Middle East Technical University
Institute of Applied Mathematics

Single Server Private Information Retrieval and SimplePIR Protocol

Fatma Sıla MENEKAY
(Cryptography)

Advisor: Assoc. Prof. Dr. Oğuz YAYLA

Term Project Report
July 2023

Abstract

In this project, single server private information retrieval (PIR) protocols are examined. An extensive discussion is initiated on the fundamental principles and core concepts underpinning single server PIR schemes. Furthermore, particular emphasis is placed on the introduction of computationally secure single server PIR (CPIR) and computationally symmetric single server PIR (CSPIR) constructions. Important concepts aimed at reducing the overall complexity inherent in PIR protocols are presented as well. Finally, to illustrate the latest advancements in single server PIR field, the project ends with a review of the state-of-the-art SimplePIR scheme.

Öz

Bu projede, tek sunuculu özel bilgi erişimi (private information retrieval-PIR) protokolleri incelenmektedir. Tek sunuculu PIR protokollerini oluşturan temel ilke ve kavramlar detaylıca tartışılmaktadır. Bunlara ek olarak, hesaplama güvenlikli tek sunuculu PIR ve hesaplama güvenlikli simetrik tek sunuculu PIR şemalarını oluşturma yöntemleri gösterilmektedir. Ayrıca PIR protokollerinin toplam karmaşıklığını azaltmak için kullanılan önemli kavramlar sunulmaktadır. Son olarak proje, tek sunuculu PIR alanındaki son gelişmeleri göstermek için, güncel SimplePIR tasarımının özeti ile sonlanmaktadır.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Client and server interaction is part of everyone's daily life. Almost any interaction with internet triggers a connection to a database and these interactions reveal some information about clients such as what they like or what stock they are interested in. The servers process information about client and use these information for various purposes like selling people new products or influence them in certain way. Other than these daily interactions, there are also some sensitive cases which needs privacy. In scenarios such as a client desiring to confidentially search for stock information on a dedicated server, or an individual needing to covertly query a specific patent from a patent database, preserving the privacy of both the server and the querying individuals has become a major concern. This is to prevent potential adversaries from gaining an unfair advantage—like front-running a stock or applying for a patent before the client. For all these reasons the demand for privacy in client-server interactions is increasing rapidly and it is the interest of cryptography. To address these challenges, Private Information Retrieval (PIR) was introduced as a key concept in the influential paper by Chor et al. [1]. PIR protocols let clients to extract required data from a database without exposing to the server which data is accessed. Specifically, suppose an n-bit length data, $x = (x_1, x_2, ..., x_n)$, is stored on one or more servers, and the client is interested in retrieving the i-th bit, x_i, where i belongs to $[n]$ from the database. In such a case, PIR ensures the value of i remains undisclosed to the server. A naive solution for client is to download the complete database and perform local queries. However, in most scenarios, the process of downloading and storing the entire database is resource-intensive and impractical. This solution becomes highly undesirable, particularly when dealing with large databases. Therefore, the primary objective in designing a PIR protocol is to minimize the communication complexity between the client and the server.

There are two primary approaches to Private Information Retrieval (PIR): information-theoretic PIR and computational PIR. Information theoretic PIR (ITPIR) protocols were initially introduced and developed by Chor et al. [1] in 1995. To implement ITPIR protocols with efficient communication complexity (i.e., sublinear in database size), it is necessary to have replicates of database across multiple servers that operate independently and non-colluding. In such a setup, the client can issue queries to the non-colluding servers, each of which possesses the same database. The client then combines the responses received from each server to obtain the desired result. If we consider the current state, multi server PIR with information-theoretic secure has $n^{o(1)}$ communication and with computational secure has $O(\log n)$ complexity.

Computational PIR (CPIR) was first developed by Chor and Gilboa [2] with a multi-server solution. In 1997, Kushilevitz and Ostrovsky [3] further contributed by presenting a method to construct a single-server CPIR protocol with sublinear communication. In this type of protocol, the client's query remains private from a polynomial-time bounded server. The privacy of the requested index is ensured based on the assumptions of cryptographic primitives. If we consider the current state, computationally secure single server PIR has polylog(n) communication complexity.

In the earlier solutions provided for the PIR problem, the main objective was to safeguard the client's privacy while preventing the servers from accessing any details regarding the client's interests. However, it should be noted that the client might possess additional information about specific bits of the database, referred as side information. If a protocol satisfies both the client's and the server's privacy, it is called as Symmetric Private Information Retrieval (SPIR). With this type of protocol, the server remains unaware of the client's query, and the client only learns the specific bit it requested. In 1998, Y. Gertner et al.[4] defined symmetric private information retrieval in an information-theoretic manner. Subsequently, various computational approaches were introduced to design SPIR protocols with efficient sublinear communication by Lipmaa[5], including the works of Kushilevitz and Ostrovsky [3], Naor and Pinkas [6], and Y. C. Chang [7].

Symmetric single-server Private Information Retrieval (PIR) is closely related to Oblivious Transfer (OT). OT was constructed by Rabin [8]. OT is a protocol where the client gets only piece of information it interested and the server does not know which piece was retrieved. Therefore, the concept of symmetric single-server PIR inherently includes the concept of Oblivious Transfer.

In this project, we focus on single-server PIR schemes whose security is based on the various cryptographic hardness assumptions. Although multi-server PIR constructions are more

efficient than single-server schemes due to high computational complexity of CPIR, finding non-colluding servers can be unrealistic. Replicated databases can reduce the security. Therefore developing practical single-server PIR scheme is desirable. In Chapter 2, important developments in literature regarding to single server PIR protocols are covered. Next in chapter 3, cryptographic primitives and properties are given which are commonly used in order to construct secure PIR protocols. Then some preliminaries are defined to be able to understand the requirements of single server PIR protocols. Further in Chapter 4, single-server PIR protocol scheme is constructed by using Decisional Diffie Hellman Problem and construction of symmetric single server PIR protocol with 3 different methods are given. Commonly used techniques which reduce to computational complexity of PIR protocols are described in Chapter 4 as well. Finally in Chapter 5, SimplePIR (published in 2022) [9] protocol is reviewed which is single server PIR protocol with efficient overall complexity.

Chapter 2

A Brief Literature Review on Single Server PIR

The PIR protocol was first introduced by Chor et al. [1], it was designed for multi server where the n-bit data x is copied on multiple servers without any alterations. It was assumed that servers were not colluded in order to ensure privacy. It was also shown that for single server, all the n-bits should be transferred to the client in order to satisfy information theoretic security. They showed that communication complexity of information theoretic secure single server PIR is $O(n)$.

Information-theoretic secure single server PIRs have high communication complexity for practical purposes. To overcome this problem, Kushilevitz and Ostrovsky [3] developed a computationally secure approach. They thought database as a matrix $M = (x_{ij})_{s\times t}$ (where $s \times t$ is dimension of matrix) instead of string database $x \in \{0,1\}^n$. Assume that the client is interested in x_{lm}, to receive that particular bit client creates arrays of $y = (y_1, y_2, ..., y_t)$ chosen from t random integers such that only y_m is not a quadratic residue modulo of N where N is a public composite modulus, $y_m \neq \alpha^2 \mod N$ for any integer α. Then client sends y to the server and the server calculates $z_i = \Pi_{j=1}^{t} y_j^{2-x_{ij}} \mod N$ for $1 \leq i \leq s$ and sends results to the client. Finally, the client finds out $x_{lm} = 0$ if z_l is a quadratic residue module of N and $x_{lm} = 1$ if it is not. Therefore, with this algorithm they showed the communication complexity of single server PIR drops to $O(2^{\sqrt{\log n \log\log N}})$ where N is the multiplication of two primes. It is easy to see the communication complexity is less than $O(n^{\epsilon})$ for any $\epsilon > 0$.

Cachin, Micali and Stadler [10] developed a security protocol using ϕ-hiding assumption for single server PIR. Briefly, ϕ-hiding assumption states that for a given RSA modulus N it is

computationally-hard to distinguish which of the two small primes ($p_i \ll N^{1/4}$) is a divisor of $\phi(N)$. In order to find the i-th bit of database x, client creates $p = (p_1, p_2, ..., p_n)$ such that p_j is prime for all $j \in [n]$, p_i divides $\phi(N)$ and a generator g with order divisible by p_i. After creating p, client sends RSA modulus N and p to the server. Then server calculates $r = g^{\Pi_j p_j^{b_j}}$ and sends back to the client. Finally, the client finds out $x_i = 1$ if r is a p_i-residue modulo of N or $x_i = 0$ otherwise. By applying this algorithm, they managed to reduce the complexity to $O(\log^8 n)$ which is less than $O(2^{\sqrt{\log n \log\log N}})$.

In 2000, Kushilevitz and Ostrovsky [11] came up with a single database PIR protocol which has $n - n/2 + O(K)$ communication complexity for Honest-but-Curious server and $n - n/6K + O(K^2)$ communication complexity for malicious server where K is the security parameter. They used 2-1 trapdoor functions in [12] and hard-core predicates in [13]. They changed the basic approach of treating data by bits $x = (x_1, x_2, ..., x_n)$ and they assumed the database was composed of $2l$ sub-strings $x = (z_{1,L}, z_{1,R}, z_{2,L}, z_{2_R}, ..., z_{l,L}, z_{l,R})$ and each sub-string has a size of K. The client picks 2 trapdoor functions f_L and f_R and sends these to the server while keeping trapdoors of these functions. The server applies these functions to the data such that $f_L(z_{i,L})$ and $f_R(z_{i,R})$ for each i and then sends results to the client. Assume that i-th bit the client is interested in is at block $z_{s,L}$. The next step for the client is to pick hard-core predicate functions $r_L, r_R \in \{0,1\}^K$ such that $r_L(z_{s,L}) \neq r_L(z'_{s,L})$ and $r_R(z_{s,R}) = r_R(z'_{s,R})$ and send these to the server. The server calculates $b_j = r_L(z_{j,L}) \oplus r_R(z_{j,R})$ for each $j = 1, 2, ..., l$ and then sends back to the client. It is easy for client to check b_s and calculate the x_i and all the other bits in the block $z_{s,L}$.

Gentry and Ramzan [14] took further step to design a single server PIR protocol with communication complexity $O(\log^2 n)$. They dealt with blocks instead of bits and the algorithm's underlying security is based on ϕ-hiding assumption as well. The public parameters of the scheme are the database of size n, an integer parameter l, small distinct prime numbers $p = (p_1, p_2, ..., p_t)$ where $t = \lceil n/l \rceil$ and a set $S = \{\pi_1, \pi_2, ..., \pi_t\}$ of prime powers $\pi_i = p_i^{c_i}$ where $c_i = \lceil l/\log_2 p_i \rceil$. Let the database x is divided into t blocks $x = C_1||C_2||...||C_t$ of size l at most. Knowing p and S the server can calculate $e = C_i \mod \pi_i$ with using the Chinese Remainder Theorem by treating the C_i as an integer satisfying $0 \leq C_i < 2^l < \pi_i$. The client sends an RSA modulo N such that π_i divides $\phi(N)$ and a generator g with order divisible to by π_i. Then, the server sends back $r = g^e \mod N$. If we set $q = |g|/\pi_i$, then $|g^q| = \pi_i$. Finally, the client can compute the discrete logarithm $\log_{g^q}(r^q) = e \mod \pi_i = C_i$, since we used small prime p_i.

$O(\log^2 n)$ provides almost optimal communication complexity for the PIR protocols. However, those protocols require vast amount of computation on the database and which makes

them impractical for real world use-cases. Therefore researchers mainly focused on finding a computationally-efficient PIR protocol. Around 2007-2008, C. Aguilar-Melchor and P.Gaborit [15] [16] developed a lattice based homomorphic encryption approach to tackle the computational-efficiency problem in PIR protocols. In the protocol client generates n matrices $B_1, B_2, ..., B_n$ from a secret matrix M. While doing so, the client adds soft (small) noise to all matrices (softly distributed matrices) other than the i-th one where i is the index the client is trying to learn. The client generates the B_i which is called Hardly Distributed Matrix. Then client sends these matrices to the server which encodes each element and creates encoded matrices $A_1, A_2, ..., A_n$. Next, the server computes $(A_1, A_2, ..., A_n)(B_1, B_2, ..., B_n)^T = R$ and sends R to the client. Finally, the client can calculate A_i and also i-th bit of database by knowing the M and all the soft noises and hard noises. Later in 2010 , Olumofin and Goldberg [17] reported that proposed protocol is one to three orders of magnitude more efficient than the previous approaches.

After Aguilar-Melchor and P.Gaborit [15] [16], the paradigm of the single server PIR shifted. The problem was not anymore solving only the communication complexity of the scheme but creating a scheme which optimizes both computational and communication complexity. When they try to reduce one, they usually cause other one to increase. Most of the PIR schemes emerged in the last fifteen years, used Lattice-based approaches which was inspired by Kushilevitz-Ostrovsky's idea of thinking database as a matrix. Most of the schemes used ring-learning-with-errors (Ring LWE) which was presented in [18] by Regev as an encryption scheme. Polylogarithmic computations in security parameter is required for most of these schemes instead of polynomial. As mentioned, the cost of reducing computation complexity means increasing the communication complexity and one bit is expanded to roughly 10 bit with encryption which makes expansion factor $F \approx 10$.

In 2014 XPIR was introduced by Aguilar-Melchor et al [19]. In Lattice-based schemes, d is defined as the dimension of the lattice and the expansion factor F is related to $F \sim (d-1)$. Since d is closely related to the communication cost, increasing it would end up with high communication cost. In XPIR, d was set to 2 which means they worked 2-dimensional matrices. Therefore, for each query client sends $\sqrt{N}$ (where N is database size) encrypted query index (ciphertexts). For ring-LWE, each one of these can be as large as thousands of kilobytes and results in higher communication cost (hundreds of MB for a database size of GBs).

In 2018, Angel et al.[20] developed the SealPIR in which they tried to overcome the high communication cost problem in XPIR. The idea is that client compresses the ciphertexts before sending them to the server. Then by using homomorphic operations, server decompresses

the ciphertexts to do computations with them. They managed to reduce the communication cost greatly by applying this logic, however this created another problem that server needs to store megabytes of information for each client. Storing extra data for each client on server side is not ideal for storage and also it might give hint to server about client's key whether it is switched or not.

MulPIR [21], OnionPIR [22] and Spiral [23] were published in 2021 and 2022. Instead of using ring-LWE for encyrption they all used fully homomorphic encryption presented by Gentry in 2009 [24]. If we think $N^{1/d}.F$ as communication cost, in ring-LWE's F grows roughly with $d-1$. However, switching to fully-homomorphic encrpytion (FHE) reduced F to 1 instead of $d-1$. Therefore they managed to reduce communication cost compared to the other schemes.

Henzinger et al. introduced SimplePIR [9] at 2022 which uses classical learning-with-error instead of FHE or ring-LWE proposed in other papers. By doing so they managed to eliminate the compression and decompression costs encountered by other schemes. Even though this comes with higher communication cost, the overall performance of the scheme has been enhanced with respect to other protocols until the day of publishment. In this project we will review SimplePIR and regarding construction algorithm in Chapter 5.

Chapter 3

Background

3.1 Cryptographic Primitives

The foundation of computationally Private Information Retrieval (PIR) schemes lies in the security of linearly homomorphic encryption. This type of encryption allows for the computation of additions or multiplications on ciphertexts, resulting in the encryption of the sum or product of the underlying plaintexts. Homomorphic encryption can be constructed by leveraging public key assumptions such as Decisional Diffie-Hellman (DDH), Quadratic Residue (QR), or Learning With Errors (LWE). Thus, we first explain the homomorphic encryption and then necessary tools are defined for better understanding of concepts presented in the following chapters.

For single server PIR schemes, public key cryptosystems play a vital role which are built upon the idea of one-way functions. Even though these are sufficient basis for symmetric encryption, for public key encryption, the function must be invertible and this is satisfied with trapdoor information. The trapdoor permutations first defined by Yao [28] (This concept was initially proposed by Diffie and Hellman [25] and further formalized by Goldwasser and Micali [26]). In fact, building upon previous work of Blum and Micali [27], Yao demonstrated that a set of trapdoor permutations can be employed to construct a secure public-key encryption scheme.

It has been shown by previous studies of Bellare et al. [29] that a trapdoor function can be constructed from any arbitrary one-way function. As indicated in [29], the presence of a trapdoor function with a pre-image size bounded by polynomial (referred as a many-to-one trapdoor function) is adequate for constructing trapdoor predicates and, consequently, for establishing a public key cryptosystem. The work of Goldwasser and Micali [26] demonstrated

the equivalence between trapdoor predicates and semantically secure public key cryptosystems. It should be noted that the pre-image size of trapdoor function serves as a critical parameter in the construction of a secure public key cryptosystem.

Definition 3.1.1 *(Homomorphic Encryption).* A cryptosystem is considered homomorphic when it possesses the capability to perform operations on encrypted data directly, without the requirement of decryption such that:

$$D(E(x) * E(y)) = x.y$$

where $E : G \rightarrow G'$, $x, y \in G$, G and G' are abelian groups under ., * operations respectively. Here E denotes encryption and D denotes decryption

There are many examples to homomorphic cryptosystems. RSA or ElGamal encryptions can be given as an example of multiplicatively homomorphic encryption. On the other hand Goldwasser-Micali, Paillier and Damgard-Jurik encryption systems are examples to additive homomorphic cryptosystems. If an encryption system allows both addition and multiplication on the ciphertext, then it is called *"fully homomorphic encryption".*

Definition 3.1.2 *(One-Way Function-OWF).* A function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ is one-way if the followings hold:

1. $\exists$ poly-time algorithm A that computes $f(x)$ correctly , $\forall x$
2. $\forall$ probabilistic poly-time algorithm A,

$$Pr(A(f(x)) \in f^{-1}(f(x))) \leq negl(k)$$

Here, x is randomly chosen in $\{0,1\}^*$. Consequently, the function f can not be reversed within polynomial time. The probability of successfully computing the preimage of $f(x)$ is considered negligibly small, denoted by $negl(k)$ (k is the security parameter of the algorithm).

Definition 3.1.3 *(One-Way Permutation-OWP).* A function $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ is defined as one-way permutation if the followings hold:

1. It has one-way function properties and
2. Every $y = f(x)$ has a unique preimage x (i.e., bijective)

Definition 3.1.4 *(Trapdoor Permutation-TDP).* A function $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ is considered as trapdoor permutation if the followings hold:

1. It is One-Way Permutation
2. $\exists$ poly-time algorithm I (referred as inverse) with information t and $x \in \{0,1\}^n$, $I(f(x), t) = x'$, where $f(x') = f(x)$

The findings presented in [29] say that the existence of trapdoor function families with polynomially bounded pre-image size implies the existence of a family of trapdoor predicates. These trapdoor predicates exhibit an exponentially small decryption error, bounded by $(1 - \frac{1}{O(k)}).\frac{1}{2}$, where $O(k)$ represents the pre-image size and k denotes the security parameter. This extends the work of Yao and Goldwasser-Micali [28] [26] by showing that a trapdoor permutation is not strictly necessary for achieving a semantically secure public-key cryptosystem. It highlights that the requirement of injectivity can be relaxed in certain cases.

Constructing 2-1 Trapdoor Functions.
This construction based on the paper of Naor and Yung [12] . Let G be a family of one-way trapdoor permutations over $\{0,1\}^n$ and define H such that:

$$H = (h_{a,b} : GF[2^n] \to GF[2^n] \mid h_{a,b}(x) = ax + b, \quad a, b \in GF[2^n],\ a \neq 0)$$

Naor and Yung introduced a family of functions by considering G and H such that:

$$F = (f : \{0,1\}^n \to \{0,1\}^{n-1} \mid g \in G,\ h \in H,\ f(x) = chop(\ h(g(x)))),$$

The defined family of functions involves applying the chop operator to a string, resulting in the removal of its final bit.

The properties below are fundamental characteristics of F:

1. For every function $f \in F$ is 2-1. It means that, for every $x \in \{0,1\}^n$, there is another unique bit-string x', such that $f(x) = f(x')$ and $x \neq x'$
2. With trapdoor information which makes easy to compute g^{-1}, computing x and x' from the equation $f(x) = f(x') = y \in \{0,1\}^{n-1}$ is feasible.
3. To figure out the collisions for F is hard.

There are no proof for the existence of one-way functions, yet we have candidates. Therefore the existence of one-way permutations and trapdoor permutations are not proved either. Let us share some candidates for the sake of completeness of definitions given.

Integer Multiplication as a One-Way Function Candidate.
Let f be a function such that $f(p,q) = p.q$, p and q are k-bit primes.

$$f : \mathbb{P}_k * \mathbb{P}_k \longrightarrow C$$

$\mathbb{P}_k$ is the set of k-bit primes, and C is the set of $2k$-bit numbers. Assume that $n = p.q$, there is no known polynomial time algorithm A such that $A(n)$ outputs the factorization of n. If we test all the numbers from 2 to $\sqrt{n}$ by the algorithm which outputs divisor of n, program run time will be $O(\sqrt{n})$, which is polynomial. However, n is magnitude of 2^{2k} and of size $2k$. Therefore, algorithm runs in time $O(2^k)$ which is exponential. Consequently, there is no algorithm known that can factorize n in polynomial time. While the computation of f is easy, its inversion is computationally difficult.

Modular Exponentiation as a One-Way Permutation Candidate.
The multiplicative group of $\mathbb{Z}_n$, $\mathbb{Z}_n^*$ is defined as:

$$\mathbb{Z}_n^* = \{a | a \in \mathbb{Z}_n,\ gcd(a,n) = 1\}$$

When p is prime, $\mathbb{Z}_p^*$ has at least one element "g" as generator with order $p-1$. Let f be a function such that $f(x) = g^x \bmod p$, where p is a prime number, g is a generator and $x \in \mathbb{Z}_p^*$. Since g is generator, this function $f : \mathbb{Z}_p^* \longrightarrow \mathbb{Z}_p^*$ is a permutation. Computing $y = g^x \bmod p$ takes polynomial time. However finding x such that $g^x = y \bmod p$, considered to be hard when y, p, g values are given because of the known problem which is Discrete-Log Problem. $f_{p,g}$ is hard to invert. Thus, modular exponentiation is a candidate for one-way permutation.

RSA as a Trapdoor Permutation Candidate.
An RSA function is defined as: $f : \mathbb{Z}_n^* \longrightarrow \mathbb{Z}_n^*$, $f(x) = y = x^e \bmod n$, where $n = p.q$, p and q are primes, $e, d \in \mathbb{Z}_{\phi(n)}$, $e^{-1} = d \bmod \phi(n)$, $\phi(n) = (p-1).(q-1)$
$f(x) = x^e \bmod n$ can be computed in polynomial time in size of n, which is k (i.e.,n is k-bit number).

In order to reverse RSA encryption, it involves decomposing the number "n" into its prime factors. This factorization enables the determination of $\phi(n)$, which in turn helps calculate

a value for "d" and retrieve "x" from "y," as. It is known that this factorization process is challenging, as no polynomial-time algorithm has been discovered thus far. Notably, alternative methods for reversing the function "f," such as directly obtaining $\phi(n)$ or "d," have been proven to be equally difficult as factorization. Consequently, inverting f is defined to be hard as it takes exponential time. But with the value "d" (which is known as trapdoor information), RSA is easy to invert. x can be computed back by doing:

$$x^{e.d} = x \bmod n$$

Therefore RSA is a candidate for trapdoor permutation.

Definition 3.1.5 *(Hard-Core Predicate).* A predicate $h : \{0,1\}^n \to \{0,1\}$ is a hard-core predicate for f, if h is efficiently computable when given x and there exists a negligible function of n such that for every non-uniform probabilistic-polynomial time adversary A:

$$Pr[x \xleftarrow{R} \{0,1\}^n : A(1^n, f(x)) = h(x)] \leq \frac{1}{2} + negl(n)$$

This implies that when given a function $f(x)$ with x chosen uniformly random, a computationally bounded adversary who tries to determine $h(x)$ cannot do make accurate predictions beyond a random guess of 0 or 1, with equal probabilities.

One-way functions (OWFs) are often not particularly useful on their own. While they do ensure that $f(x)$ conceals the original value of x, they may not necessarily hide specific subsets of x's bits. As a result, f could potentially disclose several bits of x, but it fails to reveal the hard-core bit. In other words, obtaining knowledge of the hard-core bit of x, even when given $f(x)$, is just as difficult as inverting the function $f(x)$ itself.

According to Goldreich and Levin [13], every one-way function can be transformed to a one-way function that has a specific hard-core predicate such that:

Let f be an OWF, and define another OWF g as:

$g(x,r) = (f(x), r)$ where the length of x and r are equal and r is chosen randomly.

Then h function is defined as:

$$h(x,r) = \langle x, r \rangle, \quad (\langle x, r \rangle = \sum x_i.r_i \mod 2, \text{ i.e., inner product})$$

For a random r, the value of $\langle x, r \rangle$ is hard to compute when $f(x)$ and r are given. An adversary cannot figure out the hardcore bit of g with probability greater than $1/2$. Therefore

h is hard-core predicate for OWF g.

Definition 3.1.6 *(Pseudo Random Function-PRF).* A keyed function F is a two-input function F :$\{0,1\}^*$x$\{0,1\}^* \rightarrow \{0,1\}^*$. $F_k(x) = F(k,x)$. We call F_k is a PRF if (key k is chosen randomly) it is indistinguishable from any other uniformly random chosen function from set of all functions.

Definition 3.1.7 *(Pseudo Random Generator-PRG).* A deterministic polynomial time function $G : \{0,1\}^m \rightarrow \{0,1\}^n$ is PRG if $m < n$ and two probability ensembles $\{A_n\}_{n\in\mathbb{N}}, \{B_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable. $\{A_n\}$ defined as the output of G on a uniformly selected seed $s \in \{0,1\}^m$. $\{B_n\}$ defined as uniformly distributed on $\{0,1\}^n$

Hastad et al. [30] proved that pseudo random generators exist iff one-way functions exist. We can construct a pseudorandom generator with the help of hardcore predicates from any one-way function. If h is a hardcore predicate of an OWF g and s is a random seed, then $\{h(g^n(s))\}_n$ is a pseudorandom sequence. As an example, following process can be thought: Let seed s as $\langle x, r\rangle$, where x_0 is set to x. Then one calculates x_i as $g(x_{i-1})$ iteratively, and the output bits h_i's are obtained by inner product of x_i with r.

Definition 3.1.8 *(Decisional Diffie Hellman Problem-DDH Problem).* Let $\mathbb{G}$ is a cyclic group of prime order q generated by $g \in \mathbb{G}$. $\mathbb{G}$ is subgroup of $\mathbb{Z}_p^*$. Given $X, Y, Z \in \mathbb{G}$, it is hard to distinguish between random DH-triplets and random tuples. Any PPT adversary A has negligible advantage to determine if X, Y, Z is a DH-triplet or non-DH triplet.

X, Y, Z is called DH-triplet if:

$$X = g^x \bmod p, \quad Y = g^y \bmod p, \quad Z = g^{x.y} \bmod p$$

Otherwise, X, Y, Z is called non-DH triplet.
In this definition, x or y is a "trapdoor information" that enables to efficiently solve the DDH problem.

Definition 3.1.9 *(Learning With Errors Assumption-LWE).* Learning with errors problem is introduced by Regev [18]. The assumption is that it is hard to recover s $\in \mathbb{Z}_q^n$ given approximate linear equations in the form:

$$a_{1,i}s_1 + ... + a_{n,i}s_n \approx b_i$$

where $i = [1, ..., m]$ with $m > n$. To create these approximate equations, $n \geq 1$ as size parameter, $q \geq 2$ as modulus and χ defined on $\mathbb{Z}_q^n$ as error probability distribution should be set. A random vector $\mathrm{a} \in \mathbb{Z}_q^n$ is chosen along with an $e \in \mathbb{Z}_q$ according to the probability distribution χ and $(\mathrm{a}, \langle \mathrm{a}, \mathrm{s} \rangle + e)$ is computed. Iterating this step m times one can create probability distribution $A_{\mathrm{s},\chi} \in \mathbb{Z}_q^m$. Solving this system is thought to be hard for multiple reasons. LWE is an extension of learning parity with noise problem and that problem is thought to be hard. Until now, the algorithms developed for learning parity with noise problem run in exponential time. The hardness of LWE is also based on the hardness of the worst-case hardness of standard lattice problems.

Definition 3.1.10 *(Secret-key Regev Encryption).* It is used in SimplePIR, for that reason we will briefly show this scheme. In 2009 Regev introduced this scheme in [18], and it is secure under LWE assumption. The parameters n, q and χ should be set before starting and we have p as the plaintext modulus.

Secret Key: The secret key s is randomly chosen vector from $\mathbb{Z}_q^n$.

$$\mathrm{s} \xleftarrow{R} \mathbb{Z}_q^n$$

Public Key: Public key represents the m approximate equalities in the form of (a_i, b_i) where $i = (1, ..., m)$. b_i is calculated as

$$b_i = \langle \mathrm{a}_i, \mathrm{s} \rangle + e$$

where $e \xleftarrow{R} \chi$

Encryption: The public key and ciphertext pair of a message $m \in \mathbb{Z}_p$ is shown as below

$$(\mathrm{a}, c) = (\mathrm{a}, \langle \mathrm{a}, \mathrm{s} \rangle + e + \lfloor q/p \rfloor .m)$$

Decryption: In order to decrypt one needs the secret key s so that she can calculate $c - \mathrm{a}^T\mathrm{s}$ mod q. Finally she needs to find the nearest multiple of $\lfloor q/p \rfloor$.

For this scheme to work as it should, the error e should be sampled s.t. $e < 1/2\lfloor q/p \rfloor$ and it can be enforced while sampling χ.

It is important to note that Regev encryption is additively homomorphic s.t. $(\mathrm{a}_1, c_1) + (\mathrm{a}_2, c_2) = (\mathrm{a}_1 + \mathrm{a}_2, c_1 + c_2)$ where errors are small enough.

3.2 Preliminaries

In this section, some definitions and theorems are presented which are useful for construction of single server PIR schemes.

Definition 3.2.1 *(Polynomial Time Algorithm).* If a problem can be solved in polynomial time, it means that there exists an algorithm A capable of solving it. This algorithm's complexity is bounded by a polynomial function of the input length 'n'. The number of steps required by the algorithm to solve the problem remains within a reasonable limit, directly related to the size of the input.
An algorithm A, input of size n, it is defined as polynomial time algorithm if it runs in $O(n^c)$ time, where c is a constant.

Definition 3.2.2 *(Probabilistic Polynomial Time Algorithm).* If an algorithm A is randomized and operates in polynomial time, it is considered probabilistic. By utilizing randomness in its computation, the algorithm produces non-deterministic outputs. As a result, the prediction of specific outputs is only possible with certain probabilities. This means that the algorithm is allowed to incorporate coin flips during its execution.

Definition 3.2.3 *(Negligible n (negl(n))).* An arbitrary function $f(n)$-type of probability function is negl(n) if for all $c \in \mathbb{R}^+$, there exists $n' \in \mathbb{N}$ such that:

$$(\forall n \geq n') \left[f(n) < \frac{1}{n^c} \right], \text{ where } n \text{ is input size}$$

Definition 3.2.4 *(Indistinguishable Distributions).* Let $S = \{S^n\}_{n\geq 1}$ be a probability distribution ensemble on some finite domain. Given two distribution ensembles $S = \{S^n\}$ and $T = \{T^n\}$ are indistinguishable if every PPT algorithm A cannot distinguish whether the element of ensemble is generated corresponding to S or T:

$$|Pr[A(s, 1^n) = 1] - Pr[A(t, 1^n) = 1]| \leq \frac{1}{n^c} = negl(n),\ s \in \{S^n\},\ t \in \{T^n\},\ \forall c \geq 1,$$

Before moving to other concepts, let us show the requirements of PIR. There are two requirements to have a successful PIR scheme which are "correctness" and "security". We have 3 efficient algorithms in PIR, Q as a query algorithm which takes desired index of client as an input. A as an answer algorithm of the server which takes database and query of the client as inputs. Finally we have reconstruction algorithm R for the client which takes answers of server as an input:
$Q(1^n, i, r) = q,$ is a query algorithm,where r is random string and n is database size
$A(q, x) = a,$ is an answer algorithm,where x is database
$R(1^n, i, r, a) = x_i,$ is the reconstruction algorithm

Client's Computational Privacy (Security).
To give brief explanation, malicious server that does not follow the protocol can learn nothing about the adaptive indices that the client's reading. For every two adaptive sequences of indices $\{i_n\}_{n\geq 1}$ and $\{j_n\}_{n\geq 1}$, where $1 \leq i_n, j_n \leq n$ the distribution ensembles, server's view is computationally indistinguishable whether the client is making query sequences $\{Q(1^n, i_n, r)\}$ or $\{Q(1^n, j_n, r)\}$:

$$\{\text{server's view on query sequence:}\{Q(1^n, i_n, r)\}\} \approx_c \{\text{server's view on: } \{Q(1^n, j_n, r)\}\}$$

Correctness of PIR.
If we have an honest client and a honest server that execute the protocol faithfully, then for any database held by the server and for any adaptive indices queried by the client, the client can correctly recover all the database bits that it wants to read with overwhelming probability. If client always outputs the correct value of x_i, then there is a correctness of the protocol. For every database of length n, database $x \in \{0,1\}^n$, every index $i \in \{1, 2, ..., n\}$, every random string r :

$$Pr[R(1^n, i, r, a_i) = x_i] = 1, \text{ where } q_i = Q(1^n, i, r),\ a_i = A(i, q_i, x), \text{ for } i = 1, ..., n$$

Communication Complexity of a PIR Protocol.
When examining a PIR protocol, the communication complexity can be defined as the upper limit on the number of bits exchanged between the client and the server. This value is determined from the size of query sent by client and the size of answer sent by server.

Honest-but-Curious Model.
There are typically two protocol settings when it comes to interactions between a client and server. The first setting is known as the "Honest-but-Curious Model," while the second one is

referred to as the "Malicious Model." In 1987, Goldreich [31] proposed transformations that can convert protocols designed for security against the Honest-but-Curious Model into protocols that provide security against the Malicious Model. But, it is worth noting that these transformations often come at the expense of increased communication complexity.

In the Honest-but-Curious Model, participants in the communication strictly adhere to the defined protocol but actively attempt to extract all possible information from the legitimately received messages.

In order to give formally description, let data string $x \in \{0,1\}^n$, for every $i, j \in [n], (i \neq j)$. The objective is to ensure that the distribution of the client's index "j" cannot be distinguished from the distribution of query when the index is "i" for the server's view. In this model, it is crucial to note that the server is strictly prohibited from altering the data string throughout the execution of the protocol.

Malicious Model.

In a communication scenario involving a malicious server, it is important to note that the server is not obligated to adhere to the protocol. The server has the freedom to either refuse participation in the protocol or manipulate the contents of the database. Regardless of the actions taken by the server, it is necessary that the identity of the interest index "i" remains secret in the PIR protocol. More formally, for any given indices "i" and "j" within the range $[n]$ $(i \neq j)$ the distributions of any two indices are indistiguishable to the server.

Chapter 4

Basic Single Server PIR Schemes and Their Communication Complexities

In this chapter, communication complexities of single server PIR schemes and how to improve them by using balancing and recursion techniques instructed by Lipmaa [5] are shown. Then an example of CPIR protocol scheme which is based on Diffie-Hellman assumption is given. Further the construction of Symmetric PIR protocol via three different ways are presented. The first is using CPIR scheme, the second way is using log$n$ iteration of 1-out-of-2 Oblivious Transfer protocol and the last way is using Oblivious Pseudo-Random Function(OPRF). In order to understand the techniques used, two different 1-out-of-2 OT protocol schemes are given, one is based on DDH assumption constructed by Chou and Orlandi[32] and the other one proposed by Rivest [33]. At the end of the chapter, OPRF scheme is given using by DDH assumption and its semantically security is shown.

4.1 Single Server PIR Scheme With Different Complexities

The communication complexity refers to the number of bits exchanged between the client and the server. In the naive solution, the total complexity is equivalent to the size of the entire database, 'n'. Although this approach ensures perfect privacy, it becomes prohibitively costly in terms of complexity, especially when dealing with large databases. A single server PIR scheme is presented with 3 different communication complexities. First, the naive approach is presented with $O(n)$ communication complexity. Second, balancing techniques based on matrices are used to reduce the communication complexity to $O(\sqrt{n})$. Finally, recursion

method is applied to further reduce communication complexity to polylogaritmic.

4.1.1 First Approach

DDH assumption is used to construct CPIR protocol with $O(n)$ communication complexity. The construction can be seen in table 4.1. Here, client's input: an index $i \in \{1, ..., n\}$ and server's input: a database $x \in \{0,1\}^n$

Table 4.1: A single server PIR protocol with $O(n)$ communication complexity

Client				Server		
Process	Secret	Public		Public	Secret	Process
chooses generator and prime numbers		p,q,g	$\Rightarrow$			
prepares n-triplets only i-th triplet is non-DH tuple	$x \xleftarrow{R} \mathbb{Z}_q$ $y_1, ...y_n \xleftarrow{R} \mathbb{Z}_q$ $z_i \neq x.y_i$					
calculates public values sends Y_j, Z_j, p, q, g for $j = 1, ...n$		$X = g^x \bmod p$ $Y_j = g^{y_j} \bmod p \;\; j = 1, ..., n$ $Z_j = g^{x.y_j} \bmod p \;\; j = 1, ..i-1, i+1, ..n$ $Z_i = g^{z_i} \bmod p$	$\Rightarrow$			
			$\Leftarrow$	$Y = \prod_{(j:x_j=1)} Y_j \bmod p$ $Z = \prod_{(j:x_j=1)} Z_j \bmod p$		sends Y, Z to the client

In this scheme, the client prepares n triplets such that only the desired index ith triplet is non-DH triplet, the others are DH-triplets. The server calculates the multiplication of the triplets corresponding to database values if $x_j = 1$ for $j = 1, ..., n$. If desired value x_i is equal to 0 then it is not multiplicated with DH-triplets, since multiplication of DH-triplets is again DH-triplet, client can figure out x_i value. On the other hand if x_i is equal to 1, then non-DH triplet which is constructed by client participates in the multiplication. Since multiplication of DH-triplet with non-DH triplet is non-DH triplet, client easily observes that $x_i = 1$. According to DDH assumption, the server can not distinguish between tuples sended by client. Thus, client's security based on the DDH assumption.
The client sends $2n$ bits to server and server sends only 2 bits so total communication costs $2n + 2$ bits. In this protocol, complexity is $O(n)$.

4.1.2 Second Approach

In the protocol mentioned above, the communication between the client and the server is unbalanced. The client transmits significantly larger number of bits compared to the server. To address this imbalance, matrices can be used as in [1]. By treating the data bit string as a two-dimensional array, we can achieve a more balanced communication scheme. Let define

(x_{i_1}, x_{i_2}) as a database element of $\sqrt{n} \times \sqrt{n}$ matrix, $1 \leq i_1, i_2 \leq \sqrt{n}$.
Let assume that the desired index is $i = (i_1, i_2)$.

$$i_1 = \lceil (i/\sqrt{n}) \rceil$$

$$i_2 = ((i-1) \bmod \sqrt{n}) + 1$$

Client prepares queries such that i_2-th triplet is non-DH triplet. On the server side each row of matrix is considered as database and server makes calculations according to elements of each row with given query. Therefore answer of the server in the above scheme is the total number of the rows of matrix multiplied with 2(because there are Y and Z to be calculated), i.e.,$2.\sqrt{n}$. And the total communication between client and server is $4\sqrt{n}$. Thus communication complexity becomes $O(\sqrt{n})$ with balancing method.

In order to understand the the idea, a simple example is given:

Example For Balancing Method.
Assume that $n = 25$, then the two-dimensional array of database is:

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & x_{3,5} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} & x_{4,5} \\ x_{5,1} & x_{5,2} & x_{5,3} & x_{5,4} & x_{5,5} \end{bmatrix}$$

Let the interested index of client be $i = 8$, then $i = (2, 3)$:

$$i_1 = \lceil (i/\sqrt{n}) \rceil = 2$$

$$i_2 = ((i-1) \bmod \sqrt{n}) + 1 = 3$$

The client prepares a query vector which is blinded under DDH assumption such that only 3-rd row value is non-DH triplet. In this example non-DH triplet denoted as 1 and DH-triplet as 0 to understand the concept:

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

The server calculates $\hat{Y}_j$ and $\hat{Z}_j$ where $j = 1, ..., \sqrt{n}$ for each row and sends resulting vector to the client. Note that $\hat{Y}_j$ and $\hat{Z}_j$ are calculated similarly to Y and Z in previous approach.

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & x_{3,5} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} & x_{4,5} \\ x_{5,1} & x_{5,2} & x_{5,3} & x_{5,4} & x_{5,5} \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \hat{Y}_1, \hat{Z}_1 \\ \hat{Y}_2, \hat{Z}_2 \\ \hat{Y}_3, \hat{Z}_3 \\ \hat{Y}_4, \hat{Z}_4 \\ \hat{Y}_5, \hat{Z}_5 \end{bmatrix}$$

The client wants to learn $x_{2,3}$, so it takes the $(\hat{Y}_2, \hat{Z}_2)$ from answer of server. If $(\hat{Y}_2, \hat{Z}_2)$ is DH- triplet then $x_{2,3} = 0$, else 1.

Note that in the CPIR protocol, client's query is based on public key cryptosystems. Therefore server cannot distinguish the queries between random and desired one. So it does not have any idea of client's requested data index. However, the client has side informations, i.e., $(x_{1,3}, x_{3,3}, x_{4,3}, x_{5,3})$ in this example. There is no requirement to have server's privacy in the above CPIR scheme. Later we describe PIR protocol to have privacy for server as well.

4.1.3 Third Approach

In the above approach, server gives long response to client although the client only needs one element in the database. In order to reduce the response size, recursion method is described. In this method, client and server operates a new PIR protocol with interested index i_1 and the database becomes server's previous answer. By doing this iteratively, communication complexity reduces to sublinear in database size. It was also shown that by choosing right parameters and encryption scheme polylogarithmic communication complexity can be achieved
In order to understand the method, following example is given:

Example of Recursive Method.
Assume that the client chooses large enough prime number p with generator g and let DDH assumption holds for a cyclic group $\mathbb{G}$ with order of q.
Consider the database as $a \times b$ matrix with length n. Assume that entries of matrix are log$p$-bit string block. Let a be $n^{1/5}$.log$p$, then b is equal to $\frac{n^{4/5}}{\log p}$.
As we know from the above construction, the answer of the server consists of the number of the rows which is $b = \frac{n^{4/5}}{\log p}$. The new database in the second process of the protocol is

the previous answer of server. Note that the answer has not been sent the client. In this recursion, client and server uses same $a = n^{1/5}.\log p$. But b is changed to $(\frac{n^{4/5}}{\log p})/a = \frac{n^{3/5}}{\log^2 p}$. After the 4 iteration, the response is consist of $\frac{n^{1/5}}{\log^4 p}$ blocks with $\log^4 p$- bit string. Therefore, communication complexity is $O(n^{1/5}.\log^4 p)$.

By selecting the parameters appropriately, the communication complexity of the PIR protocol can be significantly improved, leading to an efficient solution:

Let m be equal to $\sqrt{\frac{\log n}{\log\log p}}$ and b equal to $n^{1/m}$ for each iteration.

Query length in each iteration is: $O(b) = O(n^{1/m}) = O(n^{1/\sqrt{\frac{\log n}{\log\log p}}}) = O(2^{\sqrt{\log n \log\log p}})$

Since we have m iteration, total communication is:

$O(m.2^{\sqrt{\log n \log\log p}})$ which is less than $O(\log n.2^{\sqrt{\log n \log\log p}})$ and:

$\frac{\lim_{n\to\infty} \log n.2^{\sqrt{\log n \log\log p}}}{\lim_{n\to\infty} n^{\epsilon}} = 0 < 1, \quad \text{thus } \log n.2^{\sqrt{\log n \log\log p}} < n^{\epsilon}, (\log p < n)$

Therefore, the resulting complexity is smaller than $n^{\epsilon},\ \epsilon > 0$

4.2 Symmetric Single Server PIR Schemes and Their Construction Methods

In SPIR protocol, server learns no information about the client's desired bit, and the client only learns the bit it wants and nothing else. Such protocols can be thought of as Oblivious Transfer protocols.
3 methods are described to construct SPIR. First one is a variant of the presented as protocol above. Then a general transformation from PIR to SPIR is showed by recursively using 1-out-of-2 OT. Finally, OPRF protocol is used to have server's privacy from any standard PIR.

4.2.1 First Method-From CPIR to CSPIR

Assume that the same protocol used above is utilized. In order to provide privacy for server, client should not know which Y_j's are participated in Y. For example if database consists of zero-bit string then $Y = 1$. To overcome this problem, server chooses random $r \in \mathbb{Z}_p^*$ and blind Y with it. Thus, there is additional step in this protocol on the server side.
After receiving the answer of server, clients computes if X, Y, Z is DH-triplet or not by using its trapdoor information x. A summary of protocol can be seen in table 4.2. If the tuple

is DH-triplet then requested $x_i = 0$ otherwise $x_i = 1$. Server's privacy is coming from the randomness of r, since r is chosen uniformly random, so the value of Y is also distributed randomly in $\mathbb{Z}_p^*$.

Table 4.2: A symmetric PIR protocol from CPIR scheme

Client				Server		
Process	Secret	Public		Public	Secret	Process
chooses generator and prime numbers		p,q,g	$\Rightarrow$			
prepares n-triplets only i-th triplet is non-DH tuple	$x \overset{R}{\leftarrow} \mathbb{Z}_q$ $y_1, ...y_n \overset{R}{\leftarrow} \mathbb{Z}_q$ $z_i \neq x.y_i$					
calculates public values sends Y_j, Z_j, p, q, g for $j = 1, ...n$		$X = g^x \bmod p$ $Y_j = g^{y_j} \bmod p \;\; j = 1, ..., n$ $Z_j = g^{x.y_j} \bmod p \;\; j = 1, ..i-1, i+1, ..n$ $Z_i = g^{z_i} \bmod p$	$\Rightarrow$			
			$\Leftarrow$	$Y = g^r . \prod_{(j:x_j=1)} Y_j \bmod p$ $Z = X^r . \prod_{(j:x_j=1)} Z_j \bmod p$	$r \overset{R}{\leftarrow} \mathbb{Z}_p^*$	sends Y, Z to the client

4.2.2 Second Method-General Transformation Using OT_1^2

In order to describe the technique, OT_1^2 definition is given and 2 examples are shown. First one is instructed by Rivest [33], and the other one is constructed by Chou and Orlandi [32] which is based on DDH assumption.

Definition 4.2.2.1 *(OT_1^2 Protocol) .* Assume that sender has 2 messages m_0 and m_1 and it wants to make sure that only one message is retrieved by the receiver while the other one remains unknown. As a result of the protocol, receiver gets the message of its choice and does not know the other value. Sender does not have any idea of which message is taken.

OT_1^2 Protocol Proposed by Rivest [33].
'Trusted Initializer(TI)' is needed for initialization stage. It generates randomly $r_0, r_1 \in \{0,1\}^n$ and chooses randomly $b \in \{0,1\}$. In table 4.3, process is shown.

Table 4.3: Rivest OT_1^2 Protocol

Receiver				Sender		
Process	Secret	Public		Public	Secret	Process
wants to have m_c $c \in \{0,1\}$	c				$r_0, r_1 \in \{0,1\}^n$ $m_0, m_1 \in \{0,1\}^n$	TI gives r_0, r_1 to sender
TI randomly chooses $b \in \{0,1\}$ sends b and r_b to receiver	b r_b					
computes a and send it to sender		$c \oplus b = a$	$\Rightarrow$ $\Leftarrow$ $\Leftarrow$	 $t_0 = m_0 \oplus r_a$ $t_1 = m_1 \oplus r_{1-a}$		computes t_0 and t_1 and sends to receiver

At the end of the protocol, receiver computes $m_c = t_c \oplus r_b$. Receiver can not learn the m_{1-c} because of the $\oplus$ of random values of r_0, r_1. Also, sender does not have any idea of which message is retrieved.

OT_1^2 Protocol by Chou and Orlandi [32].
Let DDH assumption holds for a cyclic group $\mathbb{G}$ with generator g, order of q and $\mathbb{G}$ is subgroup of $\mathbb{Z}_p^*$.

Table 4.4: Chou and Orlandi OT_1^2 Protocol

Receiver				Sender		
Process	Secret	Public		Public	Secret	Process
wants to obtain m_c $c \in \{0,1\}$	c $y \xleftarrow{R} \mathbb{Z}_q$		$\Leftarrow$	$X = g^x \bmod p$	$x \xleftarrow{R} \mathbb{Z}_q$ $m_0, m_1 \in \{0,1\}^n$	
Prepares $K_c = H(X^y)$ Send Y to sender		$Y = g^y \bmod p$, if $c = 0$ $Y = X.g^y \bmod p$, if $c = 1$	$\Rightarrow$			
			$\Leftarrow$	$C_0 = E(K_0, m_0)$ $C_1 = E(K_1, m_1)$	$K_0 = H(Y^x)$ $K_1 = H((Y/X)^x)$	Computes K_0 and K_1 encrypts messages with these

At the end of the protocol, receiver can compute $m_c = D(K_c, C_c)$. The process is given in table 4.4. If receiver chooses c as 0, it can not compute K_1 as x is not known to client. On the other hand if it chooses c as 1 then K_0 can not be computed since x is unknown. On the sender side, since DDH assumption holds, sender can not distinguishes between g^y and g^{x+y} without knowing $y \in \mathbb{Z}_q$.

Now, construction of SPIR can be shown by iteratively using OT_1^2 Protocol. This construction has 2 steps. First, client and server operates a standart PIR protocol which has efficient communication complexity. In regarding PIR protocol, each bit of database is encrypted by different keys which are chosen by server. At the end of the PIR protocol, the client has its

desired bit with the encrypted version and lots of encrypted side information of database. To be able to decrypt the wanted value, the corresponding key need to be owned by the client, however other keys should be kept secret. In the second step, OT_1^2 protocol is used recursively between the client and the server log$n$ times. The index of interest j is represented as log$n$-bit string $j = (j_1, ..., j_{\log n})$ and key $k_j = (k_{1,j_1}, ..., k_{\log n, j_{\log n}})$ created from randomly chosen 2.log$n$ keys $k_{1,0}, k_{1,1}, ..., k_{\log n,0}, k_{\log n,1}$. In the protocol the client need to obtain which key is used without letting the server know the retrieved key. Obviously, $n = 2^{\log n}$ is the number of subset of the keys to encrypt the each value of the database.

Database $x \in \{0,1\}^n$, index $i \in \{1, ..., n\}, i = (i_1, ..., \log n)$

- Server prepares random keys $k_{1,0}, k_{1,1}, ..., k_{\log n,0}, k_{\log n,1}$, for $j \in \{1, ...n\}$ and $j = (j_1, ..., j_{\log n})$ With these keys server encrypt each data value such that:
  $y_j = E_{k_{1,j_1}}(...E_{k_{\log n, j_{\log n}}}(x_j))$
- Server and client operates the standard PIR protocol and at end of the protocol, client has desired y_i value with the encrypted side informations.
- To retrieve the corresponding key (i.e., which encrypts the x_i value) from pair of keys $(k_{i,0}, k_{i,1})$, server and client executes OT_1^2 log$n$ times.
- After OT_1^2 process, client has the corresponding key that encrypted the data x_i. Owing to OT_1^2, client does not know the other pair of keys and server does not know which key pair is retrieved, either.

4.2.3 Third Method-General Transformation Using OPRF

In order to describe this method, we give definition of OPRF protocol and scheme of it by using the DDH assumption.

Definition 4.2.3.1 *(OPRF Protocol.)* An Oblivious PRF protocol is a protocol that the client only has the output value $y = F(k, x)$ without learning input value k of PRF F and the server does not know which x value is interested in by client.

OPRF Protocol Scheme with DDH assumption.
The protocol is shown in table 4.5. Let DDH assumption holds for a cyclic group $\mathbb{G}$. $F_k = H(x)^k$ is PRF and $k \in \mathbb{Z}_q$.

Table 4.5: OPRF protocol

Client				Server		
Process	Secret	Public		Public	Secret	Process
choses random r and takes x as input for hash	$r \xleftarrow{R} \in \mathbb{Z}_q$	$y = H(x)^r \in \mathbb{G}$	$\Rightarrow$		k	chooses random encryption key $k \in \mathbb{Z}_q$
			$\Leftarrow$	$z = y^k \in \mathbb{G}$		
Computes $z^{1/r} = H(x)^k$						

$$z^{1/r} = y^{k^{1/r}} = ((H(x)^r)^k)^{1/r} = H(x)^k$$

At the end of the protocol, client has output of the OPRF without letting server know x and without learning the value of key k. Thus, client can not decrypt other data values.

Security Of Above Protocol.

If both of the client and server are honest, then client correctly computes the output of the function. Suppose that client is honest but server is malicious. It means that server sends z' which is not equal to y^k. After taking $1/r$ power of z', client has wrong output of PRF and cannot decrypt the output of PIR process. However server cannot learn anything about the input value.

Assume that the client is malicious. We want to find out how much side information can be retrieved from server. The point is to show output of F_k cannot be retrieved by a client at t inputs unless if there are at least t interactions between client and server. Security of F_k is based on the assumption called 1MDH (one more DDH). Attack game of the 1MDH assumption is given in table 4.6:

Table 4.6: 1MDH attack game

Adversary				Challenger		
Process	Secret	Public		Public	Secret	Process
adversary is given A and arbitrary B			$\Leftarrow$	$A = g^\alpha \in \mathbb{G}$	$\alpha \xleftarrow{R} \mathbb{Z}_q$	
			$\Leftarrow$	$B \xleftarrow{R} \mathbb{G}$		chooses random B
makes t queries to the challenge oracle			$\Leftarrow$	B_i		prepares $B_i \in^R \mathbb{G}$
makes $t-1$ queries to the Diffie Hellman oracle		B_i	$\Rightarrow$ $\Leftarrow$	C_i		computes $C_i = g^{\alpha.\beta_i}$

At the end of the attack game, adversary needs to find all t solutions which are $C_i = g^{\alpha.\beta_i}$. If the adversary can output more C_i elements than has been sent as responses of DH oracle

queries with non-negligible advantage, then the adversary wins the game. However, adversary can not produce α power of any different element of $\mathbb{G}$ even if it sees the lots of values that powered with α. Since $F_k = H(x)^k$, 1MDH assumption can be thought such as:
Adversary is given $(g, g^k, g_1, g_2, ..., g_n)$ and it makes $t-1$ calls to a k-exponentiation oracle (i.e., which computes $(.)^k$). Under 1MDH assumption, the adversary cannot output ${g_i}^k$ for more than $t-1$ elements in $(g_1, g_2, ..., g_n)$. Thus, 1MDH assumption implies the security of above protocol (it is known that 1MDH assumption implies the CDH assumption). It means that malicious client can not learn the output of F_k at t inputs unless if there are at least t interactions between itself and server.

From OPRF to SPIR.
The server encrypts the database record by record with different key. Each value of database $(x_j, j = 1, ..., n)$ is encrypted by a secret key s_j which is output of OPRF:

$$y_j = E(s_j, x_j), \text{ for } j = 1, ..., n$$

$$s_j = F_k(k, j), \text{ where } k \text{ is chosen randomly by server at the beginning.}$$

In the first step, client and server operates a standard PIR protocol that does not guarantee the privacy of server. At the end of the protocol client has some side informations that are encrypted besides the interested data value. In order to decrypt the y_i, client has to have s_i key value. In the second step, server and client runs the OPRF protocol with client's input i and server's secret value k. Finally, client can get output of F_k which is s_i. And owing to privacy property of OPRF client can not have other s_j values and server can not learn the i value.
Security of such SPIR scheme comes from the security of OPRF and PIR protocol itself.

4.3 Reducing the Computation Complexity of Single Server PIR

In the computational PIR, protocols take much more server time than information-theoretic PIR protocols. For example if we think the scheme which is based on DDH assumption, there is approximately $O(\log^3 p)$ cost to do one exponentiation in $O(n)$ operations. According to Beimel et al. [34], server must do linear amount of work to respond a single query (i.e., $\Omega(n)$). The intuition behind this, lower bound is that if while responding to a query the server does not touch a particular bit in the database, than the server may learn that client can not be reading that record of interest. This also holds for non-colluding multi-server

option and it is irrespective of cryptographic assumptions. Thus server needs to scan the whole database linearly in the size of it.

Prior works has shown that there is a chance to get sublinear server time in many queires setting(i.e., sublinear amortized time). In the many queries setting, we can build PIR schemes where the amortized server time per query is sublinear in the database size n. There are mainly two classical approaches to reduce the computational cost of PIR scheme which are Batch PIR and Preprocessing PIR. Also there is a recent approach which is called Offline/Online PIR. Brief information about these three methods is given in next subsections.

4.3.1 Batch PIR

Batch PIR first introduced by Ishai [35]. In this scheme, the server and the client pre-agree on a random partition of database into Q buckets with equal size via hash function. This scheme allows the client to read multiple records from database more efficiently than run the PIR scheme one by one on each index. There is negligible probability that one of these buckets contain more than the number of input indices of the client. That is, no bucket contains more than $\lambda.\log n$ indices (for $i_1, ..., i_Q \in [n]$), where λ is security parameter. (This follows from the standard balls into bins result-there are $O(\log n)$ bins in the heaviest loaded bucket). Thus, client can query each bucket $\lambda.\log n$ times. It is sufficient to read all the indices of interest. Note that server sees $\lambda.\log n$ (number of queries per bucket) queries for each bucket regardless of client inputs.

$$\text{Server time} \quad : Q.\lambda.\log n.T(n/Q) = n.\lambda.\log n,$$

where $T(n/Q)$ is time of answering one query of the database with n/Q size and $\lambda.\log n$ is the number of queries per bucket.

Thus, running PIR on much smaller databases (databases of size n/Q) make the cost of computation be much lower. However, this approach only holds when the client makes all queries at once, in other words non-adaptive batch of queries.

4.3.2 Preprocessing PIR

This method is constructed by Beimel et al.[34]. With this scheme, server saves time by pre-computing the responses to all possible queries of the PIR scheme. Server takes the database and encodes it with relative to some PIR scheme by storing the explicit response for PIR queries on that database. Server does this on offline preprocess step. On the online

phase when client needs its query, server can look up the answer from the table of responses and sends the response back to the client. It reads one element from the table, it does not need to read the entire database. Although there is an advantage to have sublinear server time, it is disadvantage to have superlinear server storage since server saves all the possible responses on its database.

If there is a base PIR scheme with communication of $U(n)$ upload cost and $D(n)$ download cost, this preprocessing scheme needs to have space with S and needs to have time T (which is faster than standard PIR scheme) such that :

$$S = D(n).2^{U(n)}, \quad T = U(n) + D(n)$$

For example assume that there is a PIR scheme which has 2-server with the following upload and download size:

$$U(n) = O(\log n) \quad \text{and} \quad D(n) = n^{1/2+\epsilon}$$
$$\xrightarrow{\textit{with preprocessing}} \quad S = \text{poly}(n) \ \text{ and } \ T = n^{1/2+\epsilon}$$

Note that there is polynomial size with the storage which is disadvantage of this scheme. Preprocessing method described is for multi-server settings. Recently, there are developments for analogs in the single server setting like PANDA constructed by Hamlin et al. [36], but also it faces same barrier which is huge storage of the server.

There is also another preprocessing approach for single server PIR schemes. It is quite similar to offline/online method. In offline/online method client does the calculation for hint by sending some queries the database and receiving answers to them. However, in single server preprocessing, server does the calculation for the hint and sends it to client.

4.3.3 Offline/Online PIR

This method consists of two phases- offline and online steps. On online step, each of the Q queries (Q is the bound on the number of adaptive queries) run in a sublinear server time and on the offline step, it takes linear server time which is done only once.

According to work of Corrigan-Gibbs et al.[37], the client first runs offline phase, then it obtains piece of information called-'hint'(it has size sublinear in n) about the database. Although this offline phase still contains lots of computation, it only runs once per client. Client stores the hint given and it can run many queries more efficiently with the help of it. Assume that client wants to read index $i \in [n]$. After sending a query on online phase, client gets a response from server and uses this reply in hint to recover the x_i bit. Then it can send many queries to the server adaptively. These queries are independent of the server's

prior answers. Even by deviating from the protocol, the server can not learn anything about what it is that the client is reading.

Communication and server time is sublinear in the database size on the online phase. Comparing to Batch PIR, client can repeat this online phase to read multiple items adaptively. So, there is no need to commit to a non-adaptive batch process. Comparing to Preprocessing PIR, this scheme stores the exact size of database since client stores hint.

According to [37], after making $n^{1/4}$ adaptive queries, for the single server PIR setting online phase takes time on server side: $n^{3/4}$ (if the factors poly(λ) and log$n$ are ignored). And amortized communication cost is : $n^{1/2}$. This scheme can be built by using any linearly homomorphic encryption (DDH, Quadratic Residue, LWE, etc.) on offline phase with linear time. If fully homomorphic encryption systems are used, server time takes $n^{1/2}$. It is important to note that there is no public key operations on the online phase, yet offline phase has public key operations because of being a single server setting.

Chapter 5

Review of SimplePIR

Henzinger et al.[9] introduced SimplePIR on 2022 which has the best throughput rate to date. They used plain learning with errors instead of ring learning with errors and used Regev encryption [18] which was explained previously in this project.

Since LWE and Regev encryption was already covered, the construction of SimplePIR can be presented now. The database $x = (x_1, x_2, ..., x_N)$ is considered to be a $\sqrt{N} \times \sqrt{N}$ matrix as in the Kushilevitz and Ostrovsky [3] and the client is interested in the entry x_{ij}. The LWE parameters are set at the beginning of the protocol. These parameters are secret key dimension n , error probability distribution χ, LWE modulus q such that $p << q$ where p is the plaintext modulus and matrix $A \in \mathbb{Z}_q^{\sqrt{N} \times n}$.

SimplePIR has a one-time offline preprocessing on the server side. Let us define $D \in \mathbb{Z}_p^{\sqrt{N} \times \sqrt{N}}$ as the matrix form of the database. LWE matrix A is public and fixed for all clients. The matrix $D.A$ (figure 5.1) is calculated by server once at start. This new matrix is also referred as one-time hint and it greatly reduces the processing time on the server side. There is one assumption worth noting at this point, in SimplePIR we assume that the database does not change so that one time hint is valid and same for all clients at all times.

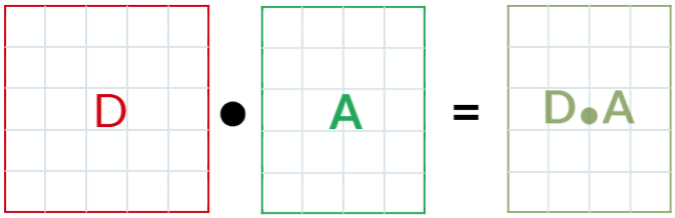

Figure 5.1: Preprocessing

Whenever a client initiate a communication, the server sends client preprocessed $D.A$ as a first step (figure 5.2).

Figure 5.2: The server sends $D.A$

In the next step client prepares its query and sends it to the server. In order to create query, client needs to sample $s \xleftarrow{R} \mathbb{Z}_q^n$ and $e \xleftarrow{R} \chi$. Upon sampling the s and e, the client prepares the query $b = A.s + e + (q/p)u_i$ where u_i is the unit vector with 1 only at the interested i-th entry and 0 elsewhere (figure 5.3).

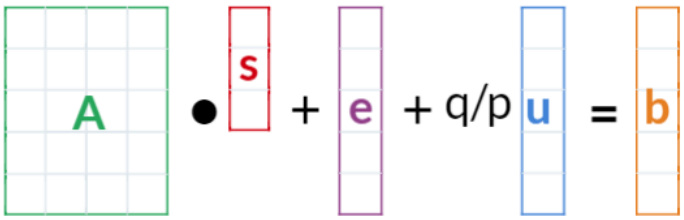

Figure 5.3: Client query creation

The client sends its query b to server. Server creates the response by computing the inner product of the database D with the query b after it receives the query b. (figure 5.4)

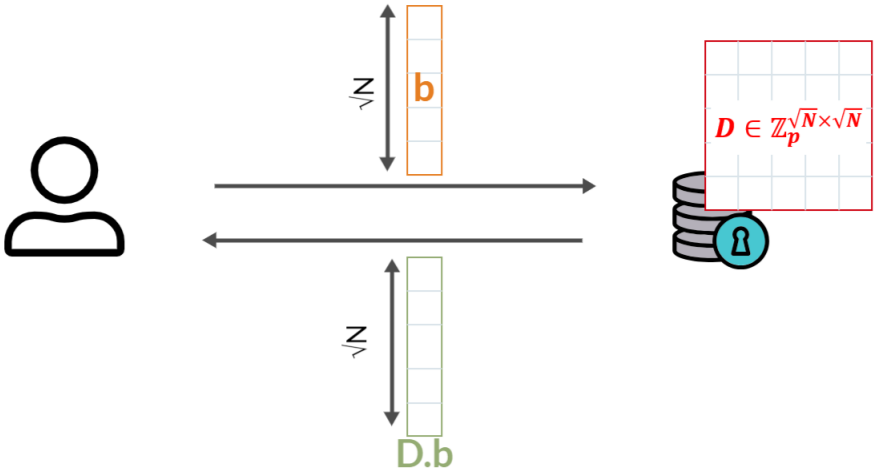

Figure 5.4: The communication between the client and the server

At the end of the communication the client has both $D.A$ and $D.b$ and due to the homomorphism of Regev encryption they are enough to find the interested information d. The client needs to do computations only on i-th row of the $D.b$ and $D.A$. After subtracting $D.A$ from $D.b$, client needs to round result to the nearest multiple of q/p. Finally, dividing the result to q/p reveals the desired information with high probability.

$$\begin{aligned} d' &= D.b_i - D.A_i, \\ d &= \frac{round_{\frac{q}{p}}(d')}{\frac{q}{p}} \end{aligned} \tag{5.1}$$

Let us briefly compute the complexity of the SimplePIR scheme described. At first, server needs to calculate $D.A$ at preprocessing step, it is important to note that this will be done

only once. Since $D \in \mathbb{Z}_p^{\sqrt{N}\times\sqrt{N}}$ and $A \in \mathbb{Z}_q^{\sqrt{N}\times n}$ where $n << N$, $2n.N$ computations (additions and multiplications) are done in this step. So the computational complexity of one-time hint is linear $O(N)$. In the next step, the resulting matrix $D.A \in \mathbb{Z}_q^{\sqrt{N}\times n}$ is transferred to client. Therefore, communication complexity of this step is $n\sqrt{N}$. Then, client calculates the b vector which requires matrix multiplication of A and s with addition of $e+(q/p).u$. The operations required for this step is $(2n+3)\sqrt{N}$. Upon calculating the b, the client uploads this vector to server and its communication complexity is simply $\sqrt{N}$. Server receives the query and then calculates its answer $D.b$ and this requires matrix multiplications with $2N$ operations. Downloading the server's answer has the same communication complexity with upload process. Finally client needs to decrypt the answer which has only three operation. The complexities of each step is summarized in table 5.1.

Table 5.1: SimplePIR Computation&Transfer complexity at each step

Step	**Computation/transfer**
$D.A$ (One-time Hint) calculation	$2nN$ computations $\in \mathbb{Z}_q$
$D.A$ (One-time Hint) transfer	$n\sqrt{N}$ elements $\in \mathbb{Z}_q$
Client query preparation	$(2n+3)\sqrt{N}$ computations $\in \mathbb{Z}_q$
Client to Server (Upload)	$\sqrt{N}$ elements $\in \mathbb{Z}_q$
Server query calculation	$2N$ computations $\in \mathbb{Z}_q$
Server to Client (Download)	$\sqrt{N}$ elements $\in \mathbb{Z}_q$
Client Decryption	3 computations

SimplePIR shares a lot of common steps with other PIR protocols [19],[20],[21],[22],[23]. Server's on demand work is not sublinear in SimplePIR unlike other works with preprocessing [37], [38], [39], [40], [41], [42], [43], [44] . However, SimplePIR performs better than all in the benchmarks and this is mainly due to one-time preprocessing unlike others and the simple calculations used in the Regev's LWE encryption scheme. It allowed server to work more efficiently than the other scheme and results in better performance.

	Communication				
	Offline (MB)		Online (KB)		**Throughput**
	Up.♣	Down.♠	Up.	Down.	(MB/s)
SealPIR	5	0	91	181	97
FastPIR	0.06	0	33 000	64	217
OnionPIR	5	0	256	128	60
Spiral	15	0	14	20	259
SpiralPack	19	0	14	20	260
SpiralStream	0.34	0	15 000	20	485
SpiralStreamPack	15	0	29 000	99	1,370★
SimplePIR (§4)	0	121	121	121	10,138
DoublePIR (§5)	0	16	313	32	7,622

Figure 5.5: PIR Scheme Comparison - *Note.* Reprinted from One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval, by Henzinger A., Hong M.M., Corrigan-Gibbs H., Meiklejohn S. and Vaikuntanathan V., 2022, Cryptology ePrint Archive, Paper 2022/949

As it can be seen from figure 5.5 it is not the best scheme in the online and offline communication especially from server to the client side, yet it managed to achieve the highest throughput values by few times better than the next best. However, preprocessing is done only once with the assumption that the database is static and does not change, this scheme might lose it's competitive advantage to the other schemes in dynamic databases. Further development for PIR might be done that consider dynamic databases which we see almost everywhere in the real world.

Chapter 6

Conclusion

In this project, we thoroughly explored single server Private Information Retrieval protocols, reviewing particularly the recently developed SimplePIR scheme. Our primary aim was to understand the strategies devised to minimize communication complexity, while concurrently reducing the overall complexity of single server PIR protocols.

We initiated our discussion with a brief literature on the Private Information Retrieval problem. We found that in single server PIR schemes, the single server information-theoretically secure method was to download the entire database—a highly impractical solution for evident reasons. This finding motivated the search for computationally secure single server PIR schemes. We further showed several key strategies aimed at lowering the communication complexity of PIR schemes. These strategies used various hardness assumptions such as Quadratic Residuosity Problem, ϕ-hiding, trapdoor permutations and diverse approaches to database structures such as considering database as string or as matrix. Consequently, we discovered that communication complexity was reduced to polylogarithmic in database size, which represents the lower bound.

We continued our exploration by presenting key tools used on the evolution of single server PIR schemes in the context of communication complexity and introduced the fundamental elements utilized in our literature research. These included the construction of $2-1$ trapdoor functions, the hardcore predicates problem utilized in early single server PIR schemes, the Pseudo Random Generator, and the Decisional Diffie Hellman Problem. These elements were instrumental in constructing computationally secure PIR (CPIR) and symmetric PIR (SCPIR) schemes. We also gave brief information about the Learning With Errors (LWE) problem and Regev's LWE encryption, which have been employed in more recent PIR schemes.

We then detailed the construction of a computationally secure PIR in three approaches, beginning with the $O(n)$ communication complexity—a scenario similar to the naive solution. By structuring the database in matrix form, we managed to reduce the complexity to $O(\sqrt{n})$, where n is the database size, and further reduced it to polylogarithmic complexity using a recursion method. This construction guaranteed client-side security. Additionally, we introduced symmetric PIR scheme which ensures the server side security along with client security. We discussed three methods of creating a symmetric PIR scheme, thereby satisfying the security of both the client and the database. In the first method the server encrypts the answer of the query with public key cryptography in order to secure its answer. The second method utilized Oblivious Transfer OT^2, and the third employed an OPRF scheme.

In the subsequent stages, we deliberated on methods to decrease the system's overall complexity, going beyond mere communication complexity. We briefly mentioned Batch PIR, which, despite its success, proved impractical as it allows only non-adaptive queries from client. Consequently, we mentioned preprocessing and online/offline communications in PIR schemes, which have demonstrated increased system performance and are now commonly incorporated into new PIR schemes.

Finally, we discussed the SimplePIR and its state-of-the-art construction to date. We observed how plain-LWE and one-time hint(preprocessing) enabled simple calculations, facilitating the highest throughput values, despite its communication complexity still being far from the lower bound proposed in earlier works. We also noted that the SimplePIR assumes a static database, making it inapplicable for dynamic databases. This observation highlights a potential area for further research in practical PIR development—namely, PIR schemes applicable to dynamic databases, which are common in the real world.

Bibliography

[1] B.Chor, O.Goldreich, E.Kushilevitz,and M.Sudan, Private information retrieval. In *Proc. of the 36th Annu. IEEE Symp. on Foundations of Computer Science*, pages 41–51, 1995. Journal version: *J. of the ACM*,45:965–981,1998.

[2] B.Chor and N. Gilboa. Computationally private information retrieval. In *Proc. of the 29th ACM Symp. on the Theory of Computing* pages 304–313,1997.

[3] E.Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. of the 38th IEEE Symp. on Foundations of Computer Science*, pages 364–373,1997.

[4] Y.Gertner, Y.Ishai, E.Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *J. of Computer and System Sciences*, 60(3):592–629,2000. Conference version in *Proc. of the 30th ACM Symp. on the Theory of Computing*, pages 151–160, 1998.

[5] H.Lipmaa. An oblivious transfer protocol with log-squared communication. In J. Zhou and J. Lopez, editors, *the 8-th Information Security Conference (ISC'05)*, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328. Springer-Verlag, 2005.

[6] M.Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. of the 31st ACM Symp. on the Theory of Computing*, pages 245–254, 1999.

[7] Y.C. Chang. Single database private information retrieval with logarithmic communication. *In Information Security and Privacy: 9th Australasian Conference*, ACISP 2004, volume 3108 of *Lecture Notes in Computer Science*, pages 50–61. Springer-Verlag, 2004.

[8] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981. Available online in the Cryptology ePrint Archive, Report 2005/187.

[9] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, Vinod Vaikuntanathan. One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval. In textit USENIX Security Symposium, 2023

[10] C. Cachin, S.Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *J.Stern, editor, Advances in Cryptology–EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer-Verlag,1999.

[11] E. Kushilevitz and R. Ostrovsky. One-Way Trapdoor Permutations Are Sufficient for Non-Trivial Single-Server Private Information Retrieval. *Proc. 19th Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT '00)*, pp. 104-121, 2000.

[12] M.Naor and M. Yung. Universal one-way functions and their cryptographic applications. In *Proc. of the 21st Annu. ACM Symp. on the Theory of Computing*, pages 33–43, 1989.

[13] O.Goldreich and L.Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages25–32, Seattle, Washington, 15–17 May 1989.

[14] C.Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proc. of the 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815. Springer-Verlag,2005.

[15] C.Aguilar-Melchor and P. Gaborit. A Fast Private Information Retrieval Protocol, in *The 2008 IEEE International Symposium on Information Theory (ISIT'08)*,Toronto, Ontario, Canada, pp. 1848–1852, IEEE Computer Society Press, 2008.

[16] C. Aguilar-Melchor and P. Gaborit. A Lattice-Based Computationally-Efficient Private Information Retrieval Protocol. *Proc. Western European Workshop Research in Cryptology (WEWORC '07)*, 2007.

[17] F. Olumofin and I. Goldberg. Revisiting the Computational Practicality of Private Information Retrieval. *Technical Report CACR 2010-17*, Univ. of Waterloo, 2010.

[18] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography *Journal of the ACM*, 2009.

[19] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR: Private information retrieval for everyone. *PoPETs*,2016.

[20] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. PIR with compressed queries and amortized query processing. In *SandP*, 2018.

[21] Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. Communication–Computation trade-offs in PIR. In *USENIX Security*, 2021.

[22] Muhammad Haris Mughees, Hao Chen, and Ling Ren. OnionPIR: Response efficient single-server PIR. In *CCS*, 2021.

[23] Samir Jordan Menon and David J. Wu. Spiral: Fast, high-rate single-server PIR via FHE composition. In *S and P*, 2022.

[24] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *STOC*, 2009.

[25] W.Diffie and M.E. Hellman. *New Directions in Cryptography. IEEE Transactions on Information Theory*, IT-22 (Nov.1976), pages644–654.

[26] S.Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, Vol. 28, No.2, pages 270–299, 1984. Preliminary version in 14th STOC, 1982.

[27] M.Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo Random Bits. *SIAM Journal on Computing*, Vol. 13, pages 850–864, 1984. Preliminary version in 23rd FOCS, 1982.

[28] A.C . Yao. Theory and Application of Trapdoor Functions. In *23 rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[29] M.Bellare, S.Halevi, A.Sahai, S.Vadhan. Many-to-one Trapdoor Functions and their Relation to Public-key Cryptosystems. .*Advances in Cryptology–Crypto 98 Proceedings*, Lecture Notes in Computer Science Vol.1462, H.Krawczyked., Springer-Verlag,1998.

[30] J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby. Construction of pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[31] O.Goldreich, S.Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th ACM Symp.on the Theory of Computing*, pages218–229,1987.

[32] Tung Chou and Claudio Orlandi. The Simplest Protocol for Oblivious Transfer. In *Proceedings of the 4th International Conference on Progress in Cryptology* LATINCRYPT 2015 - Volume 9230 August

[33] Ronald L. Rivest. Unconditionally Secure Commitment and Oblivious Transfer Schemes Using Private Channels and a Trusted Initializer. Technicalreport ,M.I.T., 1999. theory. lcs. mit. edu/rivest/Rivest-commitment. pdf

[34] A.Beimel, Y.Ishai, and T. Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. *J. Cryptol.*, 17(2):125–151, 2004.

[35] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Batch codes and their applications. *STOC* 2004.

[36] A.Hamlin, R.Ostrovsky, M. Weiss, and D.Wichs. Private anonymous data access. *Cryptology ePrint Archive*, Report 2018/363, 2018.

[37] Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan. Single-Server Private Information Retrieval with Sublinear Amortized Time. *Advances in Cryptology – EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Norway, 2022.

[38] Amos Beimel, Yuval Ishai,and Tal Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. *J.Cryptol.*, 2004.

[39] Elette Boyle, Yuval Ishai, Rafael Pass,and Mary Wootters. Can we access a database both locally and privately.In *TCC*, 2017.

[40] Ran Canetti, Justin Holmgren, and Silas Richelson. Towards doubly efficient private information retrieval. In *TCC*, 2017.

[41] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In *EUROCRYPT*, 2020.

[42] Dmitry Kogan and Henry Corrigan-Gibbs. Private blocklist lookups with Checklist. In *USENIX Security*, 2021

[43] Elaine Shi, Waqar Aqeel, Balakrishnan Chandrasekaran, and Bruce Maggs. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In *CRYPTO*, 2021.

[44] Mingxun Zhou, Wei-Kai Lin, Yiannis Tselekounis, and Elaine Shi. Optimal single-server private information retrieval. Cryptology ePrintArchive, Paper 2022/609, 2022.

[45] A. Beimel, Y. Ishai, E. Kushilevitz, and T.Malkin. One-way functions are essential for single-server private information retrieval. In *Proc.of the 31th Annu. ACM Symp. on the Theory of Computing*, 1999.