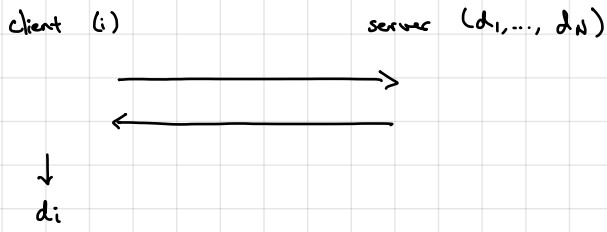


Private information retrieval (PIR): basic building block for privacy-preserving protocols

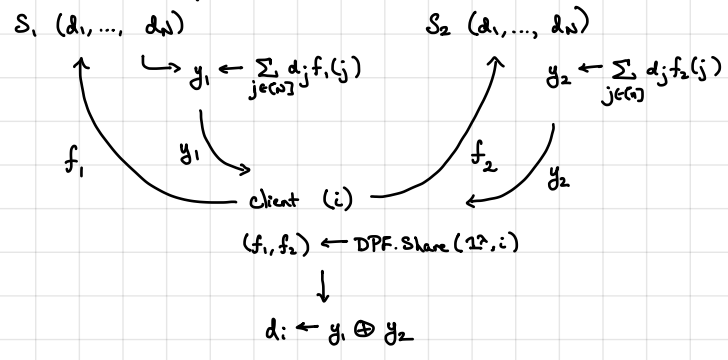


Requirements: client learns the desired database record d_i
 server does not learn anything (even if server is malicious!)
 ↳ We do not require privacy for server's database (not OT)
Trivial PIR: download the full database

Applications: private DNS lookups
 safe browsing
 private contact tracing
 contact discovery
 anonymous messaging

Goal: Minimize communication from server to client
 Trivial PIR: O(N) communication

Two-server setting: assume database is replicated across two servers (that do not collude)

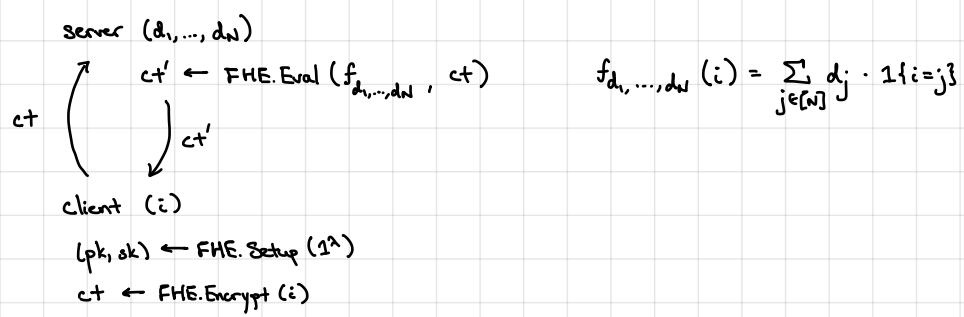


Query size: O(log N) · poly(λ)
Response size: 2 · |d_i| = O(|d_i|) } Essentially optimal with respect to N
 (in fact, can extend this to nearly optimal rate: |response| = |d_i| + poly(λ))
 ↳ Concretely: a few extra bits (e.g., 1)

Limitation: server-side work is linear in database size (possible to amortize with preprocessing)

In multi-server setting, we can also obtain information-theoretic constructions with o(N) communication
 ↳ Question closely related to locally-decodable codes

Single-server setting: database is hosted on a single server



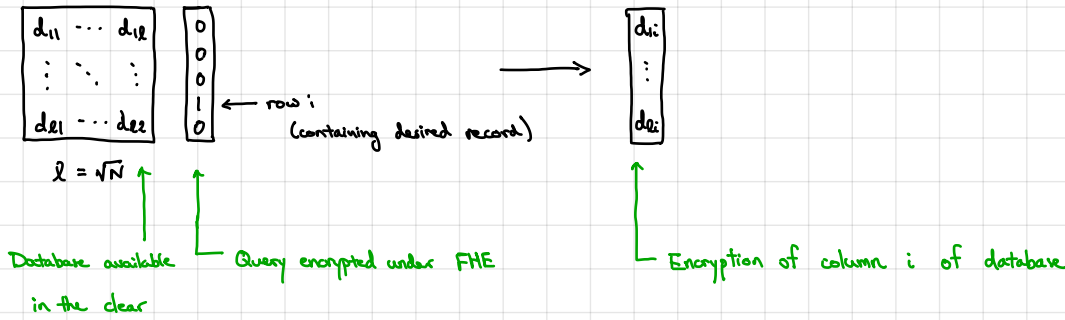
Query size: O(log N) · poly(λ)
Response size: |d_i| · poly(λ, log N)
 ↳ Can remove log N dependence with bootstrapping

FHE-based approach described above is not likely to have good concrete efficiency

↳ Requires $O(\log N)$ multiplications for each database item

↳ Typically $N \sim 2^{30}$ (or 2^{30}) so multiplicative depth is high \rightarrow poor concrete efficiency

Concretely-efficient PIR schemes follow Kushilevitz-Ostrovsky framework:



Query + response size $\sim O(\sqrt{N})$

↳ Can decrease either by recursive composition or by homomorphic multiplication \rightarrow basis of concretely-efficient constructions encoding?

↳ Open question: Further reduce compute via database

↳ Server computation is still linear

(Recent work shows how to get to \sqrt{N})
amortized with preprocessing

Improving the concrete efficiency of lattice-based schemes

Standard Regev encryption: $pk = A = \begin{bmatrix} \bar{A} \\ \bar{s}^T \bar{A} + e^T \end{bmatrix} \in \mathbb{Z}_q^{n \times m}$

large public key \sim typically $O(n^2 \log q)$ - quadratic in lattice dimension

$$sk = s^T = [-\bar{s}^T \mid 1] \in \mathbb{Z}_q^n$$

$$ct = Ar + \begin{bmatrix} 0^{n-1} \\ \mu \cdot \lfloor \frac{q}{p} \rfloor \end{bmatrix} \in \mathbb{Z}_q^n$$

need a vector of dimension n to encrypt a single scalar (high overhead)

To improve efficiency, we can instead work over polynomial rings

$\mathbb{Z}[x]$: ring of polynomials with integer coefficients

$\mathbb{Z}[x]/(x^d + 1)$: ring of polynomials modulo $x^d + 1$ (cyclotomic polynomial)

In particular $x^d = -1 \pmod{x^d + 1}$

We can view LWE as working over the ring $R = \mathbb{Z}$. Now, we consider the ring $R = \mathbb{Z}[x]/(x^d + 1)$.

RLWE assumption: Sample $a \xleftarrow{R} R_q$ ($R_q = R/qR = \mathbb{Z}_q[x]/(x^d + 1)$)

$s \xleftarrow{R} R_q$

$e \leftarrow \chi$

$u \xleftarrow{R} R_q$

analogy of discrete Gaussian (distribution over R_q where polynomials have small coefficients)

The RLWE assumption says that following distributions are computationally indistinguishable:

$(a, sa + e)$ and (a, u)

We can view ring multiplication as a matrix-vector multiplication

$$a = 2x^3 + x^2 - 3x + 1$$

$$s = x^3 - 2x + 2$$

$$R = \mathbb{Z}[x]/(x^4 + 1)$$

$$(2x^3 + x^2 - 3x + 1)(x^3 - 2x + 2)$$

first row is uniform secret is uniform

$$\begin{bmatrix} 1 & -3 & 1 & 2 \\ -2 & 1 & -3 & 1 \\ -1 & -2 & 1 & -3 \\ 3 & -1 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -2 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ -9 \\ 9 \end{bmatrix} \begin{matrix} x^3 \\ x^2 \\ x \\ 1 \end{matrix}$$

LWE with a structured matrix A

can compute a's in $O(d \log d)$ time using FFT } faster than matrix-vector multiplication in LWE (for suitable q)

Reges encryption over rings: $pk = (a, b)$ where $a \xleftarrow{r} R_q$
 $s \xleftarrow{r} R_q \rightsquigarrow b = sa + e$
 $c \leftarrow x$

$$sk = s$$

$$ct = (ar, br + \mu \cdot \lfloor \frac{\cdot}{p} \rfloor) \text{ where } r \leftarrow x \text{ and } \mu \in R_q$$

Advantages over vanilla Reges: Shorter public keys, faster key-generation

Better ciphertext rate: need 2 R_q elements to encrypt 1 R_p element

Drawback: More structured assumption (ring multiplication is commutative!)

↳ Does not have reductions to worst case lattice problems

Most constructions based on standard lattices can be translated directly to the ring setting (with better concrete efficiency)

Exploiting structure in the ring setting:

Suppose we work over the ring $R = \mathbb{Z}[x] / (x^{2^d} + 1)$ and suppose we chose the plaintext modulus $p = 1 \pmod{2^{d+1}}$.

Then, we can show that the polynomial $x^{2^d} + 1$ factors mod p as

$$x^{2^d} + 1 = \prod_{i \in [2^d]} (x - \alpha_i) \pmod{p}$$

for $\alpha_1, \dots, \alpha_{2^d} \in \mathbb{Z}_p$. Then, by Chinese Remainder Theorem (CRT):

$$R/pR = \mathbb{Z}_p[x] / (x^{2^d} + 1) \cong \mathbb{Z}_p[x] / (x - \alpha_1) \times \dots \times \mathbb{Z}_p[x] / (x - \alpha_{2^d}) \\ \cong \mathbb{Z}_p^{2^d}$$

Plaintext space is isomorphic to $\mathbb{Z}_p^{2^d}$

↳ Addition in R_p corresponds to component-wise addition in $\mathbb{Z}_p^{2^d}$
 ↳ Multiplication in R_p corresponds to component-wise multiplication in $\mathbb{Z}_p^{2^d}$ } SIMD (single instruction multiple data) support for homomorphic evaluation

We can encrypt a vector of 2^d integers

↳ Each homomorphic operation simultaneously computes on all 2^d elements

Reducing ciphertext size via modulus switching:

When we use FHE to evaluate a circuit C , parameters have to be chosen so that accumulated error is smaller than $\frac{q}{2p}$.

Useful technique: Perform all computations with respect to a modulus q and then rescale final ciphertext to a smaller modulus $q' < q$:

$$s^T c = \lfloor \frac{q}{p} \cdot \mu \rfloor + e \pmod{q}$$

$$\text{Replace } c \mapsto \lfloor \frac{q'}{q} \cdot c \rfloor \text{ (interpret } c' \text{ as element of } \mathbb{Z}_{q'})$$

To analyze this, we consider an expression over the rationals:

$$s^T c = \lfloor \frac{q}{p} \cdot \mu \rfloor + e + kq \text{ for some integer } q$$

We can write

$$c' = \lfloor \frac{q'}{q} \cdot c \rfloor = \frac{q'}{q} \cdot c + e' \text{ where } \|e'\| < \frac{1}{2} \text{ (over the rationals)}$$

Then,

$$s^T c' = \frac{q'}{q} \cdot s^T c + s^T e'$$

$$= \frac{q'}{q} \left[\left(\lfloor \frac{q}{p} \cdot \mu \rfloor + e \right) + kq \right] + s^T e'$$

$$= \frac{q'}{p} \mu + \frac{q'}{b} (e'' + e) + kg' + sTe'$$

$$= \frac{q'}{p} \mu + \frac{q'}{b} (e'' + e) + sTe' \pmod{q'}$$

require that these

components are smaller than $\frac{q'}{2p}$

will need that secret key components are small
(i.e., sampled from error distribution)

[observe original error e is scaled down by $\frac{q'}{b}$]

Take-away: After performing operations, can rescale the ciphertexts to smaller modulus q' (reduces communication concretely)