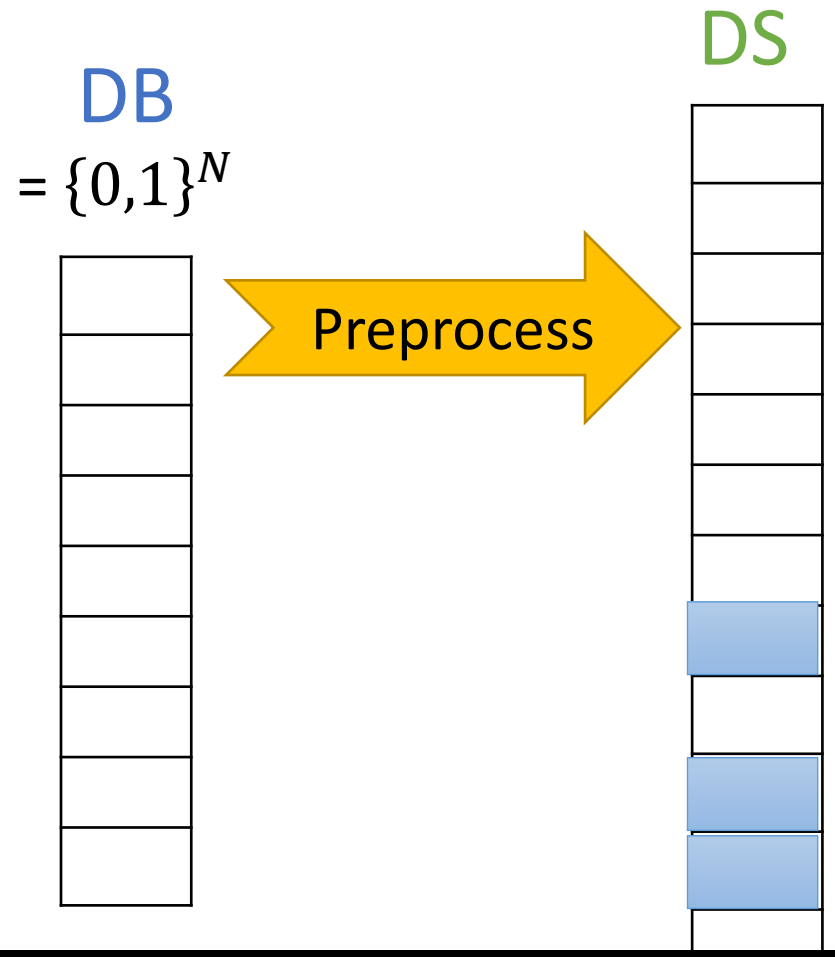
Results: DEPIR and RAM-FHE

Construction of DEPIR

Followup Results

Related: PIR and ORAM

Challenge: Preprocessing & Crypto Assump



Short time = small info

➔ Unbounded adversary can map info to DB

➔ Need crypto assump

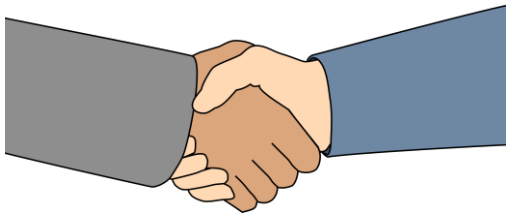
Online Time $\ll N$

Which crypto assump? How to use it?

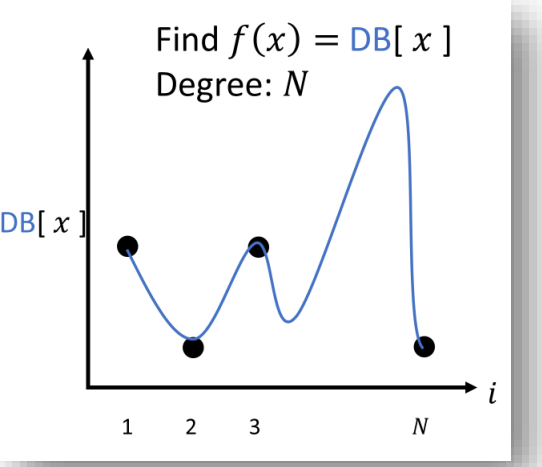
Our Doubly Efficient PIR

"Regular" PIR (linear-time) server evaluates polynomial

Key Idea:
Algebraic HE

Cryptography  Algorithms

Data structure for fast polynomial evaluation [Kedlaya-Umans'08]

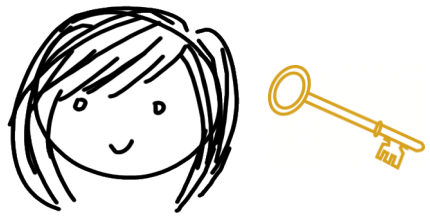


Regular PIR (linear-time)

$f_{DB}(i) = DB[\,i\,]$

$Enc(i)$

i



$Enc(f(i)) \leftarrow \mathrm{FHE.Eval}(f, \mathrm{Enc}(\,i\,))$

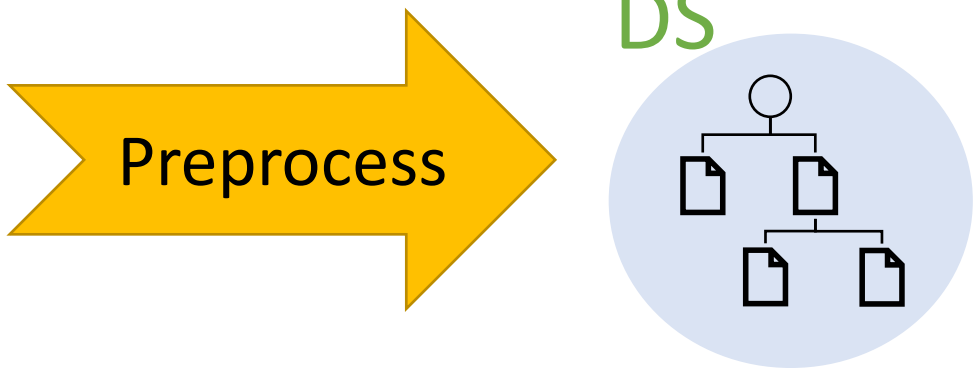
$= Enc(\,DB[\,i\,]\,)$

Linear-time

Preprocess it!

Preprocessing polynomial for Fast Evaluation [KU08]

$f(x)$
N coeffs



Space $= N^{1+\epsilon}$

Want to eval in time $\ll N$

Poly-Eval(DS, i) $\rightarrow$ $f(i)$

Time $= poly \log N$

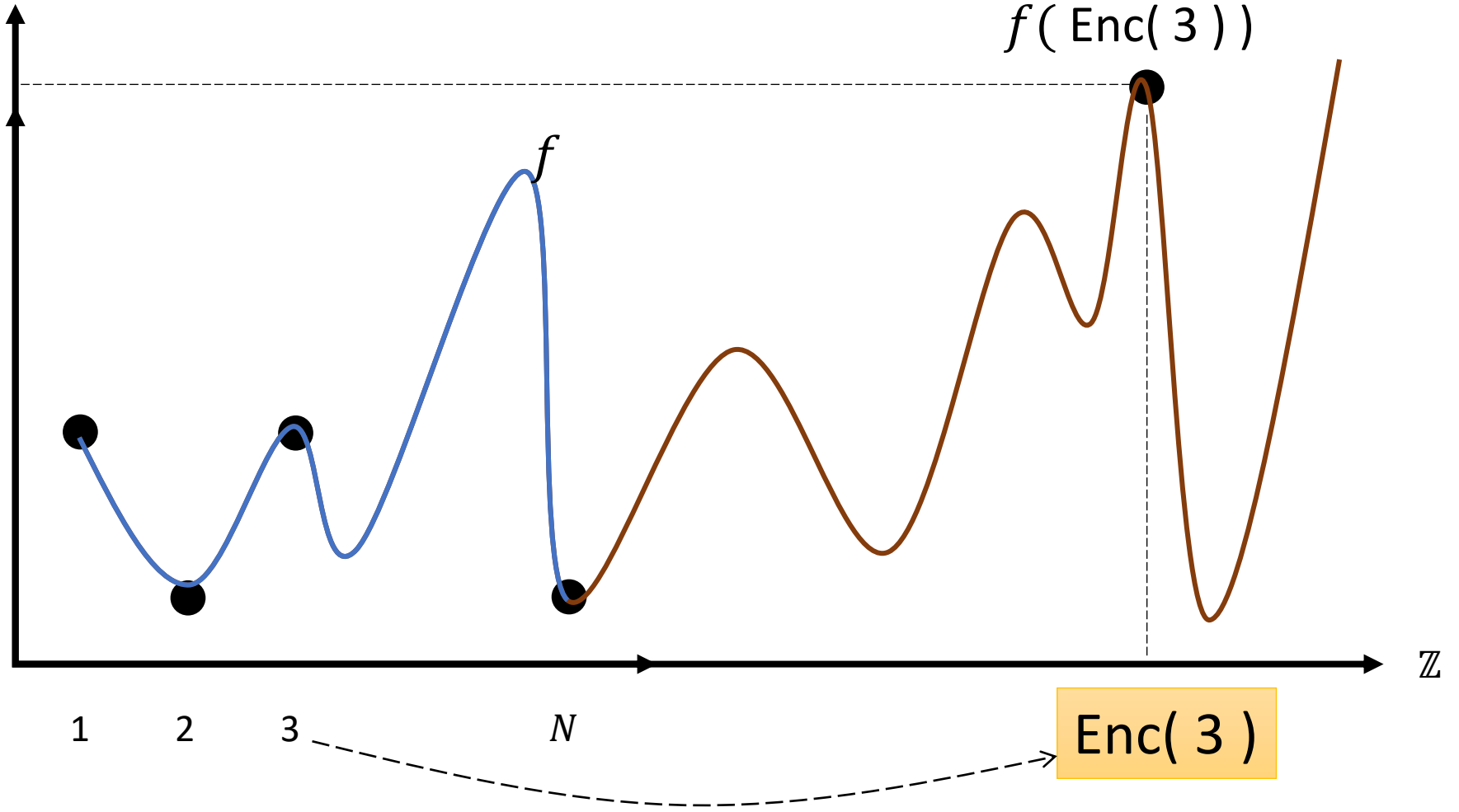
FHE.Eval(f, Enc(i))
is NOT a (small) polynomial!

Key Idea: "Algebraic" Homo. Enc.

[Brakerski-Vaikuntanathan'11]:
Based on Ring-LWE

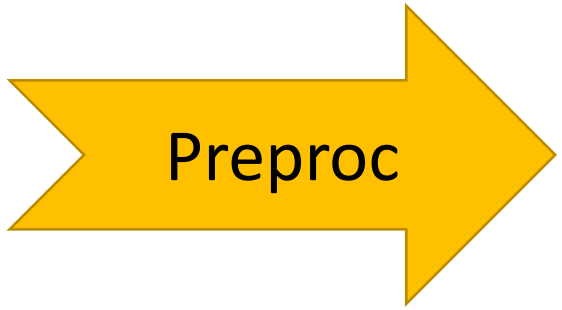
f(Enc(3))
= Enc(f(3))

New:
algebraic view,
algebraic op.

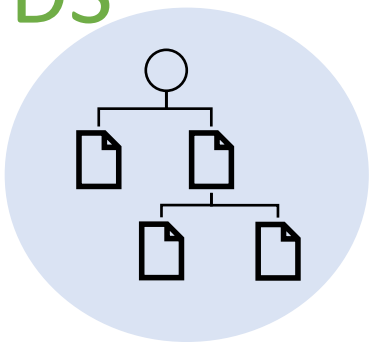


Applying [KU08] on Algebraic HE ➔ DEPIR

$f_{DB}(x)$
N coeffs



DS



Space $= N^{1+\epsilon}$

Want to eval in time $\ll N$

Poly-Eval(DS, $Enc(i)$) $\rightarrow$ $f_{DB}(Enc(i)) = Enc(f_{DB}(i))$

Time $= poly \log N$

Setting of DEPIR:
Preprocess DB,
then fast query

New Idea:

- Algebraic HE
- Polynomial eval data struct

Skipped Challenges:

- Algebraic HE from Ring-LWE
- Degree of f_{DB}:
  HE needs low deg,
  [KU08] need fewer variables
  but high deg is okay

- Preprocessing time
- Update DB
- From DEPIR to RAM-FHE

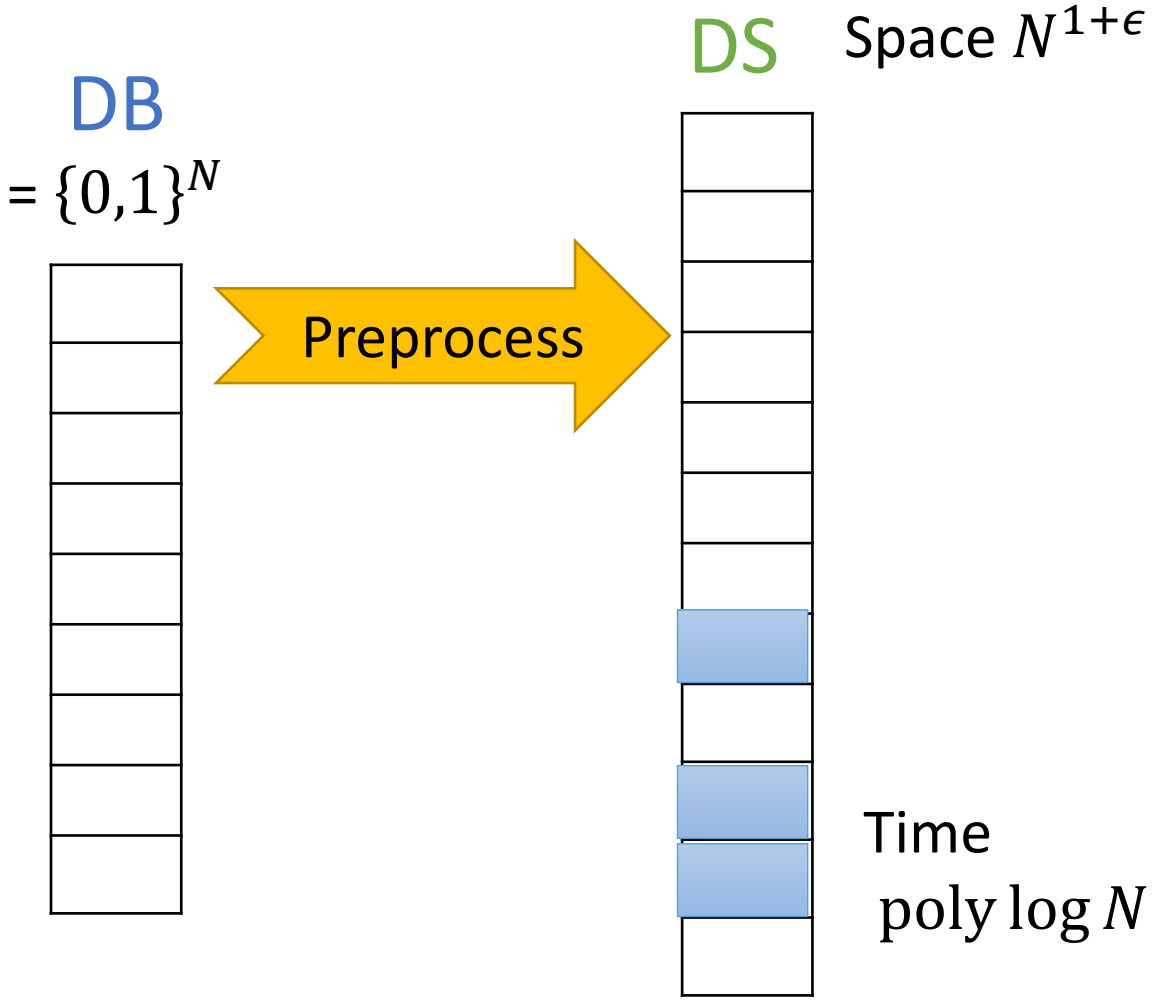
Results: DEPIR and RAM-FHE

Construction of DEPIR

Followup Results

Related: PIR and ORAM

Practical Efficiency of Doubly Efficient PIR



Theorem: For any constant $\epsilon > 0$, exists constant c s.t. … time = $O(\log^c N)$ …

Concretely, $c = \text{poly}(1/\epsilon)$.

Suppose $c = (1/\epsilon)^3$.

Choose ϵ = 0.5, then $c = 8$.

Suppose $N = 2^{30}$. Then:

DS = 2^{45}

Time = $(30)^8 = 2^{39}$

Improved DEPIR

- Practical efficiency [Okada-Player-Pohmann-Weinert'24,25]
  - Open source implementation
  - Database sizes, N: 46376 142506
  - Storage size: 44GB 2926GB
  - Running time: 1102s 6267s

- Batched queries [Ding-Malavolta-Zhang'23]

Impossibilities on DEPIR

- Using “Algebraic Homomorphic Encryptions,” the ciphertext length must be at least $\Omega(d)$ for degree d polynomials [Okada-Player-Pohmann-Weinert’25]
  - Current construction takes length $\mathrm{O}(d^2)$

- “Black-box” use of crypto does not help [**L-**Mook-Wichs'25]
  - Current construction relies on algebraic assumptions

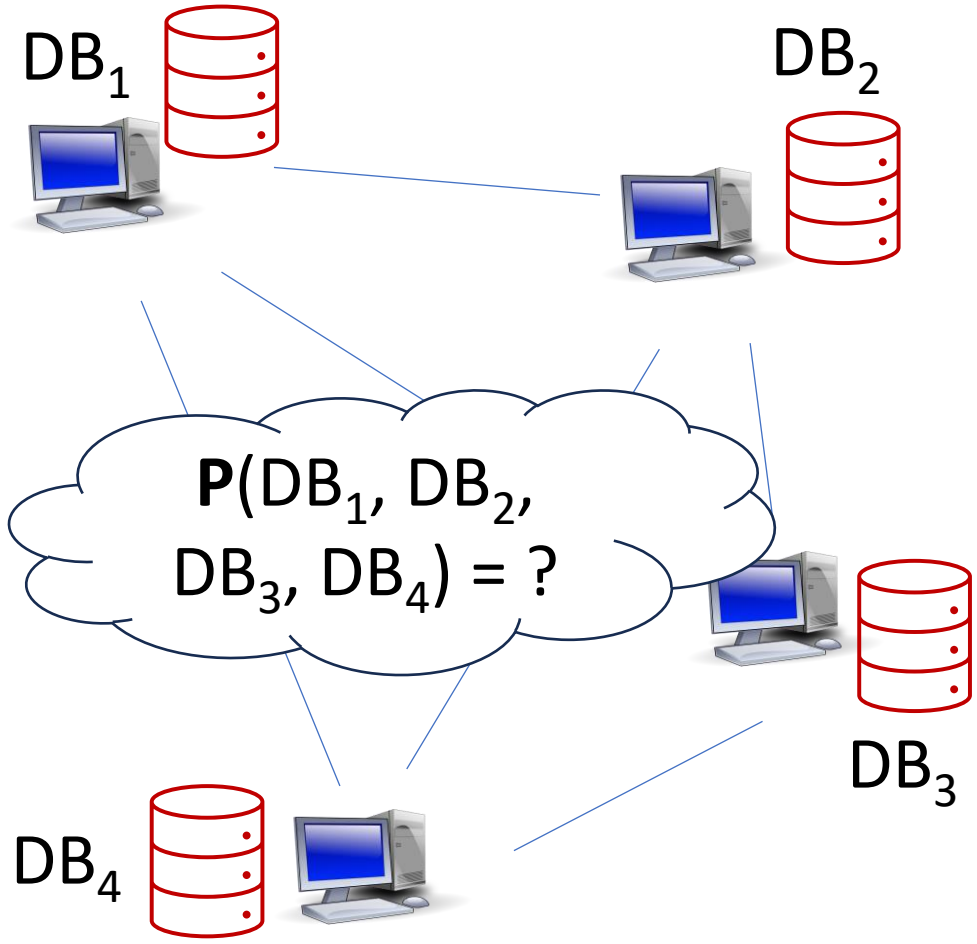
“Doubly Efficient” Crypto Protocols

Secure Multi-Party Computation (MPC):
Parties jointly compute **P** on all DBs,
reveal the output, but nothing more.

Protocols: Many [Goldreich-Micali-Wigderson’87, Ben-Or-Goldwasser-Wigderson’88 …]

Can we achieve running time $\ll |\, DB_i \,|$?

No, by the same LB argument on PIR….
YES, preprocessing circumvents LB



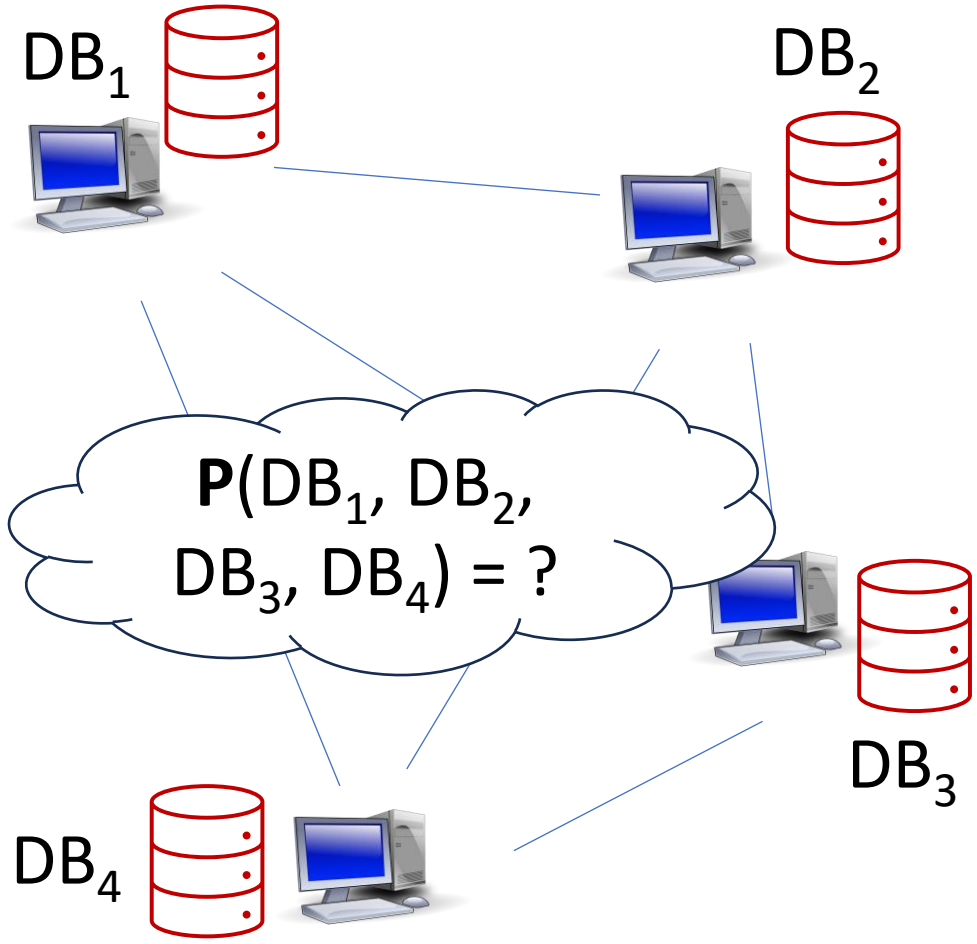
“Doubly Efficient” Crypto Protocols

Doubly Efficient MPC:
Each party i preprocess DB_i in advance;
When all parties join, compute **P** on all DB_i.

Doubly efficient: both time & commun are
~Time(**P**) ≪ | DB_i|

Semi-honest: via RAM-FHE [**L**MW’23]
Malicious: via “DE” commitment

[**L**MW’24, Bitansky-Paneth-Shamir’24]



Other protocols: Laconic function evaluation, FE, ABE, and obfuscation for RAMs
[Dong-Hao-Mook-Wichs’24, Dong-Hao-Mook-Wee-Wichs’24]

Questions?