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

Client (i) \rightarrow Server (d_1, \ldots, d_N)

\downarrow \quad d_i

Requirements: Client learns the desired database record \( d_i \);
Server does not learn anything (even if server is malicious!)
\implies \text{We do not require privacy for server's database. (not OT)}

Trivial PIR: download the full database.

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)
\( S_1 (d_1, \ldots, d_N) \quad S_2 (d_1, \ldots, d_N) \)

Query size: \( O(\log N) \cdot \text{poly}(\lambda) \)
Response size: \( 2 \cdot |d_i| = O(1d_i|) \)

Response size: \( \approx \frac{1d_i|}{\text{poly}(N)} \) with respect to \( N \)
\text{(in fact, can extend this to nearly optimal rate: \( \text{Response} = \frac{1d_i| + \text{poly}(N)}{\text{poly}(\lambda)} \))}

Limitations: 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.

\implies \text{Question closely related to locally-decodable codes}

Single-server setting: Database is hosted on a single server.

Server (d_1, \ldots, d_N)

Query size: \( O(\log N) \cdot \text{poly}(\lambda) \)
Response size: \( 1d_i| \cdot \text{poly}(\lambda, \log N) \)

\text{Can remove log N dependence with bootstrapping}
FHE-based approach described above is not likely to have good concrete efficiency

- Requires $O(n \log n)$ multiplications for each database item
- Typically $N \sim 2^n$ (or $2^{2n}$) so multiplicative depth is high $\rightarrow$ poor concrete efficiency

Concretely-efficient PIR schemes follow Kushilevitz-Ostrovsky framework:

\[
\begin{bmatrix}
\vdots & 
\vdots \\
\vdots & 
\vdots \\
\vdots & 
\vdots \\
\end{bmatrix} \rightarrow \begin{bmatrix}
\vdots \\
\vdots \\
\vdots \\
\end{bmatrix}
\]

\[ l = \sqrt{N} \quad \text{Encryption of column } i \text{ of database} \]

Dataset available $\rightarrow$ Query encrypted under FHE $\rightarrow$ Encryption of column $i$ of database in the clear

Query + response size $\sim O(\sqrt{N})$ $\rightarrow$ Open question: Further reduce compute via database

- Can decrease either by recursive computation or by homomorphic multiplication $\rightarrow$ basis of concretely-efficient constructions encoding?
- Secure 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 Ring encryption:

\[ \text{pk} = (A, A^\dagger + \epsilon I) \in \mathbb{Z}_b^{n \times n} \quad \text{large public key} \sim \text{typically } O(n \log b) \quad \text{quadratic in lattice dimension} \]

\[ s_k = s^\dagger = [-\sqrt{N} \quad 1] \in \mathbb{Z}_b^n \quad \text{need a vector of dimension } n \text{ to encrypt a single scalar} \]

\[ c_t = Ar + [0^{n-1}] \in \mathbb{Z}_b^n \quad \text{(high overhead)} \]

To improve efficiency, we can instead work over polynomial rings

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

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

In particular $x^d = -1$ (mod $x^d + 1$)

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 $a \in R_b$ $\rightarrow$ $(R_b = R/\mathbb{Z} = \mathbb{Z}_b[x]/(x^d + 1))$

\[ s \in \mathbb{Z}_b^n \quad \text{(distribution over } R_b \text{ where polynomials have small coefficients)} \]

\[ e \in x \quad \text{(analog of discrete Gaussian)} \]

\[ u \in R_b^* \]

The RLWE assumption says that following distributions are computationally indistinguishable:

\[ (a, se + e) \text{ and } (a, u) \]

We can view ring multiplication as $n$-matrix-vector multiplications

\[ a = 2x^2 + x^3 - 3x + 1 \]

\[ s = x^3 - 2x + 2 \]

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

\[ (1x^3 + x^2 - 3x + 1) (x^3 - 2x + 2) \text{ LWE with a structured matrix } A \]

\[ \text{can compute } a \odot s \text{ in } O(d \log d) \text{ time using } \text{FFT} \]

\[ \text{faster than matrix-vector multiplication in LWE} \]
Reduce encryption over rings: $\text{pk} = (a, b)$ where $a \in R_q$

$$c \in R_q \Rightarrow b = xa + e$$

$$sk = s$$

$$ct = (ar, bs + \mu \cdot \frac{q_i}{p_i}) \quad \text{where} \quad r = x \quad \text{and} \quad \mu \in R_q$$

Advantages over vanilla Regas:
- 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?)

$\Rightarrow$ 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 $R = \mathbb{Z}[x]/(x^d + 1)$, and suppose we chose the plaintext modulus $q = 1$ (mod $2^{d+1}$).

Then, we can show that the polynomial $x^d + 1$ factors mod $q$ as

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

for $\alpha_1, \ldots, \alpha_d \in \mathbb{Z}_p$. Then, by Chinese Remainder Theorem (CRT):

$$R/pR \cong \mathbb{Z}_p[x]/(x^d + 1) \cong \prod_{i \in [d]} \mathbb{Z}_p[x]/(x - \alpha_i) \cong \mathbb{Z}_p^d$$

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

$\Rightarrow$ Addition in $R_p$ corresponds to component-wise addition in $\mathbb{Z}_p^d$

$\Rightarrow$ Multiplication in $R_p$ corresponds to component-wise multiplication in $\mathbb{Z}_p^d$

SIMD (Single Instruction Multiple Data) Support for homomorphic evaluation

We can encrypt a vector of $\mathbb{Z}^d$ integers

$\Rightarrow$ Each homomorphic operation simultaneously computes on all $\mathbb{Z}^d$ 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 techniques: Perform all computations with respect to a modulus $q$ and then rescale final ciphertext to a smaller modulus $q' < q$:

$$s'c = [\frac{q'}{p} \cdot \mu] + e \quad \text{(mod} \ q)$$

Replace $c \mapsto [\frac{q'}{p} \cdot c]$ (interpret $c'$ as element of $\mathbb{Z}_{q'}$)

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

$$s'c = [\frac{q'}{p} \cdot \mu] + e + k \cdot \delta$$

We can write

$$c' = [\frac{q'}{p} \cdot \delta] = \frac{q'}{p} \cdot \delta + e'$$

where $\|e\| < \frac{1}{2}$ (over the rationals)

Then,

$$s'c' = \frac{q'}{p} \cdot s'c + s'e'$$

$$= \frac{q'}{p} \left( \left[ \frac{q'}{p} \cdot \mu \cdot e' \right] + e + k \delta \right) + s'e'$$
\[ \frac{q^t}{p} \mu + \frac{q^t}{b} (e^t + e) + kE' + s^T e' \]

\[ = \frac{q^t}{p} \mu + \frac{q^t}{b} (e^t + e) + s^T e' \pmod{q'} \]

**Takeaway:** After performing operations, can rescale the ciphertexts to smaller modulus \( q' \) (reduces communication concrete).