M.Tech. (Computer Science) Dissertation Series

On Symmetrically Private Information Retrieval

a dissertation submitted in partial fulfilment of the requirements for the M. Tech. (Computer Science) degree of the Indian Statistical Institute

By

Sanjeev Kumar Mishra

under the supervision of

Dr. Palash Sarkar

INDIAN STATISTICAL INSTITUTE

203, Barrackpore Trunk Road

Calcutta-700035

Certificate of Approval

This is to certify that the thesis entitled *On Symmetrically Private Information Retrieval* submitted by *Sanjeev Kumar Mishra*, towards partial fulfillment of the requirement for ***M. Tech.*** in *Computer Science* degree of the *Indian Statistical Institute, Calcutta*, is an acceptable work for the award of the degree.

August 8, 2000 **(Supervisor)**

(External Examiner)

Acknowledgment

" ...very little do we have and inclose which we can call our own in the deep sense of the word. We all have to accept and learn, either from our predecessors or from our contemporaries. Even the greatest genius would not have achieved much if he had wished to extract everything from inside himself."

[Goethe, in conversation with Eckermann, 17.2.1832]

My sincerest gratitude goes to *Dr. Palash Sarkar* for his guidance, advice, enthusiasm and criticism throughout the course of this dissertation.

I would also like to take this opportunity to thank all those teachers who took our classes, for their excellent courses they offered which helped me in this course.

Finally I would like to thank all my classmates, without whose cooperation and support this work would not have been a success.

Sanjeev Kumar Mishra

Abstract

In today's age of information it is very important that, information about the information which you are seeking should not be leaked even to the server who is going to provide you the desired information. On the other hand, considering information as commodity, it is age old wisdom that one should get only as much as he pays.

In this paper we essentially consider this problem and provide suitable solutions. Under a new number theoretic assumption, XOR Assumption, we give single-round symmetrically private information retrieval (SPIR) scheme for bit retrieval with communication complexity $O(n^{\epsilon})$, for any $\epsilon > 0$, where n is the number of bits in the database. We also present a block retrieval SPIR scheme which is specially efficient when number of records is of the same order as number of bits in each records. Assuming the existence of a probabilistic encryption with certain associated properties, we present an $O(K \log n)$ SPIR scheme, where K is the security parameter. Then we go on to generalize SPIR scheme to private retrieval of secrets and sharing by a group of users. We also discover and prove certain properties related to quadratic residues in particular and probabilistic encryption system in general.

Keywords : *cryptography, Private Information retrieval, PIR, SPIR, OT, Oblivious Transfer, Cryptographic Protocols, Quadratic Residuosity Assumption, Probabilistic Encryption, Secret sharing Scheme, Private Computations.*

Contents

1 Introduction **4**

1.1 Related Work 5

1.1.1 Private Information retrieval 5

1.1.2 Symmetrically Private Information Retrieval 6

1.1.3 Instance Hiding 7

1.1.4 All-Or-Nothing Disclosure of Secrets 7

1.1.5 Oblivious Transfer 8

1.2 Our Work 8

1.3 Organization 10

2 Preliminaries and Definitions **11**

2.1 Cryptographic Concepts 11

2.1.1 Probabilistic Encryption 11

2.1.2 One-way Functions 12

2.1.3 Trapdoor One-way Permutations 13

2.1.4 Quadratic Residuosity Assumption 13

2.1.5 Trapdoor Predicates 15

2.1.6 The Decisional Diffie-Hellman Assumption 16

2.1.7 Secret Sharing Scheme 17

2.2 Definitions of PIR and SPIR and related schemes 18

2.2.1 Private Information Retrieval 18

2.2.2 Symmetrically Private Information Retrieval 19

2.2.3 Oblivious transfer 20

3 Multiserver PIR scheme : Implementations. **22**

3.1 Basic Two Server Scheme 23

3.2 Multi Server Scheme 24

3.3 Covering Codes Scheme 25

4 On Security of Probabilistic Encryption **27**

4.1 Results on Quadratic Residuosity 27

4.2 Security of Probabilistic Encryption 32

5 Symmetrically PIR Schemes **39**

5.1 Introduction 39

5.2 The Basic Scheme 39

5.2.1 The Scheme 39

5.2.2 Correctness 42

5.2.3 Privacy of database 42

5.2.4 Privacy of User 44

5.2.5 Communication complexity 45

5.3 Iterative Bit SPIR Scheme 45

5.3.1 The Scheme 45

5.3.2 Privacy of Database 48

5.3.3 Privacy of User 49

5.3.4 Communication Complexity 50

5.4 Block Retrieval SPIR Scheme 51

5.4.1 The Scheme 51

5.4.2 Privacy of Database 55

5.4.3 Privacy of User 56

5.4.4 Communication Complexity 56

5.5 $O(K \cdot \log n)$ PIR and SPIR protocols 57

5.5.1 The $O(K \cdot \log n)$ PIR Scheme 59

5.5.2 Correctness 60

5.5.3 Privacy of user 61

5.5.4 Communication complexity 62

5.5.5 An $O(K \cdot \log n + K^2)$ SPIR protocol from above PIR protocol 62

5.5.6 An $O(K \cdot \log n)$ SPIR protocol from above PIR protocol 63

6 Private Secret Retrieval and Sharing by Group **65**

6.1 The Scheme 65

7 Summary **69**

7.1 Applications 69

7.1.1 Pay-Per-Access Database with Private Queries 69

7.1.2 Sending Gift 69

7.1.3 Selling of Physical Objects Obliviously 69

7.1.4 Some More Applications 70

7.2 Open Problems 70

7.3 Conclusion 71

A Appendix **72**

Chapter 1

Introduction

In the age of Internet when accessing remote database is common, and the information in itself taking the place of most sought after and costliest commodity, it is very important to protect data i.e., information. While talking about information, it may be emphasized that the identity of information in which user is interested, can itself be an information of great interest. Consider the case, when an investor wants to know value of a certain stock, but reluctant to reveal the identity of that stock, because it will expose his future intentions with regard to that stock. The scheme or protocol which facilitate user to access database and receive desired information without exposing the identity of information was first introduced by Chor et al. [3] in 1995 under the name of *Private Information Retrieval*. While protecting the interest of user, it is equally important that user should not be allowed to learn more than one information at the cost of single information. Gertner et. al. [12] extended the privacy requirement so that the privacy of database is also protected. This extended protocol was called as Symmetrically Private Information Retrieval (SPIR) scheme. Specifically, they presented a general transformation of information theoretic PIR schemes into SPIR schemes, with a logarithmic overhead in communication complexity, but their general transformation is not applicable to single server scheme.

We purpose a new number theoretic assumption, *XOR Assumption*. Under the Quadratic Residuosity Assumption and our XOR Assumption, we give single-round symmetrically private information retrieval (SPIR) scheme for bit retrieval with communication complexity $O(n^{\epsilon})$, for any $\epsilon > 0$, where n is the number of bits in the database. We also present a block retrieval SPIR scheme which is specially efficient when number of records is of the same order as number of bits in each records. Assuming the existence of a probabilistic encryption with certain associated properties, we present an $O(K \log n)$ SPIR scheme, where K is the security parameter. Then we go on to generalize SPIR scheme to private retrieval of secrets and sharing by a group of users. We also discover and prove certain properties related to quadratic residues in particular and and probabilistic encryption system in general.

1.1 Related Work

1.1.1 Private Information retrieval

Private Information Retrieval (PIR) scheme allows a user to retrieve information from a database while maintaining the query private from the database managers. More formally, the database is modeled as a array of bits x held by server(s), and the user wants to retrieve the bit x_i for some i, without disclosing any information about i to the server(s). We denote number of bits (records) in database by n. The communication complexity of such a protocol is denoted by $C(n)$. A trivial PIR scheme consists of sending the entire database to the user, resulting in $C(n) = n$. This trivial solution provide information theoretic privacy for user. Any PIR scheme with $C(n) < n$ is called non trivial. The problem of constructing PIR protocol was originally introduced by Chor et. al. [3], it was also independently studied by Cooper and Birmen [6] in context of implementing an anonymous messaging service for mobile user. In particular, Chor et al. [3] proved that under the requirement of information theoretic security, and involvement of single database in the protocol, the only solution to PIR problem is to send the whole database i.e., trivial solution is the only solution. A possible way to overcome the above mentioned impossibility result is as suggested in [3], by considering $k > 1$ servers, each holding the database x. Chor et al. [3] constructed a two-server PIR scheme with communication complexity $O(n^{\frac{1}{3}})$. In general they presented a $k > 1$ server scheme with communication complexity $O(n^{\frac{1}{k}})$. Following the work in [3] Ambainis [7] in 1997 constructed a more efficient $k > 2$ server PIR scheme and lowered the communication complexity to $O(n^{\frac{1}{(2k-1)}})$ for k servers scheme. Chor and Gilboa presented the notion of computationally private information retrieval scheme in which the privacy of user is guaranteed only against the computationally bounded servers. Their scheme achieved much better communication complexity than known information theoretic PIR schemes, but have to rely on certain intractability assumptions. They provided a two server scheme with communication complexity $O(n^{\epsilon}$ for any $\epsilon > 0$. Kushilevitz and Ostrovsky [11] showed that under the notion of computational complexity one can achieve nontrivial PIR protocol even with a single server. In particular, [11] show that assuming that assuming the hardness of quadratic residuosity problem, they can get single database PIR protocol with communication complexity $O(n^{\epsilon})$ for any $\epsilon > 0$. Further constructions of single-database PIR schemes, improving either the communication complexity or the assumptions followed [22, 23, 20, 43]. In particular, Cachin et al. [20] constructed PIR with polylogarithmic communication complexity, under so called Φ-hiding (number-theoretic) assumptions. Eran Mann [22] generalized the construction of [11] to use any trapdoor predicate with certain homomorphic properties. Specifically, such predicates exist under the decisional Diffie-Hellman assumptions and under the assumption that it is hard to approximate the shortest vector in a lattice. PIR problem was also studied in [9, 21, 52, 53, 55]. Recently, [43] have shown a single database PIR based on any one-way trapdoor permutations, though their communication, while less than n, is bigger than Schemes based on specific number theoretic assumptions [20, 11, 22, 23]. On the other hand [21] have shown that any nontrivial single database PIR implies the existence of one-way functions. Chor et. Al. [3] also defines several extensions of the PIR problem:

Private Retrieval of Blocks or Records: The database is consists of n blocks of m bits each. The user wants to retrieve the i_{th} block of the database. A trivial solution would be to retrieve bit by bit. Chor et. al. [3] shows a significant saving is possible by taking advantage of the block structure. They shows that if we have a *k-server bit retrieval PIR scheme* in which user sends to the servers queries of $\alpha(n)$ bits, and the servers send replies of $\beta(n)$ bits, then we can construct from it a scheme for retrieving blocks of size m with query complexity $\alpha(n)$ bits and reply complexity of $m \cdot \beta(n)$ bits, resulting in a communication of $k(\alpha(n) + m \cdot \beta(n))$ bits, compared to the $m \cdot k(\alpha(n) + \beta(n))$ bits of communication in trivial solution.

t-Private Information Retrieval: In the multiserver scheme, protecting privacy of query becomes important against the coalition of more than one servers. A t-Private Information Retrieval Scheme is a PIR Scheme in which privacy of query (and therefore user) can be maintained even when there is a coalition of not more than t servers. Chor et. al. [3]shows a construction which achieves communication complexity $O(t.n^{\frac{1}{d}})$ with $k = t \cdot d$ servers. All the PIR scheme mentioned earlier assume that the user knows the physical address of the information that he is interested in. A more realistic scheme in which user retrieve information based on keywords was presented by Chor and Gilboa [10]. In their scheme, Private Information Retrieval by Keywords, Same database can support private and regular non-private searches. Ostrovsky and Shoup extended the PIR scope and presented the notion of Private Information Storage. Their scheme enabled user both to read and write into the database in a private manner. They achieve this with an addition of extra server and a polylogarithimic communication overhead compared to retrieval only schemes. They use an adapt techniques of oblivious RAM introduced by Goldreich and Ostrovsky [49, 50, 51], and inherit some of the properties of their construction. In particular, the protocol is multiround and data stored in servers is in encrypted form and not same in each server. Kushilevitz and Ostrovsky [11] noted that in the single-database mode, if the data is stored in its plain form then private writing is impossible since the database can easily observe the changes in x. Thus to achieve private writing data must be stored in encrypted form and requires a preprocessing initialization step. If there are several users it requires them to coordinate their acts so that each of them will have access to the encryption key and still this must be hidden from the database. If the database obtain this key pretending to be user privacy is lost.

1.1.2 Symmetrically Private Information Retrieval

A private information retrieval scheme is called a Symmetrically Private Information Retrieval (SPIR) scheme, if in that scheme privacy of database is also maintained, where by privacy of database we mean that user should not get more than one bit of information in a single invocation of protocol, no matter how he forms his query. Gertner et. al. [12] introduced the notion of database privacy in PIR protocols. They presented general transformations of PIR schemes in multi server setting satisfying certain properties into SPIR schemes, with logarithmic overhead in communication complexity. Kushilevitz and Ostrovsky [11] noted that in a setting of single server, their PIR protocol can be converted into a SPIR protocol with a communication complexity $O(n^{\epsilon})$

for any $\epsilon > 0$, employing zero knowledge techniques to verify the validity of query to protect the privacy of database. Note that zero knowledge techniques will need a multiround of communication.

1.1.3 Instance Hiding

In the instance hiding problem, a computationally bounded user holds an input i, and wishes to compute a known function f ($f : \{0,1\}^* \to \{0,1\}$) on input i. The function f may be hard to compute, so user can query k computationally unbounded oracles to achieve this task, oracles can compute $f(j)$ for any j. Still he wants to keep its input i hidden from the oracles. This problem was introduced in Rivest et al. [16] and Abadi et al. [13] and further studied in [14, 17, 18]. This problem can be viewed as if the oracles have a string $f(1)f(2)\dots f(n)$ and user wants to retrieve i_{th} bit of this string, which is the value $f(i)$, while keeping i private. Thus any PIR scheme (information theoretic) can be used as an instance hiding scheme: The oracles will simply use all the values of f on strings of length $|i|$ as the database x, where the x_i would be $f(i)$. The PIR scheme would allow the user to retrieve x_i, while hiding the value of i from the oracles. There are subtle differences between instance hiding problem and PIR problem. In instance hiding setting the user is limited to computations polynomial in $|i|$, while in the PIR setting, he is allowed to do computations polynomial time in n. Also in the instance hiding setting the user knows the function f and could use its structure to improve upon the efficiency, while in PIR setting no prior knowledge of database content can be assumed.

1.1.4 All-Or-Nothing Disclosure of Secrets

It involves two parties, a vendor and a buyer, and allows the vendor, who holds several secrets, to disclose one of them to the buyer, with the guarantee that no information about the other secrets will be gained. Furthermore the buyer can freely choose his secret and has the guarantee that the vendor will not be able to find out which secret he picked. The All-Or-Nothing Disclosure of Secrets (ANDOS) protocol was introduced in 1986 by Brassard, Crépeeau and Robert [24]. ANDOS is also known as One-out-of-t Strings Oblivious Transfer denoted $\binom{t}{1} - OT_2^K$ when t secrets of k bits are involved. First case was considered for $t = 2$ in [28]. Then in [29] scope was further restricted to $t = 2$ and $k = 2$, and finally in [24] it was studied in its full scope. Salomaa and Santean [38] have designed a very efficient ANDOS algorithm when several buyers are involved, assuming majority of them to be honest. Another efficient ANDOS protocol was proposed in 1994 in [40], but relies on ad hoc assumptions. The problem of blind decoding introduced by Sakurai and Yamane [41] Is similar to ANDOS protocol. In blind decoding scheme, the buyer is supposed to have an encrypted secret and has it decoded by vendor in such a way that the vendor does not gain any information either on the plaintext or on the buyer's private key. However, the buyer might be able to combine several secrets in a single decoding, or might try to organize an oracle attack to recover the secret key as suggested in [42]. Furthermore one has to assume that the buyer can anonymously recover the ciphertext for some specific secrets, which on the web it seems to need in itself an ANDOS protocol. Finally the ANDOS protocol is nearest to the SPIR protocols, where buyer is a user and vendor is a database. J.P.Stern [23] presented a very efficient ANDOS protocol using the ideas of recursion introduced in [11]. His protocol needs zero knowledge proof for establishing the validity of query sent by buyer, and thus needs a multiround of communication.

1.1.5 Oblivious Transfer

The Oblivious Transfer (OT) protocol was introduced by Rabin [36], one-out-of-two Oblivious Transfer, denoted $\binom{2}{1}$-OT, was introduced in [29], and Brassard et al. [24] introduced one-out-of-n Oblivious Transfer, denoted $\binom{n}{1}$-OT. All these OT variants were shown to be equivalent to one another [35, 24]. Informally, $\binom{n}{1}$-OT consists of two players, one holding n secrets and the other wants to retrieve one of these secrets. In the information theoretic setting, the SPIR (block retrieval) model is actually a distributed version of the $\binom{n}{1}$-OT problem. In the computational setting, any computational SPIR scheme will actually be a communication efficient implementation of $\binom{n}{1}$-OT.

In [37] it was shown that OT is complete, i.e., it can be used to construct any other protocol problem. Impagliazzo and Luby [56] has shown that OT implies the existence of one-way functions (OWF). Moreover Impagliazzo and Rudich [57] have shown that assuming OT is probably stronger than assuming existence of one-way functions. They show that it is impossible to construct a block box reduction from OT to OWF. Furthermore , proving any such black box construction is as hard as separating $\mathcal{P}$ from $\mathcal{NP}$. Thus [57] gives a strong evidence that OWF are currently not sufficient to construct OT and consequently SPIR, i.e., OT or SPIR is a strictly stronger assumption.

Recently, Cresenzo, Malkin, and Ostrovsky [45] proved that PIR cannot be based on weak computational assumptions. PIR is also a complete primitive, and any nontrivial implementation of single-server PIR may be used to construct any other secure protocol. They also suggested transformation of communication efficient single-server PIR to communication efficient $\binom{n}{1}$-OT, i.e., SPIR; but resulting SPIR will involve multiround of communication and efficiency depends on the efficiency of the PIR scheme. The most general assumption based on which it is currently known how to construct OT is that one-way trapdoor permutations exist [46]. Thus in a sense, the most general assumptions one can hope to use for constructing single-server private information retrieval protocol is the assumptions that one-way trapdoor permutations exist, indeed Kushilevitz and Ostrovsky [43] shows that this is feasible. Combining their [43] results with the results of Naor and Pinkas [47] and with the known implementations of OT based on one-way trapdoor permutations [46], they suggested $\binom{n}{1}$-OT, i.e., SPIR protocol based on any one-way trapdoor permutations whose communication complexity is strictly less than n.

1.2 Our Work

In view of the result, that every PIR scheme with a single-server require $\Omega(n)$ bits of communication, there is two way to get nontrivial PIR schemes. One-way is to increase the number of server and other is to lower the security requirement from information theoretic security to computational security.

In the multiserver arena, Chor et al. [3] have provided many good solutions. We have implemented three of these schemes. The basic scheme for two servers, which although use $(2 \cdot n + 1)$ bit of communication, provide base for other efficient PIR schemes. The multiserver scheme provide the ideas behind reduction in communication in presence of more than one servers holding the copy of database. Finally we implemented Covering Code Scheme, which for two servers, till date provide the best solution for information theoretic private information retrieval.

In [11] Kushilevitz and Ostrovsky suggested that there single server PIR scheme can be converted into SPIR scheme, by using the zero knowledge proof. In their suggested scheme first user has to verify the validity of his query. Here we like to add that zero-knowledge proof although add a communication overhead only polynomial in query size but need in multiround of communication. Also communication overhead even polynomially related to query size is too costly in practical uses. Recently, Crescenzo et al. [45] provided a general transformation of a single-server PIR scheme with communication complexity $C(n) < n$, where n is the length of the database, into a SPIR scheme with security parameter K and communication complexity $C(n) \cdot q(K)$ for some polynomial q, but even there scheme use multiround of communication and for practical purpose $q(K)$ times communication is too high.

We introduce a new assumption by the name XOR Assumption, in which informally, we claim that given XOR of two numbers from the set of QR's and QNR's in particular or in general from the set of encryptions of 0's and 1's in some probabilistic encryption scheme, it is not possible to guess anything about the membership of constituent numbers even in the presence of trapdoor information. We provide some evidences in support of our assumption, which include simulations and some theorems proved by us. In particular we prove that

If probability of any bit j in a number $x \in_R \mathcal{E}_0 \bigcup \mathcal{E}_1$ being 0 is not same for $x \in_R \mathcal{E}_0$ and $x \in_R \mathcal{E}_1$, then the encryption scheme can be polynomial time in the inverse of the difference of two probabilities. The only requirement is that sampling of numbers of some subset ($\mathcal{E}_0$ or $\mathcal{E}_1$) should be possible without knowing the private key.

This is possible in all probabilistic encryption scheme, in which trapdoor predicate satisfies certain homomorphic properties, which is even otherwise necessary for PIR scheme to work.

We first discuss a basic scheme which clear main ideas required for developing efficient protocol for SPIR problem. We prove

Under the QRA, and XOR Assumption, for every $\frac{1}{4} > \epsilon > 0$, there exists a nontrivial single-round computational SPIR scheme with communication complexity $O(n^{\frac{1}{2}+\epsilon})$.

Then we utilize the idea of recursion introduced in [11], and present our main protocol, single-round efficient SPIR protocol:

Under the QRA and XOR Assumption, for every $\epsilon > 0$, there exists a single-round computational protocol with communication complexity $O(n^{\epsilon})$.

In our scheme user do not has to prove validity of his query, he will simply not get any intelligible information if his queries are not valid.

Then we go on to block retrieval schemes, which are more important from the practical point of view, and we provide a single-round block retrieval computationally SPIR scheme with communication complexity $O(m \cdot K^{L+1})$, where m is the number of bits in each record, K is a security parameter, and L depends on m and n, the number of records in database ($L = \frac{\log n}{\log m}$).

In some settings, the number of records is not substantially bigger than the length of individual records. In these settings we get specially efficient block retrieval scheme:

For $(n < m)$ *we get* $C = O(m \cdot K^2)$ *and for* $(m < n \leq m^2)$ *we get* $C = O(m \cdot K^3)$

Compare it with the $m + \log n$ bits required even without requiring any privacy.

Further, with the assumption of the existence of a probabilistic encryption scheme in which logical AND and OR operations may be done on encrypted logical values (0 and 1) without decrypting them, we provide a single round SPIR protocol with optimal communication complexity. We present an $O(K \cdot \log n)$ *PIR protocol, and then transform it into an SPIR scheme with communication complexity* $O(K \cdot \log n + K^2)$, by introducing some extra computation for database before the computation of database in PIR scheme. We then provide an even more efficient SPIR scheme, in which database do some post computation on the number obtained by him in PIR computation.

Under the assumption of the existence of an encryption scheme in which logical AND *and* OR *operations may be done on encrypted logical values without decrypting them, there exists an computationally SPIR scheme with communication complexity* $O(K \cdot \log n)$*, for any security parameter* K*, and database* x *with number of bits* n *in it.*

Finally, we extend the scope of SPIR protocol, and present an scheme in which a group of user can interactively agree on some index and then retrieve the agreed secret from some database. In the scheme we suggests, privacy of users as well as database remain preserved. As well as if any particular user illegally try to modify query, except the one (selected as group leader, who actually send query to database), it will be trapped in the query generation process itself. If the group leader tries to do cheating then they fails to get any useful information from database. Further The retrieved information can be shared in the group using any secret sharing scheme, where the role of TA is played by database. We emphasize that the group retrieving the shares can even be different than the original group who generated the query.

1.3 Organization

In chapter 2 we first describe and define some cryptographic concepts and define PIR, SPIR and related schemes. In chapter 3 we present the implementational details of multiserver schemes. Chapter 4 deals with the results on number theoretic concepts. We first prove some results on quadratic residues, provide an algorithm, which prove the necessity of certain requirement, if quadratic residues are to be used for the purpose of probabilistic encryption. Then we prove some more theorem on security of probabilistic encryption. Chapter 5 is our main work. In this chapter we present various computationally SPIR protocol, prove their correctness, and privacy requirement and find the communication complexity. In chapter 6, we extend the scope of SPIR protocol, and present a new primitive under the name of Private Secret Retrieval and Sharing by Group. Finally, in chapter 7, we mention possible applications of protocols developed by us and discuss about some open problems.

Chapter 2

Preliminaries and Definitions

In this chapter we give some definitions and facts that are used in this report.

2.1 Cryptographic Concepts

In this section we include some facts from the number-theory and state several definitions. We start with the definition of probabilistic encryption, state certain assumptions based on which probabilistic encryption system can be built. Then we go on to the trapdoor predicates.

2.1.1 Probabilistic Encryption

Probabilistic encryption was introduced by Goldwasser and Micali [1], to deal with case when size of message text is known to be very small, specially the case when message is a single bit. The idea of probabilistic encryption is that no information about the plain text should be computable from the cipher text in polynomial time. We define probabilistic encryption formally as in [60]

DEFINITION 2.1.1. *A* **probabilistic public-key cryptosystem** *is defined to be a six tuple* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{R})$*, where* $\mathcal{P}$ *is the set of* **plaintext**, $\mathcal{C}$ *is the set of* **ciphertext**; $\mathcal{K}$ *is the* **keyspace**, *and for each* **key** $K \in \mathcal{K}$, $e_K \in \mathcal{E}$ *is a public* **encryption rule** *and* $d_K \in \mathcal{D}$ *is a secret* **decryption rule**. *The following properties should be satisfied:*

1. *Each* $e_K : \mathcal{K} \times \mathcal{R} \to \mathcal{C}$ *and* $d_K : \mathcal{C} \to \mathcal{P}$ *are functions such that*

   $$d_K(e_K(b, r)) = b$$

   for every plaintext $b \in \mathcal{P}$ *and every* $r \in \mathcal{R}$. *This condition implies that* $e_K(x, r) \neq E_K(x', r)$ *if* $x \neq x'$.

2. *Let* ϵ *be a specified* **security parameter**. *For any fixed* $K \in \mathcal{K}$ *and for any* $x \in \mathcal{P}$, *define a probability distribution* $p_{K,x}$ *on* $\mathcal{C}$, *where* $p_{K,x}(y)$ *denotes the probability that* y *is the ciphertext given that* K *is the key and* x *is the plaintext (this probability is computed over all* $r \in \mathcal{R}$*). Suppose* $x, x' \in \mathcal{P}$, $x \neq x'$, *and* $K \in \mathcal{K}$. *Then the probability distributions* $p_{K,x}$ *and* $p_{K,x'}$ *are not* ϵ*-distinguishable.*

Working of above system is as follows: to encrypt a plaintext x, choose a random $r \in \mathcal{R}$ and compute $y = e_K(x, r)$. Any such y can be decrypted to x with the knowledge of secret decryption information. Informally, Property 2 in definition says, an encryption of x looks like an encryption of x'. The security parameter should be small : in practice ϵ desired to be smaller than $c/|\mathcal{R}|$ for some small $c > 0$.

2.1.2 One-way Functions

Informally, a one-way function is a function which is easy to compute but hard to invert. Formal definition follows:

DEFINITION 2.1.2. *A function $f : \{0,1\}^* \to \{0,1\}^*$ is called* **one-way** *if the following two conditions hold*

1. easy to compute: *There exists a polynomial time algorithm, $\mathcal{A}$, which on input x outputs $f(x)$. i.e., $\mathcal{A}(x) = f(x)$.*

2. hard to invert: *For every probabilistic polynomial time algorithm, $\mathcal{A}'$, for every polynomial P, and for all sufficiently large K's*

$$Pr(\mathcal{A}'(f(U_K), 1^K) \in f^{-1}f(U_K)) < \frac{1}{P(K)}$$

For extending definition of one-way function to a more useful notion as the collection of one-way function, we first define a collection of function, which is also useful in describing trapdoor permutations and trapdoor predicates.

DEFINITION 2.1.3. collection of functions: *A* **collection of functions** *consists of an infinite set of indices, denoted mathcal$N$, a finite set D_n, for each $n \in \mathcal{N}$, and a function f_n defined over D_n.*

DEFINITION 2.1.4. collection of one-way functions: *A collection of functions, $\{f_n : D_n \to \{0,1\}^*\}_{n \in \mathcal{N}}$ is called* **one-way** *if there exists three probabilistic polynomial time algorithms, $\mathcal{A}_{\mathcal{I}}, \mathcal{A}_{\mathcal{D}}$ and $\mathcal{A}_{\mathcal{F}}$, so that the following two conditions hold*

1. easy to sample and compute: *The output distribution of algorithm $\mathcal{A}_{\mathcal{I}}$, on input 1^K, is a random variable assigned values in the set $\mathcal{N} \bigcap \{0,1\}^K$. The output distribution of algorithm $\mathcal{A}_{\mathcal{D}}$, on input $n \in \mathcal{N}$, is a random variable assigned values in D_n. On input $n \in \mathcal{N}$ and $x \in D_n$, algorithm $\mathcal{A}_{\mathcal{F}}$ always outputs $f_n(x)$.*

2. hard to invert: *For every probabilistic polynomial time algorithm, $\mathcal{A}'$, for every polynomial P, and for all sufficiently large K's*

$$Pr\left(\mathcal{A}'(f_{\mathcal{I}_K}(X_K), \mathcal{N}_K) \in f_{\mathcal{I}_K}^{-1} f_{\mathcal{I}_K}(X_K)\right) < \frac{1}{P(K)}$$

   where $\mathcal{I}_K$ is a random variable describing the output distribution of algorithm $\mathcal{A}_I$ on input 1^K, and X_K is a random variable describing the output of algorithm D on input $\mathcal{I}_K$

Examples of the collection of one-way functions are RSA, factoring, and Discrete Log Problem(DLP). For details refer [2].

2.1.3 Trapdoor One-way Permutations

Informally stating, these are the collection of one-way permutations, $\{\mathcal{F}_n\}$, with the property that $\mathcal{F}_n$ can be efficiently inverted given a extra input t_n for index n, called **trapdoor** for that index. The trapdoor t_n can not be efficiently computed from n, but one can efficiently generate corresponding pairs (n, t_n). See [2] for details.

DEFINITION 2.1.5. Collection of trapdoor permutations: *A collection of functions $\mathcal{F} = \{\mathcal{F}_K\}$ is called a collections of one-way trapdoor permutations if the following hold:*

1. *There exists a probabilistic polynomial time algorithm, $\mathcal{A}_{\mathcal{I}}$, that on input 1^K output a pair (f, f^{-1}) where f is a function in $\mathcal{F}_K$ and f^{-1} is associated trapdoor string.*

2. *Each function $f \in \mathcal{F}_K$ is a permutation over $\{0,1\}^K$ and there exists an algorithm, $\mathcal{A}_{\mathcal{F}}$, that on input $f \in \mathcal{F}$ and $x \in \{0,1\}^*$ computes the value of $f(x)$ in time polynomial in $|x|$.*

3. *Each f is easy to invert given its trapdoor f^{-1}. That is there exists an algorithm $\mathcal{A}_{\mathcal{F}}^{-1}$, that on input $y \in \{0,1\}^K$ and the string f^{-1} computes the unique value x such that $f(x) = y$ (i.e., $\mathcal{A}_{\mathcal{F}}^{-1}(f^{-1}, \mathcal{A}_{\mathcal{F}}(f, x)) = x$) in time polynomial in K.*

4. *It is hard to invert the functions in $\mathcal{F}$ without having the trapdoor. That is for every probabilistic polynomial time algorithm $\mathcal{A}'$, every integer c, and sufficiently large K*

$$Pr_{f \in \mathcal{A}_{\mathcal{F}}(1^K), y \in_R \{0,1\}^K}(\mathcal{A}'(f, y) = f^{-1}(y)) < \frac{1}{K^c}$$

   where $f \in \mathcal{A}_{\mathcal{F}}(1^K)$ denotes the choosing a function f according to the probability distribution induced by the generation algorithm $\mathcal{A}_{\mathcal{F}}$.

Collection of trapdoor permutation can easily be formed from RSA collection, or Rabin collection. For details refer to [2].

2.1.4 Quadratic Residuosity Assumption

In this paper we have used the intractability of the **Quadratic Residuosity** problem. This problem was first used in [1], and since then has found many cryptographic applications.

Quadratic Residuosity Predicate

Let N be a natural number. Define
$Z_N^* = \{x | 1 \leq x \leq N, gcd(N, x) = 1\}$.
A number x is a quadratic residue modulo N if there exists a number y such that $y^2 = x \pmod{N}$.
Formally, the quadratic residuosity predicate is defined as follows :

$$QR_N(x) = \begin{cases} 0 & \text{if there exists a } y \in Z_N^* \text{ s.t. } y^2 = x \pmod{N} \\ 1 & \text{otherwise} \end{cases}$$

A number x is said to be QR (quadratic residue) *mod* N if $QR_N(x) = 0$, and QNR (quadratic non-residue) if $QR_N(x) = 1$. The Quadratic Residuosity problem is considered to be "hardest" when N is a product of two distinct primes of equal length $K/2$. Define

$$H_K = \{N|N = p \cdot q where\ p\ and\ q\ are\ K/2\text{-}bit\ primes\}$$

Even for a $N \in H_K$, if factorization is know, $QR_N(x)$ can be computed in $O(K^3)$ time. The Jacobi symbol $(\frac{x}{N})$ can be computed in $O(poly(K))$ time without knowing the factorization, even for $N \in H_K$. If $(\frac{x}{N})$ is -1 then x is always QNR and if $(\frac{x}{N})$ is 1 then it may be a QR or QNR with equal probability. We define

$$Z_N^{+1} = \{x \in Z_N^* | (\frac{x}{N}) = 1\}$$

We say x is pQR (pseudo quadratic residue) if $x \in Z_N^{+1}$ and x is QNR.

Remark 2.1.1.

1. *Any one can choose a random QR without knowing the factorization of N. To choose a random QR, pick $r \in Z_N^*$ and compute r^2. When a random pQR is required, if prime factors p and q of N were chosen such that $p \equiv q \equiv 3 \pmod 4$, then choose y a random QR, and $(N - y)$ is a pQR.*

2. *$QR_N(\cdot)$ is an additive predicate. i.e., $QR_N(xy) = QR_N(x) \oplus QR_N(y)$. Thus, it can be used to calculate xor function privately.*

Now we state the Quadratic Residuosity Assumption (QRA), which informally says that there is no family of polynomial-size circuits for computing the predicate $QR_N(\cdot)$ with probability significantly better than "guessing".

Assumption 2.1.1 Quadratic Residuosity Assumption (QRA) *[11] :*
For every constant c, and every family of polynomial size circuits $C_K(\cdot, \cdot)$, there exists an integer K_0 such that for all $K > K_0$

$$Pr_{N \in_R H_K; x \in_R Z_N^{+1}}(C_K(N, x) = QR_N(x)) < \frac{1}{2} + \frac{1}{K^c}$$

where $N \in_R H_K$ means N was computed from two $K/2$-bit primes chosen uniformly at random; and $x \in_R Z_N^{+1}$ means drawing x uniformly at random from Z_N^{+1}.

In [1] it was shown that even when $poly(K)$ many QNR's are given, it is still as difficult to compute $QR_N(x)$ as without QNR's. Many cryptographic protocols, like [59, 31], were based upon this assumption.

2.1.5 Trapdoor Predicates

In this section, we define the notion of trapdoor predicates, which is in some sense generalization of quadratic residuosity predicate. Informally, a one-way predicate is a predicate B, such that given x it is hard to compute $B(x)$, but given a value $v \in \{0,1\}$ it is easy to find a value x such that $B(x) = v$. A trapdoor predicate is a one-way predicate, such that when generating an instance of the predicate, one can generate with it a trapdoor information t, such that computing $B(x)$ given x and t becomes easy. A trapdoor predicate could be thought of as a public-key cryptosystem, formally defined as follows:

DEFINITION 2.1.6. *A Collection of Trapdoor Predicates Let $\mathcal{N} \subseteq \{0,1\}^*$ be a set of indices, and for $N \in \mathcal{N}$ let D_N be a finite set. A collection of trapdoor predicates is a set $B = \{B_N : D_N \to \{0,1\}\}_{N \in \mathcal{N}}$ satisfying the following conditions:*

1. **key generation** *There exists a polynomial P and a probabilistic polynomial time algorithm $A_{\mathcal{K}}$ which on input 1^K output (N, t_N) where $N \in_R \mathcal{N} \bigcap \{0,1\}^K$ and $|t_N| < P(K)$, t_N is the trapdoor of N.*

2. **encryption** *Let $S_N \subseteq D_N$. Denote $S_N^v = x \in S_N : B_N(x) = v$. There exists a polynomial time probabilistic algorithm $A_{\mathcal{E}}$, which on input $N \in \mathcal{N}$ and $v \in \{0,1\}$ outputs $x \in_R S_N^v$.*

3. **decryption** *There exists a polynomial time algorithm $A_{\mathcal{D}}$ such that for every $x \in D_N$: $A_{\mathcal{D}}(N, t_N, x) = B_N(x)$.*

4. **security** *For every family of polynomial-size randomized circuits $\{C_j\}$ and for every polynomial Q, there exists K_0 such that for every $K > K_0$:*

$$Pr\left(C_K(N,x) = B_N(x) | N \in_R \mathcal{N} \bigcap \{0,1\}^K; v \in_R \{0,1\}; x \in_R S_N^v\right) \leq \frac{1}{2} + \frac{1}{Q(K)}$$

   where the probability is taken on the choice of N, v and x and the internal coin tosses of C_K.

Now we go on to define a collection of homomorphic trapdoor predicates. We need the homomorphic property to build SPIR schemes based on trapdoor predicates.

DEFINITION 2.1.7. *A collection of Homomorphic Trapdoor Predicates*
Let D_N, D_N^v be as in previous definitions. A collection of trapdoor-predicates B is said to be a collection of homomorphic trapdoor predicates if the following three conditions hold:

1. *For all $N \in \mathcal{N}$, D_N forms a group with respect to certain operator $\cdot$. Also $|D_N| \leq 2^{P(K)}$ for some polynomial P. Where $K = |N|$.*

2. *For all $N \in \mathcal{N}$, for all $x_1, x_2 \in D_N^0$ it holds that $B_N(x_1 \cdot x_2) = 0$.*

3. *For all $N \in \mathcal{N}$, for all $x_1 \in D_N^0$ and $x_2 \in D_N^1$ it holds that $B_N(x_1 \cdot x_2) = 1$.*

DEFINITION 2.1.8. *A collection of Additive Homomorphic Trapdoor Predicates*
A collection of homomorphic trapdoor predicate B, is called additive if for all $N \in \mathcal{N}$ and for all $x_1, x_2 \in D_N^1$ it holds that $B_N(x_1 \cdot x_2) = 0$.

DEFINITION 2.1.9. *A collection of multiplicative Homomorphic Trapdoor Predicates*
A collection of homomorphic trapdoor predicate B, is called multiplicative if for all $N \in \mathcal{N}$ and for all $x_1, x_2 \in D_N^1$ it holds that $B_N(x_1 \cdot x_2) = 1$.

DEFINITION 2.1.10. *A collection of f-Homomorphic Trapdoor Predicates*
Let D_N, D_N^v, S_N^v be as in previous definitions, and let $f : N \to N$ be a function . A collection of trapdoor-predicates B is said to be a collection of f-homomorphic trapdoor predicates if the following three conditions hold:

1. *For all $N \in \mathcal{N}$, D_N forms a group with respect to certain operator $\cdot$. Also $|D_N| \leq 2^{P(K)}$ for some polynomial P. Where $K = |N|$.*

2. *For all K, for all $N \in \mathcal{N} \bigcap \{0,1\}^K$, for all $m \leq f(K)$ and for all $x_1, \ldots, x_m \in S_N^0$ it holds that $B_N(\prod_{j=1}^m x_j) = 0$.*

3. *For all K, for all $N \in \mathcal{N} \bigcap \{0,1\}^K$, for all $m \leq f(K)$, for all $j_0 \in \{1, \ldots, m\}$ and for all $x_1, \ldots, x_m \in S_N$ such that:*

$$B_N(x_j) = \begin{cases} 0 & \textit{if } j \neq j_0 \\ 1 & \textit{if } j = j_0 \end{cases}$$

   it holds that $B_N(\prod_{j=1}^m x_j) = 1$.

REMARK 2.1.2. *Every collection of homomorphic trapdoor predicate is also a collection of f-homomorphic trapdoor predicates for every function f.*

We can similarly define f-Additive Trapdoor Predicates, and f-Multiplicative Trapdoor Predicate. It can easily be shown [22] that quadratic Residuosity predicate $QR_N(\cdot)$, is an additive trapdoor predicate. Here we describe a f-Multiplicative trapdoor predicate based on the Decisional Diffie-Hellman Assumption.

2.1.6 The Decisional Diffie-Hellman Assumption

Let p be a prime. Define $Z_p^* = \{x | 1 \leq x \leq p, gcd(p, x) = 1\}$. Let g be a primitive element of Z_p^*. Informally, Decisional Diffie-Hellman Assumption ($DDHA$) states that given $p, g, g^a \pmod p, g^b \pmod p, g^c \pmod p$ no family of polynomial size circuits can decide with a probability significantly better than "guessing", whether $c = a \cdot b \pmod p$. We can define a trapdoor predicate based on this assumption as follows:

$$DDH_{p,g,g^a}(g^b, g^c) = \begin{cases} 0 & \text{if } c = a \cdot b \pmod p \\ 1 & \text{otherwise} \end{cases}$$

f-Multiplicative Trapdoor Predicate based on $DDHA$ We define $S^0_{p,g,g^a} = \{(g^b, g^{a\cdot b}) | b \in Z^*_p\}$ and $S^1_{p,g,g^a} = \{(g^b, g^{a\cdot b+1}) | b \in Z^*_p\}$. One can generate elements both of these sets in time polynomial in K, we would get a collection of 2^{K-1}-Multiplicative Trapdoor Predicates (assuming p is drawn uniformly among the primes of length K), as follows:

1. Consider the operator $\times$, defined as follows: $(x_1, y_1) \times (x_2, y_2) = (x_1 \cdot x_2, y_1 \cdot y_2)$ where the operation $\cdot$ is multiplication modulo p. It can easily be shown that $D_{p,g,g^a} = \{(x, y) : x, y \in Z^*_p\}$ forms a group under the operator $\times$.
2. Let $x_1, x_2 \in S^0_{p,g,g^a}$, then $DDH_{p,g,g^a}(x_1 \times x_2) = 0$.
3. Let $x_1 \in S^0_{p,g,g^a}$ and $x_2 \in S^1_{p,g,g^a}$ then $DDH_{p,g,g^a}(x_1 \times x_2) = 1$.
4. Let $x_1, x_2 \in S^1_{p,g,g^a}$, then $DDH_{p,g,g^a}(x_1 \times x_2) = 1$.

Above properties are true, even if we consider m-many x's in place of two, provided $m \leq 2^{K-1}$.

The Decisional Diffie-Hellman assumption was used as a basis for several cryptographic constructions [25, 27, 26, 33, 34]. An additive homomorphic trapdoor predicate could be used to privately calculate **XOR** of bits in the database, while a multiplicative predicate could be used to privately calculate **OR/AND** of bits in the database.

2.1.7 Secret Sharing Scheme

A **generalized perfect secret sharing scheme realizing** the access structure Γ is a method of sharing a secret S among a set of m participants (denoted by G), in such a way that the following two properties are satisfied :
1. Any authorized subset of participants $B \subseteq G$, can privately reconstruct the secret, without revealing their shares.
2. If an unauthorized subset of participants $B \subseteq G$ pool their shares, then they can determine nothing about the secret.

Formal definition follows:

DEFINITION 2.1.11. *[60] Suppose Γ is an access structure and $\mathcal{F} = \bigcup_{K\in\mathcal{K}} \mathcal{F}_K$ is a set of distribution rules. Then $\mathcal{F}$ is a perfect secret sharing scheme realizing the access structure Γ provided that the following two properties are satisfied:*

1. *For any authorized subset of participants $B \subseteq \mathcal{P}$, there do not exist two distribution rules $f \in \mathcal{F}_K$ and $f' \in \mathcal{F}_{K'}$ with $K \neq K'$, such that $f|_B = f'|_B$ (that is, any distribution of shares to the participants in an authorized subset B determines the value of the key).*
2. *For any unauthorized subsets of participants $B \subseteq \mathcal{P}$ and for any distribution of share $f_B \in \mathcal{S}_B$*
$$Pr_{\mathcal{K}}(K|f_B) = Pr_{\mathcal{K}}(K).\ \textit{for every } K \in \mathcal{K}.$$
i.e., the conditional probability distribution on $\mathcal{K}$, given a distribution of shares f_B to an unauthorized subset B, is the same as the a priori probability distribution on $\mathcal{K}$. In other words, the distribution of shares to B provides no information as to the value of the key.

One can weaken the requirement of secrecy to make it computationally secret sharing scheme.

2.2 Definitions of PIR and SPIR and related schemes

In this section we give formal definition for PIR, SPIR and OT.

2.2.1 Private Information Retrieval

Informally stating, a PIR scheme should satisfy following requirements:

1. **Correctness:** In every invocation of the scheme the user retrieves the bit he is interested in.

2. **User Privacy:** In every invocation of the scheme the server does not gain any information about the index of the bit that the user retrieves.

PIR schemes can be classified in two group on the basis of privacy they provide. Following are the two notion of privacy stated informally:

1. **Information Theoretic User Privacy** Given the query to server or database, he can not gain any information about the index user is interested in, *regardless of his computational power.* This is also called Perfect Privacy, and require that query of the user should be independent of the index he tries to retrieve.

2. **Computational User Privacy** Here server is assumed to be computationally bounded, say a polynomial size circuit. Thus for computational privacy, the distribution of the queries the user sends to database when he is interested in i_{th} bit, and the distribution of the queries when he is interested in i'_{th} bit, should be computationally indistinguishable.

Further, we can generalize a PIR scheme from a bit retrieval scheme to a block retrieval scheme, in which user retrieves a block of l bit from the database in one invocation of protocol. There database is considered to be having n such blocks or records of l bits.

REMARK 2.2.1. *There are some multi-round PIR protocols in which user and database interact many times. But we are concerned only with single-round schemes, in which user sends all his query in the beginning, and receives reply from database only once.*

Now we state formal definitions of PIR schemes:

DEFINITION 2.2.1. **Information Theoretic PIR Scheme**
A $(k \geq 1)$-server information theoretic PIR scheme is a collection of three algorithms:

1. **Query Generation Algorithm** $\mathcal{A}_Q$ *It is a polynomial time algorithm that on input $(1^n, i, r)$, where $i \in \{1, \ldots, n\}$ and $r \in_R \{0,1\}^{P(n)}$ (for some polynomial P) is a random string, outputs a k-tuple of queries $(q_1, \ldots, q_k)$, where q_j is intended for DB_j. For each j, $q_j \in \{0,1\}^{\ell_q}$, where ℓ_q is polynomial size in n.*

2. *DB* **Computation Algorithm** $\mathcal{A}_{\mathcal{D}}$ *It is a polynomial time algorithm that on input* (x, q_j)*, where* $x \in \{0,1\}^n$ *and* q_j *is a query sent by user, output a reply* R_j.

3. **Reconstruction Algorithm** $\mathcal{A}_{\mathcal{R}}$ *It is a polynomial time algorithm that on input* $(1^n, i, r, R_1, \dots, R_k)$ *outputs a bit* b.

Satisfying the following conditions:

1. **Correctness:** *Let* $q_j = \mathcal{A}_{\mathcal{Q}}{}^j(1^n, i, r)$. *For every* n*, for every* $x \in \{0,1\}^n$*, for any* $i \in \{1, \dots, n\}$ *and* $r \in \{0,1\}^{P(n)}$*:*

$$\mathcal{A}_{\mathcal{R}}(1^n, i, r, \mathcal{A}_{\mathcal{D}}(x, \mathcal{A}_{\mathcal{Q}}{}^1(1^n, i, r)), \dots, \mathcal{A}_{\mathcal{D}}(x, \mathcal{A}_{\mathcal{Q}}{}^k(1^n, i, r))) = x_i$$

2. **User Privacy** *For every* $i, i' \in \{1, \dots, n\}$*,* $j \in \{1, \dots, k\}$ *and* $q \in \{0,1\}^{\ell_q}$*,*

$$Pr(\mathcal{A}_{\mathcal{Q}}{}^j(1^n, i, r) = q) = Pr(\mathcal{A}_{\mathcal{Q}}{}^j(1^n, i', r) = q)$$

where the probabilities are taken over uniformly chosen $r \in \{0,1\}^{P(n)}$.

Definition 2.2.2. **Computational PIR scheme**
A k*-server computational PIR scheme is a collection of three algorithms* $\mathcal{A}_{\mathcal{Q}}, \mathcal{A}_{\mathcal{D}}, \mathcal{A}_{\mathcal{R}}$ *as in the definition of information theoretic PIR schemes satisfying the same correctness condition, and the following user privacy condition: For every family of polynomial-size circuits* $\{C_j\}$ *and for every polynomial* Q*, there exists an* n_0 *such that for every* $n > n_0$ *for every* $i, i' \in 1, 2, \dots, n$

$$|Pr(C_n(\mathcal{A}_{\mathcal{Q}}{}^j(1^n, i, r)) = 1) - Pr(C_n(\mathcal{A}_{\mathcal{Q}}{}^j(1^n, i', r')) = 1)| < \frac{1}{Q(n)}$$

where $r, r' \in_R \{0,1\}^{P(n)}$ *for some polynomial* P*, and the probability is taken on the choice of* r, r' *and the internal coin-tosses of* C_n.

Definition 2.2.3. The Communication Complexity of a PIR Scheme *Communication complexity of a PIR scheme is the maximum (over all possible invocations) of the sum of the lengths of the queries and the lengths of the replies. Formally let* $(\mathcal{A}_{\mathcal{Q}}, \mathcal{A}_{\mathcal{D}}, \mathcal{A}_{\mathcal{R}})$ *be a (computational or information theoretic) PIR scheme for* k *servers, as defined earlier. The communication complexity of the scheme* $C_{PIR}(n)$ *is defined as:*

$$C_{PIR}(n) = \max_{x \in \{0,1\}^n, i \in \{1,2,\dots,n\}, r \in \{0,1\}^{P(n)}} \left(\sum_{j=1}^{k} \left(|\mathcal{A}_{\mathcal{Q}}{}^j(1^n, i, r)| + |\mathcal{A}_{\mathcal{D}}(x, \mathcal{A}_{\mathcal{Q}}{}^j(1^n, i, r))| \right) \right).$$

2.2.2 Symmetrically Private Information Retrieval

A symmetrically private information retrieval (SPIR) scheme is a PIR scheme satisfying an additional requirement, the privacy of data. We can again consider the privacy to be information theoretic or computational. As we are concerned with computational privacy, we here state only that. Informally, computational data privacy requires, for each execution, there exists an index i, such that the distributions on the user's view are computationally indistinguishable for any two databases x, y such that $x_i = y_i$. That is a computationally bounded user does not receive information about more than a single bit of the data, no matter how he forms his query of given length. Formal definition follows:

DEFINITION 2.2.4. **Computational Symmetrically Private Information Retrieval Scheme** *A computational symmetrically private information retrieval scheme is a PIR scheme satisfying the following additional condition on data privacy:*

3. **Data Privacy** *There exists an n_0 such that for every $n > n_0$, for any polynomial time algorithm $\mathcal{A}_\mathcal{R}$, for any $i' \in \{1, \ldots, n\}$, there exists an $i \in \{1, \ldots, n\}$, such that for any $x, y \in \{0,1\}^n$ where $x = x_1 x_2 \ldots x_n$ and $y = y_1 y_2 \ldots y_n$, $x_l, y_l \in \{0,1\}$ for $l = 1, \ldots, n$, and such that $x_i = y_i$, for every polynomial Q, it holds that*

$$\begin{aligned} &|Pr(\mathcal{A}_\mathcal{R}(1^n, i', r, \mathcal{A}_\mathcal{D}(x, q_1), \ldots, \mathcal{A}_\mathcal{D}(x, q_k)) = 1) \\ - \quad &Pr(\mathcal{A}_\mathcal{R}(1^n, i', r, \mathcal{A}_\mathcal{D}(y, q_1), \ldots, \mathcal{A}_\mathcal{D}(y, q_k)) = 1)| < \frac{1}{Q(n)} \end{aligned}$$

where $q_1, \ldots, q_k \in \{0,1\}^{\ell_q}$, are the query of prescribed length sent by user to different servers.

2.2.3 Oblivious transfer

A $\binom{n}{1}$-oblivious transfer ($\binom{n}{1}$-OT) is an interactive protocol between Alice, holding n bits (secrets) $b = b_1, b_2, \ldots, b_n$, and Bob, holding a selection index i. At the end of the protocol, Bob should obtain the bit b_i, but no information about any bit $b_j (j \neq i)$, whereas Alice should obtain no information about i. A formal definition follows:

DEFINITION 2.2.5. *($\binom{n}{1}$-Oblivious Transfer) Let (Alice,Bob) be an interactive protocol. We say that (Alice,Bob) is a* $\binom{n}{1}$**-Oblivious Transfer ($\binom{n}{1}$-OT)** *protocol with security parameter K if there exists a collection of three probabilistic polynomial time algorithms, $\mathcal{A}_\mathcal{B}$ and $\mathcal{A}_\mathcal{R}$ run by Bob, and $\mathcal{A}_\mathcal{R}$ run by Alice, and it holds that:*

1. Correctness: *For all $b \in \{0,1\}^n$, all $i (1 \leq i \leq n)$,*

$$\mathcal{A}_\mathcal{R}(1^K, i, r_B, \mathcal{A}_\mathcal{A}(b, \mathcal{A}_\mathcal{B}(1^K, i, r_B), r_A)) = b_i$$

where $r_A, r_B \in \{0,1\}^{P(K)}$ are random bit string chosen by Alice and Bob respectively. P is any polynomial.

2. Privacy against Alice. *For all probabilistic time algorithm $\mathcal{A}'$, all $b \in \{0,1\}^n$, all constant c, and all sufficiently large K,*

$$Pr(\mathcal{A}'(1^K, b, r_{A'}, \mathcal{A}_\mathcal{B}(1^K, i, r_B)) = i) \leq \frac{1}{n} + \frac{1}{K^c}$$

where $r_{A'}$ is a random bit string chosen by Alice.

3. Privacy against Bob *For all probabilistic polynomial time* $\mathcal{B}'$, *all* $i' \in \{1,\ldots,n\}$, *there exists a* $i \in \{1,\ldots,n\}$ *such that for any* $a, b \in \{0,1\}^n$ *where* $a = a_1a_2\ldots a_n$ *and* $b = b_1b_2\ldots b_n$, $a_l, b_l \in \{0,1\}$ *for all* $l \in \{1,2,\ldots,n\}$, *and such that* $x_i = y_i$, *for all constant* c, *and all sufficiently large* K,

$$|Pr(\mathcal{B}'(1^K, i', r_{B'}, \mathcal{A}_{\mathcal{A}}(a, q_B, r_A)) = 1) - Pr(\mathcal{B}'(1^K, i', r_{B'}, \mathcal{A}_{\mathcal{A}}(b, q_B, r_A)) = 1)| \leq \frac{1}{K^c}$$

where $r_{B'}$ *is a random bit string chosen by Bob, and* $q_B = \mathcal{A}'_{\mathcal{B}}(1^K, \cdot, \cdot)$, *for any polynomial time algorithm* $\mathcal{A}'_{\mathcal{B}}$, *such that for any polynomial time algorithm* $\mathcal{A}'_{Alice}$, *for all constant* c, *for all sufficiently large* K,

$$|Pr(q_B = \mathcal{A}_{\mathcal{B}}(1^K, \cdot, \cdot)) - Pr(q_B = \mathcal{A}_{\mathcal{B}}(1^K, \cdot, \cdot))| \leq \frac{1}{K^c}$$

Communication Complexity of SPIR scheme and of an $\binom{n}{1}$-OT scheme are defined analogous to the definition 2.2.3. of communication complexity of PIR scheme.

SPIR vs. $\binom{n}{1}$-OT . It can easily be verified that SPIR with a database of length n is equivalent to $\binom{n}{1}$-OT. We need two concepts, and defined them in two different, though equivalent, ways for following reasons. There were different motivations for using these primitives, and they were historically defined in this way. Specially, when constructing a SPIR protocol, the communication complexity is a crucial parameter, while it was not so for $\binom{n}{1}$-OT.

Chapter 3

Multiserver PIR scheme : Implementations.

Chor et. al. [3] proved that under the information theoretic considerations if there is only one copy of database available then in any PIR scheme n bits must be exchanged. It means, in a single server scheme trivial solution, i.e; downloading the whole database is the optimum solution.

THEOREM 3.0.1. *[3] Lower bound on communication in any single server private retrieval scheme under information theoretic consideration is n bits where n is the number of bits in the database.*

Proof : A communication C is possible for (x, i) if when the database content is x and the user is interested in i_{th} bit there is a positive probability for C to be communication. A communication C is said to be possible for i if it is possible for some pair (x, i). Now, fix a value i and assume towards a contradiction that the number of possible communications for i is smaller than 2^n. This means that This implies that there exist $x \neq y$ and C such that C is possible for both (x, i) and (y, i). If the sets of possible communications for (x, i) on the various x's are disjoint then their must be at least 2^n such communications. By the privacy requirement, for every $j \in [n]$, C must also be possible for (x, j), or else DB distinguishes the item identities i from j on database content x. Similarly, C is possible for (y, j), for every $j \in [n]$. Thus, in particular, C is possible for both (x, j) and (y, j), where j is an index for which $x_j \neq y_j$. This yields contradiction since on the same communication, C, the user must output the same bit, and so this output cannot equal both x_j and y_j. ■

In light of the above limit, there were two possible approaches to break the barrier of lower bound of n bits. One was to go for multiple server scheme, and other to lower the security requirement from information theoretic to computational security. Chor et. al. presented several multiserver scheme under information theoretic security. Three of these schemes are being presented here, and have been implemented by us. In following sections implementational details has been described, for the proof of security etc. [3] can be referred.

Notations :

DB_k : k_{th} server or database.

U : User.
x : database content held by DB_k's.
i : index of the bit user is interested in.
x_i : i_{th} bit of database.
brand() : random bit generator routine.
RandString(length) : random bit string generator of given length, uses routine brand().

3.1 Basic Two Server Scheme

- **Query Generation**
  Input : index i, length of database n.
  Output : Two n bit string, queries for two servers.
  procedure :
  $q_1 \leftarrow$ RandString(n); **comment:** q_1 stores a random bit string of length n.
  $q_2 \leftarrow$ invert i_th bit of q_1, by xoring with 1 at the i_{th} position.
  send(q_1 to DB_1 and q_2 to DB_2).

- DB **Computation**
  Input : query from user.
  Output : a single bit, the bit wise $\oplus$'s of all those bit of database, corresponding to which query string sent by user has entry 1.
  Procedure :
  $temp \leftarrow$ bit wise AND of querystring with database string.
  temp is same as database string x but 1 at all those positions masked which do not have 1 in the query received by DB.
  $r \leftarrow$ bit wise $\oplus$ of all bits of temp.
  return(r to user)

- **Reconstruction**
  Input : r_1, r_2; Anwers of DB_1 and DB_2.
  Output : x_i, the i_{th} bit of databse.
  Procedure :
  User on receiving answer from both server $\oplus$'s the answer bits. Hence obtained bit is the i_{th} bit of database, x_i.

Communication Complexity :

$$C = 2 \cdot n + 1.$$

3.2 Multi Server Scheme

Assume
$k = 2^d$: number of databases(DB_1,$DB_2, \ldots, DB_k$).
$n = l^d$: number of bits in the database.
A routine query(index, length) : which is the query generation routine of *Basic Scheme*, it takes index and length as input and return two bit strings of given length, having all the bits same except the one indicated by index.
Expand(base, number, width) : return an array of given width, containing expansion of number in the given base.

- **Query Generation**
  Input : index i, length l.
  Output : $2 \cdot d$ query strings of length l are generated and then properly sent to DBs.
  procedure :

$$\begin{cases} ind[] \leftarrow Expand(l, i, d); \textbf{for } j \leftarrow 0 \textbf{ to } (d-1) \\ \quad \textbf{do } \left\{ query[][j] \leftarrow query(ind[j], l); \textbf{comment:}\text{two } l \text{ strings differing at only } ind[j]_{th} \text{ position.} \right. \\ \textbf{return } (\text{all the } 2 \cdot d \ query[][]) \end{cases}$$

- **Distribution of query**
  Input : $2 \cdot d$ query strings generated in previous step.
  Output : queries are sent to all the k servers.
  Procedure :

$$\begin{cases} \textbf{for } \kappa \leftarrow 0 \textbf{ to } (k-1) \ \textbf{comment:} \text{ for each of the } k \text{ servers.} \\ \quad \textbf{do } \begin{cases} DBind[] \leftarrow Expand(2, i, d); \textbf{comment:}\text{Binary expansion of } DB \text{ index.} \\ \textbf{for } j \leftarrow 0 \textbf{ to } (d-1) \\ \quad \textbf{do } \begin{cases} qlet[j] \leftarrow query[DBind[j]][j]; \\ \textbf{comment: } \text{choose one of the two queries, query[0][j] and query[1][j].} \end{cases} \\ \textbf{send}(d \text{ qlets, } qlet[0], \ldots, qlet[d-1] \text{ to server } \kappa \ ) \end{cases} \end{cases}$$

- ***DB* Computation**
  Input : d qlets, qlet[] from user.
  Output : a single bit.
  Procedure :
  Generate complete query from d qlets.

$$\left\{\begin{array}{l} temp \leftarrow qlet[d-1];\ (\textbf{comment: } temp \text{ is a bit array}). \\ \textbf{for } u \leftarrow d-2 \textbf{ downto } 0 \\ \quad \textbf{do} \left\{\begin{array}{l} \textbf{for } v \leftarrow 0 \textbf{ to } (qlength-1) \\ \quad \textbf{do} \left\{\begin{array}{l} qbit \leftarrow ReadBit(temp); \\ \textbf{if } (qbit \neq 0) \\ \quad \textbf{then } append(qlet[v], query);\ (\textbf{comment: } \text{query is a bit array.}) \\ \quad \textbf{else } append(zeros, query);\ (\textbf{comment:}\text{zeros is a } l \text{ bit string of zeros.}) \end{array}\right. \\ qlength \leftarrow qlength \cdot l; \\ temp \leftarrow query; \end{array}\right. \\ \textbf{return } (query) \textbf{ comment } \text{query is the n bit binary string} \end{array}\right.$$

$temp \leftarrow$ bit wise AND of $query$ with database string x.
$temp$ is same as database string x but 1 at all those positions masked which do not have 1 in the $query$ string generated from $qlet[]'s$ received by DB.
$r \leftarrow$ bit wise $\oplus$ of all bits of $temp$.
return(r to user)

- **Reconstruction**
  Input : r_κ's, Anwers of DB_κ's , the k DB's.
  Output : x_i, the i_{th} bit of databse.
  Procedure :
  User on receiving answers from k DB's $\oplus$'s them. Hence obtained bit is the i_{th} bit of database, x_i.

Communication Complexity :

$$\begin{aligned} C &= k \cdot (l \cdot d + 1). \\ &= 2^d \cdot (d \cdot n^{\frac{1}{d}} + 1) \end{aligned}$$

3.3 Covering Codes Scheme

This algorithm is based on covering codes. In this scheme each server cover the job of some of the other server. And therefore Multiserver Scheme presented in above section can be simulated for 2^d DB's only by $k < 2^d$ DB's. We implemented the 2^3 Multiserver scheme with only 2 DB's. We name two DB as DB_{000} and DB_{111}. In the 2^3 multiserver scheme there were six other DB's. DB_{000} covers DB_{001},DB_{010},$DB100$ and DB_{111} covers DB_{110},DB_{101}, and DB_{011}.

- **Query Generation**
  Same as in above scheme with $d = 3$.

- **Distribution of query**
  send($query[0][0], query[0][1], query[0][2]$ to DB_{000})
  send($query[1][0], query[1][1], query[1][2]$ to DB_{111})

- DB **Computation**

  Let b be the binary variable taking value 0 and 1 only. DB_{bbb}, has received query[b][0],query[b][1],query[b][2] from user. So using the same algorithm as in Multiserver scheme he can form complete query for himself. Thus he do the computation and obtain a single bit as his part of answer. Observe that if one know the index j then from the partial query[b][δ], one can obtain query[$\bar{b}$][δ]. Now DB_{bbb} covers $DB_{bb\bar{b}}$ by taking $j = 0$ to $j = l - 1$ and for each value of j he calculate query[$\bar{b}$][2]. Thus he obtains l possible query for $DB_{bb\bar{b}}$ and compute l possible answers he return all these answers to user. Similarly DB_{bbb} simulate $DB_{b\bar{b}b}$ and $DB_{\bar{b}bb}$

- **Reconstruction**

  User on receiving answers from DB_{000} and DB_{111} can select right answer for the DB covered by them from l answers sent in lieu of the DB being covered as user knows the correct index. User $\oplus$ all the 8 bit thus obtained and hence obtained bit is the i_{th} bit of database, x_i.

Communication Complexity :

$$\begin{aligned} C &= 2 \cdot (l \cdot d + 3 \cdot l + 1). \\ &= 12 \cdot n^{\frac{1}{3}} + 2 \end{aligned}$$

COROLLARY 3.3.1. *There is a two server private information retrieval information theoretic secure scheme for n bits data, with communication complexity* $O(n^{\frac{1}{3}})$

Chapter 4

On Security of Probabilistic Encryption

4.1 Results on Quadratic Residuosity

Let p and q be two prime number and $N = pq$. We define four subsets of the set Z_N^* as follows :

$$\begin{array}{lcllcl} QR & = & \{y : (\frac{y}{p}) = (\frac{y}{q}) = 1, y \in Z_N^*\} & = & A_0 \\ pQR & = & \{y : (\frac{y}{p}) = (\frac{y}{q}) = -1, y \in Z_N^*\} & = & A_1 \\ pNQR & = & \{y : (\frac{y}{p}) = 1, (\frac{y}{q}) = -1, y \in Z_N^*\} & = & A_2 \\ qNQR & = & \{y : (\frac{y}{p}) = -1, (\frac{y}{q}) = 1, y \in Z_N^*\} & = & A_3 \end{array}$$

Here $(\frac{y}{N})$ denotes the Jacobi symbol.

REMARK 4.1.1. *The set A_1 consists of pseudo quadratic residues, i.e. for any $x \in A_1$ the Jacobi symbol $(\frac{y}{N}) = 1$, but x is a quadratic non residue modulo N.*

THEOREM 4.1.1. *Set QR forms a Abelian subgroup of the group Z_N^* with respect to the composition, multiplication modulo N.*

Proof : Z_N^* is a finite abelian group and hence to show A_0 is a subgroup it is enough to show the closure property. Let $x_1, x_2 \in QR$, then $(\frac{x_1}{p}) = (\frac{x_1}{q}) = (\frac{x_2}{p}) = (\frac{x_2}{q}) = 1$. Then $(\frac{x_1 \cdot x_2}{p}) = (\frac{x_1 \cdot x_2}{q}) = 1$. Thus $x_1 \cdot x_2 \in QR$.
From the closure property existence of inverse and identity can be proved. Commutativity follows from the closure property and the fact that Z_N^* is abelian group and associativety is inherited from the parent group. ■

THEOREM 4.1.2. *Cardinalities of four set defined above are equal, and these sets form a partition of Z_N^*.*

Proof : From the definition of subsets above it is clear that any element $y \in Z_N^*$ belongs to one and only one of the four subsets. Now take an element $y \in A_1$ and form the coset yA_0 of $A_0 = QR$.

$yA_0 = \{z : z = y \cdot x \pmod N, x \in A_0\}$. For any $z \in yA_0$, their exists $x \in A_0$ such that $z = y \cdot x$. Thus $(\frac{z}{p}) = (\frac{y \cdot x}{p}) = (\frac{y}{p})(\frac{x}{q}) = (-1)(1) = -1$ and similarly $(\frac{y \cdot x}{q}) = -1$. Therefore $z \in A_1$. Thus $yA_0 \subseteq A_1$ and leads to the result $|A_0| \leq |A_1|$.

Now consider the set $yA_1 = \{z : z = y \cdot x \pmod N, x \in A_1\}$ and $y \in A_1$. As $y \in Z_N^*$, thus for all $x_1, x_2, x_1 \neq x_2 \pmod N$, $y \cdot x_1 \neq y \cdot x_2 \pmod N$, thus $|yA_1| = |A_1|$. Now for any $z \in yA_1$, their exists $x \in A_1$ such that $z = y \cdot x$. Thus $(\frac{z}{p}) = (\frac{y \cdot x}{p}) = (\frac{y}{p})(\frac{x}{q}) = (-1)(-1) = 1$ and similarly $(\frac{y \cdot x}{q}) = 1$. Therefore, $z \in yA_1 \Rightarrow z \in QR$. Thus $yA_1 \subseteq A_0$, leading us to $|A_1| \leq |A_0|$. Therefore we arrive at the result $|A_0| = |A_1|$.

Similarly we can prove that $|A_0| = |A_2|$ and $|A_0| = |A_3|$. ■

REMARK 4.1.2. *The four sets defined above are the coset of subgroup QR in the group Z_N^*. These are the only possible coset, and form the coset decomposition of Z_N^*.*

Since $|Z_N^*| = \phi(p)\phi(q) = (p-1)(q-1)$, we have

COROLLARY 4.1.1. $|QR| = |pQR| = |pNQR| = |qNQR| = \frac{(p-1)(q-1)}{4}$

PROPOSITION 4.1.1. *If for any j, $1 \leq j \leq K$*

$$|Pr(x[j] = 0|x \in QR) - Pr(y[j] = 0|y \in pQR)| \geq \epsilon$$

then there exist a polynomial time (polynomial in $\frac{1}{\epsilon}$) algorithm that can decide the quadratic residuosity of a given number with success probability $1 - \delta$, where $\delta > 0$ can be made as small as desired.

Proof : Let $Pr(x[j] = 0|x \in QR) = p_0$ and $Pr(y[j] = 0|y \in pQR) = p_1$. Without loss of generality assume that $p_0 \leq p_1$. Let $p_1 - p_0 = 2\epsilon$

Now consider the following algorithm :

Algorithm 4.1.1: QUADRES($y \pmod N, j, p_{QR}, p_{pQR}$)

comment: Given a number $y \in A_0 \bigcup A_1$, this orcle decide its quadratic character. Return value "YES" implies QR and "NO" implies pQR

comment: estimate p_0 and p_1.
choose m QR's uniformly at random.
Count numbers having a 0 in j_{th} bit, say CNT.
$\hat{p_0} \leftarrow \frac{CNT}{m}$
Similarly calculate $\hat{p_1}$.

comment: Sample t numbers $x_1, x_2, \ldots, x_t$ uniformly at random from the set Z_N^* having same quadratic Character as y.
for $i \leftarrow 1$ **to** t
do
  $temp \leftarrow Rand() \pmod N$
  comment: Rand() chooses a number uniformly at random from Z_N^*
  $temp \leftarrow temp^2 \pmod N$
  $x_i \leftarrow y \cdot temp \pmod N$
comment: CNT counts the number of x_i's having a 0 in the j_{th} bit
$CNT \leftarrow 0$
for $i \leftarrow 1$ **to** t
do
  if j_{th} bit of x_i is 0
    then $CNT \leftarrow CNT + 1$;
$\epsilon \leftarrow \frac{1}{2} \cdot (\hat{p_1} - \hat{p_0})$
if $CNT \le (\hat{p_0} + \epsilon) \cdot t$
  then return ("YES")
  else return ("NO")

First we discuss on the estimation of probabilities p_0 and p_1. The proposed algorithm estimate p_0 by choosing m QR's uniformly at random. The estimation of p_0 from this sample of m QR's can be viewed as choosing m boxes uniformly at random from a set of $n = (p-1)(q-1)/4$ boxes where p and q are prime factor of N (Boxes are basically j_{th} bit of the QR's under consideration), and then finding the ratio of number of boxes among chosen set of m boxes containing 0. From the assumption about p_0, $n \cdot p_0$ boxes out of n boxes are containing 0 and rest are containing 1. The expectation of $\hat{p_0} \cdot m$ (estimate of p_0) follows the binomial distribution with mean $\mu = mp_0$ and variance $\sigma = \sqrt{mp_0q_0}$. As in our case m is going to be a sufficiently large number and p_0 is sufficiently away from 0 and 1, thus we can approximate expectation of $\hat{p_0}$ with normal distribution and then we have following result in Appendix A:

$$Pr(\mu - 3 \cdot \sigma \le \hat{p_0} \cdot m \le \mu + 3 \cdot \sigma) \ge 0.99$$

Thus with a probability 0.99, we can say that estimated value of p_0 lies between $p_0 - 3 \cdot \sigma/m$ and $p_0 + 3 \cdot \sigma/m$, i.e.,

$$\begin{aligned} |\hat{p_0} - p_0| &< \frac{3 \cdot \sqrt{m p_0 q_0}}{m} \\ &= \frac{3 \cdot \sqrt{p_0 q_0}}{\sqrt{m}} \end{aligned}$$

Thus by choosing m polynomial in $\frac{1}{\epsilon}$, we can estimate p_0, sufficiently good for this experiment. Similar argument validate the estimation of p_1. Now onward we use $p_0(p_1)$ to mean both $p_0(p_1)$ and its estimate $\hat{p_0}(\hat{p_1})$, assuming that the sample size for estimation was large enough (polynomial in $\frac{1}{\epsilon}$), so that error in estimation is negligible in comparison to ϵ. Now we calculate the probability P_S that above algorithm return "YES" given number y, a QR.

We denote by $p_0(i)$, the probability that i numbers out of a sample of t numbers chosen uniformly at random from the set QR, will have a 0 in j_{th} bit.

$$p_0(i) = p_0^i q_0^{t-i} \binom{t}{i}$$

where $q_0 = 1 - p_0$.
Thus $p_0(i)$ follows the binomial distribution.

Above algorithm QuadRes(x) return "YES" if $i \leq r = (p_0 + \epsilon)t$. Thus,

$$P_S = \sum_{i=0}^{r} p_0(i) = \sum_{i=0}^{r} p_0^i q_0^{t-i} \binom{t}{i}$$

We denote by $F(r)$ the probability distribution function for a binomial distribution $p(i)$.

$$F(r) = \sum_{i=0}^{r} p(i)$$

If t is sufficiently large and p and q are not very small, then we can approximate $p(i)$ with normal distribution. The parameters of resulting normal distribution are , mean $\mu = tp$ and variance $\sigma^2 = tpq$. We get,

$$\begin{aligned} F(r) &= \sum_{i=0}^{r} p(i) \\ &\simeq \sum_{i=0}^{r} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(i-\mu)^2}{2\sigma^2}} \\ &\simeq \int_0^r \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} dx \end{aligned}$$

Thus $F(\mu) = \frac{1}{2}$.
As t is sufficiently large and p_0 and q_0 are not very small, in this algorithm, so

$$\begin{aligned} P_S &= F(\mu_0) + F(r) - F(\mu_0) \\ &\simeq \frac{1}{2} + \frac{1}{\sqrt{2\pi\sigma_0^2}} \int_{\mu_0}^{r} e^{\frac{-(x-\mu_0)^2}{2\sigma_0^2}} dx \end{aligned}$$

where $\mu_0 = tp_0$ and $\sigma_0^2 = tp_0 q_0$. A substitution of independent variable x in integration leads to:

$$P_S \simeq \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \int_0^{(r-\mu_0)/\sigma_0} e^{\frac{-z^2}{2}} dz$$

where $z = (x - \mu_0)/\sigma_0$ is the substitution for x.

The function $\phi(z) = \frac{1}{\sqrt{2\pi}} e^{\frac{-z^2}{2}}$ is well studied in statistics and known as error function. We define a function in terms of error function: $G(\alpha) = \int_\alpha^\infty \phi(z)dz$. Then,

$$P_S \simeq \frac{1}{2} + \int_0^\infty \phi(z)dz - G(\epsilon t/\sigma_0)$$

where we have used the fact that $r = \epsilon t + \mu_0$. Also from Appendix A $\int_0^\infty \phi(z)dz = \frac{1}{2}$. Thus

$$P_S \simeq 1 - G(\epsilon t/\sigma_0)$$

Now we calculate the probability P_E, that the algorithm QuadRes() reply "YES" given a number y a pQR. We denote by $p_1(i)$, the probability that i numbers out of a sample of t numbers chosen uniformly at random from the set pQR, will have 0 in j_{th} bit.

$$p_1(i) = p_1^i q_1^{t-i} \binom{t}{i}$$

where $q_1 = 1 - p_1$.
Above algorithm QuadRes() return "YES" if $i \leq r = (p_0 + \epsilon)t = (p_1 - \epsilon)t$. Thus,

$$P_E = \sum_{i=0}^{r} p_1(i) = \sum_{i=0}^{r} p_1^i q_1^{t-i} \binom{t}{i}$$

Again approximating the binomial distribution with normal distribution assuming large t and not so small p_1 and q_1, we get normal distribution parameters $\mu_1 = tp_1$ and $\sigma_1^2 = tp_1 q_1$

$$\begin{aligned}
P_E &= F(\mu_1 - \epsilon t) \\
&\simeq \int_0^{\mu_1 - \epsilon t} \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{\frac{-(x-\mu_1)^2}{2\sigma_1^2}} dx \\
&= \frac{1}{\sqrt{2\pi\sigma_1^2}} \int_0^{\mu_1} e^{\frac{-(x-\mu_1)^2}{2\sigma_1^2}} dx - \frac{1}{\sqrt{2\pi\sigma_1^2}} \int_{\mu_1 - \epsilon t}^{\mu_1} e^{\frac{-(x-\mu_1)^2}{2\sigma_1^2}} dx \\
&= \frac{1}{2} - \frac{1}{\sqrt{2\pi\sigma_1^2}} \int_{\mu_1 - \epsilon t}^{\mu_1} e^{\frac{-(x-\mu_1)^2}{2\sigma_1^2}} dx \\
&= \frac{1}{2} - \frac{1}{\sqrt{2\pi\sigma_1^2}} \int_{-t\epsilon}^{0} e^{\frac{-x^2}{2\sigma_1^2}} dx \\
&= \frac{1}{2} - \frac{1}{\sqrt{2\pi\sigma_1^2}} \int_0^{\epsilon t} e^{\frac{-x^2}{2\sigma_1^2}} dx \\
&= G(\epsilon t/\sigma_1)
\end{aligned}$$

where the function $G(\cdot)$ is defined as above. We have following result on function $G(\alpha)$ in Appendix A:

1. For $\alpha = 1.0$, $G(\alpha) < 0.16$.

2. For $\alpha = 4.0$, $G(\alpha) < 10^{-4}$.

3. For $\alpha > 1$, $G(\alpha) < \sqrt{\frac{1}{2\pi}} e^{-\alpha/2}$.

Above result on function $G(\alpha)$ implies following result for P_S and P_E:

1. For $t \geq \max\{(p_0 q_0/\epsilon^2), (p_1 q_1/\epsilon^2)\}$, $P_S > 0.84$ and $P_E < 0.16$.

2. For $t \geq \max\{(16 p_0 q_0/\epsilon^2), (16 p_1 q_1/\epsilon^2)\}$, $P_S > 0.9999$ and $P_E < 0.0001$.

3. For $t \geq \max\{(p_0 q_0/\epsilon^2), (p_1 q_1/\epsilon^2)\}$,

$$\begin{aligned} P_S &> 1 - \sqrt{\frac{1}{2\pi}}\, e^{-\epsilon\sqrt{t}/2p_0q_0} \\ P_E &< \sqrt{\frac{1}{2\pi}}\, e^{-\epsilon\sqrt{t}/2p_1q_1} \end{aligned}$$

Thus if $\epsilon > \frac{1}{K^c}$ for some integer c, then above algorithm can decide the quadratic residuosity with success probability $1 - \delta$ in time polynomial in K, where $\delta = \sqrt{\frac{1}{2\pi}}\, e^{-\epsilon\sqrt{t}/2p_0q_0}$. We can choose t polynomial in $\frac{1}{\epsilon}$ and make δ smaller than any given value larger than zero. Thus without knowing the factorization quadratic nature of a random number can be determined with a probability not significantly away from one. ■

REMARK 4.1.3. *From simulation for small values of N it is found that for any bit j, $Pr(y[j] = 0 | y \text{ is a } QR)$,$Pr(y[j] = 0 | y \text{ is } pQR)$, $Pr(y[j] = 0 | y \in Z_N^*)$ is equal to each other within reasonable error limits.*

REMARK 4.1.4. *The algorithm described above implies that, if probability of any bit j in a number $x \in_R \mathcal{E}_0 \bigcup \mathcal{E}_1$ being 0 is not same for $x \in_R \mathcal{E}_0$ and $x \in_R \mathcal{E}_1$, then the encryption scheme can be broken in polynomial time. The only requirement is that sampling of numbers of same subest to which given number belong should be possible without knowing the key. It is possible in almost all currently known algorithm including Goldwasser and Micalis and Diffie-Hellman's scheme.*

4.2 Security of Probabilistic Encryption

In this section, we consider set of plaintext $\mathcal{P} = \{0,1\}$, set of ciphertext $\mathcal{C} = \{0,1\}^K$, and $\mathcal{R} = \{0,1\}^r$ for the probabilistic encryption cryptosystem 2.1.1.. The encryption rule $\mathcal{E}$ can then be defined as $\mathcal{E} : \{0,1\} \times \{0,1\}^r \to \{0,1\}^K$. Thus $\mathcal{E}(b, x_r) = y$, b is the bit to be encrypted, and x is a random r-bit string. We denote $\mathcal{E}(b, x_r)$ by $\mathcal{E}_r(b)$. Probabilistic encryption scheme $\mathcal{E}_r$, should satisfy $\mathcal{E}_r(0) \bigcap \mathcal{E}_r(1) = \emptyset$ for unique decryption. Let us denote $\mathcal{E}_r(0)$ by $\mathcal{E}_0$ and $\mathcal{E}_r(1)$ by $\mathcal{E}_1$. By the notation $x \in_R \mathcal{E}$ we mean that x has been chosen from $\mathcal{E}$ uniformly at random, and by $x \oplus y$ we mean bit wise xor of two numbers x and y.

THEOREM 4.2.1. **Main theorem**
Consider a probabilistic encryption scheme $\mathcal{E}_r$, $x, y \in_R \mathcal{E}_0 \bigcup \mathcal{E}_1$ *and* $z = x \oplus y$.
If for all j, $1 \leq j \leq K$, *where* K *is the number of bits in the encrypted number in said scheme,* $Pr(x[j] = 0 | x \in \mathcal{E}_0) = Pr(x[j] = 0 | x \in \mathcal{E}_1) = \delta_j$. *Then,* $Pr(z[j] = b | x \in \mathcal{E}_0, y \in \mathcal{E}_0) = Pr(z[j] = b)$ *and where* $b \in \{0, 1\}$. *More importantly,*

$$\begin{aligned}
Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_0 | z[j] = 0) &= Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_0 | z[j] = 1) = Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_0) \\
Pr(x \in \mathcal{E}_1, y \in \mathcal{E}_0 | z[j] = 0) &= Pr(x \in \mathcal{E}_1, y \in \mathcal{E}_0 | z[j] = 1) = Pr(x \in \mathcal{E}_1, y \in \mathcal{E}_0) \\
Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_1 | z[j] = 0) &= Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_1 | z[j] = 1) = Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_1) \\
Pr(x \in \mathcal{E}_1, y \in \mathcal{E}_1 | z[j] = 0) &= Pr(x \in \mathcal{E}_1, y \in \mathcal{E}_1 | z[j] = 1) = Pr(x \in \mathcal{E}_1, y \in \mathcal{E}_1)
\end{aligned}$$

Proof : As $x \in_R (\mathcal{E}_0 \bigcup \mathcal{E}_1)$, we have $Pr(x \in \mathcal{E}_0) + Pr(x \in \mathcal{E}_1) = 1$, we get

$$\begin{aligned}
Pr(x[j] = 0) &= Pr(x[j] = 0 | x \in (\mathcal{E}_0 \bigcup \mathcal{E}_1)) \\
&= Pr(x[j] = 0 | x \in \mathcal{E}_0) \cdot Pr(x \in \mathcal{E}_0) + Pr(x[j] = 0 | x \in \mathcal{E}_1) \cdot Pr(x \in \mathcal{E}_1) \\
&= \delta_j (Pr(x \in \mathcal{E}_0) + Pr(x \in \mathcal{E}_1)) \\
&= Pr(x[j] = 0 | x \in \mathcal{E}_0) \qquad (4.2.1)
\end{aligned}$$

Now using equation 4.2.1,

$$\begin{aligned}
&Pr(z[j] = 0 | x \in \mathcal{E}_0, y \in \mathcal{E}_0) \\
&= Pr(x[j] \oplus y[j] = 0 | x \in \mathcal{E}_0, y \in \mathcal{E}_0) \\
&= Pr(y[j] = 0 | y \in \mathcal{E}_0) \cdot Pr(x[j] = 0 | x \in \mathcal{E}_0) + Pr(y[j] = 1 | y \in \mathcal{E}_0) \cdot Pr(x[j] = 1 | x \in \mathcal{E}_0) \\
&= Pr(y[j] = 0) \cdot Pr(x[j] = 0) + Pr(y[j] = 1) \cdot Pr(x[j] = 1) \\
&= Pr(z[j] = 0)
\end{aligned}$$

Here we have proved first claim in the theorem.

Now using Bayes' theorem on conditional probability,

$$\begin{aligned}
Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_0 | z[j] = 0) &= \frac{Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_0) \cdot Pr(z[j] = 0 | x \in \mathcal{E}_0, y \in \mathcal{E}_0)}{Pr(z[j] = 0)} \\
&= Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_0)
\end{aligned}$$

Similarly, other three results also follows. ■

REMARK 4.2.1. *If probability of any bit* j *in a number* $x \in_R \mathcal{E}_0 \bigcup \mathcal{E}_1$ *being* 0 *is same for* $x \in_R \mathcal{E}_0$ *and* $x \in_R \mathcal{E}_1$, *then from the bit wise* xor *of two numbers* $x, y \in_R \mathcal{E}_0 \bigcup \mathcal{E}_1$, *one can not get any information about the membership of* x *and* y.

THEOREM 4.2.2. **Generalisation of Main theorem**
Consider a probabilistic encryption scheme $\mathcal{E}_r$, $x, y \in_R \mathcal{E}_0 \bigcup \mathcal{E}_1$ *and* $z = x \oplus y$.
If for all j, $1 \leq j \leq K$, *where* K *is the number of bits in the encrypted number in said scheme,*

1. $Pr(x[j]=0|x\in\mathcal{E}_0) = Pr(x[j]=0|x\in\mathcal{E}_1) = \delta_j$.

2. $Pr(z[j_m]=c_m|x\in\mathcal{E}_0, z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_{m-1}]=c_{m-1})$
$= Pr(z[j_m]=c_m|z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_{m-1}]=c_{m-1})$

where $m \le K$ *and* $j_1, j_2, \ldots, j_m$ *are distinct numbers from* $\{1,2,\ldots,K\}$ *and* $c_1, c_2, \ldots, c_m \in \{0,1\}$
Then,

$$\begin{aligned} Pr(x\in\mathcal{E}_0|z) &= Pr(x\in\mathcal{E}_0) \\ Pr(x\in\mathcal{E}_1|z) &= Pr(x\in\mathcal{E}_1) \end{aligned}$$

same is true for y.

Proof : Applying Bayes' theorem on conditional probability,

$$\begin{aligned} & Pr(x\in\mathcal{E}_0|z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_m]=c_m) \\ &= \frac{Pr(z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_m]=c_m|x\in\mathcal{E}_0)\cdot Pr(x\in\mathcal{E}_0)}{Pr(z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_m]=c_m)} \\ &= \frac{Pr(z[j_1]=c_1|x\in\mathcal{E}_0)\cdot Pr(z[j_2]=c_2,\ldots,z[j_m]=c_m|x\in\mathcal{E}_0, z[j_1]=c_1)\cdot Pr(x\in\mathcal{E}_0)}{Pr(z[j_1]=c_1)\cdot Pr(z[j_2]=c_2,\ldots,z[j_m]=c_m|z[j_1]=c_1)} \end{aligned}$$

using the result of main theorem, namely $Pr(z[j_1]=c_1|x\in\mathcal{E}_0) = Pr(z[j_1]=c_1)$, we obtain,

$$\begin{aligned} & Pr(x\in\mathcal{E}_0|z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_m]=c_m) \\ &= \frac{Pr(z[j_2]=c_2,\ldots,z[j_m]=c_m|x\in\mathcal{E}_0, z[j_1]=c_1)\cdot Pr(x\in\mathcal{E}_0)}{Pr(z[j_2]=c_2,\ldots,z[j_m]=c_m|z[j_1]=c_1)} \end{aligned}$$

Repeated application of above steps leads to,

$$\begin{aligned} & Pr(x\in\mathcal{E}_0|z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_m]=c_m) \\ &= \frac{Pr(z[j_m]=c_m|x\in\mathcal{E}_0, z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_{m-1}]=c_{m-1})\cdot Pr(x\in\mathcal{E}_0)}{Pr(z[j_m]=c_m|z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_{m-1}]=c_{m-1})} \end{aligned}$$

But, from the assumption,

$$\begin{aligned} & Pr(z[j_m]=c_m|x\in\mathcal{E}_0, z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_{m-1}]=c_{m-1}) \\ &= Pr(z[j_m]=c_m|z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_{m-1}]=c_{m-1}) \end{aligned}$$

Thus

$$Pr(x\in\mathcal{E}_0|z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_m]=c_m) = Pr(x\in\mathcal{E}_0)$$

Taking $m=K$ in above result we get

$$Pr(x\in\mathcal{E}_0|z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_K]=c_K) = Pr(x\in\mathcal{E}_0)$$

But, $j_1, j_2, \ldots, j_m$ are distinct numbers from $\{1, 2, \ldots, K\}$. Hence

$$\begin{aligned}
& Pr(x \in \mathcal{E}_0 | z[j_1] = c_1, z[j_2] = c_2, \ldots, z[j_K] = c_K) \\
= \; & Pr(x \in \mathcal{E}_0 | z[K] = b_K, z[K-1] = b_{K-1}, \ldots, z[1] = b_1) \\
= \; & Pr(x \in \mathcal{E}_0 | z[K]z[K-1] \ldots z[1] = b_K b_{K-1} \ldots b_1) \\
= \; & Pr(x \in \mathcal{E}_0 | z)
\end{aligned}$$

where $b_l = c_m$ such that $l = j_m$. Therefore,

$$Pr(x \in \mathcal{E}_0 | z) = Pr(x \in \mathcal{E}_0)$$

Similarly by considering $x \in \mathcal{E}_1$ in above derivation we get

$$Pr(x \in \mathcal{E}_1 | z) = Pr(x \in \mathcal{E}_1)$$

■

THEOREM 4.2.3. *Converse of Main Theorem*
If for all j $(1 \leq j \leq K)$ $Pr(x \in \mathcal{E}_0 | z[j] = 0) = Pr(x \in \mathcal{E}_0)$, $Pr(x \in \mathcal{E}_1 | z[j] = 0) = Pr(x \in \mathcal{E}_1)$, and $Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_0 | z[j] = 0) = Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_0)$, then for any $j \in \{1, 2, \ldots, K\}$,

$$Pr(x[j] = 0 | x \in \mathcal{E}_0) = Pr(x[j] = 0 | x \in \mathcal{E}_1) = Pr(x[j] = 0).$$

where $x, y, z, \mathcal{E}_0, \mathcal{E}_1$ are same as in Theorem 4.2.1.

Proof : Using Bayes' theorem on conditional probability,

$$Pr(x \in \mathcal{E}_0 | z[j] = 0) = \frac{Pr(x \in \mathcal{E}_0) \cdot Pr(z[j] = 0 | x \in \mathcal{E}_0)}{Pr(z[j] = 0)}$$

and the given relation $Pr(x \in \mathcal{E}_0 | z[j] = 0) = Pr(x \in \mathcal{E}_0)$ we obtain,

$$\begin{aligned}
Pr(z[j] = 0 | x \in \mathcal{E}_0) &= Pr(z[j] = 0) \\
&= Pr(x[j] \oplus y[j] = 0) \\
&= Pr(x[j] = 0) Pr(y[j] = 0) + Pr(x[j] = 1) Pr(y[j] = 1) \qquad (4.2.2)
\end{aligned}$$

But,

$$\begin{aligned}
Pr(z[j] = 0 | x \in \mathcal{E}_0) &= Pr(z[j] = 0 | x \in \mathcal{E}_0) \\
&= Pr(x[j] \oplus y[j] = 0 | x \in \mathcal{E}_0) \\
&= Pr(x[j] = 0 | x \in \mathcal{E}_0) Pr(y[j] = 0 | x \in \mathcal{E}_0) + Pr(x[j] = 1 | x \in \mathcal{E}_0) \cdot \\
&\quad\; Pr(y[j] = 1 | x \in \mathcal{E}_0) \\
&= Pr(x[j] = 0 | x \in \mathcal{E}_0) Pr(y[j] = 0) + Pr(x[j] = 1 | x \in \mathcal{E}_0) \cdot \\
&\quad\; Pr(y[j] = 1) \qquad (4.2.3)
\end{aligned}$$

Let $Pr(y[j]=0)=p$, $Pr(x[j]=0|x\in\mathcal{E}_0)=p_0$ and $Pr(x[j]=0|x\in\mathcal{E}_1)=p_1$. Thus from equations 4.2.2 and 4.2.3 we get

$$p^2+(1-p)^2=p_0p+(1-p_0)(1-p)$$

Similarly, by considering the given relation $Pr(x\in\mathcal{E}_1|z[j]=0)=Pr(x\in\mathcal{E}_1)$ we obtain

$$p^2+(1-p)^2=p_1p+(1-p_1)(1-p)$$

Solving them we get,
$p_0=p$ or $p=\frac{1}{2}$, and $p_1=p$ or $p=\frac{1}{2}$. Also if $p=\frac{1}{2}$ then $p_0+p_1=1$. Thus we get two case: either $p_0=p_1=p$ or $p=\frac{1}{2}$ and $p_0=1-p_1$. Now we use the third information i.e., $Pr(x\in\mathcal{E}_0, y\in\mathcal{E}_0|z[j]=0)=Pr(x\in\mathcal{E}_0, y\in\mathcal{E}_0)$, This on applying Bayes' Theorem gives: $Pr(z[j]=0|x\in\mathcal{E}_0, y\in\mathcal{E}_0)=Pr(z[j]=0)$ from which we derive the equation: $p_0^2+(1-p_0)^2=p^2+(1-p)^2$, thus for $p=\frac{1}{2}$ we get $p_0=\frac{1}{2}$. Thus we establish $p=p_0=p_1$. ■

THEOREM 4.2.4. *Generalisation of Converse Theorem*
If $Pr(x\in\mathcal{E}_0|z)=Pr(x\in\mathcal{E}_0)$ and $Pr(x\in\mathcal{E}_1|z)=Pr(x\in\mathcal{E}_1)$, then for all $j_1,j_2,\ldots,j_m\in\{1,2,\ldots,K\}$ and $c_1,c_2,\ldots,c_m\in\{0,1\}$
where $m\le K$ and for any $i\ne i'$ $j_i\ne j_{i'}$

$$\begin{aligned}&Pr(z[j_m]=c_m|x\in\mathcal{E}_0, z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_{m-1}]=c_{m-1})\\ =\;& Pr(z[j_m]=c_m|z[j_1]=c_1, z[j_2]=c_2,\ldots,z[j_{m-1}]=c_{m-1})\end{aligned}$$

Proof : Applying Bayes' theorem on conditional probability,

$$\begin{aligned}&Pr(x\in\mathcal{E}_0|z[j_1]=c_1,z[j_2]=c_2,\ldots,z[j_m]=c_m)\\ =\;&\frac{Pr(x\in\mathcal{E}_0, z[j_1]=c_1,z[j_2]=c_2,\ldots,z[j_m]=c_m)}{Pr(z[j_1]=c_1,z[j_2]=c_2,\ldots,z[j_m]=c_m)}\\ =\;&\frac{Pr(z[j_m]=c_m|x\in\mathcal{E}_0,z[j_1]=c_1,\ldots,z[j_{m-1}]=c_{m-1})\cdot Pr(x\in\mathcal{E}_0,z[j_1]=c_1,\ldots,z[j_{m-1}]=c_{m-1})}{Pr(z[j_m]=c_m|z[j_1]=c_1,\ldots,z[j_{m-1}]=c_{m-1})\cdot Pr(z[j_1]=c_1,z[j_2]=c_2,\ldots,z[j_{m-1}]=c_{m-1})}\\ =\;&\frac{Pr(z[j_m]=c_m|x\in\mathcal{E}_0,z[j_1]=c_1,\ldots,z[j_{m-1}]=c_{m-1})\cdot Pr(x\in\mathcal{E}_0|z[j_1]=c_1,\ldots,z[j_{m-1}]=c_{m-1})}{Pr(z[j_m]=c_m|z[j_1]=c_1,\ldots,z[j_{m-1}]=c_{m-1})}\end{aligned}$$

Now, we are given $Pr(x\in\mathcal{E}_0|z)=Pr(x\in\mathcal{E}_0)$. Thus,

$$\begin{aligned}&Pr(x\in\mathcal{E}_0|z[j_1]=c_1,z[j_2]=c_2,\ldots,z[j_m]=c_m)\\ =\;&Pr(x\in\mathcal{E}_0)\\ =\;&Pr(x\in\mathcal{E}_0|z[j_1]=c_1,z[j_2]=c_2,\ldots,z[j_{m-1}]=c_{m-1})\end{aligned}$$

Hence,

$$\begin{aligned}&Pr(z[j_m]=c_m|x\in\mathcal{E}_0,z[j_1]=c_1,\ldots,z[j_{m-1}]=c_{m-1})\\ =\;&Pr(z[j_m]=c_m|z[j_1]=c_1,\ldots,z[j_{m-1}]=c_{m-1})\end{aligned}$$

Replacement of $\mathcal{E}_0$ by $\mathcal{E}_1$ in above derivation yields

$$\begin{aligned}
& Pr(z[j_m] = c_m | x \in \mathcal{E}_1, z[j_1] = c_1, \ldots, z[j_{m-1}] = c_{m-1}) \\
= \quad & Pr(z[j_m] = c_m | z[j_1] = c_1, \ldots, z[j_{m-1}] = c_{m-1})
\end{aligned}$$

■

Now we perpose an assumption based on which we develop some protocol in later chapters. First we state the assumption and then support it with theoretical arguments and evidence from the simulations.

Assumption 4.2.1 XOR Assumption
Let $x, y \in_R QR \bigcup QNR$ and $z = x \oplus y$.

$$\begin{aligned}
Pr(x \in QR, y \in QR | z) &= p_0 \\
Pr(x \in QR, y \in QNR | z) &= p_1 \\
Pr(x \in QNR, y \in QR | z) &= p_2 \\
Pr(x \in QNR, y \in QNR | z) &= p_3 \\
p_0 = p_1 = p_2 = p_3 &= 0.25.
\end{aligned}$$

Experimental evidence supports this. If N is a K-bit string, and QR,QNR are taken from Z_N^*, then the number of K-bit string, then the number of K-bit strings z, s.t.,

$$p_0, p_1, p_2, p_3 < 0.25 - \delta(K)$$

or

$$p_0, p_1, p_2, p_3 > 0.25 + \delta(K)$$

is almost zero. Further as K increases $\delta(K)$ decreases.

REMARK 4.2.2.

1. *From simulation it is observed that δ depends on $\frac{N-2^{K-1}}{2^{K-1}}$. As this ratio goes from 0 to $\eta(K) > 0$, δ falls rapidly, where $\eta(K)$ decreases exponentially with increasing K.*
2. *Also the ratio $\frac{p}{q}$ of two prime factor of N is nearer to 1, smaller is δ.*

Assumption 4.2.2 Generalised XOR Assumption
Let for some probabilistic encryption scheme $\mathcal{E}$, $x, y \in_R \mathcal{E}_0 \bigcup \mathcal{E}_1$ and $z = x \oplus y$.

$$\begin{aligned}
Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_0 | z) = p_0 \\
Pr(x \in \mathcal{E}_0, y \in \mathcal{E}_1 | z) = p_1 \\
Pr(x \in \mathcal{E}_1, y \in \mathcal{E}_0 | z) = p_2 \\
Pr(x \in \mathcal{E}_1, y \in \mathcal{E}_1 | z) = p_3 \\
p_0 = p_1 = p_2 = p_3 = 0.25.
\end{aligned}$$

Our Main Theorem, Theorem 4.2.1. and Converse Theorem, Theorem 4.2.3. supports our generalisation.

THEOREM 4.2.5. *Let $x_1, x_2, \ldots, x_m \in_R (\mathcal{E}_0 \bigcup \mathcal{E}_1)$ for some probabilistic encryption scheme $\mathcal{E}_r$ and $z = x_1 \oplus x_2 \oplus \ldots \oplus x_m$ for $m > 2$. Then for all i $(1 \leq i \leq m)$, under the XOR assumption:*

$$\begin{aligned} Pr(x_i \in \mathcal{E}_0 | z) &= Pr(x_i \in \mathcal{E}_0) \\ Pr(x_i \in \mathcal{E}_0 | z) &= Pr(x_i \in \mathcal{E}_0) \end{aligned}$$

Proof : If $m = 2$ then proof follows from XOR assumption. Let for some $m > 2$, and some i

$$Pr(x_i \in \mathcal{E}_0 | z) = Pr(x_i \in \mathcal{E}_0) + \epsilon$$

for some $\epsilon \neq 0$. Then consider $z' = x'_1 \oplus x'_i$. construct Z'' as follows,
1. Generate $(m-2)$ numbers $y_1, y_2, \ldots, y_{m-2} \in_R (\mathcal{E}_0 \bigcup \mathcal{E}_1)$.
2. $z'' = z \oplus y_1 \oplus y_2 \oplus \ldots \oplus y_{m-2}$
3. By the associativety and commutative property of $\oplus$ operation x'_i can be assumed to be at i_{th} position in z''.

Now, from the construction of z'' and the assumption made in the beginning, we get

$$\begin{aligned} Pr(x_i \in \mathcal{E}_0 | z'') &= Pr(x_i \in \mathcal{E}_0 | z') \\ &= Pr(x_i \in \mathcal{E}_0) + \epsilon \end{aligned}$$

Which is in contradiction to XOR assumption. Hence we must have $\epsilon = 0$. ■

Chapter 5

Symmetrically PIR Schemes

5.1 Introduction

In this Chapter we first introduce a Basic Symmetrically Private Information Retrieval scheme for bit retrieval. This scheme in itself will not provide an efficient scheme, but will make the main ideas clear which will be used in efficient bit and block symmetrically private information retrieval scheme.

5.2 The Basic Scheme

Here we present a scheme based on intractability of Quadratic residue problem. Actually any homomorphic trapdoor predicate can be used instead. For clarity, we use DB to mention server or database program which handle database.

5.2.1 The Scheme

The basic scheme is described below. The databse x held by DB is considered to be a n bit string. User is interested in i_{th} bit of the string.

1. **Setup**
   We view the database x held by server as a $(s \times t)$ matrix of bits denoted as D. The i_{th} bit x_i of the database x is $(a, b)_{th}$ entry in matrix D, where $a \leftarrow \lfloor \frac{i}{t} \rfloor + 1$ and $b \leftarrow i \pmod t$ if $b = 0$ then $a \leftarrow a - 1$ and $b \leftarrow t$.

2. **Query Generation**

   - User generate two $\frac{K}{2}$ bit prime numbers p and q, such that $p \equiv q \equiv 3 \pmod 4$, and product of two prime is sufficiently away from 2^{K-1}. Here K is the security parameter. Calculate $N = pq$.
     User sends N to DB and keep its factorization secret.

- User chooses t number at random $y_1, \ldots, y_t \in Z_N^*$. such that y_b is a QNR and y_j $(j \neq b)$ is a QR. User sends these t numbers to DB.
- User chooses s number at random $\gamma_1, \ldots, \gamma_t \in Z_N^*$. such that γ_a is a QNR and γ_j $(j \neq b)$ is a QR. User sends these s numbers to DB.

3. **Database Computation**

- ***Pre Processing*** *(Column Control)*
  DB chooses at random a number χ which is a QNR (which he can do because $p \equiv q \equiv 3 \pmod 4$) and $(t-1)$ QR's $\chi_1, \ldots, \chi_{t-1}$, and from that he forms a $t \times t$ matrix q as follows:

$$q = \begin{bmatrix} \chi & \chi_1 & \cdot & \cdot & \cdot & \chi_{t-2} & \chi_{t-1} \\ \chi_1 & \chi & \cdot & \cdot & \cdot & \chi_{t-2} & \chi_{t-1} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \chi_1 & \chi_2 & \cdot & \cdot & \cdot & \chi & \chi_{t-1} \\ \chi_1 & \chi_2 & \cdot & \cdot & \cdot & \chi_{t-1} & \chi \end{bmatrix}$$

  He computes for every row r of the matrix D, and every row τ of the matrix q, a number $\psi(r,\tau) \in Z_N^*$, as follows :

$$\psi(r,\tau) = \prod_{l=1}^{t} (q(\tau,l))^{D(r,l)}$$

  where $1 \leq r \leq s$ and $1 \leq \tau \leq t$. Thus, DB computes $s \times t$, K-bit numbers $\psi(r,\tau)$.
  DB forms K matrices of size $s \times t$, $D^{(\kappa)}$ $:\kappa = 1, 2, \ldots, K$ as follows:
  Each $\psi(r,\tau)$ is a K-bit number. Denote by $\psi(r,\tau,\kappa)$ the κ_{th} bit of $\psi(r,\tau)$. The matrix D_κ is defined as follows:

$$D_\kappa(r,\tau) = \psi(r,\tau,\kappa)$$

  Thus for the original matrix D, DB has formed K matrices $D_{(\kappa)}$ of same size.

- For each of the K matrices $D_{(\kappa)}$, DB computes for every row r of D_κ, a number $z(r,\kappa)$ as follows :

$$z(r,\kappa) = \nu_r \left(\prod_{l=1}^{t} (y_l)^{D_\kappa(r,l)} \right)$$

  where ν_r is a randomly chosen QR by DB. Thus, DB computes $s \times K$, numbers $z(r,\kappa)$ where each $z(r,\kappa)$ is a K-bit number. For, $1 \leq c \leq K$, denote by $z(r,\kappa,c)$ the c_{th} bit of $z(r,\kappa)$.

- ***Post Processing*** *(Row Control)*
  DB forms K matrices $\Delta_{(\kappa)}$, $1 \leq \kappa \leq K$, such that

  $$\Delta_{(\kappa)}(r, c) = z(c, \kappa, r)$$

  Now, for each of the K matrices $\Delta_{(\kappa)}$ $(1 \leq \kappa \leq K)$, the database DB, computes for every row r $(1 \leq r \leq K)$ a number $\zeta(r, \kappa)$ as follows :

  $$\zeta(r, \kappa) = \nu_r \left(\prod_{l=1}^{s} (\gamma_l)^{\Delta_\kappa(r,l)} \right)$$

  where ν_r is a randomly chosen QR by DB. Thus for each of the K matrices Δ_κ, DB computes K, numbers $\zeta(1, \kappa)$,$\zeta(2, \kappa)$,...,$\zeta(s, \kappa)$, where each of these numbers in itself a K-bit number. DB sends these K^2 numbers (a total of K^3 bits) to user.

4. **Bit Reconstruction**
   User retrieve desired bit as follows:

   (a) Observe that $\zeta(r, \kappa)$ is a QR iff $\Delta_\kappa(r, a)$ is 0. Thus by determining the quadratic characters of K^2 numbers, $\zeta(r, \kappa)$, he gets $\Delta_\kappa(1, a), \Delta_\kappa(2, a), \ldots, \Delta_\kappa(K, a)$.

   (b) From the construction of matrix Δ_κ, we see that,

   $$\begin{aligned} z(a, \kappa, r) &= \Delta_\kappa(r, a) \\ z(a, \kappa, 1), \ldots, z(a, \kappa, K) &= z(a, \kappa) \text{ for all } 1 \leq \kappa \leq K \ . \end{aligned}$$

   Thus for all $1 \leq \kappa \leq K$, user gets $z(a, \kappa)$.

   (c) $z(a, \kappa)$ is a QR iff $D_\kappa(a, b)$ is 0. Thus by determining the quadratic characters of K numbers $z(a, \kappa)$ $(1 \leq \kappa \leq K)$, he gets $D_1(a, b), \ldots, D_K(a, b)$. From the construction of matrices D_κ, it is clear that

   $$\begin{aligned} D_\kappa(a, b) &= \psi(a, b, \kappa) \\ \psi(a, b, 1), \ldots, \psi(a, b, K) &= \psi(a, b) \end{aligned}$$

   (d) $\psi(a, b)$ is a QR iff $D(a, b)$ is 0.

REMARK 5.2.1. *The security parameter K must be such that*

$$K \geq \max\{K_0, \quad poly(\log n)\}$$

where n is the number of bits in database, K_0 is the smallest number such that encryption scheme under consideration (encryption by QR's and QNR's here) is secure. The $poly(\log n)$ *factor comes because, DB is assumed to be resourceful enough to do a computation of $O(n)$.*

REMARK 5.2.2. *In place of computing $\psi(\sigma, \tau)$, as suggested in* Preprocessing phase *of Database Computation, DB can just choose a QR if $(\sigma, \tau)_{th}$ bit of database is* 0 *or a QNR if it is* 1 *uniformly at random and call it $\psi(\sigma, \tau)$.*

5.2.2 Correctness

LEMMA 5.2.1. *Let $x = [x_1, x_2, \ldots, x_t]$ be a bit array of length t and let $\chi_1, \ldots, \chi_t$ be chosen such that $\chi_1, \ldots, \chi_{i-1}, \chi_{i+1}, \ldots, \chi_t$ are QR's and χ_i is a QNR. Let*

$$y = \prod_{l=1}^{t} (\chi_l)^{x_l}.$$

Then y is a QR iff $x_i = 0$.

Proof : . We defined a number as QR if it is from the set Z_N^* and its Jacobi Symbol is 1 with respect to both the prime factor of composite number N. And here by QNR, we mean only those numbers from the set Z_N^*, which have their Jacobi symbol -1 with respect to both of the prime factors of N (i.e., Jacobi symbol with respect to N is 1). We have already shown in section 4.1 product of any two QR's is a QR, product of any two QNR's is a QR and product of a QR with a QNR is a QNR. As among t χ_j's only χ_i is a QNR thus y is a QNR iff χ_i is included in the product odd number of times. Now from the rule of computation for y it is clear that χ_i is included in the product once if $x_i = 0$ otherwise it is not included. ■

Correctness of the protocol described above can be established by using above lemma in the reconstruction process described in the scheme.

5.2.3 Privacy of database

1. **Honest but Curious User** It is clear from the argument given for establishing correctness of protocol, and can easily be verified following the presented scheme, that as long as user send only one QNR in each set of numbers in his query, he will be able to reconstruct one and only one bit of database, the index of bit being determined by the position of QNR's in query. Hence privacy of database established for honest user.

2. **Dishonest User** A dishonest user can deviate from the protocol to possibly gain any extra information in the following ways:

   (a) N is a product of more than two primes. It is not clear that the user can gain extra information by using such N. Hence we will assume that N is indeed a product of two primes.

   (b) Assuming that N is a product of two primes, the numbers that the user sends to DB must be in Z_N^* and their Jacobi Symbol must evaluate to 1. This is necessary since the DB can perform this computation and reject a query not confirming this specification. Hence the only way a dishonest user can cheat is to send more than one QNR in each query set. We now argue that this provides the user with no information at all, i.e., even if one query set has two QNR's then the user does not get even one bit of the database.

LEMMA 5.2.2. *Let $x_1, \ldots, x_t$ be in $QR \bigcup QNR$ and $b_1, \ldots, b_t$ be a bit string and a number z is computed as:*

$$z = \prod_{l=1}^{t} (x_l)^{b_l}.$$

Suppose $x_{i_1}, \ldots, x_{i_s}$ are QNR's and rest are QR's, then z is a QR iff $b_{i_1} \oplus \ldots \oplus b_{i_s} = 0$.

Proof : We defined a number as QR if it is from the set Z_N^* and its Jacobi Symbol is 1 with respect to both the prime factor of composite number N. And here by QNR, we mean only those numbers from the set Z_N^*, which have their Jacobi symbol -1 with respect to both of the prime factors of N (i.e., Jacobi symbol with respect to N is 1). We have already shown in section 4.1 product of any two QR's is a QR, product of any two QNR's is a QR and product of a QR with a QNR is a QNR. Rest of the proof then follows. ∎

Now we consider three different cases which can occur from the possible character of query send by user:

i. *First set contains more than one QNR* Let the first set in the query sent by user contain s QNR's at positions $b_1, \ldots, b_s$. Then lemma 5.2.2., implies that in the reconstruction phase a number $z(a, \kappa)$, $(1 \leq \kappa \leq K)$ obtained by user is a QR iff $D_\kappa(a, b_1) \oplus \ldots \oplus D_\kappa(a, b_s)$. Thus user be able to reconstruct $\psi(a, b_1) \oplus \ldots \oplus \psi(a, b_s)$. Theorem 4.2.5. says from the $\oplus$ of two or more numbers from the set $QR \bigcup QNR$, it is impossible to know anything about the membership of the constituent numbers. Thus user fails to know the quadratic character of any of the $psi(a, b_i)$. Hence if he sends more than one QNR's in the first set of his query he fails to get any bit of the database x.

ii. *Second set contains more than one QNR* Consider that second set of numbers in the query sent by user contains QNR's at positions $a_1, a_2, \ldots, a_m$. Again using lemma 5.2.2. in post processing computation, we see that a number $\zeta(r, \kappa)$ received by user is a QR iff $\Delta_\kappa(r, a_1) \oplus \ldots \oplus \Delta_\kappa(r, a_s) = 0$, $(1 \leq r \leq K)$, and $(1 \leq \kappa \leq K)$. Thus user will be able to reconstruct $z(a_1, \kappa) \oplus z(a_2, \kappa) \oplus \ldots \oplus z(a_m, \kappa)$ and from which he will be unable to gain anything about the membership of any $z(a_\mu, \kappa)$. Thus he will not gain any information about database.

iii. *Both sets contain more than one QNR* Let first set of numbers in the query sent by user contains more than one QNR's at positions $b_1, \ldots, b_s$ and the second set of numbers in the query sent by user contains QNR's at positions $a_1, a_2, \ldots, a_m$. Then a number $\xi(r, \kappa)$ received by user is a QR iff $\Delta_\kappa(r, a_1) \oplus \ldots \oplus \Delta_\kappa(r, a_s) = 0$, $(1 \leq r \leq K)$, and $(1 \leq \kappa \leq K)$. Thus user will be able to reconstruct $z(a_1, \kappa) \oplus z(a_2, \kappa) \oplus \ldots \oplus z(a_m, \kappa)$ $(1 \leq \kappa \leq K)$. In this case even after knowing the quadratic nature of say $z(a_j, \kappa)$ he can get only $\psi(a_j, b_1) \oplus \ldots \oplus \psi(a_j, b_s)$. Thus obtaining any information in this case is even difficult than that in case 2(b)i and case 2(b)ii.

5.2.4 Privacy of User

To prove the privacy of user, we follow the technique suggested by Kushilevitz and Ostrovsky [11]. Let Q_i be the distribution of queries sent by user when he is interested in the i_{th} bit of database. Suppose that for some indices i and i' the database DB can distinguish the queries on index i from the query on index i'. The fact that that database can distinguishes these two queries implies that there exists a family of polynomial time circuits C_n that distinguishes the two query distribution Q_i and $Q_{i'}$ with probability larger than $\frac{1}{n^\ell}$ for some integer ℓ. Assume $Pr(C_n(q \in Q_i) = 1) = p$ and $Pr(C_n(q \in Q_{i'}) = 1) = p + \frac{1}{n^\ell}$ In our matrix notation $i = (a, b)$ and $i' = (a', b')$. we emphasize here that in our protocol for any $i \neq i'$, Q_i and $Q_{i'}$ are not identical. By protocol, in case of index i, query consists of a number N uniformly picked from H_K and two set of t numbers and s numbers respectively, from the set z_N^{+1}. First set consists of $t-1$ QR's and a QNR at position b, and second set consists of $s-1$ QR's and a QNR at position a. Query distribution $Q_{i'}$ is similar to Q_i except that two QNR's are at position b' in first set and at position a' in second set, instead of a and b.

We can construct another family of polynomial time circuit C'_n from C_n, which will, on input $N \in H_K$ and $y \in Z^{+1}$ computes the quadratic residuosity predicate with probability at least $\frac{1}{2} + \frac{1}{8 \cdot n^\ell}$.

Let $I_i = \{a, b\}$ and $I_{i'} = \{a', b'\}$. If $i \neq i'$ then $I_i \neq I_{i'}$. The construction of C'_n is as follows :

1. Construct a sequence as expected by the circuit C_n. Pick at random one of the subsets I_i and $I_{i'}$. In every position not in the chosen set put a random QR from Z_N^{+1}. In every position which is in the chosen subset put the product of y and a random QR.

2. With probability $\frac{1}{4 \cdot n^\ell}$ answer 0 and stop.

3. Run C_n on the input constructed in (1). If the chosen subset was $I_{i'}$ output the same answer as C_n; If the chosen subset was I_i flip C_n's output.

Let us now calculate probability that C'_n outputs 1 on input N, y such that y is a QR modulo N. In this case no matter which subset was chosen in step(1), the input to C_n is a sequence of random QR's in Z_N^{+1}. On such an input there is a probability q that C_n output 1. Thus the probability that C'_n output 1 in this case is $\left(1 - \frac{1}{4 \cdot n^\ell}\right)\left(\frac{1}{2}q + \frac{1}{2}(1-q)\right) = \frac{1}{2} + \frac{1}{8 \cdot n^\ell}$

Now, we compute the probability that C'_n outputs 1 on input N, y such that y is a QNR modulo N. By assumptions on C_n, the probability that C'_n output 1 is :

$$\begin{aligned}
\left(1 - \frac{1}{4 \cdot n^\ell}\right)\left(\frac{1}{2} \cdot \left(p + \frac{1}{n^\ell}\right) + \frac{1}{2} \cdot (1-p)\right) &= \left(1 - \frac{1}{4 \cdot n^\ell}\right)\left(1 + \frac{1}{2 \cdot n^\ell}\right) \\
&= \frac{1}{2} + \frac{1}{4 \cdot n^\ell} - \frac{1}{8 \cdot n^{2l}} \\
&\geq \frac{1}{2} + \frac{1}{8 \cdot n^\ell}
\end{aligned}$$

So, indeed if any family of polynomial-size circuits breaks the privacy of SPIR scheme presented here, then we can use it to decide quadratic residuosity predicate.

5.2.5 Communication complexity

User sends a K-bit number N followed by two set of K-bit numbers, first set having t numbers and second set having s, to DB. Thus total communication from user to database DB in this scheme is $(1 + t + s)$ K-bit numbers $(N, y_1, \ldots, y_t, \gamma_1, \ldots, \gamma_s)$. while database returns K^2 K-bit numbers ζ_r^κ $(1 \leq r \leq K, 1 \leq \kappa \leq K)$ obtained after the postprocessing to user. Thus the communication complexity is $(1 + t + s + K^2) \cdot K$ bit, which can be minimized by choosing $t = s = \sqrt{n}$, and the communication complexity is :

$$(2\sqrt{n} + 1 + K^2) \cdot K$$

Therefore, even with the weaker assumption on security parameter, i.e; $K = n^\epsilon$ for some constant $\epsilon > 0$, we get a communication complexity $O(n^{\frac{1}{2}+\epsilon})$, provide $\epsilon < \frac{1}{4}$. Under the similar assumptions over security parameter Kushilevitz and Ostrovsky [11] obtained the communication complexity $(2 \cdot \sqrt{n} + 1) \cdot K$ for their basic PIR scheme. A closer look reveals that, with an extra communication of K^3 bit over the basic PIR scheme presented by Kushilevitz and Ostrovsky [11], we have successfully obtained a SPIR scheme.

Thus we have proved the following theorem :

THEOREM 5.2.1. *Under the QRA, and XOR Assumption, for every $\frac{1}{4} > \epsilon > 0$, there exists a cSPIR protocol with communication complexity $O(n^{\frac{1}{2}+\epsilon})$.*

REMARK 5.2.3. *The communication complexity for basic SPIR scheme presented here is less than n for all $n > n_0$ for some n_0, which depends on security parameter K.*

5.3 Iterative Bit SPIR Scheme

In this section we develop a improved scheme using the ideas developed in our basic scheme and we manage to bring down the communication complexity. We essentially achieve this by inducing DB to do a multiround of computation iteratively. Thereby we successfully reduce total communication size. We put stress on the fact that scheme involves only single round of communication and security of database as well as user remain preserved.

5.3.1 The Scheme

The databse x held by DB is considered to be a n bit string. User is interested in i_{th} bit of the string.

1. **Setup**
   We view the database x held by server as a $(s \times t)$ matrix of bits denoted as D. The i_{th} bit x_i of the database x is $(a_1, b_1)_{th}$ entry in matrix D, where $a_1 \leftarrow \lfloor \frac{i}{t} \rfloor + 1$ and $b_1 \leftarrow i \pmod t$ if $b = 0$ then $a \leftarrow a - 1$ and $b \leftarrow t$.

2. **Query Generation**

- User U generate two $\frac{K}{2}$ bit prime number p and q. Calculate $N = pq$
  User sends N to DB and keep its factorization secret.
- user calculate t such that $t^{L+1} = n$, where L is the number of round database is going to do his iterative computation, beside a special last round of computation. L is chosen such that communication complexity is minimized (as we will see later).
- User generates a sequence of the pair of indices (a_j, b_j)
  $$\textbf{for } j \leftarrow 1 \textbf{ to } L-1 \begin{cases} a_{j+1} \leftarrow \lfloor a_j/t \rfloor + 1 \\ b_{j+1} \leftarrow a_j \% t \end{cases}$$
  $$\text{if } b_{j+1} = 0 \text{ then } a_{j+1} \leftarrow a_{j+1} - 1 \text{ and } b_{j+1} \leftarrow t$$
  (a_j, b_j) correspond to the row and column index of the relevant bit in the matrices in j_{th} round of DB computation. Also define $s_j = \frac{n}{t^j}$ and $t_j = t$ for $1 \leq j \leq L$. (s_j, t_j) are number of rows and columns in each matrix in the j_{th} round of DB computation.
- User generates an $L \times t$ matrix y, where for $1 \leq \sigma \leq L$, $1 \leq \beta \leq t$:
  $y(\sigma, \beta)$ is a QNR if $\beta = b_\sigma$, else it is a QR. clearly each row of y contains exactly one QNR. User sends the matrix y do DB.
- User also chooses s_L numbers $\gamma_1, \ldots, \gamma_{s_L} \in Z_N^*$ uniformly at random, such that γ_{a_L} is a QNR and γ_j $(j \neq a_L)$ is a QR.
  U sends these s_L numbers to DB.

3. **Database Computation**
   Database DB performs a $L+2$ round of computation in three phases. First phase is the preprocessing round of basic scheme, while the third phase is the post processing round. The middle step of the basic scheme is being computed iteratively over L rounds.

   - ***Pre Processing*** *(Column Control)*
     Database x is assumed to be arranged in a matrix form as follows :
     $$x \equiv \{x_i\} \rightarrow \{D(\alpha, \beta)\} \qquad \alpha t + \beta = i; 0 \leq \beta < t$$
     As in the basic scheme DB, forms K matrices $D_\kappa(\alpha, \beta)$ from the original matrix $D(\alpha, \beta)$. Again the user requires exactly one bit from each of the matrices D_κ's.
   - ***Iterative Computation***
     The database DB performs a L round of iterative computation according to following algorithm:
     From main: for each D_κ $(1 \leq \kappa \leq K)$ formed in the preprocessing round perform the call $DFSCompute(D_\kappa, s, t, y_1, 1)$.
     The algorithm $DFSCompute$ is described below.
     $DFSCompute(M, s, t, y_l, l)\{$
     /*
     - y_l is the l_{th} row of the matrix of numbers sent by user. Remember $y_l[i] = y(l, i)$ is a QR if $i \neq b_l$ and $y_l[i]$ is a QNR if $i = b_l$.

– M is an $s \times t$ matrix of bits and we want to retrieve the bit $M[a_l, b_l]$.

– l denoted the level of the recursion.

*/

(a) Set for $1 \le i \le s$

$$z[i] = \nu_i \left(\prod_{j=1}^{t} (y_l[j])^{M[i,j]} \right)$$

where ν_i for each i is a QR chosen by DB uniformly at random.

/* Each $z[i]$ is a K-bit string. For $1 \le j \le K$, let $z[i,j]$ denote the j_{th} bit of $z[i]$. We require the string $z[a_l]$ */

(b) For $1 \le j \le K$, form K matrices M_j, where each M_j is an $\frac{s}{t} \times t$ matrix formed from the column vector,

$$z[*,j] = z[1,j], \ldots, z[s,j]$$

by breaking $z[*,j]$ into t-bit blocks and arranging the blocks in a row wise fashion. /* The string $z[a_l]$ is distributed over the K matrices M_j, i.e, the string $z[a_l]$ is equal to $M_1[a_{l+1}, b_{l+1}], \ldots, M_K[a_{l+1}, b_{l+1}]$, where $a_{l+1} = \lfloor \frac{a_l}{t} \rfloor + 1$ and $b_{l+1} = a_l \pmod t$. If $b_{l+1} = 0$, $a_{l+1} \leftarrow a_{l+1} - 1$ and $b_{l+1} = t$. */

(c) $for(1 \le j \le K)\{$

$if(l < L - 1)$

$DFSCompute(M_j, \frac{s}{t}, t, y_{l+1}, l+1)$

$else$

$PostCompute(M_j, \frac{s}{t}, t, y_L, \gamma)$

$\}$

$\}$

$PostCompute(\cdot)$ include the postprocessing step of our Basic SPIR Scheme. The algorithm $PostCompute(\cdot)$ is described below:

- ***Post Processing*** *(Row Control)*

$PostCompute(M, s, t, y, \gamma)\{$

/*

– As in $DFSCompute$ M is an $s \times t$ matrix of bits and we want to retrieve the bit $M[a,b]$, where the index a and b is hidden in the y and γ.

– $y[j]$ $(1 \le j \le t)$ is an array of t numbers (L_{th} row of the matrix y sent by user). $y[j]$ is a QNR for $j = b$ else it is a QR.

– $\gamma[j]$ $(1 \le j \le s)$ is an array of s numbers (γ sent by user). $\gamma[j]$ is a QNR for $j = a$ else it is a QR.

*/

(a) Set for $1 \le i \le s$

$$z[i] = \nu_i \left(\prod_{j=1}^{t} (y[j])^{M[i,j]} \right)$$

where ν_i for each i is a QR chosen by DB uniformly at random.
/* Each $z[i]$ is a K-bit string. For $1 \leq j \leq K$, let $z[i,j]$ denote the j_{th} bit of $z[i]$. We require the string $z[a]$ */

(b) Set $M'[i,j] = z[j,i]$ for $1 \leq j \leq s$, $1 \leq i \leq K$. /* M' is an $K \times s$ matrix of bits. */

(c) Set for $1 \leq i \leq K$

$$\zeta[i] = \nu_i \left(\prod_{j=1}^{s} (\gamma[j])^{M'[i,j]} \right)$$

where ν_i for each i is a QR chosen by DB uniformly at random.

(d) Output the strings $\zeta[1], \ldots, \zeta[K]$.

}

4. **Reconstruction Phase and Correctness**
Here we show that if user receives all the numbers obtained by DB in the end of routine $PostCompute(\cdot)$ then he can reconstruct the $(a_1, b_1)_{th}$ bit of matrix D i.e., i_{th} bit of database x. This will establish the correctness of protocol as well provide the method using which user can reconstruct the i_{th} bit of database x.

   (a) Suppose the call $PostCompute(M, s, t, y, \gamma)$ produces the strings $\zeta[1], \ldots, \zeta[K]$ and the QNR's of y and γ are the b_{th} and a_{th} position respectively. $\zeta[i]$ is a QR iff $M'[i,a]$ is 0, so it is possible to reconstruct the column vector $M'[*,a]$ which is equal to the row vector $z[a,*] = z[a]$. Again $z[a]$ is a QR iff $M[a,b]$ is 0. thus it is possible to find $M[a,b]$. So it is possible to get the bits at level $L-1$. Now we use backward induction on the depth of recursion.

   (b) Suppose the call $DFSCompute(M, s, t, y_l, l)$ produces the matrices $M_1, \ldots, M_K$, where the bits $M_i[a_{l+1}, b_{l+1}]$ are required. By induction hypothesis it is possible to get these bits. Since $z[a_l]$ is equal to $M_1[a_{l+1}, b_{l+1}], \ldots, M_K[a_{l+1}, b_{l+1}]$, it is possible to get $z[a_l]$. $z[a_l]$ is a QR iff $M[a_l, b_l]$ is 0. Thus we get the bit $M[a_l, b_l]$. Hence our hypothesis established, and user can get the $(a_1, b_1)_{th}$ bit of all the K matrices D_κ passed in the routines $DFSCompute(M, s, t, y_1, 1)$ from the main in DB computation provided the query matrix y sent by user had only one QNR per row at correct position and array γ had the only QNR at position a_L. Thus user gets the bits $D_1[a_1, b_1], \ldots, D_K[a_1, b_1]$, which on concatenation gives the number $\psi(a_1, b_1)$ by which DB had replaced the $(a_1, b_1)_{th}$ bit of matrix D, i.e., $\psi(a_1, b_1)$ is a QR iff $D(a_1, b_1)$ is 0.

Hence user can reconstruct the i_{th} bit of database x from the numbers he receives from DB.

5.3.2 Privacy of Database

Privacy of the database can be shown to be maintained in the iterative cSPIR protocol presented above, following the similar arguments we used to prove the privacy of database in our basic scheme. As already discussed, while proving the privacy of database in basic scheme, that user can send only QR's and QNR's in the set of numbers in his query. As long as user sends only one QNR in each row of the query matrix, and only one QNR in the array for postprocessing round, he will be able to retrieve a single physical bit of the database x, the index of which being determined by the sequence formed by the position of QNR in each row of query matrix and position of QNR in the query array. Thus the only way user can try to gain more information about the original database x is by sending more than one QNR in some of the row's of query matrix or in the query array. Now we consider different possible cases:

1. *Query array γ sent by user for post processing round contains more than one QNR's* Let position of QNR's are $a_1, a_2, \ldots, a_m$. Then lemma 5.2.2. implies that user will be able to reconstruct $z[a_1] \oplus \ldots \oplus z[a_m]$ in place of $z[a]$. Theorem 4.2.5. says that it is not possible to know anything about the quadratic nature of numbers from their $\oplus$'s even if the factorization is known. Thus user fails to reconstruct any of the bit of matrix $m[a, b]$ and his reconstruction process halted without getting any bit of the database.

2. *At least one row in the query matrix y contain more than one QNR* Let at least l_{th} row of the matrix y contain more than one QNR's at positions $b_1^{'}, b_2^{'}, \ldots, b_m^{'}$. Assuming that user could successfully obtain $M_1[a_{l+1}, b_{l+1}], \ldots, M_K[a_{l+1}, b_{l+1}]$, he can get $z[a_l]$ but now $z[a_l]$ is QR iff $M[a_l, b_1^{'}] \oplus \ldots \oplus M[a_l, b_m^{'}]$. Thus he can reconstruct only $z[a_{l-1_1}] \oplus \ldots \oplus z[a_{l-1_m}]$ where $a_{l-1_j} = t \cdot a_l + b_j^{'}$. From that as said before, he fails to determine quadratic nature of any of the $z[a_{l-1_j}]$'s and thus his reconstruction process terminate without reconstructing any bit of database x.

3. *Query array γ as well as at least one row in query matrix y contain more than one QNR's* From the explanation in above two cases, it is clear that, in this case it is even harder to obtain any information than the previous two cases. Following the explanation in case 1 reconstruction process halts in the first step itself. As he fails to know the quadratic character of any of the $z[a_j]$'s from the $z[a_1] \oplus \ldots \oplus z[a_m]$ which he obtains because there were QNR's at positions $a_1, a_2, \ldots, a_m$ in query array γ.

Thus if user sends query as per the protocol he will get one physical bit of the database and the moment he deviate from the protocol he fails to get any information at all about database x.

5.3.3 Privacy of User

The proof for privacy of user can be built over the ideas we used to proof the privacy of user in case of our scheme. Again, we assume towards a contradiction that there exists a distinguisher which for some index i and i' can distinguish the distribution of queries Q_i from the distributing of queries $Q_{i'}$ with probability larger than $\frac{1}{n^\ell}$ for some integer ℓ.

The query sent by user is a number N followed by $(L+1)$ set of numbers from Z_N^{+1}, each set consisting of $(t-1)$ QR's and a QNR. Thus we can view this as a sequence of numbers $y_1, y_2, \ldots, y_\beta$ from Z_N^{+1}, where $\beta = (L+1) \cdot t$. From the calculation t is equal to $n^{\frac{1}{L+1}}$. For an index i denote by

I_i the subset of $\{1, 2, \ldots, \beta\}$ of all positions that contain QNRs. By the protocol, the subset I_i is fixed and is independent of the random choices made by the user. Also for $i \neq i'$, $I_i \neq I_{i'}$ as the subset I_i uniquely determine the index i.

As before, we denote by C_n the family of polynomial time circuits that distinguishes between Q_i and $Q_{i'}$. Assume $Pr(C_n(q \in Q_i) = 1) = p$ and $Pr(C_n(q \in Q_{i'}) = 1) = p + \frac{1}{n^\ell}$

We construct another family of polynomial time circuit C'_n from C_n, which, on input $N \in H_K$ and $y \in Z_N^{+1}$ computes the quadratic residuosity predicate with probability at least $\frac{1}{2} + \frac{1}{8 \cdot n^\ell}$, as follows :

1. Construct a sequence as expected by the circuit C_n. Pick at random one of the subsets I_i and $I_{i'}$. In every position which is not in the chosen set put a random QR from Z_N^{+1}. In every position which is in the chosen subset put the product of y and a random QR.
2. With probability $\frac{1}{4 \cdot n^\ell}$ answer 0 and stop.
3. Run C_n on the input constructed in (1). If the chosen subset was $I_{i'}$ output the same answer as C_n; If the chosen subset was I_i flip C_n's output.

The Let us now calculate probability that C'_n outputs 1 on input N, y such that y is a QR modulo N. The observation is that, in this case no matter which subset was chosen in step(1), the input to C_n is a sequence of random QR's in Z_N^{+1}. On such an input there is a probability q that C_n output 1. Thus the probability that C'_n output 1 in this case is $(1 - \frac{1}{4 \cdot n^\ell})(\frac{1}{2}q + \frac{1}{2}(1-q)) = \frac{1}{2} + \frac{1}{8 \cdot n^\ell}$

Now, we compute the probability that C'_n outputs 1 on input N, y such that y is a QNR modulo N. By assumptions on C_n, the probability that C'_n output 1 is :

$$\begin{aligned}
(1 - \frac{1}{4 \cdot n^\ell})(\frac{1}{2}(p + \frac{1}{n^\ell}) + \frac{1}{2}(1-p)) &= (1 - \frac{1}{4 \cdot n^\ell})(1 + \frac{1}{2 \cdot n^\ell}) \\
&= \frac{1}{2} + \frac{1}{4 \cdot n^\ell} - \frac{1}{8 \cdot n^{2l}}) \\
&\geq \frac{1}{2} + \frac{1}{8 \cdot n^\ell}
\end{aligned}$$

So, indeed if any family of polynomial-size circuits breaks the privacy of iterative cSPIR scheme presented here, then we can use it to decide quadratic residuosity predicate.

5.3.4 Communication Complexity

Communication from user to DB is K-bit number N followed by a of $L \times t$ query matrix y of K-bit numbers, followed by a array of t K-bit numbers. Thus total communication from user to database DB in this scheme is $(1 + (L+1) \cdot t) \cdot K$. DB returns numbers computed at the end of $PostCompute(\cdot)$ routine. We analyse the tree structure formed from the computation process of DB. Root of the computation tree is the matrix D formed from original database x. Now in pre processing computation DB obtains K matrices D_κ's $(1 \leq \kappa \leq K)$ from the matrix D. Each of these K matrices becomes the child of the root. Thus root node, designated at level 0 has K children (all at level 1). Call of routine $DFSCompute(\cdot)$ at l_{th} level of recursion takes a matrix at l level as input and produces K matrices as output. Thus each of the node at level $l < L$ has K children. Matrices at level L becomes the input for routine $PostCompute(\cdot)$ which produces K numbers of K-bit each, these numbers are returned to user. Thus for each of the K^L matrices (leaf nodes of computation tree), user receives K^2. Thus total communication from DB to user is K^{L+2} bits. Thus the communication complexity is $(1 + (L+1) \cdot t + K^{(L+1)}) \cdot K$ bit, where $t = n^{\frac{1}{L+1}}$, and the communication complexity is :

$$(1 + (L+1) \cdot n^{\frac{1}{L+1}} + K^{L+1}) \cdot K$$

If we choose, $L = \sqrt{\frac{\log n}{\log K}} - 1$, then the communication complexity becomes

$$C = (1 + (\sqrt{\frac{\log n}{\log K}} + 1) \cdot 2^{\sqrt{\log n \cdot \log K}}) \cdot K \tag{5.3.1}$$

compare this with the communication complexity $O(2^{2 \cdot \sqrt{\log n \cdot \log K}})$ obtained by Kushilevitz and Ostrovsky [11]. for their cPIR scheme.

Thus we have transformed the cPIR scheme in [11] into a single-round cSPIR scheme with the communication complexity even smaller than what they got for their cPIR scheme.

Even with the weaker assumption on security parameter, i.e; $K = n^\epsilon$ for some constant $\epsilon > 0$, we get a communication complexity $O(\frac{1}{\sqrt{\epsilon}} \cdot n^{\sqrt{\epsilon}+\epsilon}) = O(n^{\sqrt{\epsilon}+\epsilon})$. which is better than the communication complexity $(n^{2 \cdot \sqrt{\epsilon}})$ obtained in [11] under the same assumption. If we take $K = \log^\epsilon n$, then we get the communication complexity of $O(\sqrt{\frac{\log n}{\epsilon \cdot \log \log n}} \cdot \log^\epsilon n \cdot 2^{\sqrt{\epsilon \cdot \log n \cdot \log \log n}})$.

Thus we have proved the following theorem :

THEOREM 5.3.1. *Under the QRA, and XOR Assumption, for every $\epsilon > 0$, there exists a cSPIR protocol with communication complexity $O(n^\epsilon)$.*

5.4 Block Retrieval SPIR Scheme

In previous section we presented scheme for retrieving a bit from a database modeled as a array of bits. But a more realistic view of a database is to assume it partitioned into blocks rather than bits. We see that that the scheme presented in previous section (for bit retrieval) is better suited to block retrieval.

In this section we present our block retrieval SPIR scheme. For simplicity we assume that number of bits in each record is fixed, m bits per record.

5.4.1 The Scheme

The Scheme is described below: We view database x as a array of records, each of size m, having n records in total. User wants to retrieve i_{th} record. Number of records n and number of bits in a

record m are determine L, which will determine the number of round in database computation.
for $m < n$ $\quad m^{L-1} \leq n \leq m^L \quad$ for some integer $L > 1$
for $l \geq n \quad L = 1$
Thus, for a given n, m : $\quad L = \lceil \frac{\log n}{\log m} \rceil$.
We also define $t = n^{\frac{1}{L}}$.

1. **Setup**
   User define two sequences of L integers which are going to be the row and column index for query generation.

$$a_1 \leftarrow \lfloor i/t \rfloor + 1 \qquad (1 \leq i \leq n)$$
$$b_1 \leftarrow i \pmod t$$
$$\text{if } b_1 = 0 \text{ then } a_1 \leftarrow a_1 - 1 \text{ and } b_1 \leftarrow t$$

   and

$$\textbf{for } j \leftarrow 2 \textbf{ to } L \begin{cases} a_j \leftarrow \lfloor a_{j-1}/t \rfloor + 1 \\ b_j \leftarrow a_{j-1} \% t \end{cases}$$
$$\text{if } b_j = 0 \text{ then } a_j \leftarrow a_j - 1 \text{ and } b_j \leftarrow t$$

   Note that $a_L = 1$ and $b_L = a_{L-1}$. For the purpose of DB computation database x held by DB is viewed as a stack of m matrices D_μ $(1 \leq \mu \leq m)$, where each matrix D_μ is an $s \times t$ matrix of bits and user wants to retrieve the bits $D_\mu(a_1, b_1)$.

2. **Query Generation**

   - User U generate two $\frac{K}{2}$ bit prime number p and q, such that $p \equiv q \equiv 3 \pmod 4$ and $N = pq$, such that N is sufficiently away from 2^{K-1}.
     U sends N to DB and keep its factorization secret.
   - As in BRSPIR (Bit Retrieval PIR) Scheme, user generatees an $L \times t$ matrix y, where for $1 \leq \sigma \leq L$, $1 \leq \beta \leq t$:
     $y(\sigma, \beta)$ is a QNR if $\beta = b_\sigma$, else it is a QR. clearly each row of y contains exactly one QNR. User sends the matrix y do DB.

3. **Database Computation**
   The database DB performs a $L + 1$ round of iterative computation as described ahead.

$$\textbf{for } j \leftarrow 1 \textbf{ to } L \begin{cases} s_j = \frac{n}{t^j} \\ t_j = t \end{cases}$$

   (s_j, t_j) are number of rows and columns in each matrix in the j_{th} round of DB computation. With our choice of t, s_L is equal to 1

   Database DB performs a $L + 1$ round of computation in two phases. First phase is preprocessing round as in the basic scheme, while in the second phase $L - 1$ round of computation is follwed by postprocessing.

   - ***Pre Processing*** *(Column Control)*

Database x is assumed to be arranged in a stack of matrices as follows :

$$x \equiv \{x_i\} \to \{D_\mu(\alpha, \beta)\} \qquad \alpha t + \beta = i; 0 \le \beta < t$$

Thus there are m matrices d_μ $(1 \le \mu \le m)$. As in the bit retrieval scheme DB, forms K matrices $D_{\mu,\kappa}(\alpha, \beta)$ from the each of the original matrix $D_\mu(\alpha, \beta)$ in the stack. Again the user requires exactly one bit from each of the matrices $D_{\mu,\kappa}$'s.

- ***Iterative Computation***

  The database DB performs a L round of iterative computation according to following algorithm:

  From main: for each of the K stack of m matrices D_κ $(1 \le \kappa \le K)$ formed in the preprocessing round perform the call $DFSCompute(D_\kappa, s, t, y_1, 1)$.

  The algorithm $DFSCompute$ is described below.

  $DFSCompute(M, s, t, y_l, l)\{$

  /*

  - y_l is the l_{th} row of the matrix of numbers sent by user. Remember $y_l[i] = y(l, i)$ is a QR if $i \ne b_l$ and $y_l[i]$ is a QNR if $i = b_l$.
  - M is a stack of m matrices, where each matrix is an $s \times t$ matrix of bits and we want to retrieve the bit $M_1[a_l, b_l], \ldots, M_m[a_l, b_l]$ from these m matrices.
  - l denoted the level of the recursion.

  */

  (a) Set for $1 \le i \le s$

  $$z_\mu[i] = \nu_i \left(\prod_{j=1}^{t} (y_l[j])^{M_\mu[i,j]} \right)$$

  where ν_i for each i is a QR chosen by DB uniformly at random.

  /* Each $z_\mu[i]$ is a K-bit string. For $1 \le j \le K$, let $z_\mu[i, j]$ denote the j_{th} bit of $z[i]$. We require the strings $z_\mu[a_l]$ */

  (b) For $1 \le j \le K$, form K matrices $M_{\mu,j}$, where each $M_{\mu,j}$ is an $\frac{s}{t} \times t$ matrix formed from the column vector,

  $$z_\mu[*, j] = z_\mu[1, j], \ldots, z_\mu[s, j]$$

  by breaking $z_\mu[*, j]$ into t-bit blocks and arranging the blocks in a rowwise fashion.

  /*

  i. The string $z_\mu[a_l]$ is distributed over the K matrices $M_{\mu,j}$, i.e, the string $z_\mu[a_l]$ is equal to $M_{\mu,1}[a_{l+1}, b_{l+1}], \ldots, M_{\mu,K}[a_{l+1}, b_{l+1}]$, where $a_{l+1} = \lfloor \frac{a_l}{t} \rfloor + 1$ and $b_{l+1} = a_l \pmod t$. If $b_{l+1} = 0$, $a_{l+1} \leftarrow a_{l+1} - 1$ and $b_{l+1} = t$.

  ii. We have formed K stacks of m matrices $M_j (1 \le j \le K)$ from one stack of m matrices M.

iii. To get the bits $M_\mu[a_l, b_l]$, we need bits $M_{\mu,1}[a_{l+1}, b_{l+1}], \ldots, M_{\mu,K}[a_{l+1}, b_{l+1}]$

*/

(c) $for(1 \leq j \leq K)\{$

$if(l < L-2)$

$DFSCompute(M_j, \frac{s}{t}, t, y_{l+1}, l+1)$

$else$

$PostCompute(M_j, \frac{s}{t}, t, y_{L-1}, y_L)$

$\}$

$\}$

$PostCompute(\cdot)$ include a modified postprocessing step of our Basic SPIR Scheme. The algorithm $PostCompute(\cdot)$ is described below:

- ***Post Processing*** *(Row Control)*

$PostCompute(M, s, t, y, \gamma)\{$

/*

- As in $DFSCompute$ M is a stack of m matrices, where each matrix M_μ is an $s \times t$ matrix of bits and we want to retrieve the bits $M_\mu[a, b]$, where the index a and b is hidden in the y and γ.
- $y[j]$ $(1 \leq j \leq t)$ is an array of t numbers (L_{th} row of the matrix y sent by user). $y[j]$ is a QNR for $j = b$ else it is a QR.
- $\gamma[j]$ $(1 \leq j \leq s)$ is an array of s numbers (γ sent by user). $\gamma[j]$ is a QNR for $j = a$ else it is a QR.

*/

(a) Set for $1 \leq i \leq s$

$$z_\mu[i] = \nu_i \left(\prod_{j=1}^{t} (y[j])^{M_\mu[i,j]} \right)$$

where ν_i for each i is a QR chosen by DB uniformly at random.

/* Each $z_\mu[i]$ is a K-bit string. For $1 \leq \kappa \leq K$, let $z[i, \kappa]$ denote the κ_{th} bit of $z_\mu[i]$. We require the strings $z_\mu[a]$ $(1 \leq \mu \leq m)$.*/

(b) Set $M'_\kappa[\mu, i] = z_\mu[i, \kappa]$ for $1 \leq \mu m$, $1 \leq i \leq s$, and $1 \leq \kappa \leq K$. /* M'_κ is an $m \times s$ matrix of bits. Thus we obtain K matrices of size $m \times s$ from the stack of m matrices of size $s \times t$. From each of these K matrices we need a_{th} column.*/

(c) Set for $1 \leq \kappa \leq K$, and for $1 \leq \mu \leq m$,

$$\zeta_\kappa[i] = \nu_i \left(\prod_{j=1}^{s} (\gamma[j])^{M'_\kappa[i,j]} \right)$$

where ν_i for each i is a QR chosen by DB uniformly at random.

(d) Output the strings $\zeta_1[1], \ldots, \zeta_K[m]$.

}

4. **Reconstruction Phase and Correctness**
   Here we show that if user recieves all the numbers obtained by DB in the end of routine $PostCompute(\cdot)$ then he can reconstruct the $(a_1, b_1)_{th}$ bit of all the matrices in stack D i.e., i_{th} record of database x. This will establish the correctness of protocol as well provide the method using which user can reconstruct the i_{th} record of database x.

   (a) Suppose the call $PostCompute(M, s, t, y, \gamma)$ produces the strings $\zeta_1[1], \ldots, \zeta_K[m]$ and the QNR's of y and γ are the b_{th} and a_{th} position respectively. $\zeta_\kappa[\mu]$ is a QR iff $M'_\kappa[\mu, a]$ is 0, so it is possible to reconstruct all the bits $M'_\kappa[\mu, a]$ from which we can obtain row vectors $z_\mu[a, *] = z_\mu[a]$. Again $z_\mu[a]$ is a QR iff $M_\mu[a, b]$ is 0. thus it is possible to find $M_\mu[a, b]$. So it is possible to get the m bits at level $L - 2$. Now we use backward induction on the depth of recursion.

   (b) Suppose the call $DFSCompute(M, s, t, y_l, l)$ produces the stack of matrices $M_1, \ldots, M_K$, where the bits $M_{\mu,i}[a_{l+1}, b_{l+1}]$ are required. By induction hypothesis it is possible to get these bits. Since $z_\mu[a_l]$ is equal to $M_{\mu,1}[a_{l+1}, b_{l+1}], \ldots, M_{\mu,K}[a_{l+1}, b_{l+1}]$, it is possible to get $z_\mu[a_l]$. $z_\mu[a_l]$ is a QR iff $M_\mu[a_l, b_l]$ is 0. Thus we get the bit $M_\mu[a_l, b_l]$. Hence our hypothesis established, and user can get the $(a_1, b_1)_{th}$ bit of all the m matrices in all the K stack D_κ passed in the routines $DFSCompute(M, s, t, y_1, 1)$ from the main in DB computation provided the query matrix y sent by user had only one QNR per row at correct position. Thus user gets the bits $D_{\mu,1}[a_1, b_1], \ldots, D_{\mu,K}[a_1, b_1]$, which on concatenation gives the numbers $\psi_\mu(a_1, b_1)$ by which DB had replaced the bits of $(a_1, b_1)_{th}$ record of matrix D, i.e., $\psi_\mu(a_1, b_1)$ is a QR iff $D_\mu(a_1, b_1)$ is 0.

   Hence user can reconstruct the i_{th} record of database x from the numbers he recieves from DB.

5.4.2 Privacy of Database

While establishing the correctness who showed that, K^L K-bit number returned by DB to U is sufficient for reconstructing one record of the database D. Here we show that from those K^L numbers he can get no more than one record of database D, no matter how he send his query. As we have already discussed, user can not send any thing other than QRs and QNRs in his L set of t numbers in the query. It is also clear from the proof of correctness that K^L K-bit number returned by DB to U is necessary for reconstructing one record of the database D, and as long as he is sending only one QNR in each set of t numbers in the query, he will be able to reconstruct only one record. Thus the only way database can gain more information about database D is by sending more than one QNR in some of the sets of t numbers in his query. But then, if σ_{th} set contain m QNR's at positions $0 < \beta_1, \beta_2, \ldots, \beta_m < t$, then user will be able to reconstruct only xors of the numbers computed by DB in round $(\sigma - 1)$, provided all other set of numbers for round higher than σ in his query contained only one QNR at correct position. We have already proved that from the *xor* of more than one number one can not gain any information about the membership of the constituent numbers. Thus user fails to get any information at all. In fact he lost even the one record which he deserved. It is clear that same will be the result if he sends more than one QNRs in any set of t numbers in his query. For the completeness, consider the case, in which he send more than one QNRs in first round of his query. Then user can at most obtain ℓ xors of more than one K-bit numbers used by DB to prepare K database $D^{(\kappa)}$ in preprocessing phase, provided all other set in his query had only one QNR in each. From the xors he fails to know any thing about the quadratic nature of the constituent numbers, as already have been proved. We conclude, if user sends query as per the protocol he will get one record of the database and the moment he deviate from the protocol he fails to get any information at all about database x.

5.4.3 Privacy of User

The proof for the privacy of user in this scheme is similar to the proof we presented for the same in case of bit retrieval SPIR scheme. Again, we assume towards a contradiction that there exists a distinguisher which for some index i and i' can distinguish the distribution of queries Q_i from the distributing of queries $Q_{i'}$ with probability larger than $\frac{1}{n^m}$ for some integer m. The observation is that, in this protocol the query sent by user is a number N followed by L set of numbers from Z_N^{+1}, each set consisting of $(t-1)$ QR's and a QNR. Thus we can view this as a sequence of numbers $y_1, y_2, \ldots, y_\beta$ from Z_N^{+1}, where $\beta = L \cdot t$. From the calculation t is equal to $n^{\frac{1}{L}}$. For an index i denote by I_i the subset of $\{1, 2, \ldots, \beta\}$ of all positions that contain QNRs. By the protocol, the subset I_i is fixed and is independent of the random choices made by the user. Also for $i \neq i'$, $I_i \neq I_{i'}$ as the subset I_i uniquely determine the index i. As before, we denote by C_n the family of polynomial time circuits that distinguishes between Q_i and $Q_{i'}$. Assume $Pr(C_n(q \in Q_i) = 1) = p$ and $Pr(C_n(q \in Q_{i'}) = 1) = p + \frac{1}{n^m}$

We can construct another family of polynomial time circuit C'_n from C_n which will, on giving the input $N \in H_K$ and $y \in Z_N^{+1}$ computes the quadratic residuosity predicate with probability at least $\frac{1}{2} + \frac{1}{8 \cdot n^m}$. The construction of C'_n and proof follows exactly as in the case of bit retrieval iterative SPIR scheme.

5.4.4 Communication Complexity

Total communication from user to database DB in this scheme is $(1 + L \cdot t)$ K-bit numbers, the composite number N and the L set of t numbers from Z_N^{+1}. While database returns $m \cdot K^L$ numbers each of size K-bit, obtained at the end of $PostCompute$ round of computation. Thus the communication complexity is $(1 + L \cdot t + m \cdot K^L) \cdot K$ bit, where $t = n^{\frac{1}{L}}$, and $L = \lceil \frac{\log n}{\log m} \rceil$, and the communication complexity is :

$$C = (1 + L \cdot n^{\frac{1}{L}} + m \cdot K^L) \cdot K$$

Therefore, we proved following theorem:

THEOREM 5.4.1. *Under the Qudratic Residuosity Assumption and XOR Assumption, there exist a block retrieval SPIR protocol with communication complexity linear in number of bits in the record* m *and polynomial in security parameter* K, *i.e., we have* $O(m \cdot K^{L+1})$, *where* m *is the number of bits in the record and* $L = \lceil \frac{\log n}{\log m} \rceil$.

COROLLARY 5.4.1. *For* $n \leq m$, *i.e., number of records not more than number of bits in any record, we get* $L = 1$, *and communication complexity:*

$$\begin{aligned} C &= (1 + n + m \cdot K) \cdot K \\ &< m \cdot (K + K)K \end{aligned}$$

i.e., $C = O(mK^2)$.
For $m < n \leq m^2$), *we get* $L = 2$ *and communication complexity* $C = O(m \cdot K^3)$.
In general, $n^{\frac{1}{L}} = m$, *and* $L < K^L$, *thus communication complexity* $C = O(m \cdot K^{L+1})$.

REMARK 5.4.1. *Although we have presented our schemes based on the Quadratic Residuosity Assumption, but it could be based on more general assumption. Following the approach of Eran Mann [22], we can replace Quadratic Residuosity Predicates with any Homomorphic Trapdoor Predicates. We emphasize that Xor Assumption introduced by us is equally valid for any homomorphic trapdoor predicates.*

5.5 $O(K \cdot \log n)$ PIR and SPIR protocols

We assume the existence of a probabilistic encryption scheme $\mathcal{E}$, with two associated operations, denoted by $\textcircled{\vee}$ and $\textcircled{\wedge}$, having the following properties :

$$\mathcal{E}^{(b)}(x) \;\textcircled{\vee}\; \mathcal{E}^{(b)}(y) = \mathcal{E}^{(b)}(x \vee y)$$
$$\mathcal{E}^{(b)}(x) \;\textcircled{\wedge}\; \mathcal{E}^{(b)}(y) = \mathcal{E}^{(b)}(x \wedge y)$$

for $x \in \{0, 1\}$, where $\vee$ and $\wedge$ are, us usual, logical OR and AND operator respectively.

b is the public key, the associated private key being a and

$$\mathcal{D}^{(a)}\mathcal{E}^{(b)}(x) = x \tag{5.5.2}$$

THEOREM 5.5.1. *The number* z *which is* $\textcircled{\wedge}$ *of* d *numbers* $x_1, \ldots, x_d$, *represents encryption of* 1, *if and only if all the* d *numbers used are encryption of* 1, *otherwise it is an encryption of* 0.

Proof :

$$\begin{aligned} z &= \mathcal{E}(x_0) \;\textcircled{\wedge}\; \mathcal{E}(x_1) \;\textcircled{\wedge}\; \mathcal{E}(x_2) \;\textcircled{\wedge}\; \ldots \;\textcircled{\wedge}\; \mathcal{E}(x_{d-1}) \\ &= \mathcal{E}(x_0 \wedge x_1) \;\textcircled{\wedge}\; \mathcal{E}(x_2) \;\textcircled{\wedge}\; \ldots \;\textcircled{\wedge}\; \mathcal{E}(x_{d-1}) \end{aligned}$$

where we have used the property of the operator $\textcircled{\wedge}$. Thus, we get

$$z = \mathcal{E}(x_0 \wedge x_1 \wedge x_2 \wedge \ldots \wedge x_{d-1})$$

Now $x_0 \wedge x_1 \wedge x_2 \wedge \ldots \wedge x_{d-1}$ is 1 if and only if all the x_δ's are 1, else it is 0. ∎

THEOREM 5.5.2. *The number z which is $\circledvee$ of d numbers $x_1, \ldots, x_d$, represent encryption of* 1, *if any of the d numbers used are encryption of* 1, *it is an encryption of* 0 *if and only if all the d numbers are encryption of* 0.

Proof :

$$\begin{aligned} z &= \mathcal{E}(x_0) \circledvee \mathcal{E}(x_1) \circledvee \mathcal{E}(x_2) \circledvee \ldots \circledvee \mathcal{E}(x_{d-1}) \\ &= \mathcal{E}(x_0 \vee x_1) \circledvee \mathcal{E}(x_2) \circledvee \ldots \circledvee \mathcal{E}(x_{d-1}) \end{aligned}$$

where we have used the property of the operator $\circledwedge$. Thus, we get

$$z = \mathcal{E}(x_0 \vee x_1 \vee x_2 \vee \ldots \vee x_{d-1})$$

Now $x_0 \vee x_1 \vee x_2 \vee \ldots \vee x_{d-1}$ is 1 if and only if all the x_δ's are 1, else it is 0. ∎

Under the assumption of the existence of such encryption scheme, we first present an $O(K \cdot \log n)$ PIR protocol and then we convert this protocol into a SPIR protocol by introducing some interesting extra computation for DB, for which we need following theorem.

THEOREM 5.5.3. *If there exist a probabilistic encryption scheme $\mathcal{E}$, with two associated operators $\circledvee$ and $\circledwedge$ such that*

$$\begin{aligned} \mathcal{E}(x) \circledvee \mathcal{E}(y) = \mathcal{E}(x \vee y) \\ \mathcal{E}(x) \circledwedge \mathcal{E}(y) = \mathcal{E}(x \wedge y) \end{aligned}$$

Then there also exists a probabilistic encryption scheme $\mathcal{E}'$ in which any logical operation can be carried out on the encryption of 0 *and* 1 *without decrypting them. Specifically, there exists an operator $\circledplus$ such that*

$$\mathcal{E}(x) \circledplus \mathcal{E}(y) = \mathcal{E}(x \oplus y)$$

Proof : : Consider a probabilistic encryption scheme $\mathcal{E}'$ built using probabilistc encryption scheme $\mathcal{E}$. Rules of encryption for $\mathcal{E}'$ are :

$$\begin{aligned} \mathcal{E}'(0) = (\mathcal{E}(0),\ \mathcal{E}(1)) \\ \mathcal{E}'(1) = (\mathcal{E}(1),\ \mathcal{E}(0)) \end{aligned}$$

Let $x, y \in \{0, 1\}$, and by $\overline{x}$ we mean complement of x. Then define
$NOT(\mathcal{E}(x),\ \mathcal{E}(y)) = (\mathcal{E}(y),\ \mathcal{E}(x))$.
We have following result:

$$\begin{aligned} \mathcal{E}'(\overline{x}) &= NOT\mathcal{E}'(x) \\ &= NOT(\mathcal{E}(x),\ \mathcal{E}(\overline{x})) \\ &= (\mathcal{E}(\overline{x}),\ \mathcal{E}(x)) \end{aligned}$$

Thus we can successfully carry out NOT operation over the encrypted values under this new encryption scheme $\mathcal{E}'$ without knowing the decryption. Now we show that operators $\textcircled{\vee}$ and $\textcircled{\wedge}$ associated with $\mathcal{E}$ will be available in $\mathcal{E}$ as well. Here we show the rules for evaluating $\textcircled{\vee}$ and $\textcircled{\wedge}$ under the encryption scheme $\mathcal{E}'$

$$\begin{aligned}
\mathcal{E}'(x) \ \textcircled{\vee}\ \mathcal{E}'(y) &= (\mathcal{E}(x),\ \mathcal{E}(\overline{x}))\ \textcircled{\vee}\ (\mathcal{E}(y),\ \mathcal{E}(\overline{y}))\\
&= (\mathcal{E}(x)\ \textcircled{\vee}\ \mathcal{E}(y),\ \mathcal{E}(\overline{x})\ \textcircled{\wedge}\ \mathcal{E}(\overline{y}))\\
&= (\mathcal{E}(x \vee y),\ \mathcal{E}(\overline{x} \wedge \overline{y}))\\
&= (\mathcal{E}(x \vee y),\ \mathcal{E}(\overline{x \vee y}))\\
&= \mathcal{E}'(x \vee y)
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{E}'(x) \ \textcircled{\wedge}\ \mathcal{E}'(y) &= (\mathcal{E}(x),\ \mathcal{E}(\overline{x}))\ \textcircled{\wedge}\ (\mathcal{E}(y),\ \mathcal{E}(\overline{y}))\\
&= (\mathcal{E}(x)\ \textcircled{\wedge}\ \mathcal{E}(y),\ \mathcal{E}(\overline{x})\ \textcircled{\vee}\ \mathcal{E}(\overline{y}))\\
&= (\mathcal{E}(x \wedge y),\ \mathcal{E}(\overline{x} \vee \overline{y}))\\
&= (\mathcal{E}(x \wedge y),\ \mathcal{E}(\overline{x \wedge y}))\\
&= \mathcal{E}'(x \wedge y)
\end{aligned}$$

and finally

$$\begin{aligned}
\mathcal{E}'(x)\ \textcircled{\oplus}\ \mathcal{E}'(y) &= \mathcal{E}'(x)\ \textcircled{\wedge}\ NOT(\mathcal{E}'(y)\ ))\ \textcircled{\vee}\ (NOT(\mathcal{E}'(x))\ \textcircled{\wedge}\ \mathcal{E}'(y)\\
&= \mathcal{E}'(x)\ \textcircled{\wedge}\ \mathcal{E}'(\overline{y}))\ \textcircled{\vee}\ (\mathcal{E}'(\overline{x})\ \textcircled{\wedge}\ \mathcal{E}'(y)\\
&= \mathcal{E}'(x \wedge \overline{y})\ \textcircled{\vee}\ (\mathcal{E}'(\overline{x} \wedge y)\\
&= \mathcal{E}'(\ (x \wedge \overline{y})) \vee (\overline{x} \wedge y)\ )\\
&= \mathcal{E}'(x \oplus y)
\end{aligned}$$

■

In following sections, by x is a $\mathcal{E}_0$ we mean $x \in \mathcal{E}_0$ and similarly by x is a $\mathcal{E}_1$ we mean $x \in \mathcal{E}_1$. Also we assume a function $Expand : \{1, \ldots, t^d\} \rightarrow \{1, \ldots, t\}^d$. Thus on giving input i to function expand, it returns a d tuple. There is a one-to-one mapping between i and its d tuple representation returned by $Expand(\cdot)$.

5.5.1 The $O(K \cdot \log n)$ PIR Scheme

1. **Setup**

   Database x is modelled as a d dimensional array D. We can imagine a d dimensional grid in which each of the crosspoint is holding one bit of data. Each side of the grid is considered to be of equal size, say having t crosspoint. Thus this grid or the d dimensional array can hold t^d bits. Take number of bits in database $n = t^d$. Thus fixing n and d determine t, as $t = n^{\frac{1}{d}}$.

2. **Query Generation**
   Let user is interested in i_{th} bit of the database x. Then the query generation process is as follows :

   (a) Express i in radix t to get a d-radit number :
   $(i_d, i_{d-1}, \ldots, i_0) \leftarrow Expand(i)$
   where i_δ $(0 \leq \delta < d) \in \{1, \ldots, t\}$.

   (b) create a pair of private and public key a and b following the protocol of underlying encryption scheme.
   Send the public key b to database, while keep private key a secret.

   (c) Generate d ordered set of t numbers, each set having $(t-1)$ $\mathcal{E}_0$'s, numbers encrypting 0 and one $\mathcal{E}_1$, the encryption of 1. The position of $\mathcal{E}_1$ in set δ is i_δ. User send these d sets of t numbers $y(\delta, \tau)$ $1 \leq \delta \leq d$ and $1 \leq \tau \leq t$, to database.

3. **Database Computation**
   The DB performs following operations after receiving the query comprising the public key of user b and the d set of t numbers $y(\delta, \tau)$. from user:

$$\begin{cases} \textbf{for } j \leftarrow 1 \textbf{ to } n \\ \quad \textbf{do} \begin{cases} \textbf{if } (x_j = 1) \\ \quad \textbf{then} \begin{cases} (j_d, j_{d-2}, \ldots, j_1) \leftarrow Expand(j) \\ z_j \leftarrow y(d, j_d) \; \textcircled{\small$\wedge$} \; y(d-1, j_{d-1}) \; \textcircled{\small$\wedge$} \; \ldots \; \textcircled{\small$\wedge$} \; y(1, j_1) \end{cases} \end{cases} \\ z \leftarrow \; \textcircled{\small$\vee$}_{\,(1 \leq j \leq n : x_j = 1)} z_j \\ \textbf{return } (z) \end{cases}$$

   The DB returns a single number z to user.

4. **Reconstruction**
   User receives a single number z in reply from database and decrypt it using his private key. The decrypted value is the i_{th} bit of database x.

$$x_i \leftarrow D^{(a)}(z) \tag{5.5.3}$$

5.5.2 Correctness

First we prove following theorem:

THEOREM 5.5.4. *Number z returned by DB is an encrption of* 1 *iff $x_i = 1$, where i is such that its expansion tuple $(i_d, \ldots, i_1)$ represent the positions of $\mathcal{E}_1$ in d rows of query matrix y sent by user.*

Proof : The number z_j is an encryption of 1 only if all of the number $y(d, j_d), \ldots, y(1, j_1)$ are $\mathcal{E}_1$, this follows from the Theorem 5.5.1.. Now the $one-to-one$ relationship between $(j_d, j_{d-1}, \ldots, j_1)$

and j means there is only one index i for which all the numbers $y(d, j_d), \ldots, y(1, j_1)$ are in $\mathcal{E}_1$, provided there was only one $\mathcal{E}_1$ in each row of matrix y sent by user. Thus only z_i is $\mathcal{E}_1$ all other z_j's are $\mathcal{E}_0$'s. Now the number z which is $\textcircled{v}$ of z_j's corresponding to which x_j is 1. Thus z_i is included in the $\textcircled{v}$ iff $x_i = 1$. Thus by Theorem 5.5.2., z is $\mathcal{E}_1$ iff $x_i = 1$. ∎

Above theorem establishes the correctness of protocol.

5.5.3 Privacy of user

We prove the privacy of user, by contradiction. Assume, that there exists a distinguisher, a family of polynomial time circuits C_n, which for some index i and i' can distinguish the distribution of queries Q_i from the distributing of queries $Q_{i'}$ with probability larger than $\frac{1}{n^\ell}$ for some integer ℓ. The query sent by user consists of a number b followed by d ordered set of t numbers, each set consisting $t-1$ $\mathcal{E}_0$'s and one $\mathcal{E}_1$. Thus we can view this as a sequence of numbers b followed by $y_0, y_1, \ldots, y_{\beta-1}$ from $\mathcal{E}_0 \bigcup \mathcal{E}_1$, where $\beta = d \cdot t$. From the calculation t is equal to $n^{\frac{1}{d}}$. For an index i denote by I_i the subset of $\{0, 1, 2, \ldots, \beta - 1\}$ of all positions occupied by $\mathcal{E}_1$s. By the protocol, the subset I_i is fixed and is independent of the random choices made by the user. Also for $i \neq i'$, $I_i \neq I_{i'}$ as the subset I_i uniquely determine the index i. Assume $Pr(C_n(q \in Q_i) = 1) = p$ and $Pr(C_n(q \in Q_{i'}) = 1) = p + \frac{1}{n^\ell}$

We can construct another family of polynomial time circuit C'_n from C_n which will, on giving the input b the public key, corresponding to the private key held by user and $y \in \mathcal{E}_0 \bigcup \mathcal{E}_1$ decrypt y with probability at least $\frac{1}{2} + \frac{1}{8 \cdot n^\ell}$. The construction of C'_n follows :

1. Construct a sequence as expected by the circuit C_n. Pick at random one of the subsets I_i and $I_{i'}$. In every position which is not in the chosen set put a random $\mathcal{E}_0$ from $\mathcal{E}_1$. In every position which is in the chosen subset put the $\textcircled{v}$ of y and a random $\mathcal{E}_0$.

2. With probability $\frac{1}{4 \cdot n^\ell}$ answer 0 and stop.

3. Run C_n on the input constructed in (1). If the chosen subset was $I_{i'}$ output the same answer as C_n; If the chosen subset was I_i flip C_n's output.

Let us now calculate probability that C'_n outputs 1 on input b, y such that y is an encryption of 0 with public key b. The observation is that, in this case no matter which subset was chosen in step(1), the input to C_n is a sequence of random $\mathcal{E}_0$'s. There is a probability q that C_n output 1 on such input. Then the probability that C'_n output 1 in this case is $(1 - \frac{1}{4 \cdot n^\ell})(\frac{1}{2}q + \frac{1}{2}(1-q)) = \frac{1}{2} + \frac{1}{8 \cdot n^\ell}$

Now, we compute the probability that C'_n outputs 1 on input b, y such that y is an encryption of 1 with public key b. By assumptions on C_n, the probability that C'_n output 1 is :

$$\begin{aligned}
(1 - \frac{1}{4 \cdot n^\ell})(\frac{1}{2}(p + \frac{1}{n^\ell}) + \frac{1}{2}(1-p)) &= (1 - \frac{1}{4 \cdot n^\ell})(1 + \frac{1}{2 \cdot n^\ell}) \\
&= \frac{1}{2} + \frac{1}{4 \cdot n^\ell} - \frac{1}{8 \cdot n^{2l}}) \\
&\geq \frac{1}{2} + \frac{1}{8 \cdot n^\ell}
\end{aligned}$$

So, indeed if any family of polynomial-size circuits breaks the privacy of this PIR scheme, then we can construct a family of polynomial time circuit, which can decrypt with significant probability away from half, given any encrypted number and only the public key.

5.5.4 Communication complexity

Total communication from user to database DB in this scheme is $(1 + d \cdot t)$ numbers, the public key b and the d set of t numbers from $\mathcal{E}_0 \bigcup \mathcal{E}_1$, each K-bit in size. While database returns one K-bit numbers obtained at the end of computation. Thus the communication complexity is $(1+d{\cdot}t+1){\cdot}K$ bit, where $t = n^{\frac{1}{d}}$, and the communication complexity is :

$$(1 + d \cdot n^{\frac{1}{d}} + 1) \cdot K$$

The Communication complexity is minimized by taking $d = \log_e n$ *(the natural base to the log).* But we must have d an integer for our purpose. For t restricted to integer domain, we get minimum complexity by choosing base 3 to the log i.e; $d = \log_3 n$, then $t = n^{\frac{1}{d}} = 3$the communication complexity becomes

$$C = (1 + 3 \cdot \log_3 n + 1) \cdot K = (2 + 1.88 \cdot \log_2 n) \cdot K \qquad (5.5.4)$$

By choosing $d = \log_2 n$, we would have got

$$C = (2 + 2 \cdot \log_2 n) \cdot K \qquad (5.5.5)$$

5.5.5 An $O(K \cdot \log n + K^2)$ SPIR protocol from above PIR protocol

Intordeuce a preprocessing computation before the computation performed by DB in above PIR scheme. In this preprocessing computation DB replaces each of the bit of databse x by $\mathcal{E}_0$ if the bit is 0 and by $\mathcal{E}_0$ if it is 1. Then he forms K databases $D_1, \dots, D_K$, where D_j is formed from the j_{th} bit of $\mathcal{E}_0$ and $\mathcal{E}_1$ used by DB. Thus to retrieve i_{th} bit of original database user needs i_{th} bit of all the K database so that he can obtain the number used by DB to replace the i_{th} of database x. Then by decrypting that number user gets x_i. It is clear from the PIR protocol described above that if user send more than one $\mathcal{E}_1$ in any of the row of his query matrix y he will get **OR**'s of several bits (having index whose tuple representation corresponds to the sequence, corresponding to which numbers in the query matrix are all $\mathcal{E}_1$). Now DB applies computation of PIR scheme over all the K databases prepared by him and return K numbers to user. User still needs to send only one query matrix, as same query matrix can be applied to all the K databases D_j, as user needs bit with same index from all of them. Now if user try to send more than ane $\mathcal{E}_1$ in any row of his query matrix he will obtains a bit wise **OR** of encrypted numbers and by decrypting that user fails to get anything useful. Thus we achieves a SPIR protocol. Here the communication complexity is

$$C = (1 + d \cdot n^{\frac{1}{d}} + K) \cdot K$$

By taking $d = 2$ we get $C = (K + 1 + 2 \log n) \cdot K$.

Now we describe anothe method of obtaining $SPIR$ protocol from above PIR protocol.

5.5.6 An $O(K \cdot \log n)$ SPIR protocol from above PIR protocol

A careful look into the computation of DB reveals that if user is honest he will get only one physical bit of database. Further we assume that user can send numbers in his query only from the set $\mathcal{E}_0 \bigcup \mathcal{E}_1$. This assumption is based on the fact that DB can always choose random $\mathcal{E}_0$s and $\ovee$ them with the numbers in user's query, thereby randomising the number sent by user, at the same because of the property of operator $\ovee$ an $\mathcal{E}_0$ will remain $\mathcal{E}_0$ and so does $\mathcal{E}_1$, and if user is sending some number $\notin \mathcal{E}_0 \bigcup \mathcal{E}_1$, the result of DB computation will be unpredictable for user.

Theorem 5.5.3. asserts that DB can compute $\bigoplus$ of two encrypted number, the operation $\bigoplus$ is defined as follows :

$$z = \mathcal{E}(x) \bigoplus \mathcal{E}(y) = \mathcal{E}(x \oplus y) \tag{5.5.6}$$

Thus we assume that only way user can gain more information from the DB is by sending more than one $\mathcal{E}_1$ in any set, (as long as he sends only one $\mathcal{E}_1$ per set he will get only one bit). The construction of SPIR protocol is as follows :
Take $d = \log_2 n$ in the PIR protocol described in previous section, then $t = 2$, i.e; each of the d set of numbers sent by user consists of 2 numbers only.

- **Query Generation**
  User use the query generation algorithm of PIR scheme. Generate the query and send to DB.

- DB **Computation**
  DB uses the query sent by user and apply the Computation algorithm of the PIR scheme. Thus DB computes a number Z. In place of sending the number Z, DB does following post processing computation :

$$\begin{cases} z \leftarrow DBcompPIR(query) \\ z' \leftarrow z\textbf{for } \delta \leftarrow 0 \textbf{ to } (d-1) \\ \quad \textbf{do} \begin{cases} r \leftarrow rand\{0,1\} \\ \textbf{if } (r = 1) \\ \quad \textbf{then} \begin{cases} y \leftarrow y_{\delta,0} \owedge y_{\delta,1} \\ z' \leftarrow z' \bigoplus y \end{cases} \end{cases} \\ \textbf{return } (z') \end{cases}$$

  DB returns z' to user.

- **Reconstruction**
  User on receiving reply from DB, decrypt it to get desired bit as in PIR scheme.

Thus the only change is post processing computation by DB. We analyse the effect of this extra computation, in two case. First assume that user's query had only one $\mathcal{E}_1$ in each of the d set of numbers, and the query correspond to index i. Then $z = \mathcal{E}(x_i)$ i.e; z is the encryption of i_{th} bit of database x. Also for any δ, $y = y_{\delta,0} \bigotimes y_{\delta,1}$ is $\mathcal{E}_0$ because one of $y_{\delta,0}$ and $y_{\delta,0}$ is $\mathcal{E}_0$ and other one is $\mathcal{E}_1$. Then z' and $z' \bigoplus y$ are both $\mathcal{E}_{x_i}$. Thus user obtain x_i. Hence the **correctness** of protocol established.

Consider the case in which, query sent by user contain both the number from the set $\mathcal{E}_0$. Then the number z, returned by $DBcompPIR$ is $\mathcal{E}_0$ regardless of the database content. Thus in this case user gains nothing.

Now consider the case in which, query sent by user contain d' set out of d set having both the number from set $\mathcal{E}_1$. If any of the set having both number from $\mathcal{E}_1$ is get selected by DB in post processing computation, then the number y obtained for this set is an $\mathcal{E}_1$ and $\bigoplus$ ing y with z' complements the membership of z' i.e; if z' was an $\mathcal{E}_0$ then it becomes $\mathcal{E}_1$ and vice-versa. Let d'' number of set from d' set having both number from $\mathcal{E}_1$ is been selected by DB in post processing computation. Then

$$Pr(z' \in \mathcal{E}_0 | z \in \mathcal{E}_0) = Pr(z' \in \mathcal{E}_1 | z \in \mathcal{E}_1) = Pr(d'' \text{ is even})$$
$$Pr(z' \in \mathcal{E}_0 | z \in \mathcal{E}_1) = Pr(z' \in \mathcal{E}_1 | z \in \mathcal{E}_0) = Pr(d'' \text{ is odd })$$

As each set have probability of being selected is $\frac{1}{2}$. Thus $Pr(d''iseven) = Pr(d''isodd) = \frac{1}{2}$. Which implys

$$\begin{aligned} Pr(z' \in \mathcal{E}_0 | z \in \mathcal{E}_0) &= Pr(z' \in \mathcal{E}_1 | z \in \mathcal{E}_0) \\ &= Pr(z' \in \mathcal{E}_0 | z \in \mathcal{E}_1) \\ &= Pr(z' \in \mathcal{E}_1 | z \in \mathcal{E}_0) \end{aligned}$$

Thus the **Privacy of database** established.

As user is sending same query as he was in underlying PIR scheme, thus his privacy is implied by the **Privacy of User** provide by PIR protocol.

Communication from user to DB and DB to user is same as in the PIR protocol.

REMARK 5.5.1. *We have described above SPIR protocol for $t = 2$, it can easily be generalized for any value of t. The idea is that for each row in the query matrix sent by user we can form $\binom{t}{2}$ pairs and compute $\bigotimes$ of both number in each pair. In total from d rows we get $d \times \binom{t}{2}$ such numbers, say y_j $(1 \leq j \leq d \cdot \binom{t}{2})$. If no row contain more than one $\mathcal{E}_1$ then all these numbers are $\mathcal{E}_0$ otherwise some of them are $\mathcal{E}_1$. Now for number y_1, we choose at random 0 or 1, and $\bigoplus$ it with z (number computed by DB by applying users query on database x) and obtain a new z, say z'. He repeats same process for all the y_j's taking z' obtained with previous number as z for $\bigoplus$ ing. Thus we can get SPIR protocol for any value of t with the communication complexity same as PIR protocol.*

Chapter 6

Private Secret Retrieval and Sharing by Group

In this chapter we present an interesting application of SPIR protocol. Here we describe an generalized secret sharing scheme in which we get rid of a central weakness of all the secret sharing scheme presently known. In any known secret sharing scheme, a trusted authority (TA) is assumed to be present and he is suppose to distribute the secret to different user. Thus TA always knows the secret. We combine the SPIR scheme with secret sharing scheme to get rid of the assumption about the honesty of TA.

Let there is a group G of m users u_1,u_2, $\ldots$, u_m. We also assume that there exists a public encryption scheme $\mathcal{E}$, in which the encryption and decryption operators are commutative. that is :

$$\mathcal{D}_K(\mathcal{E}_K(x)) = \mathcal{E}_K(\mathcal{D}_K(x)) = x \tag{6.0.1}$$

For each user $u - j$, d_j is his private Key known only to him, while the corresponding public key e_j is known to all including the DB, who in our scheme replace the TA.

We also assume the existence of a secret sharing scheme.

6.1 The Scheme

DB is assumed to have n numbers of secrets, the index of secret is assumed to be known to everyone.

- **Registration of Group**
  m users (u_1,u_2, $\ldots$, u_m), intended to jointly share some secrets, forms a group G, and select u_1 as group leader. The information about the access structure Γ and the ordering of users is provided to DB.

- **Agreement on Index**
  When any user u_μ feels the need of having the i_{th} secret from the DB, he informs the group leader u_1, about the index i of the secret. User u_1 generate the necessary input to the query

generation algorithm and communicate it to u_2, all other user u_j, on receiving such request from u_{j-1}, if agree with it pass it on to next user u_{j+1}, except the last user of the group. If u_m also agree with it, then they go into next step.

Algorithm 6.1.1: INDEXAGREEMENT($index$)

u_1 : receives request for retrieving secret s_i from u_μ;
if (u_1 agrees)
  then
    comment: u_1 performs
    $u_1 : r_0 \leftarrow QueryInput(Index)$;
    $u_1 : r_1 \leftarrow E_{e_2}(ir_0)$;
    u_1 : sends r_1 to u_2;
    comment: ir_0 is the concatenation of index i, and input r_0
for $j \leftarrow 2$ **to** $m-1$
  do
    u_j : receives r_{j-1} from u_{j-1};
    if (u_j agrees)
      then
        $u_j : ir_0 \leftarrow \mathcal{D}_{d_j}(r_{j-1})$;
        u_j : Note down i and r_0;
        $u_j : r_j \leftarrow E_{e_{j+1}}(ir_0)$;
        u_j : sends r_j to u_{j+1};
u_m : receives r_{m-1} from u_{m-1};
if (u_m agrees)
  then
    $u_m : ir_0 \leftarrow \mathcal{D}_{d_m}(r_{m-1})$;
    u_m : Note down i and r_0;

- **Query Generation**

  1. User u_m use the input received by him in previous step, as parameters to the query generation algorithm of SPIR scheme and generate the query for retrieving the i_{th} secret s_i.
  2. u_m first encrypt the query using his *private key* and then encrypt it using public key of u_{m-1} and then send it to user u_{m-1}. Any user u_j $(2 \leq j \leq m-1)$ on receiving the encrypted query decrypt it and verify whether it is same as the query generated by him using input available with him. Rest of the protocol will be clear from the query generation algorithm below.

Algorithm 6.1.2: GENERATEQUERY(i, r_0)

$$
\begin{cases}
u_m: \text{ When all agree for retrieving } s_i, \text{ get } i \text{ and } r_0;\\
\textbf{comment: } u_m \text{ computes}\\
u_m : q_0 \leftarrow QuerySPIR(i, r_0);\\
\textbf{comment: } \text{encryption is being done using private key } u_m : q_m \leftarrow \mathcal{E}_{d_m}(q_0);\\
u_m : q'_m \leftarrow \mathcal{E}_{e_{m-1}}(q_m);\\
u_m : \text{ sends } q'_m \text{ to } u_{m-1};\\
\textbf{for } j \leftarrow m-1 \textbf{ downto } 2\\
\quad \textbf{do} \begin{cases}
u_j : \text{ receives } q'_{j+1} \text{ from } u_{j+1};\\
u_j : q_{j+1} \leftarrow \mathcal{D}_{d_j}(q'_{j+1});\\
u_j : \text{From } q_{j+1} \text{ repeated decryption in correct order using}\\
\text{the public keys of } u_{j+1}, \ldots, u_m \text{ extracts } q_0;\\
u_j : \text{using index and input available to him in agreement round generate}\\
q_0 \text{ and verifies it againest the extracted } q_0;\\
u_j : q_j \leftarrow \mathcal{E}_{d_j}(q_{j+1});\\
u_j : q'_j \leftarrow \mathcal{E}_{e_{j-1}}(q_j);\\
u_j : \text{ sends } q'_j \text{ to } u_{j-1};
\end{cases}\\
u_1 : \text{ receives } q'_2 \text{ from } u_2;\\
u_1 : q_2 \leftarrow \mathcal{D}_{d_1}(q'_2);\\
u_1 : \text{From } q_2 \text{repeated decryption in correct order using}\\
\text{the public keys of } u_2, \ldots, u_m \text{ extracts } q_0;\\
u_1 : \text{using index and input available to him in agreement round generate}\\
q_0 \text{ and verifies it againest the extracted } q_0;\\
u_1 : q_1 \leftarrow \mathcal{E}_{d_j}(q_{j+1});
\end{cases}
$$

- **Query transmission to DB**

  After completion of query generation round, u_1 has with him q_1 which has been encrypted by the private keys of $u_1 \ldots, u_m$ in reverse order. u_1 send q_1 to DB

- **Query decryption by DB**

  DB decrypt q_1, to get the query q_0 intended for SPIR scheme, as follows :

$$q_0 \leftarrow \mathcal{D}_{e_m}\mathcal{D}_{e_{m-1}} \ldots \mathcal{D}_{e_1}(q_1); \tag{6.1.2}$$

- **Computation on database by DB**

  DB uses the query q_0 extracted by him as input to $DB_c ompSPIR$ algorithm and obtain the answer A.

$$A \leftarrow DBcompSPIR(q_0); \tag{6.1.3}$$

- **Partition of Secret by DB**

  DB uses the answer A obtained in previous step along with the access structure Γ supplied by the group during registration as input to the secret partition algorithm $Partition()$ of secret sharing scheme. Thereby, he obtains m partitions of secret one for each user in the group.

$$\{A_1, A_2, \ldots, A_m\} \leftarrow Partition(A, \Gamma); \tag{6.1.4}$$

- **Partial Secret Transmission by DB**

  DB encrypt the share of each user using users public key and send it to him.

$$\textbf{for } j \leftarrow 1 \textbf{ to } m \quad \textbf{do} \begin{cases} A'_j \leftarrow \mathcal{E}_{e_j}(A_j); \\ \text{send } A'_j \text{ to } u_j; \end{cases}$$

- **Users gets their share**

  When user u_j receives his share from DB, he just decrypt it, using his private key. The decrypted value A_j is his share.

- **Computation over retrieved secret**

  If a subset of user from group G wants to reconstruct the secret they have, first they follow the reconstruction algorithm of secret sharing scheme. Then they get the answer A computed by DB provided the set of user qualifies the criteria of access structure. Then the next step is to use the reconstruction algorithm of SPIR scheme to reconstruct the secret s_i. After this they can compute any function over this.

Chapter 7

Summary

In this chapter first we list some of the possible applications of protocols developed by us. Then we go on to discuss some open problems that arises from our work, and finally present our conclusion.

7.1 Applications

7.1.1 Pay-Per-Access Database with Private Queries

This is a very straight forward application. User can register himself to a site which is having some n type of information's which may be updated on regular basis. For regularly retrieving one particular type of information he can just send query once at the time of registration and then can be accessed regularly, maintaining the privacy of database and user both.

7.1.2 Sending Gift

Let U_1 wants to give a gift to U_2, and DB has n gifts, which we assume are indexed and transferable over Internet. U_1 generate proper query for i_{th} gift according to SPIR protocol and send it to DB along with the address of gift receiver. U_1 also sends the associated trapdoor information needed to decrypt the answer send by DB, to U_2, after encrypting it with U_2's public key. DB send answer to U_2. U_2 use the trapdoor information sent by U_1 and recover his gift from the answer sent by DB to him.

7.1.3 Selling of Physical Objects Obliviously

We assume that there exists a warehouse with n_0 objects locked in different immovable enclosures having number keys. Let c_i be the cost of i_{th} object, we assume all c_i's to be integers, then let α be the largest number such that, for all i $\alpha \mid c_i$. Let $\sigma_i = \frac{c_i}{\alpha}$. Assume c to be some security parameter. Then put the i_{th} object in an enclosure which has $c \cdot \sigma_i$ bits in its number key. DB stores all the keys in a database, keys are assumed to be stored sequentially, we can consider each record c bit long, and a key number with $c \cdot \sigma$-bits needs σ many records in database. Number of records in database are $n = \sum_{i=1}^{n_0} \sigma_i$. Now user wants to purchase j_{th} object he can retrieve records related to the keys of j_{th} object from the database, for which he needs to retrieve σ_j many records, starting from $\sum_{i=1}^{j-1} \sigma_i + 1$ upto $\sum_{i=1}^{j} \sigma_i$. Payments can be done online as number of records retrieve is directly proportional to the cost of the object. The object itself can be retrieved by user anytime from the warehouse as numberkeys for that object is with him.

7.1.4 Some More Applications

- Implementation of multiplayer mental games, for reference see [24].
- Electronic Voting, the basic idea is to use an SPIR protocol to distribute "eligibility tokens", which are later used to prove one's right to vote [39]
- Asymmetric traiter tracing. The user can use the SPIR protocol to get a part of his key, say the half, while the other half would be chosen and kept by the database along with the query string. Tracing with be performed by comparing a set of recovered keys with the known halves, trial can be achieved by exhibiting the recovered key and the query string. An innocent user can prove himself innocent by revealing his cleartexts to show they do not match the recovered key. Details about these families of protocols, resistance against collusions, and good choices for the sets of keys can be found in [8, 5].
- For fingerprinting purposes. This is similar to above application, but in place of sending back the the selected secret (Chosen by user), the database insert the encrypted secret inside the data to be fingerprinted, using the homomorphic properties of the encryption function and send this data to the user. Beihl et al. [4] have given the details about the existence of collusion-secure asymmetric fingerprinting with the help of "Committed" ANDOS protocol, which means that the user has to commit to the secret he is willing to buy at the beginning of the protocol, which is the case in our scheme.

7.2 Open Problems

Perhaps the most important open problem is, is to understand the place of XOR Assumption, say with respect to Quadratic Residuosity Assumption. We conjecture that XOR of two numbers from the set of QR's and QNR's do not provide any information at all about the membership of constituent numbers. In other words privacy for database in our scheme is information theoretic, but this has to solved firmly and formally. We also presented generalized XOR Assumption, but the verification was done for only quadratic residues. It has to be seen whether it is valid for all probabilistic encryption schemes.

Equally important open problem is to find a probabilistic encryption system in which one can define rules for evaluating logical AND and OR operations, without decrypting the logical value. We point out that for homomorphic trapdoor predicates we can either define rules for obtaining XOR and NOT (Additive f-Homomorphic Trapdoor Predicates) or we can define rules for AND

or OR (Additive f-Homomorphic Trapdoor Predicate). But we need both AND and OR logical operations at the same time.

Another interesting question is related to the new primitive introduced by us. The place the primitive, Private Secret Retrieval and Sharing by Group (PSRSG) problem, is not very clear, it has to be understood more clearly, and other application it may offer is to be uncovered.

7.3 Conclusion

The primary contribution of this work is that we have successfully presented single-server bit and block retrieval SPIR protocols, in which query has to be send in the beginning only and database returns answer only once. We emphasize that verification of query is no longer required in our schemes. The fallout of this work is a new number theoretic assumption, XOR Assumption, which may lead to many new cryptographic protocols.

Also if one can find an probabilistic encryption scheme with certain associated operations, then we have provided SPIR scheme which are essentially optimal in terms of communication complexity.

We have also introduced a new primitive, Private Secret Retrieval and sharing by Group Problem, this may also be seen as extension of secret sharing schemes. We hope this will be useful in various cryptographic scenario.

Finally, in course of this work, we have proved certain number theoretic properties related to quadratic residues and we have also proved certain theorems on the security of probabilistic encryption.

Appendix A

Theorem A.0.1. *Let $(\Omega, \mathcal{S}, Pr)$ be a probability space, and let $\{H_i\}$ be a disjoint sequence of events such that $Pr(H_i) > 0$ $(i = 1, 2, \ldots)$ and $\sum_{i=1}^{\infty} H_i = \Omega$. Let $A \in \mathcal{S}$ with $Pr(A) > 0$, then*

$$\begin{aligned} Pr(H_i|A) &= \frac{Pr(H_i \bigcap A)}{Pr(A)} \\ &= \frac{Pr(A|H_i)Pr(H_i)}{\sum_{i=0}^{\infty} Pr(H_i)Pr(A|H_i)} \end{aligned}$$

Corollary A.0.1. *Let $A, B \in \mathcal{S}$ with $Pr(A)Pr(B) > 0$, $Pr(A \bigcap B) = 0$ and $Pr(A \bigcup B) = 1$, i.e., A and B are two disjoint event and together makes the whole sample space. Then*

$$Pr(H_i|A \cup B) = Pr(H_i|A)Pr(A) + Pr(H_i|B)P(B)$$

Proof :

$$\begin{aligned} Pr(H_i|A \cup B) &= \frac{Pr(A \cup B|H_i)Pr(H_i)}{Pr(A \cup B)} \\ &= Pr((A \cup B) \cap H_i) \\ &= Pr((A \cap H_i) \cup (B \cup H_i)) \\ &= Pr(A \cap H_i) + Pr(B \cap H_i) \\ &= Pr(H_i|A)Pr(A) + Pr(H_i|B)Pr(B) \end{aligned}$$

Definition A.0.1. **Binomial Distribution**
We say that a discrete random variable X has a binomial distribution with parameter p if its probability mass function is given by

$$p(i) = Pr(X = i) = \binom{t}{i}\ p^i\ (1-p)^{t-i}$$

where $i = 0, 1, 2, \ldots, t$; and $0 \le p \le 1$.

Remark A.0.1. *If X is a binomially distributed random variable with parameter p it is written as $X \sim b(1, p)$. For such a X we have*
Expectation of X:

$$E(X) = \mu = t \cdot p$$

Variance of X:

$$Var(X) = \sigma^2 = t \cdot p \cdot q$$

where $q = 1 - p$.

DEFINITION A.0.2. *A random variable X is said to have a normal distribution with parameters $\mu(-\infty < \mu < \infty)$ and $\sigma(> 0)$ if its probability distribution function is given by*

$$\Phi(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\; e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

where $-\infty < x < \infty$; $\sigma > 0$; $-\infty < \mu < \infty$.

REMARK A.0.2. *If X is a normally distributed random variable with parameters μ and σ, it is written as $X \sim \mathcal{N}(\mu, \sigma^2)$. For $X \sim \mathcal{N}(\mu, \sigma^2)$*

$$Pr(\mu - 3 \cdot \sigma \leq X \leq \mu + 3 \cdot \sigma) \geq 0.99$$

EXAMPLE A.0.1. *Let $X_1, X_2, \ldots, X_m$ be identically independently distributed $b(1, p)$ random variables. Then $S_m = \sum_{i=0}^{m} X_i$ is asymptotically $\mathcal{N}(mp, mp(1-p))$. Thus for large enough m we can estimate $Pr(S_m \leq x)$ by*

$$Pr\left(\frac{S_m - mp}{\sqrt{mp(1-p)}} \leq \frac{x - mp}{\sqrt{mp(1-p)}}\right) \approx \Phi\left[\frac{x - mp}{\sqrt{mp(1-p)}}\right]$$

In practice $m \geq 20$ suffices. A somewhat better approximation results if we apply the continuity correction. If X is an integer-valued random variable, then $Pr(x_1 \leq X \leq x_2)$, where x_1 and x_2 are integers, is exactly the same as $Pr(x_1 - \frac{1}{2} < X < x_2 + \frac{1}{2})$. This amounts to making the discrete space of X continuous by considering intervals of length 1 with midpoints at integers.

Let $G(\alpha) = \int_{\alpha}^{\infty} \phi(z)dz)$ where $\phi(z) = \frac{1}{\sqrt{2\pi}}\; e^{\frac{-z^2}{2}}$. Then we have following results on $G(\alpha)$:

1. $G(0) = \frac{1}{2}$
2. $G(1.0) < 0.16$
3. $G(4.0) < 10^{-4}$
4. $G(\alpha) < \frac{1}{\sqrt{2\pi}}\; e^{\frac{-\alpha}{2}}$, for $\alpha > 1.0$.

Bibliography

[1] S. Goldwasser, and S. Micali. *Probabilistic encryption.* Journal of Computer and System sciences, vol. 28(2), 1984, pp.270-299.

[2] O. Goldreich. *Foundations of Cryptography (fragments of a book).* Electronic Colloquium on Computational Complexity, 1995. Electronic publication :
http://www.eccc.uni-trier.de/eccc-local/ECCC-Books/eccc-books.html.

[3] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. *Private information retrieval* . In Proceedings of 36th Annual Symposium on Foundation of Computer Science. IEEE. Milwaukee, Wisconsin, 23-25 October 1995, pp.41-50. *Journal version in JACM*, vol. 45(6), 1998, pp.965-981.

[4] I. Biehl, and B. Meyer. *Protocols for collusion secure asymmetric fingerprinting.* In STACS'97, 1997, pp 399-412.

[5] D. Boneh, and J. Shaw. *Collusion-secure fingerprinting for digital data.* Lecture Notes in Computer Science. vol. 963. Springer-Verlag, 1995, pp.452-465.

[6] David A. Cooper, and Kenneth P. Birman. *Preserving privacy in a network of mobile computers.* Proc. IEEE Symposium on Security and Privacy,1995, pp.26-38.

[7] A. Ambainis. *An uuper bound on the communication complexity of private information retrieval.* In Proc. of the 24th ICALP. Lecture Notes in Computer Science. vol. 1256. Springer-Verlag, New York, 1997, pp.401-407.

[8] B. Chor, A. Fiat, and M. Naor. *Tracing traiters.* In Proc. of the CRYPTO'95. Lecture Notes in Computer Science. vol. 839. Springer-Verlag, 1994, pp.257-270.

[9] R. Ostrovsky, and V. Shoup. *Private information storage.* In Proceedings of the 29th Annual Symposium on Theory of Computing, El Paso, Tex., May 4-6, 1997. ACM, New York, pp294-303.

[10] B. Chor, and N. Gilboa. *Computationally private information retrieval* . In Proceedings of the 29th Annual Symposium on Theory of Computing, El Paso, Tex., May 4-6, 1997. ACM, New York, pp294-303.

[11] E. Kushilevitz, and R. Ostrovsky. *Replication is not needed : single database computationally-private information retrieval* . In Proceedings of 38th Annual Symposium on Foundation of Computer Science. IEEE Computer Society Press, Los Alamitos, Calif., 1997, pp. 364-373.

[12] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. *Protecting data privacy in private information retrieval schemes* . In Proceedings of the 30th Annual Symposium on Theory of Computing, Dallas, Tex., May 23-26, 1998. ACM, New York, pp.151-160.

[13] M. Abadi, J. Feigenbaum, and J. Kilian. *On hiding information from an oracle* . Journal of Computer and System sciences, vol. 39, 1984, pp.21-50.

[14] D. Beaven, and J. Feigenbaum. *Hiding instances in multioracle queries.* In Proceedings of the STACS. Lecture Notes in Computer Science. vol. 415. Springer-Verlag, New York, 1990, pp.37-48.

[15] D. Beaven, J. Feigenbaum, J. Kilian, and P. Rogaway. *Security with low communication overhead.* In Proceedings of the CRYPTO'90. Lecture Notes in Computer Science. vol. 537. Springer-Verlag, New York, 1991, pp.62-76.

[16] R. L. Rivest, L. Adelman, and M. L. Detrouzos. *On data banks and privacy homomorphisms.* In R. Demillo, D. Dobkin, A. Jones, and R. Lipton, editors, Foundation of Secure Computation. Academic Press, 1978.

[17] D. Beaven, J. Feigenbaum, J. Kilian, and P. Rogaway. *Locally random reductions : Improvements and applications.* Journal of Cryptology. vol. 10(1), 1997. pp.17-36.

[18] D. Beaven, J. Feigenbaum R. Ostrovsky, and V. Shoup. *Instance hiding proof systems.* Technical Report TR-93-65, DIMACS, 1993.

[19] B. Chor, N. Gilboa, and M. Naor. *Private information retrieval by keywords.* Technical Report TR CS0917. Dept. Compt. Science. Technion, Israel. 1997.

[20] C. Cachin, S. Micali, and M. Stadler. *Computationally Private information retrieval with polylogarithmic communication.* In Advances in Cryptology - EUROCRYPT'99, 1999.

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

[22] E. Mann. *Private access to distributed information.* Master's thesis, Technion, Israel Institute of Technology, Hafia, 1998.

[23] J. P. Stern. *A new all or nothing disclosure of secrets protocol.* In ASIACRYPT'98, 1998. Lecture Notes in Computer Science. vol. 1514. Springer-Verlag, pp.357-371.

[24] G. Brassard, C. Crépeau, and Jean-Marc Robert. *All-or-nothing disclosure of secrets.* In CRYPTO'86. Lecture Notes in Computer Science. vol. 263. Springer-Verlag, 1987, pp.234-238.

[25] M. Bellare, and S. Micali. *Non-interactive oblivious transfer and applications.* In Proceedings of CRYPTO'89, 1989, pp 547-557.

[26] M. Fraklin, and S. Haber. *Joint encryption and message-efficient secure computation.* Journal of Cryptography, vol 9(4), 1996, pp 217-232.

[27] R. Canneti. *Toward realizing random oracles: Hash functions that hide all partial information.* In Proceedings of CRYPTO'97, 1997, pp 455-469.

[28] S. Wiesner. *Conjugate coding.* In sigact news, 1983, vol. 18, pp.78-88.

[29] S. Even, O. Goldreich, and A. Limpel. *A randomized protocol for signing contracts.* In CRYPTO'82, pp.205-210, New York, 1983. Plenum Press, Communications of the ACM, vol. 28, 1985, pp.437-447.

[30] G. Brassard, C. Crépeau, and M. Santha. *Oblivious transfer and intersecting codes.* In IEEE Transactions on Information Theory, 1996, pp.1769-1780.

[31] G. Brassard, and C. Crépeau. *Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for sat and beyond.* In Proceedings of 27th FOCS, 1986, pp 188-195.

[32] C. Cachin, C. Crépeau, and S. Marcil. *Oblivious transfer with a memory bounded receiver.* In Proceedings of 39th FOCS, 1998.

[33] M. Naor, and O. Reingold. *Number-theoretic constructions of efficient pseudo-random functions.* In Proceedings of 39th FOCS, 1998.

[34] M. Steiner, G. Tsudik, and M. Waidner. *Diffie-Hellman Key distribution extended to group communication.* In Proceedings of 3rd CCCS, 1996, pp 31-37.

[35] C. Crépeau. *Equivalence between two flavors of oblivious transfer.* In Proceedings of CRYPTO'87, 1987.

[36] M. O. Rabin. *How to exchange secrets by oblivious transfer.* Technical Report, TR-81.

[37] J. Kilian. *Founding cryptography on oblivious transfer.* In Proceedings of 20th STOC, 1988.

[38] A. Salomaa, and L. Santean. *Secret selling of secrets with several buyers.* In 42th EATCS Bulletin, 1990, pp.178-186.

[39] H. NurmiA. Salomaa, and L. Santean. *Secret ballot elections in computer networks.* In Computers and Security, vol. 10, 1991, pp. 553-560.

[40] V. Niemi, and A. Renvall. *Cryptographic protocols and voting.* In Results and Trends in Theoretical Computer science. Lecture Notes in Computer Science. vol. 812. Springer-Verlag, 1994, pp.307-316.

[41] K. Sakurai, and Y. Yamane. *Blind decoding, blind undeniable signatures, and their applications to privacy protection.* In Information Hiding. Lecture Notes in Computer Science. vol. 1174. Springer-Verlag, 1997, pp.257-264.

[42] K. Ohta. *Remarks on blind decryption.* In Information Security Workshop, 1997, pp.59-64.

[43] E. Kushilevitz, and R. Ostrovsky. *One-way trapdoor permutations are sufficient for single database computationally-private information retrieval* . In Proceedings of EUROCRYPT'00, 2000.

[44] A. C. Yao. *Theory and application of trapdoor functions.* In Proceedings of 23th Annual IEEE Symposium on Foundation of Computer Science. 1982, pp.80-91.

[45] G. Di Crescenzo, T. Malkin, and R. Ostrovsky. *Single database private information retrieval implies oblivious transfer.* In Proceedings of EUROCRYPT'00, 2000.

[46] O. Goldreich, S. Micali, and A. Wigderson. *How to play any mental game or a completeness theorem for protocol with honest majority.* In Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987. pp218-229.

[47] M. Naor, and B. Pinkas. *Oblivious transfer and polynomial evaluation.* In Proceedings of the 31th Annual ACM Symposium on Theory of Computing, 1999. pp245-254.

[48] A. Beimel, Y. Ishai, and T. Malkin. *Reducing server computation in private information retrieval : PIR with preprocessing.*

[49] O. Goldreich, and R. Ostrovsky. *Software protection and simulation by Oblivious RAMs.* JACM, 1996.

[50] O. Goldreich. *Towards a theory of software protection and simulation by Oblivious RAMs.* STOC'87, 1987.

[51] R. Ostrovsky. *Software protection and simulation on Oblivious RAMs.* M.I.T. Ph.D. Thesis in Computer Science, June 1992.

[52] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. *Universal service-providers for database private information retrieval.* In Proceedings of 17th PODC, 1998.

[53] Y. Gertner, S. Goldwasser, and T. Malkin. *A random server model for private information retrieval schemes.* In Proceedings of the 2nd RANDOM,1998.

[54] E. Kushilevitz, S. Micali, and R. Ostrovsky. *Reducibility and completeness in multi-party private computations.* In Proceedings of 35th FOCS'94, 1994.

[55] Y. Ishai, and E. Kushilevitz. *Improved upper bounds on information theoretic private information retrieval* In Proceedings of 31st STOC, 1999.

[56] R. Impagliazzo, and M. Luby. *One-way functions are essential for complexity based cryptography.* In Proceedings of FOCS, 1989.

[57] R. Impagliazzo, and S. Rudich. *Limits on the provable consequences of one-way permutations.* In Proceedings of 21st STOC, 1989.

[58] J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. *Reducibility and completeness in private computations.* In SIAM Journal on Computing, include [54].

[59] M. Blum,M. Blum, and M. Shum. *A simple unpredictable pseudo-random number generator.* In SIAM Journal on Computing, vol 15, 1986, pp 364-383.

[60] D. Stinson. *CRYPTOGRAPHY Theory and Practice.* 1995 by CRC Press, Inc.

[61] V. K. Rohtagi, Bowling Green State University. *An Introduction to Probability Theory and Mathematical Statistics.* 1976 Wiley Eastern Limited