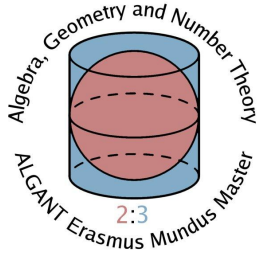


MASTER ALGANT

University of Padova
and
University of Bordeaux 1

Master Thesis in Mathematics

A Study of Computational Private Information Retrieval Schemes and Oblivious Transfer

Valentina Settimi

Supervisor: Prof. Gilles Zemor

26 june 2007

Contents

List of Tables

Notations and Abbreviations

$$
\begin{aligned}
\mathcal{I}_n &= \{1, \ldots, n\} \\
\mathcal{J}_{n-1} &= \{0, \ldots, n-1\} \\
u \in_R U &= u \text{ is an element chosen at random with the uniform distribution over } U \\
U \subseteq_R V &= U \text{ is a randomly and uniformly chosen subset of } V \\
\log u &= \log_2 u \\
U \oplus u &= \begin{cases} U \cup \{u\} & \text{if } u \notin U \\ U \setminus \{u\} & \text{if } u \in U \end{cases} \text{ when } U \text{ set and } u \text{ element} \\
\text{PIR} &= \text{Private Information Retrieval} \\
\text{itPIR} &= \text{Information-Theoretic Private Information Retrieval} \\
\text{cPIR} &= \text{Computational Private Information Retrieval} \\
\text{SPIR} &= \text{Symmetrically Private Information Retrieval} \\
\text{OT} &= \text{Oblivious Transfer} \\
\mathbb{Z}_N &= \mathbb{Z}/N\mathbb{Z} \\
\mathbb{Z}_N^* &= \text{the multiplicative group of } \mathbb{Z}/N\mathbb{Z} \\
\mathbb{Z}_N^1 &= \text{the subset of } \mathbb{Z}_N^* \text{ of the elements having Jacobi symbol } +1 \\
\mathscr{QR}_N &= \text{set of quadratic residues modulo } N \\
\mathscr{PQR}_N &= \text{set of pseudo-quadratic residues modulo } N \\
\mathscr{4R}_N &= \text{set of quartic residues modulo } N \\
\mathscr{2}^m\mathscr{R}_N &= \text{set of } 2^m\text{-th residues modulo } N \\
\mathscr{NR}_{N^2} &= \text{set of } N^{\text{th}} \text{ residues modulo } N^2 \\
\text{QRA} &= \text{Quadratic Residuosity Assumption} \\
\text{CRA} &= \text{Composite Residuosity Assumption} \\
\text{DLP} &= \text{Discrete Logarithm Problem}
\end{aligned}
$$

Chapter 1

Introduction

The core of the networked computing environments (Internet, intranets) is passing information; a typical situation is the one in which many servers distribute information to the public.

In the past a lot of efforts were devoted to finding methods that protect servers' privacy: for instance to protect the servers from non-legitimate users (*e.g.* by authentication of the users [9]) or from eavesdroppers (*e.g.* by encryption). On the contrary, the issue of protecting users' privacy against the servers is quite recent: *Private Information Retrieval* (or *PIR*) schemes, introduced by Chor, Goldreich, Kushilevitz and Sudan [7], allow a user to retrieve information from a database without exposing the identity of his interest. The cost of such protocols is their communication complexity, measured as the number of bits transmitted between the user and the servers.

Formally, the PIR problem is stated as follows: Let the database be modeled by a bit string x of length n held by some server(s); we want to fulfill a user who wishes to retrieve the i^{th} bit x_i, for some $i \in \mathcal{I}_n$, without disclosing any information about i to the server(s).

The trivial solution to the PIR problem is sending the whole database from the server(s) to the user. In realistic setting the size of the database is very large, therefore this solution, although preserving user's privacy, is impractical. The goal of PIR protocols is to allow the user to privately retrieve data from a public database of length n with communication complexity strictly smaller than n (*i.e.* with less communication than just downloading the whole database).

The PIR protocols are divided into two main classes, according to the kind of privacy they guarantee to the user:

Information-Theoretic PIR (itPIR) : They guarantee *information-theo-*

retic privacy to the user, *i.e.* privacy against computationally unbounded server(s);

Computational PIR (cPIR) : They guarantee *computational privacy* to the user, *i.e.* privacy against computationally bounded server(s); in this case we will need to define appropriate intractability assumptions.

The main difference between these two classes is that information-theoretic privacy can be efficiently achieved only if the database is replicated at $k \geq 2$ non-communicating servers: Chor et al. proved [7] that if we have only a single server holding the database, than the best solution to the itPIR problem is the trivial one. On the contrary, in computational privacy setting, the replication of the database is not needed. This result, proved by Kushilevitz and Ostrovsky [17], is very relevant since, though possibly viable, the assumption of the existence of non-communicating servers may not be too practical.

The PIR problem is only concerned with users' privacy, without requiring any protection of server(s)' privacy; indeed PIR protocols can allow the user to obtain x_i and some additional information, such as other bits of the database [17, 19] or the exclusive-or of some bits of x. Gertner, Ishai, Kushilevitz and Malkin [11] introduced *Symmetrically-Private Information Retrieval* (or *SPIR*) schemes which are PIR protocols that also maintain the server(s)' privacy. For instance,this means that if the user payed for a single bit of the database, he will not be able to obtain more information than what he payed for. Server(s)' privacy can be guaranteed either against honest-but-curious user (*i.e.* the user follows the protocol, but he tries to deduce extra information), or against dishonest user (*i.e.* the user tries to cheat). A protocol that meets the second requirement is also called an *Oblivious Transfer* scheme.

1.1 Organization of the work and contributions

In Chapter 2 we provide some definitions (*e.g.* of itPIR and cPIR schemes) that are used throughout the work.

In Chapter 3 we carefully describe and analyze the cPIR protocol by Kushilevitz and Ostrovsky [17]; here we also provide a more precise estimation of the communication complexity of this scheme.

In Chapter 4 we present our protocols: our idea is to modify Kushilevitz and Ostrovsky scheme in order to obtain cPIR protocols which enable the user to

retrieve a block of bits or several bits in any position, without using stronger assumptions. In particular, we use quartic residues to construct two cPIR schemes which allows the user to retrieve either 2 bits in any position or 2 consecutive bits (that is a block of 2 bits). Subsequently, we generalize theme: using 2^m-th residues, we construct two cPIR protocols which enable the user to obtain either m bits in any position or a block of m bits. All these protocols are based on the Quadratic Residuosity Assumption (QRA), as the scheme by Kushilevitz and Ostrovski: we actually prove that QRA is enough to guarantee the privacy of our schemes.

In Chapter 5 we describe with details two cPIR protocols based on more sophisticated assumption: the first is by Chang [6] and the second is by Ostrovsky and Skeith III [21]. We give a more precise estimation of the communication complexity of Chang's protocol and we fill Ostrovsky and Skeith III's work providing the missing details.

In Chapter 6 we define SPIR and Oblivious Transfer protocols and we show how to transform the cPIR scheme by Kushilevitz and Ostrovsky into a SPIR scheme secure against honest-but-curious users (the modification we present is actually a special case of Ostrovsky and Skeith III's protocol). Finally we trasform it into an Oblivious Transfer protocol (the same construction can be found in a paper of Mishra and Sarkar [20]).

In Appendix A, for completeness, we shortly describe some itPIR schemes.

1.2 Acknowledgment

I would like to thank Prof. Gilles Zemor for his instructive guidance.

I also thank Prof. Alessandro Languasco for his invaluable support and Prof. Adrian Iovita.

Grazie anche ai miei amici perché il loro affetto è la mia forza.

Infine ringrazio la mia famiglia a cui dedico questo lavoro, come del resto tutto quello che faccio.

Chapter 2

PIR Schemes

Private Information Retrieval (or *PIR*) schemes have been introduced in [7] to solve the so called

PIR problem : A user want to retrieve some information from a database *without* exposing his interest in that information.

Thus a PIR protocol involves two parties, the *user* and the *server(s)*, each having a secret input: server(s)' secret input is an n-bit string x (called *database*) and user's secret input is an integer i between 1 and n (called *index*). A PIR protocol has to be communication-efficient (*i.e.* its communication complexity must be strictly smaller then n) and it has to meet two main requirements:

Correctness : In every invocation of the protocol the user retrieves the bit he is interested in ($i.e$ x_i);

Privacy : In every invocation of the protocol each server does not gain any information about the index of the bit retrieved by the user (*i.e.* i).

	PARTIES	
	User $\mathcal{U}$	Servers $\mathcal{S}_1, \ldots, \mathcal{S}_k$, for $k \geq 1$
Secret Input	index i, for $i \in \mathcal{I}_n$	database $x \in \mathbb{Z}_2^n$ $(x = x_1, \ldots, x_n)$
Final information	x_i	none information about i

Table 2.1: PIR scheme

The privacy could be defined in two different ways:

Information-theoretic privacy (or **perfect privacy**): The distribution of the queries the user sends to any server is *independent* of the index he wishes to retrieve. This means that each server cannot gain any information about user's interest regardless of his computational power.

Computational privacy : The distributions of the queries the user sends to any server are computationally *indistinguishable* by varying the index. This means that each server cannot gain any information about user's interest provided that he is computationally bounded.

According to the kind of privacy they guarantee, the PIR schemes are divided into two classes.

Definition 2.0.1 (itPIR). *A* (k-server) information-theoretic PIR *(or* itPIR*) scheme is a triple of algorithms* $(\mathcal{Q}, \mathcal{A}, \mathcal{R})$ *where:*

$\mathcal{Q}$ = query generator *is such that:*

- $\mathcal{Q}$ *is a polynomial-time algorithm run by* $\mathcal{U}$,
- $\texttt{INPUT} \leftarrow (1^n, i, r)$
  with $\begin{cases} n \in \mathbb{N} & \text{the length of the database,} \\ i \in \mathcal{I}_n & \text{the index,} \\ r \in_R \mathbb{Z}_2^{P(n)} & \text{for some fixed } P \text{ polynomial,} \end{cases}$
- $\texttt{OUTPUT} \rightarrow (q_1, \ldots, q_k)$
  with q_j query *intended for the server* $\mathcal{S}_j$*;*

$\mathcal{A}$ = answer generator *is such that:*

- $\mathcal{A}$ *is a polynomial-time algorithm run by each server* $\mathcal{S}_j$,
- $\texttt{INPUT} \leftarrow (x, q)$
  with $\begin{cases} x \in \mathbb{Z}_2^n & \text{the database,} \\ q & \text{a query,} \end{cases}$
- $\texttt{OUTPUT} \rightarrow (a)$
  with a answer *intended for* $\mathcal{U}$*;*

$\mathcal{R}$ = reconstruction algorithm *is such that:*

- $\mathcal{R}$ *is a polynomial-time algorithm run by* $\mathcal{U}$,

- $\texttt{INPUT}\leftarrow (1^n, i, r, a_1, \ldots, a_k)$
  with a_j *the answer sent by* $\mathcal{S}_j$,
- $\texttt{OUTPUT}\rightarrow (b)$
  with $b \in \mathbb{Z}_2$.

The triple $(\mathcal{Q}, \mathcal{A}, \mathcal{R})$ *has to satisfy the following two conditions:*

1. (Correctness) *Let* $\mathcal{Q}(1^n, i, r)^j$ *be the* j^{th} *element of* $\mathcal{Q}(1^n, i, r)$. *Then:*
$$\mathcal{R}\Big(1^n, i, r, \mathcal{A}\big(x, \mathcal{Q}(1^n, i, r)^1\big), \ldots, \mathcal{A}\big(x, \mathcal{Q}(1^n, i, r)^k\big)\Big) = x_i$$
for every $n \in \mathbb{N}$, *for every* $x \in \mathbb{Z}_2^n$, *for every* $i \in \mathcal{I}_n$ *and for every* $r \in \mathbb{Z}_2^{P(n)}$.

2. ((information-theoretic) Privacy) *Let* D_i^j *be the distribution of queries from* $\mathcal{U}$ *to* $\mathcal{S}_j$ *when* $\mathcal{U}$ *wants to retrieve the* i^{th} *bit of the database (*i.e. D_i^j *is the distribution of* $\mathcal{Q}(1^n, i, r)^j$, *by varying* $r \in_R \mathbb{Z}_2^{P(n)}$*). Then:*
$$D_i^j = D_{i'}^j$$
for every $i, i' \in \mathcal{I}_n$ *and for every* $j \in \mathcal{I}_k$.

The correctness condition means that, if $\mathcal{U}$ sends queries computed on index i to all the servers, then, using all the answers he receives and his own inputs, he must be able to reconstruct the desired bit.

The information-theoretic privacy condition means that, fixed a server $\mathcal{S}_j$, the queries he receives when $\mathcal{U}$ wants the i^{th} bit and that ones he receives when the desired bit is the i'^{th} are identically distributed (the distribution is taken on the random input r). It implies that each server, analyzing his queries, cannot infer any information about the index, however powerful he is.

Definition 2.0.2 (cPIR). *A* (k-server) computational PIR *(or* cPIR*) scheme is a triple of algorithm* $(\mathcal{Q}, \mathcal{A}, \mathcal{R})$ *as in definition 2.0.1 satisfying the same correctness condition and the following privacy condition:*

2'. ((computational) Privacy) *Let* D_i^j *and* $D_{i'}^j$ *be as above. For any family of polynomial-time circuits* $\{\mathcal{C}_j\}$ *and for every non-constant polynomial* $P'(X) \in \mathbb{N}[X]$, *there exists* $n_0 \in \mathbb{N}$ *such that, for every* $n > n_0$*:*
$$\left|\Pr\left[\mathcal{C}_n(Q) = 1 | Q \in D_i^j\right] - \Pr\left[\mathcal{C}_n(Q) = 1 | Q \in D_{i'}^j\right]\right| < \frac{1}{P'(n)}$$
for every $i, i' \in \mathcal{I}_n$ *and for every* $j \in \mathcal{I}_k$.

The computational privacy condition means that the queries a fixed server receives when the user wants different indices must be indistinguishable in polynomial (in $n = |x|$) time.

Remark 2.1. In our setting:

1. Each server $\mathcal{S}_j$ holds the same database x.
2. Each server $\mathcal{S}_j$ can communicate with the user but not with $\mathcal{S}_{j'}$, for any $j' \neq j$.
3. The database x is stored in *plain form*; this implies that the server(s) can serve both PIR users and users who do not require privacy and these non-private queries can be served with minimal communication complexity.
4. The scheme is a *single-round* protocol: The communication consists only of queries from the user to the server(s) and one replay from each server. Note that this property already assures that user's privacy does not depend on the behavior of the server(s). Moreover single-round schemes free the server(s) from storing any information but the database itself.

The main cost measure for an any PIR protocol is its communication complexity, obtained by counting the number of bits transmitted between the user and the server(s). Intuitively the communication complexity of a protocol is the maximum (over all possible invocations) of the sum of the lengths of queries and answers.

Definition 2.1.1 (Communication complexity). *Let $\mathcal{P} = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a PIR protocol (itPIR or cPIR). For any $n \in \mathbb{N}$, its* communication complexity *is:*

$$\mathcal{CC}_{\mathcal{P}}(n) := \max_{\substack{r \in \mathbb{Z}_2^{P(n)} \\ x \in \mathbb{Z}_2^n \\ i \in \mathcal{I}_n}} \left\{ |\mathcal{Q}(1^n, i, r)| + \sum_{j=1}^{k} \left| \mathcal{A}\big(x, \mathcal{Q}(1^n, i, r)^j\big) \right| \right\}.$$

Suppose $k = 1$ (*i.e.* there is only one server $\mathcal{S}$). It is quite obvious and easy to prove [7] that any itPIR scheme needs the exchange of n bits (with $n = |x|$), hence the trivial solution is optimal in this case. This bound is clearly due to the information-theoretic privacy constraint; indeed, there exist cPIR protocols with only one single server and communication complexity strictly smaller than n [17, 6].

Chapter 3

Description of $\mathcal{P}$: Basic cPIR Protocol

The most important result about cPIR schemes is due to Kushilevitz and Ostrovsky who proved that the replication of the database is not needed in computational privacy setting: Indeed in [17] they present a method for constructing a 1-server cPIR protocol, say $\mathcal{P}$, with communication complexity $\mathcal{O}(n^{\epsilon})$, for any $\epsilon > 0$ (in this work we show that actually the communication complexity is $e^{c\sqrt{\ln n}}$, for some $c > 0$ depending on a security parameter). In this chapter we give a detailed presentation and analysis of this protocol $\mathcal{P}$.

Since it is based on the *Quadratic Reciprocity Assumption* [13], we start giving a short description of this well known number-theoretic assumption.

3.1 Quadratic Residuosity Assumption

Let $N \in \mathbb{N}$; consider the multiplicative group $\mathbb{Z}_N^*$.

Definition 3.1.1 (Quadratic residue). *An integer $x \in \mathbb{Z}_N^*$ is said to be a* quadratic residue modulo N *if there exists an integer $y \in \mathbb{Z}_N^*$ such that $x = y^2 \mod N$. We denote with $\mathscr{QR}_N$ the set of quadratic residues modulo N.*

If $N = p$ is a prime number, then exactly half of the numbers in $\mathbb{Z}_p^*$ are in $\mathscr{QR}_N$ and they can be easily found using the *Jacobi symbol* [14].

If $N = p_1 p_2$ with $p_1 \neq p_2$ prime numbers, then $\mathbb{Z}_N^*$ is isomorphic to $\mathbb{Z}_{p_1}^* \times \mathbb{Z}_{p_2}^*$ and so, by *Chinese Reminder Theorem*, for every $x \in \mathbb{Z}_N^*$:

$$x \in \mathscr{QR}_N \iff x \in \mathscr{QR}_{p_1} \cap \mathscr{QR}_{p_2}.$$

Using the *Quadratic Reciprocity Law*, we can easily calculate the Jacobi symbol of any element of $\mathbb{Z}_N^*$ even when the factorization of N is unknown. However,

for composite moduli the Jacobi symbol cannot be used to the detect quadratic residues: Half of the numbers in $\mathbb{Z}_N^*$ have Jacobi symbol -1 and they actually are not quadratic residues modulo N; the other half have Jacobi symbol $+1$ but only half of them are quadratic residues. We denote with $\mathbb{Z}_N^1$ the subset of $\mathbb{Z}_N^*$ of the numbers having Jacobi symbol $+1$.

Definition 3.1.2 (Pseudo-quadratic residue). *An integer $x \in \mathbb{Z}_N^*$, with $N = p_1 p_2$ as above, is said to be a* pseudo-residue modulo N *if has Jacobi symbol $+1$ (*i.e. *it is in $\mathbb{Z}_N^1$) but it is not a quadratic residue modulo N. We denote with $\mathscr{PQR}_N$ the set of pseudo-quadratic residue modulo N.*

Thus: $|\mathscr{QR}_N| = |\mathscr{PQR}_N| = \frac{1}{2}|\mathbb{Z}_N^1| = \frac{1}{4}|\mathbb{Z}_N^*| = \frac{\phi(N)}{4}$, with ϕ the *Euler's totient function* and $\phi(N) = (p_1 - 1)(p_2 - 1)$ in the present case.

The *Quadratic Residuosity Problem* modulo N, with $N = p_1 p_2$ as above, is: Given $x \in \mathbb{Z}_N^1$, determine whether $x \in \mathscr{QR}_N$ or $x \in \mathscr{PQR}_N$ [13]. Clearly it can be easily solved if the factorization of N is known; on the contrary it is believed that solving the Quadratic Residuosity Problem modulo N without knowing the factorization of N is computationally hard.

Conjecture 3.1.3 (Quadratic Residuosity Assumption (QRA)). *Let p_1 and p_2 be distinct prime numbers such that $|p_1| = |p_2|$ (large enough) and let $N = p_1 p_2$. If the factorization of N is not known, there is no efficient procedure for solving the Quadratic Residuosity Problem modulo N.*

The Quadratic Residuosity Problem has some useful properties:

1. For every $x, y \in \mathbb{Z}_N^1$:

$$xy \in \mathscr{QR}_N \iff (x, y \in \mathscr{QR}_N) \vee (x, y \in \mathscr{PQR}_N). \tag{3.1}$$

2. Picking a random element in $\mathscr{QR}_N$ is easy: pick $r \in_R \mathbb{Z}_N^*$ and compute r^2 mod N (for doing it, we do not need to know the factorization of N).

3. Under QRA, the Quadratic Residuosity Problem modulo N is hard not only for some special $x \in \mathbb{Z}_N^1$, but it is hard on average (that is it is either everywhere hard or everywhere easy) [13]. We refer to this property saying that the Quadratic Residuosity Problem is *random-self reducible.*

3.2 Basic scheme

In this section we present a 1-server cPIR scheme with communication complexity $\mathcal{O}(n^{1/2+\epsilon})$, for any $\epsilon > 0$. In the next section we will use it to construct a recursive protocol with less communication complexity.

We consider the database $x \in \mathbb{Z}_2^n$ as a $R \times C$ matrix of fixed dimensions: we associate the string x with a matrix $(x_{r,c})_{r \in \mathcal{I}_R, c \in \mathcal{I}_C}$ and each position $j \in \mathcal{I}_n$ with a pair $(r, c) \in \mathcal{I}_R \times \mathcal{I}_C$. In particular, the index i of the desired bit is associated with the pair (r^*, c^*).

Protocol $\mathcal{B}$

Let $k \in \mathbb{N}$ be the security parameter.

$\mathcal{Q}$: – $\texttt{INPUT}\leftarrow (1^n, i = (r^*, c^*))$
(we omit the random input, since we have to specify how $\mathcal{U}$ use randomness),
– Choose at random $p_1 \neq p_2$ prime numbers such that $|p_1| = |p_2| = k/2$,
– Let $N = p_1 p_2$,
– For $1 \leq c \leq C$, choose $q_c \in_R \mathbb{Z}_N^1$ such that:
$$\begin{cases} q_{c^*} \in \mathscr{P}\mathscr{Q}\mathscr{R}_N, \\ q_c \in \mathscr{Q}\mathscr{R}_N \qquad \forall c \neq c^*, \end{cases}$$
– $\texttt{OUTPUT}\rightarrow Q = (N, q_1, \ldots, q_C)$;

$\mathcal{A}$: – $\texttt{INPUT}\leftarrow (x, Q)$
with $\begin{cases} x = (x_{r,c})_{r \in \mathcal{I}_R, c \in \mathcal{I}_C}, \\ Q = (N, q_1, \ldots, q_C) \in \mathbb{N} \times (\mathbb{Z}_N^*)^C \text{query sent by } \mathcal{U}, \end{cases}$
– For $1 \leq r \leq R$, let:
$$a_r = \prod_{c=1}^{C} (q_c)^{x_{r,c}} \mod N,$$
– $\texttt{OUTPUT}\rightarrow (a_1, \ldots, a_R)$;

$\mathcal{R}$: – $\texttt{INPUT}\leftarrow (1^n, i = (r^*, c^*), a_1, \ldots, a_R)$
with $a_1, \ldots, a_R$ the answers sent by $\mathcal{S}$,
– Let $b \in \mathbb{Z}_2$ be such that:
$$b = 0 \Leftrightarrow a_{r^*} \in \mathscr{Q}\mathscr{R}_N,$$

– OUTPUT$\longrightarrow (b)$.

Theorem 3.2.1. *The protocol $\mathcal{B}$ defined above is a cPIR scheme such that $\mathcal{CC}_{\mathcal{B}}(n) = \mathcal{O}(n^{\epsilon+1/2})$, for any $\epsilon > 0$.*

Proof. We have to prove that $\mathcal{B}$ verifies the definition 2.0.2 and that it has the required communication complexity.

1. (Correctness) By 3.1, a_{r^*} is in $\mathscr{QR}_N$ if and only if there are an even number of pseudo-quadratic residues modulo N among the $(q_c)^{x_{r^*,c}}$, with $1 \leq c \leq C$. Since only q_{c^*} is in $\mathscr{PQR}_N$, we have that $a_{r^*} \in \mathscr{QR}_N \Leftrightarrow x_{r^*,c^*} = 0$. Therefore $b = x_{r^*,c^*}$. Remark that, knowing the factorization of N, $\mathcal{U}$ can efficiently determine the quadratic residuosity of a_{r^*}.

2. (Privacy) Assume by contradiction that for some indices $i = (r^*, c^*)$ and $i' = (r'^*, c'^*)$ the server can distinguish the queries on i from the ones on i'. That is, if we denote with D_i and $D_{i'}$ the distributions of the queries on i and i' respectively, then there exists a family of polynomial-time circuits $\{\mathcal{C}_j\}$ such that:

$$\left|\Pr\left[\mathcal{C}_n(Q) = 1 | Q \in D_i\right] - \Pr\left[\mathcal{C}_n(Q) = 1 | Q \in D_{i'}\right]\right| \geq \frac{1}{P(n)}$$

for some non-constant P polynomial. We can suppose, without loss of generality, that $\Pr[\mathcal{C}_n(Q) = 1 | Q \in D_{i'}] - \Pr[\mathcal{C}_n(Q) = 1 | Q \in D_i] \geq \frac{1}{P(n)}$.

By construction, a query in D_i consists of N followed by C elements of $\mathbb{Z}_N^1$ such that only the $c^{*\text{th}}$ is in $\mathscr{PQR}_N$. A query in $D_{i'}$ is similar except that the pseudo-quadratic residue is located in position c'^*. Therefore we must have $c^* \neq c'^*$, otherwise there is no way to distinguish D_i from $D_{i'}$.

We now use $\mathcal{C}_n$ to construct a circuit $\mathcal{C}'$ which solves the Quadratic Residuosity Problem for N, that is $\mathcal{C}'$ takes in input $y \in \mathbb{Z}_N^1$ and it computes the quadratic residuosity of y with probability at least $\frac{1}{2} + \frac{1}{8P(n)}$.

Circuit C'

- INPUT$\longleftarrow (y)$ as above,
- Choose $\bar{c} \in_R \{c^*, c'^*\}$,

- For $1 \leq c \leq C$, choose $q_c \in \mathbb{Z}_N^1$ such that:
$$\begin{cases} q_{\overline{c}} = y, \\ q_c \in_R \mathscr{QR}_N \quad \forall c \neq \overline{c}, \end{cases}$$
- Let $b \in \mathbb{Z}_2$ be such that:
$$\begin{cases} b = 1 \oplus \mathcal{C}_n\big((q_1, \ldots, q_C)\big) & \text{if } \overline{c} = c^*, \\ b = \mathcal{C}_n\big((q_1, \ldots, q_C)\big) & \text{otherwise,} \end{cases}$$
- $\texttt{OUTPUT} \rightarrow (b)$.

If $y \in \mathscr{QR}_N$, then $(q_1, \ldots, q_C)$ is a sequence of elements in $\mathscr{QR}_N$, whatever is the value of $\overline{c}$. Let $p = \Pr[\mathcal{C}_n(Q) = 1 | Q \in (\mathscr{QR}_N)^C]$, then:

$$\Pr\left[\mathcal{C}'(y) = 1 | y \in \mathscr{QR}_N\right] = \frac{1}{2}p + \frac{1}{2}(1-p) = \frac{1}{2}.$$

On the contrary:

$$\begin{aligned} \Pr\left[\mathcal{C}'(y) = 1 | y \in \mathscr{PQR}_N\right] &= \frac{1}{2}\Big(1 - \Pr\left[\mathcal{C}_n(Q) = 1 | Q \in D_i\right]\Big) + \\ &\quad + \frac{1}{2}\Pr\left[\mathcal{C}_n(Q) = 1 | Q \in D_{i'}\right] \geq \\ &\geq \frac{1}{2} + \frac{1}{2P(n)}. \end{aligned}$$

It is not exactly what we need, since we want $\Pr[\mathcal{C}'(y) = 1 | y \in \mathscr{QR}_N] < \frac{1}{2}$. To this end we add to $\mathcal{C}'$ an initial step in which with probability $\frac{1}{4P(n)}$ it outputs 0 and stops. We now have:

$$\begin{aligned} \Pr[\mathcal{C}'(y) = 1 | y \in \mathscr{QR}_N] &= \left(1 - \frac{1}{4P(n)}\right)\frac{1}{2} = \frac{1}{2} - \frac{1}{8P(n)}. \\ \Pr[\mathcal{C}'(y) = 1 | y \in \mathscr{PQR}_N] &\geq \left(1 - \frac{1}{4P(n)}\right)\left(\frac{1}{2} + \frac{1}{2P(n)}\right) = \\ &= \frac{1}{2} + \frac{1}{2P(n)} - \frac{1}{8P(n)} - \frac{1}{8P^2(n)} \geq \\ &\geq \frac{1}{2} + \frac{1}{8P(n)}. \end{aligned}$$

Therefore $\mathcal{C}'$ solves the Quadratic Residuosity Problem modulo N with non-negligible probability without knowing the factorization of N, in contradiction to QRA.

3. (Communication complexity) The security parameter is $k = \lceil \log N \rceil$. $\mathcal{U}$ sends $Q = (N, q_1, \ldots, q_C)$, with $q_c \in \mathbb{Z}_N^1$, to $\mathcal{S}$; so $\mathcal{U}$ sends $(C+1)k$ bits. $\mathcal{S}$ replies sending $(a_1, \ldots, a_R)$, with $a_r \in \mathbb{Z}_N^1$; so $\mathcal{S}$ sends Rk bits. Thus the total amount of communication is $k(R + C + 1)$ bits.

   By construction $RC = n$, hence the better choice for R and C is $R = C = \sqrt{n}$. For every $\epsilon > 0$, if we choose as the security parameter $k = n^\epsilon$, we have:

$$\mathcal{CC}_{\mathcal{B}}(n) = n^\epsilon(2\sqrt{n} + 1) = \mathcal{O}\big(n^{1/2+\epsilon}\big).$$

□

3.3 Protocol $\mathcal{P}$: Recursive scheme

In this section we use the idea of the basic scheme to construct a cPIR protocol with communication complexity $\mathcal{O}(e^{c\sqrt{\ln n}})$, for some $c > 0$ depending on the security parameter. The new protocol is based on the observation that in $\mathcal{B}$ $\mathcal{U}$ is only interested in one of the numbers he receives from $\mathcal{S}$. However $\mathcal{U}$ cannot reveal what is the item he needs, as this will violate the privacy constraint. It is therefore natural to see the Rk-bit string $(a_1, \ldots, a_R)$ as a new database of which $\mathcal{U}$ wants k bits; $\mathcal{U}$ retrieves them using k invocations of the cPIR protocol itself.

Remark that k invocations of the cPIR scheme require k new queries to be sent. In order to minimize the increase in communication, we consider $(a_1, \ldots, a_R)$ not as a Rk-bit string but as k R-bit strings such that the j^{th} string is the ordered concatenation of the j^{th} bit of each a_r. In this way the user does not need to send k different queries, but it is enough he sends only one query since he wants the same bit (the $r^{*\text{th}}$) from all the databases.

For clarity, we will not present the scheme as a triple of algorithm $(\mathcal{Q}, \mathcal{A}, \mathcal{R})$ but rather as a recursive scheme. However it is important to notice that the user can compute in advance all parts of the query he needs to send and send all of them at once. Hence the new protocol can still be implemented in a single round.

The protocol would consist of L level of recursion, we will denote the l^{th} level by Level_l (we will use the subscript l referring to Level_l, but when it is impossible we will use superscripts). Let Level_1 be the basic scheme $\mathcal{B}$ described

above. We set:

$$\begin{cases} x^L = x \text{ viewed as a } R_L \times C_L \text{ bit-matrix,} \\ n_L = |x^L| = n, \\ j_L \in \mathcal{I}_{n_L} \text{ a generic position in the database. It is associated with a pair} \\ \qquad (r_L, c_L) \in \mathcal{I}_{R_L} \times \mathcal{I}_{C_L} \text{ such that } (r_L - 1)C_L + c_L = j_L, \\ i_L = (r_L^*, c_L^*) \text{ the index of the desired bit.} \end{cases}$$

For every $L \geq l \geq 1$ do Level$_l$:

- View the database x^l as a $R_l \times C_l$ bit-matrix (thus $R_l C_l = n_l$). Let $i_l = (r_l^*, c_l^*)$ be the index of the bit $\mathcal{U}$ wants to retrieve. $\mathcal{U}$ and $\mathcal{S}$ simulate protocol $\mathcal{B}$ with this setting.
- If $l > 1$, then $\mathcal{S}$ does not send his answer $(a_1^l, \ldots, a_{R_l}^l)$ to $\mathcal{U}$ but he considers it as k new databases:

  For every $1 \leq j \leq k$:

  - Let $x^{l-1} = \big((j^{\text{th}} \text{ bit of } a_1^l), \ldots, (j^{\text{th}} \text{ bit of } a_{R_l}^l)\big)$
    Thus $n_{l-1} = |x^{l-1}| = R_l$,
  - $i_{l-1} = r_l^*$ (since $\mathcal{U}$ is only interested in $a_{r_l^*}^l$),
  - $\mathcal{U}$ and $\mathcal{S}$ go to Level$_{l-1}$ with x^{l-1} as database and i_{l-1} as index.
- If $l = 1$, $\mathcal{S}$ sends his answer $(a_1^1, \ldots, a_{R_1}^1)$ to $\mathcal{U}$.
- $\mathcal{U}$ uses $a_{r_l^*}^l$ as in $\mathcal{B}$ to retrieve $x_{i_l}^l$.

Remark that $n_{l-1} = R_l$ and i_{l-1} are the same for each invocation of Level$_{l-1}$. This implies that the length of the database decreases at each step ($n_{l-1} = R_l = n_l/C_l$) and that $\mathcal{U}$ sends only one query for all the invocations of Level$_{l-1}$.

Remark 3.4. In the original paper [17] the authors prove that the communication complexity of $\mathcal{P}$ is $\mathcal{O}(n^\epsilon)$, for any $\epsilon > 0$. In this work we give a more precise estimation, proving that $\mathcal{CC}_\mathcal{P}(n) = \mathcal{O}\big(e^{c\sqrt{\ln n}}\big)$, for some constant $c > o$ depending on the security parameter k. Throughout the work we use this new value of $\mathcal{CC}_\mathcal{P}(n)$ in our analysis.

Theorem 3.4.1. *The protocol $\mathcal{P}$ defined above is a cPIR protocol such that $\mathcal{CC}_\mathcal{P}(n) = \mathcal{O}(e^{c\sqrt{\ln n}})$, for some $c > 0$.*

Proof. We have to prove that $\mathcal{P}$ verifies the definition 2.0.2 and that it has the required communication complexity.

1. (Correctness) It follows immediately from the correctness of $\mathcal{B}$. Formally, we prove by induction that at the end of Level$_l$ $\mathcal{U}$ retrieves the $i_l{}^{\text{th}}$ bit of x^l, for every $1 \leq l \leq L$.

   For $l = 1$ it is trivial since Level$_1$ is $\mathcal{B}$.

   Suppose it is true for $l-1$, then the j^{th} invocation of Level$_{l-1}$ allows the user to retrieve the $r_l^{*\text{th}}$ bit of $\big((j^{\text{th}} \text{ bit of } a_1^l), \ldots, (j^{\text{th}} \text{ bit of } a_{R_l}^l)\big)$, that is the j^{th} bit of $a_{r_l^*}^l$. Thus $\mathcal{U}$ obtains $a_{r_l^*}^l$ and he can use it as in $\mathcal{B}$ to retrieve $x_{i_l}^l$.

2. (Privacy) The proof is similar to the one of the basic scheme $\mathcal{B}$, but it is a little bit more delicate. Again, we assume by contradiction that there exists a family of polynomial-time circuits $\{\mathcal{C}_j\}$ that for some indices i and i' can distinguish the distribution D_i of the queries on i from the distribution $D_{i'}$ of the queries on i' with probability at least $\frac{1}{P(n)}$, for some non-constant P polynomial. That is:

$$\big|\Pr\left[\mathcal{C}_n(Q) = 1 | Q \in D_i\right] - \Pr\left[\mathcal{C}_n(Q) = 1 | Q \in D_{i'}\right]\big| \geq \frac{1}{P(n)}.$$

   As before, we can suppose $\Pr[\mathcal{C}_n(Q) = 1 | Q \in D_{i'}] - \Pr[\mathcal{C}_n(Q) = 1 | Q \in D_i] \geq \frac{1}{P(n)}$.

   Analyzing the protocol we see that a query consists of N followed by a (fixed-length) sequence of numbers $q_1, \ldots, q_m$ in $\mathbb{Z}_N^1$. Let I (resp. I') be the subset of $\mathcal{I}_m$ of all positions containing pseudo-quadratic residues modulo N when the index is i (resp. i'). Clearly i and i' must be such that $I \neq I'$, otherwise the protocol works in a identical way and so it is impossible to distinguish D_i from $D_{i'}$.

   We now use $\mathcal{C}_n$ to construct a circuit $\mathcal{C}'$ which solves the Quadratic Residuosity Problem on N.

   Circuit: C'

   - `INPUT`$\leftarrow (y)$, with $y \in \mathbb{Z}_N^1$,
   - Choose $\overline{I} \in_R \{I, I'\}$,

- For every $1 \leq j \leq m$, let $q_j \in \mathbb{Z}_N^1$ be such that:
$$\begin{cases} q_j \in_R \mathscr{QR}_N, & \text{if } j \notin \overline{I}, \\ q_j = yr^2 \mod N, \text{ with } r \in_R \mathbb{Z}_N^*, & \text{if } j \in \overline{I}, \end{cases}$$
- Let $b \in \mathbb{Z}_2$ be such that:
$$\begin{cases} b = 1 \oplus \mathcal{C}_n\big((q_1, \ldots, q_m)\big), & \text{if } I = \overline{I}, \\ b = \mathcal{C}_n\big((q_1, \ldots, q_m)\big), & \text{otherwise,} \end{cases}$$
- $\texttt{OUTPUT} \rightarrow (b)$.

If $y \in \mathscr{QR}_N$, then $(q_1, \ldots, q_m)$ is a sequence of elements in $\mathscr{QR}_N$, whatever is $\overline{I}$. On the contrary, when $y \in \mathscr{PQR}_N$, $(q_1, \ldots, q_m) \in D_i$ if $\overline{I} = I$ and $(q_1, \ldots, q_m) \in D_{i'}$ otherwise. Thus we can proceed exactly as in the proof for $\mathcal{B}$.

3. (Communication complexity) For all the executions of Level_l, $\mathcal{U}$ sends $(q_1^l \ldots, q_{C_l}^l)$, with $q_c^l \in \mathbb{Z}_N^*$ (we stress that he sends only one query valid for all the executions); so $\mathcal{U}$ sends kC_l bits. $\mathcal{S}$ replies sending $(a_1^l, \ldots, a_{R_l}^l)$, with $a_r^l \in \mathbb{Z}_N^*$; so $\mathcal{S}$ sends kR_l bits.

   To compute the communication complexity we need to fix R_l and C_l. We set $C_l = n^{1/(L+1)}$ for every $l \in \mathcal{I}_L$. We can see by induction that $R_l = n^{l/(L+1)}$ for every $L \geq l \geq 1$.

   If $l = L$, then $R_L = \frac{n_L}{C_L} = \frac{n}{n^{1/L+1}} = n^{L/(L+1)}$.

   Suppose it is true for $l+1$, then $R_l = \frac{n_l}{C_l} = \frac{R_{l+1}}{C_l} = \frac{n^{(l+1)/(L+1)}}{n^{1/(L+1)}} = n^{l/(L+1)}$.

   In this setting, $\mathcal{U}$ sends $kn^{1/(L+1)}$ bit for each level, that is he sends $Lkn^{1/(L+1)}$ bits. $\mathcal{S}$ sends his answer only when he performs Level_1 and for each execution of Level_1 he sends $kR_1 = kn^{1/(L+1)}$ bits. To conclude we have to calculate how many executions of Level_1 are needed.

   Remark that for each execution of Level_l, we need k executions of Level_{l-1}. We prove by induction that Level_l is executed k^{L-l} times, for every $L \geq l \geq 1$.

   If $l = L$, then it is trivial since Level_L is clearly executed once.

   Suppose it is true for $l+1$, then Level_l is executed $kk^{L-(l+1)} = k^{L-l}$ times.

   Therefore we have k^{L-1} executions of Level_1 and so $\mathcal{S}$ sends $k^{L-1}kn^{1/(L+1)} = k^L n^{1/(L+1)}$ bits. Hence the total amount of communication is:
$$\mathcal{CC}_{\mathcal{P}}(n) = k + kLn^{1/(L+1)} + k^L n^{1/(L+1)} = n^{1/(L+1)}(kL + k^L) + k.$$

The security parameter k is fixed and small (with respect to n). We have to choose the number of levels of recursion L in such a way that it minimizes the communication complexity. Assuming that the $\mathcal{CC}_{\mathcal{P}}(n)$ is differentiable with respect to L and considering k and n as parameters, we can look for a minimum studying the first derivative of $\mathcal{CC}_{\mathcal{P}}(n)$ with respect to L:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{CC}_{\mathcal{P}}(n)}{\mathrm{d}L} &= -\frac{n^{1/(L+1)}\ln n}{(L+1)^2}(Lk+k^L)+n^{1/(L+1)}(k+k^L\ln k)= \\ &= kn^{1/(L+1)}\left(1+k^{(L-1)}\ln k-\frac{(L+k^{(L-1)})\ln n}{(L+1)^2}\right).\end{aligned}$$

We have:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{CC}_{\mathcal{P}}(n)}{\mathrm{d}L}>0 &\iff 1+k^{(L-1)}\ln k-\frac{(L+k^{(L-1)})\ln n}{(L+1)^2}>0 \\ &\iff (L+1)^2(1+k^{L-1}\ln k)>(L+k^{L-1})\ln n \\ &\iff |L+1|=L+1>\sqrt{\frac{(L+k^{L-1})\ln n}{1+k^{L-1}\ln k}}\approx \\ &\approx\sqrt{\frac{(L+k^{L-1})\ln n}{k^{L-1}\ln k}}\approx\sqrt{\frac{\ln n}{\ln k}}.\end{aligned}$$

Thus the best choice is $L\approx\sqrt{\frac{\ln n}{\ln k}}-1$. With such a L we have:

$$\begin{aligned}n^{1/(L+1)} &\approx n^{\sqrt{\ln k/\ln n}}, \\ k^L &\approx k^{\sqrt{\ln n/\ln k}-1}=(n^{\log_n k})^{\sqrt{\ln n/\ln k}-1}=n^{\frac{\ln k}{\ln n}\frac{\sqrt{\ln n}}{\sqrt{\ln k}}}=n^{\sqrt{\ln k/\ln n}}.\end{aligned}$$

Since $n\gg k$, $n^{1/(L+1)}\approx k^L$. Therefore:

$$\begin{aligned}\mathcal{CC}_{\mathcal{P}}(n) &= n^{1/(L+1)}(kL+k^L)+k\approx n^{1/(L+1)}\left(kL+n^{1/(L+1)}\right)= \\ &= n^{2/(L+1)}+kLn^{1/(L+1)}=\mathcal{O}\big(n^{1/L}\big)=\mathcal{O}\big(n^{\sqrt{\ln k/\ln n}}\big)= \\ &= \mathcal{O}\big(e^{\ln n\cdot\sqrt{\ln k/\ln n}}\big)=\mathcal{O}\big(e^{\sqrt{\ln n\ln k}}\big).\end{aligned}$$

If we choose as security parameter $k=e^{c^2}$, we have $c=\sqrt{\ln k}$ and so $\mathcal{CC}_{\mathcal{P}}(n)=\mathcal{O}\big(e^{c\sqrt{\ln n}}\big)$ as wanted.

□

At each level, $\mathcal{U}$ sends $n^{1/(L+1)}$ elements of $\mathbb{Z}_N^1$ and all but one are in $\mathscr{QR}_N$. Since $|\mathscr{QR}_N| = \frac{\phi(N)}{4}$, in order to avoid repetitions (which would reveal where the pseudo-quadratic residue is not), we must choose the security parameter k in such a way that $\frac{\phi(N)}{4} \geq n^{1/(L+1)}$. We have $k = \lceil \log N \rceil = 2\lceil \log p_1 \rceil = 2\lceil \log p_2 \rceil$ and $\phi(N) = N - p_1 - p_2 + 1$, thus, we must have:

$$\frac{2^k - 2^{k/2+1} + 1}{4} = 2^{k-2} - 2^{k/2-1} + \frac{1}{4} \geq n^{1/(L+1)}.$$

At each execution of Level$_1$, $\mathcal{S}$ sends $n^{1/(L+1)}$ elements of $\mathbb{Z}_N^*$ to $\mathcal{U}$, while $\mathcal{U}$ is only interested in one of them. We will see that on one hand it make possible to retrieve blocks of bits, but on the other hand it makes the protocol insecure against honest-but-curious users (that is the protocol enables the user to retrieve also other bits of the database).

To well understand how the protocol works consider the following example.

Example 3.4.2. *Let $p_1 = 3$ and $p_2 = 5$; therefore $N = 15$, $k = \lceil \log N \rceil = 4$, $\mathscr{QR}_{15} = \{1, 4\}$ and $\mathscr{PQR}_{15} = \{2, 8\}$.*

Let $n = 16$ and let $i = 14$ be the index of the bit $\mathcal{U}$ wants to retrieve. Let $x = (0111100010111\mathbf{1}01)$.

Let $L = 3$; thus $C_l = 16^{1/4} = 2$ for every $l = 1, 2, 3$ and $R_3 = 16^{3/4} = 8$, $R_2 = 16^{2/4} = 4$ and $R_1 = 16^{1/4} = 2$.

At Level$_3$ we consider x as a 8×2 bit-matrix x^3; so the index of the desired bit $i_3 = 14$ is associated with the pair $(r_3^*, c_3^*) = (7, 2)$ (in fact $(7-1)2+2 = 14$).

At Level$_2$ we view $\mathcal{S}$'s answer as four 4×2 bit-matrices x^2; the index of the desired bit is $i_2 = r_3^* = 7$ and it is associated with the pair $(r_2^*, c_2^*) = (4, 1)$.

At Level$_1$ we see each $\mathcal{S}$'s answer coming from Level$_2$ as four 2×2 bit-matrices x^1; the index of the desired bit is $i_1 = r_2^* = 4$ and it is associated with the pair $(r_1^*, c_1^*) = (2, 2)$.

Therefore $\mathcal{U}$ must send a query $(q_1^3, \mathbf{q_2^3}, \mathbf{q_1^2}, q_2^2, q_1^1, \mathbf{q_2^1}) \in_R (\mathscr{QR}_N \times \mathscr{PQR}_N) \times$

$(\mathscr{P}\mathscr{Q}\mathscr{R}_N \times \mathscr{Q}\mathscr{R}_N) \times (\mathscr{Q}\mathscr{R}_N \times \mathscr{P}\mathscr{Q}\mathscr{R}_N)$. *Let* $(q_1^3, \boldsymbol{q_2^3}, \boldsymbol{q_1^2}, q_2^2, q_1^1, \boldsymbol{q_2^1}) = (1, \mathbf{8}, \mathbf{8}, 4, 4, \mathbf{2})$.

$$x^3 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & \mathbf{1} \\ 0 & 1 \end{pmatrix} \overset{\mathscr{S}computes}{\longrightarrow} \begin{cases} a_1^3 = 1^0 8^1 = 8 = (1000) \\ a_2^3 = 1^1 8^1 = 8 = (1000) \\ a_3^3 = 1^1 8^0 = 1 = (0001) \\ a_4^3 = 1^0 8^0 = 1 = (0001) \\ a_5^3 = 1^1 8^0 = 1 = (0001) \\ a_6^3 = 1^1 8^1 = 8 = (1000) \\ \boldsymbol{a_7^3 = 1^1 8^1 = 8 = (1000)} \\ a_8^3 = 1^0 8^1 = 8 = (1000) \end{cases}$$

Answer: $(1000\,1000\,0001\,0001\,0001\,1000\,\mathbf{1000}\,1000)$

1. $x^2 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ \mathbf{1} & 1 \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^2 = 8^1 4^1 \equiv 2 \\ a_2^2 = 8^0 4^0 = 1 \\ a_3^2 = 8^0 4^1 = 4 \\ \boldsymbol{a_4^2 = 8^1 4^1 \equiv 2} \end{cases}$ $(0010\,0001\,0100\,\mathbf{0010})$

1.1. $x^1 = \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^0 = 1 \\ \boldsymbol{a_2^1 = 4^0 2^0 = 1} \end{cases}$ $(0001\,\mathbf{0001})$

1.2. $x^1 = \begin{pmatrix} 0 & 0 \\ 1 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^0 = 1 \\ \boldsymbol{a_2^1 = 4^1 2^0 = 4} \end{cases}$ $(0001\,\mathbf{0100})$

1.3. $x^1 = \begin{pmatrix} 1 & 0 \\ 0 & \mathbf{1} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^1 2^0 = 4 \\ \boldsymbol{a_2^1 = 4^0 2^1 = 2} \end{cases}$ $(0100\,\mathbf{0010})$

1.4. $x^1 = \begin{pmatrix} 0 & 1 \\ 0 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^1 = 2 \\ \boldsymbol{a_2^1 = 4^0 2^0 = 1} \end{cases}$ $(0010\,\mathbf{0001})$

2. $x^2 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \mathbf{0} & 0 \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^2 = 8^0 4^0 = 1 \\ a_2^2 = 8^0 4^0 = 1 \\ a_3^2 = 8^0 4^0 = 1 \\ \boldsymbol{a_4^2 = 8^0 4^0 = 1} \end{cases}$ $(0001\,0001\,0001\,\mathbf{0001})$

2.1. $x^1 = \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^0 = 1 \\ \boldsymbol{a_2^1 = 4^0 2^0 = 1} \end{cases}$ $(0001\,\mathbf{0001})$

2.2. $x^1 = \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^0 = 1 \\ \boldsymbol{a_2^1 = 4^0 2^0 = 1} \end{cases}$ $(0001\,\mathbf{0001})$

2.3. $x^1 = \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^0 = 1 \\ \boldsymbol{a_2^1 = 4^0 2^0 = 1} \end{cases}$ $(0001\,\mathbf{0001})$

2.4. $x^1 = \begin{pmatrix} 1 & 1 \\ 1 & \mathbf{1} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^1 2^1 = 8 \\ \boldsymbol{a_2^1 = 4^1 2^1 = 8} \end{cases}$ $(1000\,\mathbf{1000})$

3. $x^2 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \mathbf{0} & 0 \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^2 = 8^0 4^0 = 1 \\ a_2^2 = 8^0 4^0 = 1 \\ a_3^2 = 8^0 4^0 = 1 \\ \boldsymbol{a_4^2 = 8^0 4^0 = 1} \end{cases}$ $(0001\,0001\,0001\,\mathbf{0001})$

3.1. $x^1 = \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^0 = 1 \\ \boldsymbol{a_2^1 = 4^0 2^0 = 1} \end{cases}$ $(0001\,\mathbf{0001})$

3.2. $x^1 = \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^0 = 1 \\ \boldsymbol{a_2^1 = 4^0 2^0 = 1} \end{cases}$ $(0001\,\mathbf{0001})$

3.3. $x^1 = \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^0 = 1 \\ \boldsymbol{a_2^1 = 4^0 2^0 = 1} \end{cases}$ $(0001\,\mathbf{0001})$

3.4. $x^1 = \begin{pmatrix} 1 & 1 \\ 1 & \mathbf{1} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^1 2^1 = 8 \\ \boldsymbol{a_2^1 = 4^1 2^1 = 8} \end{cases}$ $(1000\,\mathbf{1000})$

4. $x^2 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \\ \mathbf{0} & 0 \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^2 = 8^0 4^0 = 1 \\ a_2^2 = 8^1 4^1 \equiv 2 \\ a_3^2 = 8^1 4^0 = 8 \\ \boldsymbol{a_4^2 = 8^0 4^0 = 1} \end{cases}$ $(0001\,0010\,1000\,\mathbf{0001})$

4.1. $x^1 = \begin{pmatrix} 0 & 0 \\ 1 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^0 = 1 \\ \boldsymbol{a_2^1 = 4^1 2^0 = 4} \end{cases}$ $(0001\,\mathbf{0100})$

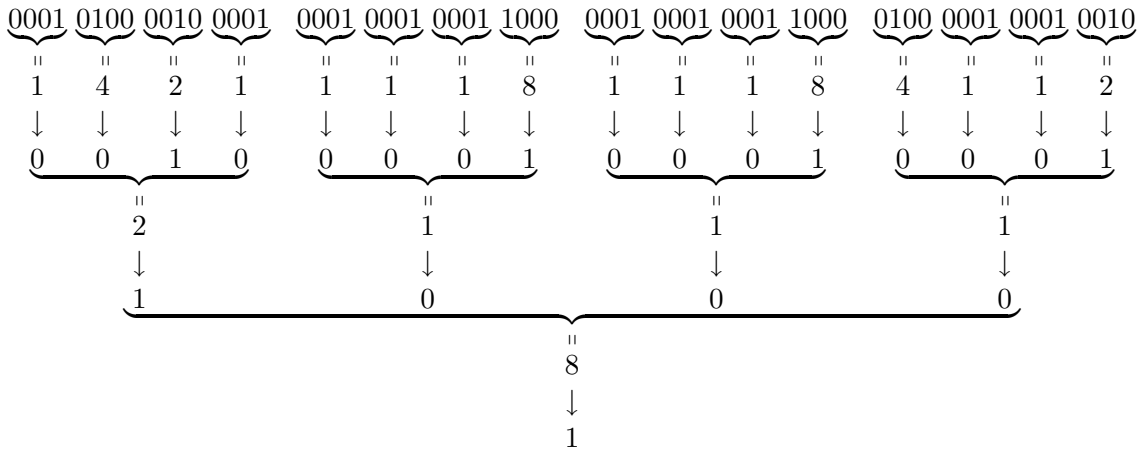
4.2. $x^1 = \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^0 = 1 \\ \boldsymbol{a_2^1 = 4^0 2^0 = 1} \end{cases}$ $(0001\,\mathbf{0001})$

4.3. $x^1 = \begin{pmatrix} 0 & 1 \\ 0 & \mathbf{0} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^0 2^1 = 2 \\ \boldsymbol{a_2^1 = 4^0 2^0 = 1} \end{cases}$ $(0010\,\mathbf{0001})$

4.4. $x^1 = \begin{pmatrix} 1 & 0 \\ 0 & \mathbf{1} \end{pmatrix} \overset{\mathscr{S}}{\rightarrow} \begin{cases} a_1^1 = 4^1 2^0 = 4 \\ \boldsymbol{a_2^1 = 4^0 2^1 = 2} \end{cases}$ $(0100\,\mathbf{0010})$

$\mathcal{U}$ receives all the answer strings of the last step, but he only needs the parts in bold type. He uses them to reconstruct the desired bit x_{14} in the following

way.



With ıı *we mean the change from binary numeral system to decimal system and with* ↓ *the computation of the quadratic residuosity: a quadratic (resp. pseudo-quadratic) residue modulo* 15 *is associated with* 0 *(resp.* 1*).*

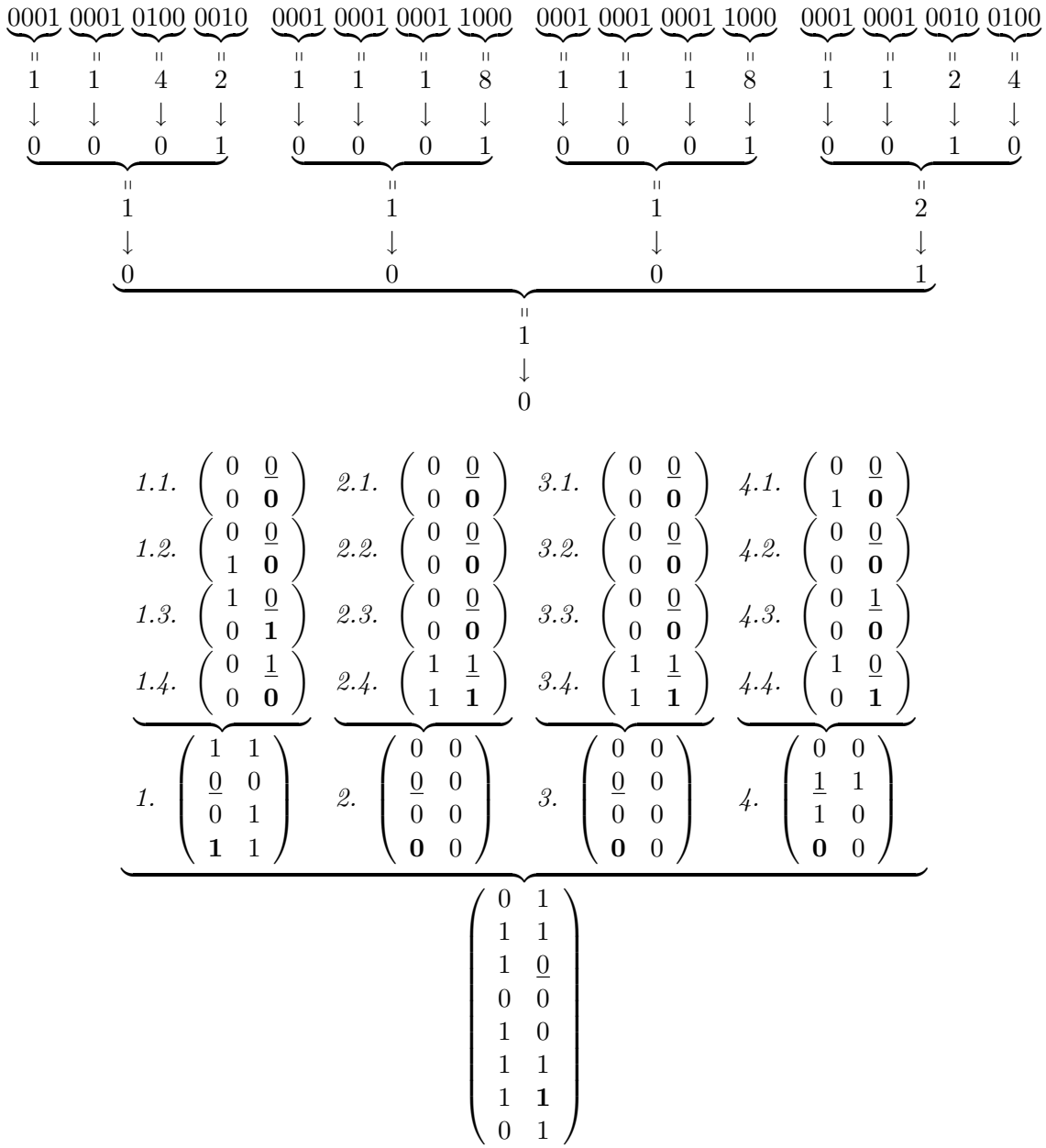
3.5 Analysis of $\mathcal{P}$

The user can cheat only sending more than one pseudo-quadratic residues in some queries. It is easy to see that if $\mathcal{U}$ sends $q_{c^*}, q_{c'^*} \in \mathscr{PQR}_N$ in the basic scheme, he can learn only the value of $x_{r^*,c^*} \oplus x_{r^*,c'^*}$, without distinguishing the single bits. Thus this behavior does not bring any remarkable advantage to the user.

We have already stressed that in each execution of Level$_1$ $\mathcal{S}$ sends $n^{1/(L+1)}$ elements of $\mathbb{Z}_N^1$ to $\mathcal{U}$, while $\mathcal{U}$ is only interested in one of them. As $\mathcal{U}$ can retrieve one bit of the database using one of the elements he receives from $\mathcal{S}$, he can retrieve $n^{1/(L+1)}$ bits of the database using all of them.

Example 3.5.1. *In example 3.4.2 in each Level*$_1$ $\mathcal{S}$ *sends* 2 *elements of* $\mathbb{Z}_{15}^1$ *to* $\mathcal{U}$*. We have shown how* $\mathcal{U}$ *uses each* 2^{nd} *element to retrieve the desired bit. We now show that* $\mathcal{U}$ *can identically use each* 1^{st} *element to retrieve another bit of*

the database.



It is possible for $\mathcal{U}$ to obtain more than one bit because there exist bits in the database that need queries of the same sort to be retrieved. Formally, using the notation i, i', I and I' as in the proof of Theorem 3.4.1, there exist distinct indices $i, i' \in \mathcal{I}_n$ such that $I = I'$. Our aim is to study how these indices must be in order that it happens.

To reduce the notation, let $C = C_1 = \ldots = C_L = n^{1/(L+1)}$. By construction I and I' are the subsets of $\mathcal{I}_{LC}$ of all positions of pseudo-quadratic residues modulo N in the queries, when the index is i and i' respectively; therefore

$I = \{c_L^*, c_{L-1}^*, \ldots, c_1^*\}$ and $I' = \{c'^*_L, c'^*_{L-1}, \ldots, c'^*_1\}$. Hence we want to know how i and i' should be to have $c_l^* = c'^*_l$ for every $l \in \mathcal{I}_L$.

For every $l \in \mathcal{I}_L$ and index $i \in \mathcal{I}_n$, we have $r_{l+1}^* = i_l = (r_l^* - 1)C + c_l^*$ with $r_l^* \in \mathcal{I}_{R_l}$ and $c_l^* \in \mathcal{I}_C$. Thus:

$$\begin{aligned} c_l^* = c'^*_l &\iff i_l - (r_l^* - 1)C = i'_l - (r'^*_l - 1)C \iff i_l - i'_l = (r_l^* - r'^*_l)C \\ &\iff r_{l+1}^* - r'^*_{l+1} = (r_l^* - r'^*_l)C. \end{aligned}$$

We prove by induction that, for every $L \geq l \geq 1$:

$$[(c_L^* = c'^*_L) \wedge (c_{L-1}^* = c'^*_{L-1}) \wedge \ldots \wedge (c_l^* = c'^*_l)] \iff i - i' = (r_l^* - r'^*_l)C^{L-l+1}.$$

If $l = L$, it is trivial since $i_L = i$ and $i'_L = i'$.

Suppose it is true for $l+1$, then:

$$\begin{aligned} (c_L^* = c'^*_L) \wedge \ldots \wedge (c_l^* = c'^*_l) &\iff \begin{cases} ((c_L^* = c'^*_L) \wedge \ldots \wedge (c_{l+1}^* = c'^*_{l+1}) \\ (c_l^* = c'^*_l) \end{cases} \\ &\iff \begin{cases} i - i' = (r_{l+1}^* - r'^*_{l+1})C^{L-(l+1)+1} \\ r_{l+1}^* - r'^*_{l+1} = (r_l^* - r'^*_l)C. \end{cases} \\ &\iff i - i' = (r_l^* - r'^*_l)CC^{L-l} = \\ &\quad = (r_l^* - r'^*_l)C^{L-l+1}. \end{aligned}$$

Therefore $c_l^* = c'^*_l$ for every $l \in \mathcal{I}_L$ if and only if $i - i' = (r_1^* - r'^*_1)C^L = (r_1^* - r'^*_1)n^{L/(L+1)}$. Hence $I' = I$ for every index $i' \in \mathcal{I}_n$ such that:

$$i' = i + \alpha n^{L/(L+1)} \mod n, \quad \forall \alpha \in \mathcal{I}_{n^{1/(L+1)}}. \tag{3.2}$$

Note that follows that $\mathcal{U}$ can retrieve exactly $n^{1/(L+1)}$ bits for each execution of protocol $\mathcal{P}$, as already claimed.

Example 3.5.2. *Let $n = 27$, $i = 11$ and $L = 2$. At Level$_2$ we view the database as a 9×3 bit-matrix and the index as the pair $(4, 2)$. Thus $\mathcal{U}$'s query at this level is $(q_1^2, q_2^2, q_3^2) \in \mathscr{QR}_N \times \mathscr{PQR}_N \times \mathscr{QR}_N$. $\mathcal{S}$'s answer consists of 9 elements of $\mathbb{Z}_N^1$. Computing their quadratic residuosity $\mathcal{U}$ could retrieve all the 9 bits situated in the same column as the desired bit (*i.e. *the whole* 2^{nd} *column). In fact the knowledge of the* 1^{st} *element of $\mathcal{S}$'s answer allows $\mathcal{U}$ to retrieve x_2, the knowledge of the 2^{nd} x_5 and so on (thus we simply denote the* 1^{st} *element of the $\mathcal{S}$'s answer as x_2, the 2^{nd} as x_5 and so on). In particular*

the knowledge of the 4^{th} *element of* $\mathcal{S}$*'s answer allows* $\mathcal{U}$ *to retrieve* x_{11}*. At* $Level_1$ *we consider* $\mathcal{S}$*'s answer as* k 3×3 *bit-matrices and* $\mathcal{U}$ *wants their* 4^{th} *element (that is the element in position* $(2,1)$*); thus at this level* $\mathcal{U}$*'s query is* $(q_1^1, q_2^1, q_3^1) \in \mathscr{PQR}_N \times \mathscr{QR}_N \times \mathscr{QR}_N$*. For each new matrix* $\mathcal{S}$ *sends his answer which consists of* 3 *elements of* $\mathbb{Z}_N^1$*.Computing their quadratic residuosity* $\mathcal{U}$ *can retrieve the the whole column containing* x_{11} *(*i.e. *the whole* 1^{st} *column).*

$$\begin{pmatrix} x_1 & \boxed{\underline{x_2}} & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \\ x_{10} & \boxed{\boldsymbol{x_{11}}} & x_{12} \\ x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} \\ x_{19} & \boxed{\underline{x_{20}}} & x_{21} \\ x_{22} & x_{23} & x_{24} \\ x_{25} & \boxed{x_{26}} & x_{27} \end{pmatrix} \qquad \begin{pmatrix} \boxed{x_2} & x_5 & x_8 \\ \boxed{\boldsymbol{x_{11}}} & x_{14} & x_{17} \\ \boxed{x_{20}} & x_{23} & x_{26} \end{pmatrix}$$

Actually $11+1\cdot 9 = 20$*,* $11+2\cdot 9 = 29 = 2 \mod 27$ *and* $11+3\cdot 9 = 11 \mod 27$*, as in formula* (3.2)*.*

3.6 Limits of $\mathcal{P}$

Formula (3.2) implies that $I = I'$ only if $|i - i'| \geq n^{L/(L+1)}$. Since $L \geq 1$, it follows that when $|i - i'| < \sqrt{n}$, it is impossible to retrieve x_i and $x_{i'}$ sending only one query. It gives a not negligible constraint on the indices.

Thus, $\mathcal{P}$ can be used to retrieve $n^{1/(L+1)}$ bits of the database, provided that are in well precise relative positions. This is not very useful because the classical situations are:

1. $\mathcal{U}$ wants some bits in any positions;

2. $\mathcal{U}$ wants a block of bits (*i.e.* some consecutive bits).

Our idea is to construct new protocols which solve these two issues, using $\mathcal{P}$ as starting point. In particular, we require our protocol to be based on QRA, without needing stronger assumptions, exactly as $\mathcal{P}$. In the next chapter we will present these protocols.

Chapter 4

Starting from $\mathcal{P}$: Some new cPIR Protocols

In this chapter we present our protocols which are constructed using the basic scheme $\mathcal{P}$ as starting point. More precisely, our idea is to modify $\mathcal{P}$ in order to obtain cPIR protocols which enable the user to retrieve a block of bits or several bits in any position, without using stronger assumptions. We actually prove that QRA is enough to guarantee the privacy of our protocols.

Throughout this chapter we use the notation introduced in Chapter 3

4.1 Variation $\mathcal{P}'$: Retrieve blocks of bits

In the PIR standard model $\mathcal{U}$ wants one bit, but it is more realistic that $\mathcal{U}$ wants a *block of bits*. The problem is stated as follows: The database is broken up into n/m blocks of m bits each and the user wants to privately retrieve a block. Of course this problem can be solved by m iteration of a PIR scheme. The question of the existence of a better solution was raised by Chor et al. [7].

In this section we show how we can transform the basic protocol $\mathcal{P}$ in order enable the user to retrieve a block of bits. Informally, what we do is to make the $n^{1/(L+1)}$ bits $\mathcal{U}$ can retrieve using $\mathcal{P}$ to form a block. In particular our idea is inverting rows with columns: At every level of the recursive scheme the user sends an element of $\mathbb{Z}_N^1$ for each row of the database matrix and we fix one number of rows for all the levels.

Variation $\mathcal{P}'$: Basic scheme

$\mathcal{Q}$: $\quad -$ INPUT$\leftarrow (1^n, i = (r^*, c^*))$,

– For every $1 \leq r \leq R$, choose $q_r \in_R \mathbb{Z}_N^*$ such that:

$$\begin{cases} q_{r^*} \in \mathscr{P}\mathscr{Q}\mathscr{R}_N, \\ q_r \in \mathscr{Q}\mathscr{R}_N \qquad \forall c \neq c^*, \end{cases}$$

– $\texttt{OUTPUT}\rightarrow Q = (q_1, \ldots, q_R)$;

$\mathcal{A}$: – $\texttt{INPUT}\leftarrow (x, Q)$,

– For every $1 \leq c \leq C$, let:

$$a_c = \textstyle\prod_{r=1}^{R} (q_r)^{x_{r,c}} \mod N,$$

– $\texttt{OUTPUT}\rightarrow (a_1, \ldots, a_C)$;

$\mathcal{R}$: – $\texttt{INPUT}\leftarrow (1^n, i = (r^*, c^*), a_1, \ldots, a_C)$,

– Let $b \in \mathbb{Z}_2$ be such that:

$$b = 0 \Leftrightarrow a_{c^*} \in \mathscr{Q}\mathscr{R}_N,$$

– $\texttt{OUTPUT}\rightarrow (b)$.

Variation $\mathcal{P}'$: Recursive scheme

For every $L \geq l \geq 1$ do Level$_l$:

- View the database x^l as a $R_l \times C_l$ bit-matrix (thus $R_l C_l = n_l$). Let $i_l = (r_l^*, c_l^*)$ be the index of the bit $\mathcal{U}$ wants to retrieve. $\mathcal{U}$ and $\mathcal{S}$ simulate the basic scheme with this setting.
- If $l > 1$, then $\mathcal{S}$ does not send his answer $(a_1^l, \ldots, a_{C_l}^l)$ to $\mathcal{U}$ but he considers it as k new databases:

  For every $1 \leq j \leq k$:

  – Let $x^{l-1} = \big((j^{\text{th}} \text{ bit of } a_1^l), \ldots, (j^{\text{th}} \text{ bit of } a_{C_l}^l)\big)$
  Thus $n_{l-1} = |x^{l-1}| = C_l$,

  – $i_{l-1} = c_l^*$ (since $\mathcal{U}$ is only interested in $a_{c_l^*}^l$),

  – $\mathcal{U}$ and $\mathcal{S}$ go to Level$_{l-1}$ with x^{l-1} as database and i_{l-1} as index.
- If $l = 1$, $\mathcal{S}$ sends his answer $(a_1^1, \ldots, a_{C_1}^1)$ to $\mathcal{U}$.
- $\mathcal{U}$ uses $a_{c_l^*}^l$ as in $\mathcal{B}$ to retrieve $x_{i_l}^l$.

Clearly correctness and privacy of $\mathcal{P}'$ are equivalent to those of protocol $\mathcal{P}$. This time we fix $R_l = n^{1/(L+1)}$, and thus $C_l = n^{l/(L+1)}$, for every $l \in \mathcal{I}_L$. Since $C_1 = n^{1/(L+1)}$, $\mathcal{P}'$ also has the same communication complexity of $\mathcal{P}$.

In this new scenario, for any indices $i, i' \in \mathcal{I}_n$, $I = I'$ if and only if $r_l^* = r'^*_l$ for every $l \in \mathcal{I}_L$. Moreover $c^*_{l+1} = i_l = (r_l^* - 1)n^{l/(L+1)} + c_l^*$, with $(r_l^*, c_l^*) \in \mathcal{I}_{n^{1/(L+1)}} \times \mathcal{I}_{n^{l/(L+1)}}$, since this time is the number of columns that changes. Therefore:

$$\begin{aligned} r_l^* = r'^*_l &\iff \frac{i_l - c_l^*}{n^{l/(L+1)}} + 1 = \frac{i'_l - c'^*_l}{n^{l/(L+1)}} + 1 \iff i_l - i'_l = c_l^* - c'^*_l \\ &\iff i_l - i'_l = i_{l-1} - i'_{l-1}. \end{aligned}$$

Thus $r_l^* = r'^*_l$ for every $l \in \mathcal{I}_L$ if and only if $i - i = i_L - i'_L = c_1^* - c'^*_1$ with $c_1^*, c'^*_1 \in \mathcal{I}_{n^{1/(L+1)}}$. Note that:

$$i = (r_L^* - 1)n^{L/(L+1)} + (r_{L-1}^* - 1)n^{(L-1)/(L+1)} + \ldots + (r_1^* - 1)n^{1/(L+1)} + c_1^*.$$

It implies that $i = c_1^* \mod n^{1/(L+1)}$. Therefore $I' = I$ for every index i' such that $i' = i - (i \mod n^{1/(L+1)}) + \alpha$, for every $\alpha \in \mathcal{I}_{n^{1/(L+1)}}$. It means that $\mathcal{P}'$ allows the user to retrieve a block of $n^{1/(L+1)}$ bits sending just one query as he would retrieve only one bit (*i.e.* without increasing the communication complexity).

Example 4.1.1. *Consider the same situation of example 3.5.2, but with the new setting, that is switching rows and columns: Let* $n = 27$, $i = 11$ *and* $L = 2$ *as before. At Level*$_2$ *this time we view the database as a* 3×9 *bit-matrix and the index* i *is associated with the pair* $(2, 2)$. $\mathcal{U}$*'s query at this level is* $(q_1^2, q_2^2, q_3^2) \in \mathscr{QR}_N \times \mathscr{PQR}_N \times \mathscr{QR}_N$. $\mathcal{S}$*'s answer consists of* 9 *elements of* $\mathbb{Z}_N^1$. *Computing their quadratic residuosity* $\mathcal{U}$ *could retrieve all the* 9 *bits situated in the same row as the desired bit (*i.e. *the whole* 2^{nd} *row). At Level*$_1$ *we consider* $\mathcal{S}$*'s answer as* k 3×3 *bit-matrices and* $\mathcal{U}$ *wants their* 2^{th} *element (that is the element in position* $(1, 2)$*); thus at this level* $\mathcal{U}$*'s query is* $(q_1^1, q_2^1, q_3^1) \in \mathscr{PQR}_N \times \mathscr{QR}_N \times \mathscr{QR}_N$. *For each new matrix* $\mathcal{S}$ *sends his answer which consists of* 3 *elements of* $\mathbb{Z}_N^1$*.Computing their quadratic residuosity* $\mathcal{U}$ *can retrieve the the whole row containing the* 2^{th} *element (*i.e. *the whole* 1^{st} *row).*

$$\left(\begin{array}{ccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \\ \underline{x_{10}} & \underline{\boldsymbol{x_{11}}} & \underline{x_{12}} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} \\ x_{19} & x_{20} & x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} \end{array}\right) \qquad \left(\begin{array}{ccc} x_{10} & \boldsymbol{x_{11}} & x_{12} \\ x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} \end{array}\right)$$

Actually $11 - (11 \mod 3) + 1 = 10$, $11 - (11 \mod 3) + 2 = 11$ *and* $11 - (11 \mod 3) + 3 = 12$, *as in the formula we have found.*

4.2 $\mathcal{N}$: a new cPIR protocol using Quartic Residues

In this section we present a new scheme: It does not need stronger assumption than QRA but it allows the user to retrieve two bits in any position sending only one query. Moreover, with a slight modification, it leads to a cPIR scheme which enables the user to obtain a block of 2 bits or, equivalently, to a cPIR scheme with communication complexity smaller than $\mathcal{P}$.

Our protocol has the same structure of $\mathcal{P}$: we consider the database as a matrix and, starting from a basic scheme, we construct a recursive scheme.

$\mathcal{P}$ is based on the partition $\mathbb{Z}_N^1 = \mathscr{2R}_N \dot\cup \mathscr{P}\mathscr{2R}_N$; our idea is generalize it considering different partitions of $\mathbb{Z}_N^1$ in order to improve the protocol, that is to enable the user to retrieve more than one bits.

Two bits form a system that can assume 4 different values; thus, to construct a protocol which allows the user to retrieve more than one bits, we must divide $\mathbb{Z}_N^1$ at least into 4 distinct subsets. The simplest case is to use quartic residues modulo N instead of quadratic residues.

Let $\mathscr{4R}_N$ be the set of quartic residues modulo N, that is:

$$\mathscr{4R}_N = \{y \in \mathbb{Z}_N^* | \exists z \in \mathbb{Z}_N^* \text{ s.t. } y = z^4 \mod N\}.$$

Clearly $\mathscr{4R}_N \subset \mathscr{2R}_N$.

Lemma 4.2.1. *Let $N = p_1p_2$, with $p_1 \neq p_2$ prime numbers; let $a \in \mathscr{P}\mathscr{2R}_N$. Then:*

$$\mathbb{Z}_N^1 = \mathscr{4R}_N \dot\cup a\mathscr{4R}_N \dot\cup a^2\mathscr{4R}_N \dot\cup a^3\mathscr{4R}_N$$

Proof. Notice that $\mathscr{4R}_N \cup a^2\mathscr{4R}_N = \mathscr{2R}_N$ and $a\mathscr{4R}_N \cup a^3\mathscr{4R}_N = \mathscr{P}\mathscr{2R}_N$. Thus it is enough to prove that $\mathscr{4R}_N \cap a^2\mathscr{4R}_N = \emptyset = a\mathscr{4R}_N \cap a^3\mathscr{4R}_N$. We split the proof in two steps.

- $\mathscr{4R}_N \cap a^2\mathscr{4R}_N = \emptyset$. An element in $a^2\mathscr{4R}_N$ is of the form a^2y^4 for some $y \in \mathbb{Z}_N^*$. It is in $\mathscr{4R}_N$ if and only if there exists $z \in \mathbb{Z}_N^*$ such that $a^2y^4 = z^4$ which is impossible since $a \in \mathscr{P}\mathscr{2R}_N$.

- $a\mathscr{4R}_N \cap a^3\mathscr{4R}_N = \emptyset$. An element in $a\mathscr{4R}_N$ is of the form ay^4 for some $y \in \mathbb{Z}_N^*$. It is in $a^3\mathscr{4R}_N$ if and only if there exists $z \in \mathbb{Z}_N^*$ such that $ay^4 = a^3z^4$. Multiplying by a we obtain $a^2y^4 = a^4z^4$ which is impossible since $a \in \mathscr{P}\mathscr{2R}_N$.

□

Clearly $|a^j 4\mathscr{R}_N| = \frac{1}{4}|\mathbb{Z}_N^1| = \frac{1}{8}|\mathbb{Z}_N^*| = \frac{\phi(N)}{8}$, for every $j \in \{0,1,2,3\}$.

4.2.1 $\mathcal{N}_1$: Retrieve 2 bits in any position

This protocol enables the user to retrieve 2 bits in any position.

$\mathcal{N}_1$: Basic scheme

Let $k \in \mathbb{N}$ be the security parameter. Let i and i' be the indices of the desired bits. We view the database as a $R \times C$ matrix and i and i' are associated with the pairs (r^*, c^*) and (r'^*, c'^*) in $\mathcal{I}_R \times \mathcal{I}_C$ respectively.

$\mathcal{Q}$: – INPUT$\leftarrow (1^n, i = (r^*, c^*), i' = (r'^*, c'^*))$,

– Choose at random $p_1 \neq p_2$ prime numbers such that $|p_1| = |p_2| = k/2$,

– Let $N = p_1 p_2$,

– For every $1 \leq c \leq C$, choose $q_c \in_R \mathbb{Z}_N^1$ such that:

$$\begin{cases} q_c \in 4\mathscr{R}_N \quad \forall c \neq c^*, c'^*, \\ \text{if } c^* \neq c'^* \Rightarrow \begin{cases} q_{c^*} \in a4\mathscr{R}_N, \\ q_{c'^*} \in a^2 4\mathscr{R}_N, \end{cases} \\ \text{if } c^* = c'^* = \overline{c} \Rightarrow q_{\overline{c}} \in a^3 4\mathscr{R}_N, \end{cases}$$

– OUTPUT$\rightarrow Q = (N, q_1, \ldots, q_C)$;

$\mathcal{A}$: – INPUT$\leftarrow (x, Q)$,

– For every $1 \leq r \leq R$, let:

$$a_r = \prod_{c=1}^{C} (q_c)^{x_{r,c}} \mod N,$$

– OUTPUT$\rightarrow (a_1, \ldots, a_R)$;

$\mathcal{R}$: – INPUT$\leftarrow (1^n, i = (r^*, c^*), i' = (r'^*, c'^*), a_1, \ldots, a_R)$,

– Let $b, b' \in \mathbb{Z}_2$ be such that:

$$\begin{cases} b = 0 \Leftrightarrow a_{r^*} \in 2\mathscr{R}_N, \\ b' = 0 \Leftrightarrow a_{r'^*} \in (4\mathscr{R}_N \dot{\cup} a4\mathscr{R}_N), \end{cases}$$

– OUTPUT$\rightarrow (b, b')$.

$\mathcal{N}_1$: Recursive scheme

We use this basic scheme to implement a recursive scheme exactly as for $\mathcal{P}$.

Theorem 4.2.2. *The protocol $\mathcal{N}_1$ defined above is a cPIR protocol which allows the user to retrieve any 2 bits of the database and such that $\mathcal{CC}_{\mathcal{N}_1}(n) = \mathcal{O}(e^{c\sqrt{\ln n}})$, for any $c > 0$.*

Proof. We have to prove that $\mathcal{N}_1$ verifies the definition 2.0.2 and that it has the required communication complexity.

1. (Correctness) To prove the correctness of the recursive scheme, we have to prove that the basic scheme is correct.

   If $c^* \neq c'^*$, then, for every $r \in \mathcal{I}_R$, $a_r = a^{x_{r,c^*}+2x_{r,c'^*}}y^4$, for some $y \in \mathbb{Z}_N^*$. Thus:

$$\begin{aligned}
a_{r^*} \in \mathscr{2R}_N &\iff x_{r^*,c^*} + 2x_{r^*,c'^*} = 0 \mod 2 \iff \\
&\iff [(x_{r^*,c^*}, x_{r^*,c'^*}) = (0,0)] \vee \\
&\quad \vee [(x_{r^*,c^*}, x_{r^*,c'^*}) = (0,1)] \iff \\
&\iff x_{r^*,c^*} = 0. \\
a_{r'^*} \in (\mathscr{4R}_N \dot\cup a\mathscr{4R}_N) &\iff [x_{r'^*,c^*} + 2x_{r'^*,c'^*} = 0 \mod 4] \vee \\
&\quad \vee [x_{r'^*,c^*} + 2x_{r'^*,c'^*} = 1 \mod 4] \iff \\
&\iff [(x_{r'^*,c^*}, x_{r'^*,c'^*}) = (0,0)] \vee \\
&\quad \vee [(x_{r'^*,c^*}, x_{r'^*,c'^*}) = (1,0)] \iff \\
&\iff x_{r'^*,c'^*} = 0.
\end{aligned}$$

   Therefore, if $c^* \neq c'^*$, $b = x_{r^*,c^*}$ and $b' = x_{r'^*,c'^*}$, whatever are r^* and r'^*.

   If $c^* = c'^* = \bar{c}$, then, for every $r \in \mathcal{I}_R$, $a_r = a^{3x_{r,\bar{c}}}y^4$, for some $y \in \mathbb{Z}_N^*$. Thus:

$$\begin{aligned}
a_{r^*} \in \mathscr{2R}_N &\iff 3x_{r^*,\bar{c}} = 0 \mod 2 \iff x_{r^*,\bar{c}} = 0. \\
a_{r'^*} \in (\mathscr{4R}_N \dot\cup a\mathscr{4R}_N) &\iff [3x_{r'^*,\bar{c}} = 0 \mod 4] \vee \\
&\quad \vee [3x_{r'^*,\bar{c}} = 1 \mod 4] \iff \\
&\iff x_{r'^*,\bar{c}} = 0.
\end{aligned}$$

   Therefore, if $c^* = c'^*$, $b = x_{r^*,c^*}$ and $b' = x_{r'^*,c'^*}$, whatever are r^* and r'^*.

2. (Privacy) We have to prove that under the QRA $\mathcal{S}$, given an element $y \in \mathbb{Z}_N^1$, cannot say anything about which set it belongs to.

   Under QRA $\mathcal{S}$ clearly cannot distinguish $4\mathscr{R}_N \dot\cup a^2 4\mathscr{R}_N = 2\mathscr{R}_N$ from $a4\mathscr{R}_N \dot\cup a^3 4\mathscr{R}_N = \mathscr{P}2\mathscr{R}_N$. Moreover $\mathcal{S}$ cannot distinguish $a4\mathscr{R}_N$ from $a^3 4\mathscr{R}_N$, since they depend on the choice of a and $\mathcal{S}$ does not know the value of a (which is randomly chosen in $\mathscr{P}2\mathscr{R}_N$ by $\mathcal{U}$). In fact, let $a, a' \in \mathscr{P}2\mathscr{R}_N$; being a pseudo-quadratic residue modulo N, $a' \in a4\mathscr{R}_N \dot\cup a^3 4\mathscr{R}_N$. Suppose $a' \in a^3 4\mathscr{R}_N$ (we can always find such an a', since $a^3 4\mathscr{R}_N \neq \emptyset$ and $a4\mathscr{R}_N \cap a^3 4\mathscr{R}_N = \emptyset$ as we have already seen) Then $a'4\mathscr{R}_N = a^3 4\mathscr{R}_N$ and so $\mathcal{S}$ cannot distinguish $a4\mathscr{R}_N$ from $a^3 4\mathscr{R}_N$ without knowing a.

   The last step is proving that $\mathcal{S}$ cannot distinguish $4\mathscr{R}_N$ from $a^2 4\mathscr{R}_N$. Remark that $4\mathscr{R}_N$ is the set of the quartic residues modulo N and $a^2 4\mathscr{R}_N$ is the set of the quadratic residues modulo N which are not quartic residues.

   By contradiction we assume that there exists a family of polynomial-time circuits $\{\mathcal{C}_j\}$ such that:

$$\left|\Pr\left[\mathcal{C}_k(y) = 1 | y \in 4\mathscr{R}_N\right] - \Pr\left[\mathcal{C}_k(y) = 1 | y \in a^2 4\mathscr{R}_N\right]\right| \geq \frac{1}{P(k)}$$

   for some non-constant P polynomial. We now use $\mathcal{C}_k$ to construct a new circuit $\mathcal{C}'$ which solves the Quadratic Residuosity Problem on N:

   - `INPUT`$\leftarrow (y)$, with $y \in \mathbb{Z}_N^1$,
   - Let $z = \mathcal{C}_k(y^2 \mod N)$,
   - `OUTPUT`$\rightarrow (z)$.

   If $y \in \mathscr{P}2\mathscr{R}_N$, then $(y^2 \mod N) \in a^2 4\mathscr{R}_N$ since it is a quadratic residue but not a quartic residue. If $y \in 2\mathscr{R}_N$, then $(y^2 \mod N) \in 4\mathscr{R}_N$. Thus:

$$\begin{aligned}&\left|\Pr\left[\mathcal{C}'(y) = 1 | y \in 2\mathscr{R}_N\right] - \Pr\left[\mathcal{C}'(y) = 1 | y \in \mathscr{P}2\mathscr{R}_N\right]\right| \;= \\ &= \left|\Pr\left[\mathcal{C}_k(y) = 1 | y \in 4\mathscr{R}_N\right] - \Pr\left[\mathcal{C}_k(y) = 1 | y \in a^2 4\mathscr{R}_N\right]\right| \;\geq\; \frac{1}{P(k)}.\end{aligned}$$

   Therefore QRA implies that for $\mathcal{S}$ all the elements of $\mathbb{Z}_N^1$ are indistinguishable, with respect to our partition. Remark that it means, not only that $\mathcal{S}$ cannot learn anything from the queries he receives, but also that he cannot infer anything from the answers he computes. Therefore the protocol $\mathcal{N}_1$ is secure.

3. (Communication complexity) We use the basic scheme exactly as in $\mathcal{P}$ to implement a recursive scheme in order to reduce the communication complexity. In this way we construct a protocol with the same complexity of $\mathcal{P}$ because $\mathcal{U}$ and $\mathcal{S}$ send the same amount of bits (the only difference is that $\mathcal{U}$ chooses the elements of his query with a different strategy). Therefore $\mathcal{CC}_{\mathcal{N}_1}(n) = \mathcal{O}\big(e^{c\sqrt{\ln n}}\big)$.

□

At each level $\mathcal{U}$ sends $n^{1/(L+1)}$ elements of $\mathbb{Z}_N^1$ such that all but one are in $\mathscr{4R}_N$. It implies that, to avoid repetitions, we must choose the security parameter k in such a way that $|\mathscr{4R}_N| = \frac{\phi(N)}{8} \geq n^{1/(L+1)}$, that is:

$$\frac{2^k - 2^{k/2+1} + 1}{8} = 2^{k-3} - 2^{k/2-2} + \frac{1}{8} \geq n^{1/(L+1)}.$$

Using protocol $\mathcal{P}$, the condition on k is $2^{k-2} - 2^{k/2-1} + \frac{1}{4} \geq n^{1/(L+1)}$, therefore, passing form $\mathcal{P}$ to $\mathcal{N}_1$, we must change the security parameter from k to $k' = k + 1$. In this way the new protocol still avoid repetitions.

Finally, it is important to remark that, since the basic scheme allows the user to retrieve 2 bits in any position, we do not need to pay attention on how the indices change in the passage from a level to the successive. Thus $\mathcal{N}_1$ allows the user to obtain 2 bits in any position. It implies that it can be equally used when $\mathcal{U}$ wants only one bit. In this case $\mathcal{N}_1$ acts exactly as $\mathcal{P}$: we have $i = i'$ and so $(r_l^*, c_l^*) = (r_l'^*, c_l'^*)$ for every $l \in \mathcal{I}_L$. Thus for each level of recursion, the query of $\mathcal{U}$ consists of $C_l - 1 = n^{1/(L+1)} - 1$ elements in $\mathscr{4R}_N \subset \mathscr{2R}_N$ and only one element in $a^3\mathscr{4R}_N \subset \mathscr{P2R}_N$.

4.2.2 $\mathcal{N}_2$: $\mathbb{Z}_4$ as alphabet (or retrive a block of 2 bits)

The partition $\mathbb{Z}_N^1 = \mathscr{4R}_N \dot\cup a\mathscr{4R}_N \dot\cup a^2\mathscr{4R}_N \dot\cup a^3\mathscr{4R}_N$ allows the user to distinguish among 4 different situations. It is natural to use it when the alphabet is $\mathbb{Z}_4$ instead of $\mathbb{Z}_2$, that is the database is $x = (x_1, \ldots, x_n)$ with $x_j \in \mathbb{Z}_4$, and the user wants only one item from it.

The structure of this protocol is the usual one: We consider the database as a matrix and, starting from a basic scheme, we construct a recursive scheme.

$\mathcal{N}_2$: Basic scheme

Let $k \in \mathbb{N}$ be the security parameter. Let i be the index of the desired item. We view the database as a $R \times C$ matrix with entries in $\mathbb{Z}_4$ and i is associated with the pair (r^*, c^*) in $\mathcal{I}_R \times \mathcal{I}_C$.

$\mathcal{Q}$: – INPUT$\leftarrow (1^n, i = (r^*, c^*))$,
– Choose at random $p_1 \neq p_2$ prime numbers such that $|p_1| = |p_2| = k/2$,
– Let $N = p_1 p_2$,
– For every $1 \leq c \leq C$, choose $q_c \in_R \mathbb{Z}_N^1$ such that:

$$\begin{cases} q_{c^*} \in a4\mathscr{R}_N, \\ q_c \in 4\mathscr{R}_N \qquad \forall c \neq c^*, \end{cases}$$

– OUTPUT$\rightarrow Q = (N, q_1, \ldots, q_C)$;

$\mathcal{A}$: – INPUT$\leftarrow (x, Q)$,
– For every $1 \leq r \leq R$, let:

$$a_r = \prod_{c=1}^{C} (q_c)^{x_{r,c}} \mod N,$$

– OUTPUT$\rightarrow (a_1, \ldots, a_R)$;

$\mathcal{R}$: – INPUT$\leftarrow (1^n, i = (r^*, c^*), a_1, \ldots, a_R)$,
– Let $b \in \mathbb{Z}_4$ be such that:

$$\begin{cases} b = 0 & \text{if } a_{r^*} \in 4\mathscr{R}_N, \\ b = 1 & \text{if } a_{r^*} \in a4\mathscr{R}_N, \\ b = 2 & \text{if } a_{r^*} \in a^2 4\mathscr{R}_N, \\ b = 3 & \text{if } a_{r^*} \in a^3 4\mathscr{R}_N, \end{cases}$$

– OUTPUT$\rightarrow (b)$.

$\mathcal{N}_2$: Recursive scheme

We use this basic scheme to implement a recursive scheme exactly as for $\mathcal{P}$.

Theorem 4.2.3. *The protocol $\mathcal{N}_2$ defined above is a cPIR protocol which works when the alphabet is $\mathbb{Z}_4$ and it allows the user to retrieve one item of the database. Its communication complexity is $\mathcal{CC}_{\mathcal{N}_2}(n) = \mathcal{O}(e^{c\sqrt{\ln n}})$, for some $c > 0$.*

Proof. We have to prove that $\mathcal{N}_2$ verifies the definition 2.0.2 and that it has the required communication complexity.

1. (Correctness) To prove the correctness of the recursive scheme, we have to prove that the basic scheme is correct.

   For any $r \in \mathcal{I}_R$, $a_r = a^{x_{r,c^*}} y^4$, for some $y \in \mathbb{Z}_N^*$. Thus:

   $$\begin{aligned} a_r^* \in 4\mathscr{R}_N &\iff x_{r^*,c^*} = 0 \mod 4 \iff x_{r^*,c^*} = 0, \\ a_r^* \in a4\mathscr{R}_N &\iff x_{r^*,c^*} = 1 \mod 4 \iff x_{r^*,c^*} = 1, \\ a_r^* \in a^2 4\mathscr{R}_N &\iff x_{r^*,c^*} = 2 \mod 4 \iff x_{r^*,c^*} = 2, \\ a_r^* \in a^3 4\mathscr{R}_N &\iff x_{r^*,c^*} = 3 \mod 4 \iff x_{r^*,c^*} = 3. \end{aligned}$$

   Therefore $b = x_{r^*,c^*}$.

2. (Privacy) Exactly as for $\mathcal{N}_1$.

3. (Communication complexity) We use the basic scheme exactly as in $\mathcal{P}$ to implement a recursive scheme in order to reduce the communication complexity. It is trivial that in this way we obtain a protocol with the same communication complexity of $\mathcal{P}$ that is $\mathcal{CC}_{\mathcal{N}_2}(n) = \mathcal{O}\big(e^{c\sqrt{\ln n}}\big)$. Remark that we compute the communication complexity with respect to the number n of element of $\mathbb{Z}_4$ contained in the database.

□

The database $x \in (\mathbb{Z}_4)^n$ actually is a bit string $x' \in (\mathbb{Z}_2)^{2n}$ since every element of $\mathbb{Z}_4$ needs two bits for its binary expansion. Therefore, to compare the complexity of $\mathcal{N}_2$ with that one of $\mathcal{P}$, we have to compute the communication complexity of $\mathcal{N}_2$ with respect to $n' = 2n$, which is the number of bits the database needs to be stored. We obtain:

$$\mathcal{CC}_{\mathcal{N}_2}(n') = \mathcal{O}\big(e^{c\sqrt{\ln (n'/2)}}\big)$$

which is smaller than $\mathcal{CC}_{\mathcal{P}}(n')$.

Hence, for any database with $n' = 2n$ bits, we can consider each block of two bits (x_{2i-1}, x_{2i}) for $i \in \mathcal{I}_n$ as an element $z_i \in \mathbb{Z}_4$ and use $\mathcal{N}_2$ instead of $\mathcal{P}$. In this way we reduce the communication complexity.

It is natural to ask if it is possible to group together more than 2 bits in order to further reduce the complexity. The following section answer this question, presenting a generalization of $\mathcal{N}_2$.

4.3 $\mathcal{M}$: a new cPIR protocol using 2^m-th Residues

In this section we present a generalization of the protocol $\mathcal{N}$: It is based on a more general partition of $\mathbb{Z}_N^1$, that is it uses 2^m-th residues modulo N instead of quartic residues.

Let $2^m\mathscr{R}_N$ be the set of 2^m-th residues modulo N, that is:

$$2^m\mathscr{R}_N = \left\{ y \in \mathbb{Z}_N^* | \exists z \in \mathbb{Z}_N^* \text{ s.t. } y = z^{2^m} \mod N \right\}.$$

Clearly $2^m\mathscr{R}_N \subset 2\mathscr{R}_N$.

Lemma 4.3.1. *Let $N = p_1p_2$, with $p_1 \neq p_2$ prime numbers; let $a \in \mathscr{P}2\mathscr{R}_N$. Then:*

$$\mathbb{Z}_N^1 = 2^m\mathscr{R}_N \dot\cup a2^m\mathscr{R}_N \dot\cup a^2 2^m\mathscr{R}_N \dot\cup \ldots \dot\cup a^{2^m-1}2^m\mathscr{R}_N.$$

Proof. Note that $2^m\mathscr{R}_N \cup a^2 2^m\mathscr{R}_N \cup \ldots \cup a^{2^m-2}2^m\mathscr{R}_N = 2\mathscr{R}_N$ and $a2^m\mathscr{R}_N \cup a^3 2^m\mathscr{R}_N \cup \ldots \cup a^{2^m-1}2^m\mathscr{R}_N = \mathscr{P}2\mathscr{R}_N$. Thus it is enough to prove that for every $t, s \in \mathcal{J}_{2^{m-1}}$ and $t < s$, we have $a^{2t}2^m\mathscr{R}_N \cap a^{2s}2^m\mathscr{R}_N = \emptyset = a^{2t+1}2^m\mathscr{R}_N \cap a^{2s+1}2^m\mathscr{R}_N$. We split the proof in two steps.

- $a^{2t}2^m\mathscr{R}_N \cap a^{2s}2^m\mathscr{R}_N = \emptyset$. An element in $a^{2t}2^m\mathscr{R}_N$ is of the form $a^{2t}y^{2^m}$ for some $y \in \mathbb{Z}_N^*$. It is in $a^{2s}2^m\mathscr{R}_N$ if and only if there exists $z \in \mathbb{Z}_N^*$ such that $a^{2t}y^{2^m} = a^{2s}z^{2^m} = a^{2t+2(s-t)}z^{2^m}$ that is $a^{2(s-t)} \in 2^m\mathscr{R}_N$. Since $a \in \mathscr{P}2\mathscr{R}_N$, we have:

  $$a^{2(s-t)} \in 2^m\mathscr{R}_N \iff 2(s-t) = 0 \mod 2^m \iff s - t = 0 \mod 2^{m-1}$$

  which is impossible because $t, s \in \mathcal{J}_{2^{m-1}}$ and $t \neq s$.

- $a^{2t+1}2^m\mathscr{R}_N \cap a^{2s+1}2^m\mathscr{R}_N = \emptyset$. An element in $a^{2t+1}2^m\mathscr{R}_N$ is of the form $a^{2t+1}y^{2^m}$ for some $y \in \mathbb{Z}_N^*$. It is in $a^{2s+1}2^m\mathscr{R}_N$ if and only if there exists $z \in \mathbb{Z}_N^*$ such that $a^{2t+1}y^{2^m} = a^{2s+1}z^{2^m} = a^{2t+2(s-t)+1}z^{2^m}$ that is $a^{2(s-t)} \in 2^m\mathscr{R}_N$. As before it impossible by our choice of t and s.

□

Clearly $|a^j 2^m\mathscr{R}_N| = \frac{1}{2^m}|\mathbb{Z}_N^1| = \frac{1}{2^{m+1}}|\mathbb{Z}_N^*| = \frac{\phi(N)}{2^{m+1}}$, for every $j \in \mathcal{J}_{2^m-1}$.

4.3.1 $\mathcal{M}_2$: $\mathbb{Z}_{2^m}$ as alphabet (or retrieve a block of m bits)

The partition $\mathbb{Z}_N^1 = 2^m\mathscr{R}_N \dot\cup a2^m\mathscr{R}_N \dot\cup a^2 2^m\mathscr{R}_N \dot\cup \ldots \dot\cup a^{2^m-1}2^m\mathscr{R}_N$ allows the user to distinguish between 2^m different situations. Thus we can use it when the alphabet is $\mathbb{Z}_{2^m}$, that is the database is $x = (x_1, \ldots, x_n)$ with $x_j \in \mathbb{Z}_{2^m}$, and the user wants only one item from it.

The structure of this protocol is the usual one: We consider the database as a matrix and, starting from a basic scheme, we construct a recursive scheme. We call it $\mathcal{M}_2$ because it is a generalization of $\mathcal{N}_2$.

$\mathcal{M}_2$: Basic scheme

Let $k \in \mathbb{N}$ be the security parameter. Let i be the index of the desired item. We view the database as a $R\times C$ matrix with entries in $\mathbb{Z}_{2^m}$ and i is associated with the pair (r^*, c^*) in $\mathcal{I}_R \times \mathcal{I}_C$.

$\mathcal{Q}$: – INPUT$\leftarrow (1^n, i = (r^*, c^*))$,

– Choose at random $p_1 \neq p_2$ prime numbers such that $|p_1| = |p_2| = k/2$,

– Let $N = p_1p_2$,

– For every $1 \leq c \leq C$, choose $q_c \in_R \mathbb{Z}_N^1$ such that:

$$\begin{cases} q_{c^*} \in a2^m\mathscr{R}_N, \\ q_c \in 2^m\mathscr{R}_N \qquad \forall c \neq c^*, \end{cases}$$

– OUTPUT$\rightarrow Q = (N, q_1, \ldots, q_C)$;

$\mathcal{A}$: – INPUT$\leftarrow (x, Q)$,

– For every $1 \leq r \leq R$, let:

$$a_r = \textstyle\prod_{c=1}^{C}(q_c)^{x_{r,c}} \mod N,$$

– OUTPUT$\rightarrow (a_1, \ldots, a_R)$;

$\mathcal{R}$: – INPUT$\leftarrow (1^n, i = (r^*, c^*), a_1, \ldots, a_R)$,

– Let $b \in \mathbb{Z}_{2^m}$ be such that:

$$b = j \Leftrightarrow a_{r^*} \in a^j 2^m\mathscr{R}_N,$$

– OUTPUT$\rightarrow (b)$.

$\mathcal{M}_2$: Recursive scheme

We use this basic scheme to implement a recursive scheme exactly as for $\mathcal{P}$.

Theorem 4.3.2. *The protocol $\mathcal{M}_2$ defined above is a cPIR protocol which works when the alphabet is $\mathbb{Z}_{2^m}$ and it allows the user to retrieve an item of the database. Its communication complexity is $\mathcal{CC}_{\mathcal{N}_2}(n) = \mathcal{O}(e^{c\sqrt{\ln n}})$, for some $c > 0$ and n the number of item contained in the database.*

Proof. We have to prove that $\mathcal{M}_2$ verifies the definition 2.0.2 and that it has the required communication complexity.

1. (Correctness) To prove the correctness of the recursive scheme, we have to prove that the basic scheme is correct.

   For any $r \in \mathcal{I}_R$, $a_r = a^{x_{r,c^*}} y^{2^m}$ for some $y \in \mathbb{Z}_N^*$. Thus:

   $$a_r^* \in a^j 2^m\mathscr{R}_N \iff x_{r^*,c^*} = j \mod 2^m \iff x_{r^*,c^*} = j.$$

   Therefore $b = x_{r^*,c^*}$.

2. (Privacy) We have to prove that under the QRA $\mathcal{S}$, given an element $y \in \mathbb{Z}_N^1$, cannot say anything about which set it belongs to, with respect to this new partition. We split the proof in some steps.

   - Even *vs.* odd. Under QRA $\mathcal{S}$ clearly cannot distinguish $2^m\mathscr{R}_N \dot\cup a^2 2^m\mathscr{R}_N \dot\cup \ldots \dot\cup a^{2^m-2} 2^m\mathscr{R}_N = 2\mathscr{R}_N$ from $a2^m\mathscr{R}_N \dot\cup a^3 2^m\mathscr{R}_N \dot\cup \ldots \dot\cup a^{2^m-1} 2^m\mathscr{R}_N = \mathscr{P}2\mathscr{R}_N$.
   - Odd. Moreover $\mathcal{S}$ cannot distinguish $a^{2t+1} 2^m\mathscr{R}_N$ from $a^{2s+1} 2^m\mathscr{R}_N$, for any $t, s \in \mathcal{J}_{2^{m-1}}$, since they depend on the choice of a and $\mathcal{S}$ does not know the value of a (which is randomly chosen in $\mathscr{P}2\mathscr{R}_N$ by $\mathcal{U}$). We want to show that there exists a pseudo-quadratic residue modulo N, say a', such that $a'^{2t+1} 2^m\mathscr{R}_N = a^{2s+1} 2^m\mathscr{R}_N$. It is enough to pick $a' \in a^{(2t+1)(2s+1)^{-1} \bmod 2^m} 2^m\mathscr{R}_N$, with $(2s+1)^{-1}$ the multiplicative inverse of $2s+1$ in $\mathbb{Z}_{2^m}$. Such a multiplicative inverse exists because $\gcd(2s+1, 2^m) = 1$, being $2s+1$ odd, and it has to be odd. Thus a' is a pseudo-quadratic residue modulo N.

   It remains to be proved that all the $a^{2t} 2^m\mathscr{R}_N$ are indistinguishable, for $t \in \mathcal{J}_{2^{m-1}-1}$.

- 2·even *vs.* 2·odd. We clearly cannot distinguish between $\dot{\bigcup}_k a^{2\cdot 2k}\mathcal{2}^m\mathscr{R}_N$ and $\dot{\bigcup}_k a^{2\cdot(2k+1)}\mathcal{2}^m\mathscr{R}_N$, otherwise we could distinguish between $\dot{\bigcup}_k a^{2k}\mathcal{2}^m\mathscr{R}_N = \mathscr{2R}_N$ and $\dot{\bigcup}_k a^{2k+1}\mathcal{2}^m\mathscr{R}_N = \mathscr{P2R}_N$ just squaring.
  Formally, we assume by contradiction that there exists a family of polynomial-time circuits $\{\mathcal{C}_j\}$ such that:
$$\left|\Pr\left[\mathcal{C}_k(y)=1|y\in\dot{\bigcup}_k a^{2\cdot 2k}\mathcal{2}^m\mathscr{R}_N\right] - \right.$$
$$\left. -\Pr\left[\mathcal{C}_k(y)=1|y\in\dot{\bigcup}_k a^{2\cdot(2k+1)}\mathcal{2}^m\mathscr{R}_N\right]\right| \;\geq\; \frac{1}{P(k)}$$
  for some non-constant P polynomial. We now use $\mathcal{C}_k$ to construct a new circuit $\mathcal{C}'$ which solves the Quadratic Residuosity Problem on N:
  - $\texttt{INPUT}\leftarrow (y)$, with $y\in\mathbb{Z}_N^1$,
  - Let $z=\mathcal{C}_k(y^2 \mod N)$,
  - $\texttt{OUTPUT}\rightarrow (z)$.

  If $y\in\mathscr{P2R}_N=\dot{\bigcup}_k a^{2k+1}\mathcal{2}^m\mathscr{R}_N$, then $(y^2 \mod N)\in\dot{\bigcup}_k a^{2\cdot(2k+1)}\mathcal{2}^m\mathscr{R}_N$. If $y\in\mathscr{2R}_N=\dot{\bigcup}_k a^{2k}\mathcal{2}^m\mathscr{R}_N$, then $(y^2 \mod N)\in\dot{\bigcup}_k a^{2\cdot 2k}\mathcal{2}^m\mathscr{R}_N$. Therefore:
$$\left|\Pr\left[\mathcal{C}'(y)=1|y\in\mathscr{2R}_N\right]-\Pr\left[\mathcal{C}'(y)=1|y\in\mathscr{P2R}_N\right]\right| \;=$$
$$=\left|\Pr\left[\mathcal{C}_k(y)=1|y\in\dot{\bigcup}_k a^{2\cdot 2k}\mathcal{2}^m\mathscr{R}_N\right]-\right.$$
$$\left.-\Pr\left[\mathcal{C}_k(y)=1|y\in\dot{\bigcup}_k a^{2\cdot(2k+1)}\mathcal{2}^m\mathscr{R}_N\right]\right| \;\geq\; \frac{1}{P(k)}.$$
  Hence QRA implies that we cannot distinguish between $\dot{\bigcup}_k a^{2\cdot 2k}\mathcal{2}^m\mathscr{R}_N$ and $\dot{\bigcup}_k a^{2\cdot(2k+1)}\mathcal{2}^m\mathscr{R}_N$.
- 2·odd. Similarly we cannot distinguish among $(a^{2\cdot(2k+1)}\mathcal{2}^m\mathscr{R}_N)_k$, otherwise we could distinguish among $(a^{2k+1}\mathcal{2}^m\mathscr{R}_N)_k$.

It remains to be proved that all the $a^{4t}\mathcal{2}^m\mathscr{R}_N$ are indistinguishable, for $t\in\mathcal{J}_{2^{m-2}-1}$, and to do this we proceed as before.

Formally we prove by induction that we cannot distinguish among $\dot{\bigcup}_k a^{2^t\cdot k}\mathcal{2}^m\mathscr{R}_N$, for every $m-1\geq t\geq 0$

If $t = m-1$, then we have to prove that $2^m\mathscr{R}_N$ and $a^{2^{m-1}}2^m\mathscr{R}_N$ are indistinguishable. It is clearly true, otherwise we could distinguish between $\mathscr{2R}_N$ and $\mathscr{P2R}_N$ just raising up to the 2^{m-1}-th power.

Suppose it is true for $t+1$, then:

$$\dot{\bigcup}_k a^{2^t\cdot k}2^m\mathscr{R}_N = \left(\dot{\bigcup}_k a^{2^t\cdot 2k}2^m\mathscr{R}_N\right) \dot{\cup} \left(\dot{\bigcup}_k a^{2^t\cdot(2k+1)}2^m\mathscr{R}_N\right).$$

- 2^t·even *vs.* 2^t·odd. We cannot distinguish between $\dot{\bigcup}_k a^{2^t\cdot 2k}2^m\mathscr{R}_N$ and $\dot{\bigcup}_k a^{2^t\cdot(2k+1)}2^m\mathscr{R}_N$, otherwise we could distinguish between $\dot{\bigcup}_k a^{2k}2^m\mathscr{R}_N = \mathscr{2R}_N$ and $\dot{\bigcup}_k a^{2k+1}2^m\mathscr{R}_N = \mathscr{P2R}_N$ just raising up to the 2^t-th power.
- 2^t·odd. Similarly we cannot distinguish among $(a^{2^t\cdot(2k+1)}2^m\mathscr{R}_N)_k$, otherwise we could distinguish among $(a^{2k+1}2^m\mathscr{R}_N)_k$.
- 2^t·even. By inductive hypothesis, we cannot distinguish among $\dot{\bigcup}_k a^{2^t\cdot 2k}2^m\mathscr{R}_N = \dot{\bigcup}_k a^{2^{t+1}\cdot k}2^m\mathscr{R}_N$.

Therefore QRA implies that $\mathcal{S}$, knowing not the factorization of N, cannot distinguish among $\dot{\bigcup}_k a^k 2^m\mathscr{R}_N$ that is, by $\mathcal{S}$'s point of view, all the elements of $\mathbb{Z}_N^1$ are indistinguishable with respect to our partition. Remark that it means, not only that $\mathcal{S}$ cannot learn anything from the queries he receives, but also that he cannot infer anything from the answers he computes. Therefore the protocol $\mathcal{M}$ is secure.

3. (Communication complexity) We use the basic scheme exactly as in $\mathcal{P}$ to implement a recursive scheme in order to reduce the communication complexity. In this way we obtain a protocol with the same communication complexity of $\mathcal{P}$ that is $\mathcal{CC}_{\mathcal{M}_2}(n) = \mathcal{O}\big(e^{c\sqrt{\ln n}}\big)$. Remark that we compute the communication complexity with respect to the number n of element of $\mathbb{Z}_{2^m}$ contained in the database.

□

The database $x \in (\mathbb{Z}_{2^m})^n$ actually is a bit string $x' \in (\mathbb{Z}_2)^{mn}$ since every element of $\mathbb{Z}_{2^m}$ needs m bits for its binary expansion. Therefore, to compare the complexity of $\mathcal{M}_2$ with that one of $\mathcal{P}$, we have to compute the communication

complexity of $\mathcal{M}_2$ with respect to $n' = mn$, which is the number of bits the database needs to be stored. We obtain:

$$\mathcal{CC}_{\mathcal{M}_2}(n') = \mathcal{O}\big(e^{c\sqrt{\ln(n'/m)}}\big)$$

which is smaller than $\mathcal{CC}_{\mathcal{P}}(n')$.

Hence, for any database with $n' = mn$ bits, we can consider each block of m bits $(x_{m(i-1)+1}, \ldots, x_{mi})$ for $i \in \mathcal{I}_n$ as an element $z_i \in \mathbb{Z}_{2^m}$ and use $\mathcal{M}_2$ instead of $\mathcal{P}$. In this way we reduce the communication complexity.

This time to avoid repetitions we must choose the security parameter k in such a way that $|2^m\mathscr{R}_N| = \frac{\phi(N)}{2^{m+1}} \geq n^{1/(L+1)}$, that is $2^{k-m-1} - 2^{k/2-m} + \frac{1}{2^{m+1}} \geq n^{1/(L+1)}$.

Using protocol $\mathcal{P}$, the condition on k is $2^{k-2} - 2^{k/2-1} + \frac{1}{4} \geq n^{1/(L+1)}$, therefore, passing form $\mathcal{P}$ to $\mathcal{M}_2$, we must change the security parameter from k to $k' = k + m - 1$. In this way the new protocol still avoid repetitions.

4.3.2 $\mathcal{M}_1$: Retrieve m bits in any position

We can generalize $\mathcal{N}_1$ and use our new partition to retrieve several (at most m) bits in any position.

The structure of this protocol is the usual one: We consider the database as a matrix and, starting from a basic scheme, we construct a recursive scheme.

$\mathcal{M}_1$: Basic scheme

Let $k \in \mathbb{N}$ be the security parameter. Let $i_0, i_1, \ldots, i_{m-1}$ be the indices of the desired bits. We view the database as a $R \times C$ matrix and i_j is associated with the pair (r_j^*, c_j^*) in $\mathcal{I}_R \times \mathcal{I}_C$, for each $j \in \mathcal{J}_{m-1}$.

$\mathcal{Q}$:
- $\texttt{INPUT}\leftarrow (1^n, i_0 = (r_0^*, c_0^*), \ldots, i_{m-1} = (r_{m-1}^*, c_{m-1}^*))$,
- Choose at random $p_1 \neq p_2$ prime numbers such that $|p_1| = |p_2| = k/2$,
- Let $N = p_1 p_2$,
- For every $1 \leq c \leq C$, choose $q_c \in_R \mathbb{Z}_N^1$ such that:

$$\begin{cases} q_{c_j^*} \in a^{2^j} 2^m\mathscr{R}_N & \text{if } c_j^* \neq c_h^* \text{ for every } h \in \mathcal{J}_{m-1} \setminus \{j\}, \\ q_{\bar{c}_g} \in a^{2^{j_1}+2^{j_2}+\ldots+2^{j_d}} 2^m\mathscr{R}_N & \text{if } \bar{c}_g = c_{j_1}^* = c_{j_2}^* = \ldots = c_{j_d}^* \\ & \text{for some } g, \\ q_c \in 2^m\mathscr{R}_N & \forall c \neq c_j^* \text{ for evey } j \in \mathcal{J}_{m-1}, \end{cases}$$

– OUTPUT$\rightarrow Q = (N, q_1, \ldots, q_C)$;

$\mathcal{A}$: – INPUT$\leftarrow (x, Q)$,

– For every $1 \leq r \leq R$, let:

$$a_r = \prod_{c=1}^{C} (q_c)^{x_{r,c}} \mod N,$$

– OUTPUT$\rightarrow (a_1, \ldots, a_R)$;

$\mathcal{R}$: – INPUT$\leftarrow (1^n, i_0 = (r_0^*, c_0^*), \ldots, i_{m-1} = (r_{m-1}^*, c_{m-1}^*), a_1, \ldots, a_R)$,

– For every $1 \leq r \leq R$, let $t_r \in \mathcal{J}_{2^m-1}$ be such that:

$$a_r \in a^{t_r} 2^m \mathscr{R}_N,$$

– For every $1 \leq r \leq R$, let $(b_r)_0, \ldots, (b_r)_{m-1} \in \mathbb{Z}_2$ be such that:

$$t_r = \sum_{j=0}^{2^m-1} 2^j (b_r)_j,$$

– OUTPUT$\rightarrow ((b_{r_0^*})_0, \ldots, (b_{r_{m-1}^*})_{m-1})$.

Remark that $\mathcal{U}$ chooses his query in such a way that $q_{c_j^*}$ is in $a^{2^j} 2^m \mathscr{R}_N$ if c_j^* is in the index of only one bit he wants to retrieve. But some desired bits can be in the same column, that is they share the column index, say $\overline{c}_g$ (we have to use the subscript g because there can be more than one subset of desired bit sharing the column index). In this case $\mathcal{U}$ must send only $q_{\overline{c}_g}$, but he has to choose it in such a way that $q_{\overline{c}_g}$ contains the information that it is the index of several bits. This is exactly what happens, in fact $q_{\overline{c}_g}$ is in $a^{2^{j_1}+\ldots+2^{j_d}} 2^m \mathscr{R}_N$, where $\{c_{j_1}, \ldots, c_{j_d}\}$ is the set of column indices represented by $\overline{c}_g$.

$\mathcal{M}_1$: Recursive scheme

We use this basic scheme to implement a recursive scheme exactly as for $\mathcal{P}$.

Theorem 4.3.3. *The protocol $\mathcal{M}_1$ defined above is a cPIR protocol which allows the user to retrieve any 2 bits of the database and such that $\mathcal{CC}_{\mathcal{M}_1}(n) = \mathcal{O}(e^{c\sqrt{\ln n}})$, for any $c > 0$.*

Proof. We have to prove that $\mathcal{M}_1$ verifies the definition 2.0.2 and that it has the required communication complexity.

1. (Correctness) To prove the correctness of the recursive scheme, we have to prove that the basic scheme is correct.

Suppose for some g, $\bar{c}_g = c^*_{j_1} = c^*_{j_2} = \ldots = c^*_{j_d}$. Then for every $r \in \mathcal{I}_R$ we have that:

$$(q_{\bar{c}_g})^{x_{r,\bar{c}_g}} = (a^{2^{j_1}+\ldots+2^{j_d}})^{x_{r,\bar{c}_g}} (y^{2^m})^{x_{r,\bar{c}_g}} = a^{2^{j_1}x_{r,c^*_{j_1}}+\ldots+2^{j_d}x_{r,c^*_{j_d}}} (y^{x_{r,\bar{c}_g}})^{2^m}$$

for some $y \in \mathbb{Z}^*_N$. On the contrary, if $c^*_j \neq c^*_h$ for every $h \neq j$ and $h, j \in \mathcal{J}_{m-1}$, we have:

$$(q_{c^*_j})^{x_{r,c^*_j}} = (a^{2^j})^{x_{r,c^*_j}} (y^{2^m})^{x_{r,c^*_j}} = a^{2^j x_{r,c^*_j}} (y^{x_{r,c^*_j}})^{2^m}$$

for some $j \in \mathbb{Z}^*_N$. Therefore, for every $r \in \mathcal{I}_R$, we have:

$$a_r = a^{\sum_{j=0}^{m-1} 2^j x_{r,c^*_j}} y^{2^m}$$

for some $y \in \mathbb{Z}^*_N$. Remark that $\sum_{j=0}^{m-1} 2^j x_{r,c^*_j} < 2^m$ because $x_{r,c^*_j} \in \mathbb{Z}_2$ and so it can be at most $2^m - 1$.

Hence $t_r = \sum_{j=0}^{m-1} 2^j x_{r,c^*_j}$ and therefore $(b_r)_j = x_{r,c^*_j}$. Thus $(b_{r^*_j})_j = x_{r^*_j,c^*_j}$.

2. (Privacy) Exactly as for $\mathcal{M}_2$.

3. (Communication complexity) We use the basic scheme exactly as in $\mathcal{P}$ to implement a recursive scheme in order to reduce the communication complexity. In this way we construct a protocol with the same complexity of $\mathcal{P}$ because $\mathcal{U}$ and $\mathcal{S}$ send the same amount of bits (the only difference is that $\mathcal{U}$ chooses the elements of his query with a different strategy). Therefore $\mathcal{CC}_{\mathcal{M}_1}(n) = \mathcal{O}\big(e^{c\sqrt{\ln n}}\big)$.

□

It is important to remark that, since the basic scheme allows the user to retrieve m bits in any position, we do not need to pay attention on how the indices change in the passage from a level to the successive. Thus $\mathcal{M}_1$ allows the user to obtain m bits in any position. It implies that it can be equally used when $\mathcal{U}$ wants only one bit. In this case $\mathcal{M}_1$ acts exactly as $\mathcal{P}$: we have $i = i_0 = \ldots = i_{m-1}$ and so we have only one index (r^*_l, c^*_l) for every $l \in \mathcal{I}_L$. Thus for each level of recursion, the query of $\mathcal{U}$ consists of $C_l - 1 = n^{1/(L+1)} - 1$ elements in $2^m\mathscr{R}_N \subset \mathscr{2R}_N$ and only one element in $a^{1+2+2^2\ldots+2^{m-1}} 2^m\mathscr{R}_N = a^{2^m-1} 2^m\mathscr{R}_N \subset \mathscr{P2R}_N$.

Chapter 5

Some cPIR Protocols Based on More Sophisticated Assumptions

All the cPIR protocols presented in the previous chapters are based on Quadratic Residuosity Assumption which is the commonest cryptographic assumption.

Indeed, after Kushilevitz and Ostrovsky's paper [17] other 1-server cPIR schemes were introduced, still having subpolynomial communication complexity, but based on other number-theoretic assumptions.

The main results obtained in this field are summarized in the following table:

Prot.	Author	Ref.	Assumption
$\mathcal{P}$	Kushilevitz, Ostrovsky (97)	[17]	Quadratic Residuosity Assumption
	Cachin et al. (99)	[5]	Φ-Assumption
	Kushilevitz, Ostrovsky (00)	[18]	Existence of One-Way Permutation
$\mathcal{CR}$	Chang (04)	[6]	Composite Residuosity Assumption
$\mathcal{HE}$	Ostrovsky, Skeith III (07)	[21]	Existence of Homomorphic Encryption Scheme

Table 5.1: Main results on cPIR schemes

In this chapter we deal with protocols $\mathcal{CR}$ and $\mathcal{HE}$, since they present structure and characteristics similar to $\mathcal{P}$'s ones. Indeed $\mathcal{HE}$ generalizes the previous works. A first generalization of $\mathcal{P}$ was done by Mann in [19]: He shows that using a construction similar to the Kushilevitz and Ostrovsky's one, it is possible to obtain a cPIR scheme based on more general assumptions, that is *homomorphic trapdoor predicates* [19]. Then Ostrovsky and Skeith III [21] give a further generalization presenting an abstract construction for cPIR protocols based upon any group homomorphic encryption scheme.

Our contribution here is to provide a more precise estimation of the communication complexity of $\mathcal{CR}$ and to fill Ostrovsky and Skeith III's work giving

a complete presentation of their general construction and showing with details how to reduce $\mathcal{P}$ and $\mathcal{CR}$ to special cases of $\mathcal{HE}$.

5.1 $\mathcal{CR}$: based on Composite Residuosity Assumption

5.1.1 Composite Residuosity Assumption

Let $p_1 \neq p_2$ be prime numbers such that $|p_1| = |p_2|$. Let $N = p_1 p_2$ and $k = \lceil \log N \rceil$ be the security parameter (that is k large enough to make the factorization of N hard). We denote by $\phi(n)$ Euler's totient function as usul and by $\lambda(n)$ *Carmichael's function* taken on n. Thus $\phi(N) = (p_1 - 1)(p_2 - 1)$ and $\lambda(N) = \text{lcm}\,(p_1 - 1, p_2 - 1)$ in the present case.

Consider the multiplicative group $\mathbb{Z}^*_{N^2}$.

Definition 5.1.1 (N^{th} residue). *An integer $x \in \mathbb{Z}^*_{N^2}$ is said to be an* N^{th} residue modulo N^2 *if there exists an integer $y \in \mathbb{Z}^*_{N^2}$ such that $x = y^N \mod N^2$. Otherwise x is said to be an* N^{th} non-residue modulo N^2. *We denote with $\mathscr{NR}_{N^2}$ the set of* N^{th} *residues modulo N^2.*

Lemma 5.1.2. *The set $\mathscr{NR}_{N^2}$ is a multiplicative subgroup of $\mathbb{Z}^*_{N^2}$ of order $\phi(N)$.*

The N^{th} *Residuosity Problem* modulo N^2, with $N = p_1 p_2$ as above, is: Given $x \in \mathbb{Z}^*_{N^2}$, determine whether $x \in \mathscr{NR}_{N^2}$ or not [22]. It can be easily solved if the factorization of N is known (as we will see later); on the contrary, solving the N^{th} Residuosity Problem modulo N^2 without knowing the factorization of N is believed to be computationally hard.

Conjecture 5.1.3 (Composite Residuosity Assumption (CRA)). *Let $p_1 \neq p_2$ be prime numbers such that $|p_1| = |p_2| = k/2$ large enough and let $N = p_1 p_2$. If the factorization of N is unknown, there is no efficient procedure for solving the* N^{th} *residuosity problem modulo N^2.*

As the Quadratic Residuosity Problem, the N^{th} Residuosity Problem is *random-self-reducible*, that is the problem is either uniformly intractable or uniformly solvable in polynomial time. Therefore the validity of CRA only depends on the choice of N [22].

We now shortly describe the number-theoretic framework underlying the cPIR protocol we are going to construct; we refer to [22] for details. For any

$\alpha \neq 0$, we denote by $\mathcal{V}_\alpha$ the subset of $\mathbb{Z}_{N^2}^*$ of elements of multiplicative order αN and by $\mathcal{V}$ their disjoint union for $\alpha \in \mathcal{I}_{\lambda(N)}$. That is:

$$\mathcal{V}_\alpha = \{y \in \mathbb{Z}_{N^2}^* | y^{\alpha N} = 1, y^\beta \neq 1 \forall \beta < \alpha N\}, \qquad \mathcal{V} = \dot{\bigcup_{\alpha=1}^{\lambda(N)}} \mathcal{V}_\alpha.$$

Remark that, since $\lambda(N) = \text{lcm}\,(p_1 - 1, p_2 - 1)$, $y^{\lambda(N)N} = 1 \mod p_1^2$ and $y^{\lambda(N)N} = 1 \mod p_2^2$. Therefore, by Chinese Remainder Theorem, $y^{\lambda(N)N} = 1 \mod N^2$ for every $y \in \mathbb{Z}_{N^2}^*$ and so the disjoint union ends with $\alpha = \lambda(N)$.

Let $y \in \mathbb{Z}_{N^2}^*$, we define the following integer-function:

$$\begin{array}{rccl} \mathfrak{E}_y : & \mathbb{Z}_N \times \mathbb{Z}_N^* & \longrightarrow & \mathbb{Z}_{N^2}^* \\ & (a, b) & \longmapsto & y^a b^N \mod N^2. \end{array}$$

Lemma 5.1.4. *If $y \in \mathcal{V}$, then $\mathfrak{E}_y$ is bijective.*

Definition 5.1.5. *Let $y \in \mathcal{V}$. For any $w \in \mathbb{Z}_{N^2}^*$, we denote by $[[w]]_y$ the unique integer in $\mathbb{Z}_N$ for which there exists (unique) $b \in \mathbb{Z}_N^*$ such that:*

$$\mathfrak{E}_y([[w]]_y, b) = w.$$

Lemma 5.1.6. *For every $w, w' \in \mathbb{Z}_{N^2}^*$ and for every $y, y' \in \mathcal{V}$ we have:*

1. $[[w]]_y = 0 \Leftrightarrow w \in \mathscr{NR}_{N^2}$,
2. $[[ww']]_y = [[w]]_y + [[w']]_y \mod N$,
3. $[[w]]_y = [[w]]_{y'}[[y']]_y \mod N$.

1. and 2. imply that the function:

$$\begin{array}{ccc} (\mathbb{Z}_{N^2}^*, \times) & \longrightarrow & (\mathbb{Z}_N, +) \\ w & \longmapsto & [[w]]_y \end{array}$$

is a group homomorphism for any $y \in \mathcal{V}$.

Theorem 5.1.7. *Let N be a composite of usual form, $w \in \mathbb{Z}_{N^2}^*$ and $y \in \mathcal{V}$. Under CRA, it is computationally intractable to compute $[[w]]_y$ without knowing the factorization of N.*

Proof. Let w, y, N be as in the statement. We prove that if we can compute $[[w]]_y$, then we can determine whether $w \in \mathscr{NR}_{N^2}$ or not. By 1. of Lemma 5.1.6, w is an N^{th} residue modulo N^2 if and only if $[[w]]_y = 0$ for every (and thus for some by 3. of Lemma 5.1.6) $y \in \mathcal{V}$. Therefore we choose an any $y \in \mathcal{V}$ and we compute $[[w]]_y$: $w \in \mathscr{NR}_{N^2} \Leftrightarrow [[w]]_y = 0$. □

Define the following function:

$$\begin{array}{rccc} L: & \mathbb{Z}_{N^2}^* & \longrightarrow & \mathbb{Q} \\ & w & \longmapsto & \frac{w-1}{N}. \end{array}$$

L induces by restriction a well-defined map from $\{w \in \mathbb{Z}_{N^2}^* | w = 1 \mod N\}$ to $\mathbb{Z}_N$.

Lemma 5.1.8. *$L(w^{\lambda(N)} \mod N^2) = \lambda(N)[[w]]_{N+1} \mod N$, for every $w \in \mathbb{Z}_{N^2}^*$.*

Proof. First of all we have to prove that $N+1 \in \mathcal{V}$; let $m = \operatorname{ord}(N+1)$, then:

$$1 = (N+1)^m = 1 + mN \mod N^2,$$

thus m must be a nonzero multiple of N. By Lemma 5.1.4, it implies that $[[w]]_{N+1}$ is well-defined for every $w \in \mathbb{Z}_{N^2}^*$. Then, for some $b \in \mathbb{Z}_N^*$:

$$\begin{aligned} w^{\lambda(N)} &= \left(\mathfrak{E}_y([[w]]_{N+1}, b)\right)^{\lambda(N)} = \left((N+1)^{[[w]]_{N+1}} b^N\right)^{\lambda(N)} = \\ &= (N+1)^{\lambda(N)[[w]]_{N+1}} b^{\lambda(N)N} \overset{(b^{\lambda(N)N}=1)}{=} (N+1)^{\lambda(N)[[w]]_{N+1}} = \\ &= 1 + \lambda(N)[[w]]_{N+1} N \mod N^2. \end{aligned}$$

Note that $w^{\lambda(N)} = 1 \mod N$, thus L taken on it has value in $\mathbb{Z}_N$. In particular:

$$\begin{aligned} L(w^{\lambda(N)} \mod N^2) &= L(1 + \lambda(N)[[w]]_{N+1} N \mod N^2) = \\ &= \frac{(1 + \lambda(N)[[w]]_{N+1} N \mod N^2) - 1}{N} = \\ &= \lambda(N)[[w]]_{N+1} \mod N \end{aligned}$$

□

Corollary 5.1.9. *For every $w \in \mathbb{Z}_{N^2}^*$, if $\gcd\left(L(w^{\lambda(N)} \mod N^2), N\right) = 1$, then $w \in \mathcal{V}$.*

Proof. We have seen proving Lemma 5.1.8 that $w^{\lambda(N)} = 1 + \lambda(N)[[w]]_{N+1}N \mod N^2$ for every $w \in \mathbb{Z}^*_{N^2}$. Let $m = \text{ord}\,(w)$, then:

$$\begin{aligned} 1 &= (w^m)^{\lambda(N)} = (w^{\lambda(N)})^m = (1 + \lambda(N)[[w]]_{N+1}N)^m = \\ &= 1 + m\lambda(N)[[w]]_{N+1}N \mod N^2. \end{aligned}$$

Thus $m\lambda(N)[[w]]_{N+1} = 0 \mod N$. Since $\gcd(\lambda(N), N) = 1$, this implies $m[[w]]_{N+1} = 0 \mod N$. That is $p_i|(m[[w]]_{N+1})$ for $i = 1, 2$.

By Lemma 5.1.8, $L(w^{\lambda(N)} \mod N^2) = \lambda(N)[[w]]_{N+1} \mod N$. Therefore $\gcd\left(L(w^{\lambda(N)} \mod N^2), N\right) = 1$ means $p_i \nmid (\lambda(N)[[w]]_{N+1} \mod N)$ for $i = 1, 2$; that is $p_i \nmid [[w]]_{N+1}$ for $i = 1, 2$ because $\gcd(\lambda(N), N) = 1$.

Therefore we must have $p_i|m$ for $i = 1, 2$, that is $N|m$. □

Theorem 5.1.10. *If we know the factorization of N, then we can efficiently compute $[[w]]_y$ for every $y \in \mathcal{V}$ and for every $w \in \mathbb{Z}^*_{N^2}$.*

Proof. By 3. of Lemma 5.1.6, setting $w = y$ and $y' = N + 1$, we have: $[[y]]_y = 1 = [[y]]_{N+1}[[N+1]]_y \mod N$ that is $[[y]]_{N+1} = [[N+1]]_y^{-1} \mod N$. By Lemma 5.1.8, $L(y^{\lambda(N)} \mod N^2) = \lambda(N)[[y]]_{N+1} \mod N$; since $\gcd(\lambda(N), N) = 1$ and $[[y]]_{N+1}$ is invertible modulo N, we have that $L(y^{\lambda(N)} \mod N^2)$ is invertible modulo N. The knowledge of the factorization of N obviously leads to the knowledge of $\lambda(N)$; therefore, for every $y \in \mathcal{V}$ and for every $w \in \mathbb{Z}^*_{N^2}$, we can compute $y^{\lambda(N)}$ and $w^{\lambda(N)}$. Thus we can compute:

$$\frac{L(w^{\lambda(N)} \mod N^2)}{L(y^{\lambda(N)} \mod N^2)} \overset{5.1.8}{=} \frac{\lambda(N)[[w]]_{N+1}}{\lambda(N)[[y]]_{N+1}} = \frac{[[w]]_{N+1}}{[[y]]_{N+1}} = [[w]]_y \mod N,$$

the last equality follows from 3. of Lemma 5.1.6, setting $y = N + 1$ and $y' = y$. □

5.1.2 Basic scheme

In this subsection we present a 1-server cPIR scheme with communication complexity $k+4k\sqrt{n}$, for $k = \lceil \log N \rceil$ security parameter. In the next subsection we will use it to construct a recursive protocol with less communication complexity.

We consider the database $x \in \mathbb{Z}_2^n$ as a $R \times C$ matrix of fixed dimensions: We associate the bit-string x with a matrix $(x_{r,c})_{r \in \mathcal{I}_R, c \in \mathcal{I}_C}$ and each position $j \in \mathcal{I}_n$ with a pair $(r, c) \in \mathcal{I}_R \times \mathcal{I}_C$. In particular, the index i of the desired bit is associated with the pair (r^*, c^*).

Protocol $\mathcal{B}$

Let $k \in \mathbb{N}$ be the security parameter.

$\mathcal{Q}$:
- $\texttt{INPUT}\leftarrow (1^n, i = (r^*, c^*))$
- Choose at random $p_1 \neq p_2$ prime numbers such that $|p_1| = |p_2| = k/2$,
- Let $N = p_1 p_2$,
- Choose $y \in_R \mathcal{V}$.
  By Corollary 5.1.9, this can be done efficiently by checking whether $\gcd\left(L(y^{\lambda(N)} \mod N^2, N)\right) = 1$,
- For every $1 \leq c \leq C$, choose $q_c \in \mathbb{Z}^*_{N^2}$ such that:
$$\begin{cases} q_{c^*} = \mathfrak{E}_y(1, z_{c^*}) & (i.e.\ q_{c^*} \notin \mathscr{NR}_{N^2}), \\ q_c = \mathfrak{E}_y(0, z_c) & (i.e.\ q_c \in \mathscr{NR}_{N^2}), \quad \forall c \neq c^*, \end{cases}$$
  with $z_c \in_R \mathbb{Z}^*_N$,
- $\texttt{OUTPUT}\rightarrow Q = (N, q_1, \ldots, q_C)$;

$\mathcal{A}$:
- $\texttt{INPUT}\leftarrow (x, Q)$
- For every $1 \leq r \leq R$, let:
$$\begin{cases} a_r = \prod_{c=1}^{C} (q_c)^{x_{r,c}} \mod N^2, \\ (u_r, v_r) \in (\mathbb{Z}_N)^2 \text{ be such that } a_r = u_r N + v_r, \end{cases}$$
- $\texttt{OUTPUT}\rightarrow (u_1, \ldots, u_R, v_1, \ldots, v_R)$;

$\mathcal{R}$:
- $\texttt{INPUT}\leftarrow (1^n, i = (r^*, c^*), u_1, \ldots, u_R, v_1, \ldots, v_R)$,
- Let $z = u_{r^*} N + v_{r^*} \in \mathbb{Z}^*_{N^2}$,
- Let $b = [[z]]_y \in \mathbb{Z}_N$,
- $\texttt{OUTPUT}\rightarrow (b)$

The fact that for each $r \in \mathcal{I}_R$ $\mathcal{S}$ sends the pair (u_r, v_r) instead of a_r could seem useless and actually it is in the basic scheme (even if it does not make the communication complexity increase, because $a_r \in \mathbb{Z}^*_{N^2}$ and $u_r, v_r \in \mathbb{Z}_N$), but we will see that it has a key role in the recursive scheme.

Theorem 5.1.11. *The protocol $\mathcal{B}$ defined above is a cPIR scheme such that $\mathcal{CC}_{\mathcal{B}}(n) = \mathcal{O}\left(n^{1/2+\epsilon}\right)$, for every $\epsilon > 0$.*

Proof. We have to prove that $\mathcal{B}$ verifies the definition 2.0.2 and that it has the required communication complexity.

1. (Correctness) For every $r \in \mathcal{I}_R$, we have $z = u_{r^*}N + v_{r^*} = a_{r^*}$ and:

$$\begin{aligned}[[z]]_y &= \left[\left[\prod_{c=1}^{C}(q_c)^{x_{r^*,c}} \mod N^2\right]\right]_y \overset{5.1.6}{=} \\ &\overset{5.1.6}{=} \sum_{c=1}^{C}\left[\left[(q_c)^{x_{r^*,c}} \mod N^2\right]\right]_y \mod N \overset{5.1.6}{=} \\ &\overset{5.1.6}{=} \sum_{c\neq c^*,c=1}^{C}([[q_c]]_y x_{r^*,c}) + [[q_{c^*}]]_y x_{r^*,c^*} \mod N = x_{r^*,c^*}\end{aligned}$$

   because $[[q_{c^*}]]_y = 1$ and $[[q_c]]_y = 0$ for every $c \neq c^*$ by construction. Therefore $b = x_{r^*,c^*}$. Remark that $\mathcal{U}$ knows the factorization of N and so he can efficiently compute $[[z]]_y$ by Theorem 5.1.10.

2. (Privacy) Suppose by contradiction that for some indices $i = (r^*, c^*)$ and $i' = (r'^*, c'^*)$ the server can distinguish the queries on i from that ones on i'. That is, if we denote by D_i and $D_{i'}$ the distributions of the queries on i and i' respectively, then $\mathcal{S}$ can distinguish D_i from $D_{i'}$. By construction a query in D_i consists of N followed by C elements in $\mathbb{Z}^*_{N^2}$ such that only the $c^{*\text{th}}$ is not in $\mathscr{NR}_{N^2}$. A query in $D_{i'}$ is similar except that the N^{th} non-residue modulo N^2 is located in position c'^*. Therefore we must have $c^* \neq c'^*$, otherwise there is no way to distinguish D_i from $D_{i'}$. By a standard argument (exactly as in the proof of Theorem 3.2.1), we obtain that if $\mathcal{S}$ can distinguish D_i form $D_{i'}$, then he can distinguish N^{th} residues modulo N^2 from N^{th} non-residues, without knowing the factorization of N, in contradiction to CRA.

3. (Communication complexity) The security parameter is $k = \lceil \log N \rceil$. $\mathcal{U}$ sends $Q = (N, q_1, \ldots, q_C)$, with $q_c \in \mathbb{Z}^*_{N^2}$; so $\mathcal{U}$ sends $k + 2kC$ bits. $\mathcal{S}$ replies sending $(u_1, \ldots, u_R, v_1, \ldots, v_R)$, with $u_r, v_r \in \mathbb{Z}^*_N$; so $\mathcal{S}$ sends $2kR$ bits. Thus the total amount of communication is $k(1 + 2(R + C))$ bits.

   By construction $RC = n$, hence the better choice for R and C is $R = C = \sqrt{n}$. For every $\epsilon > 0$, if we choose as the security parameter $k = n^\epsilon$, we have:

$$\mathcal{CC}_{\mathcal{B}}(n) = n^\epsilon(4\sqrt{n} + 1) = \mathcal{O}\big(n^{1/2+\epsilon}\big).$$

□

5.1.3 Protocol $\mathcal{CR}$: Recursive scheme

As for $\mathcal{P}$, in this subsection we use the idea of the basic scheme to construct a cPIR protocol with less communication complexity. The new protocol is based on the observation that in $\mathcal{B}$ $\mathcal{U}$ is only interested in two of the numbers he receives from $\mathcal{S}$. However $\mathcal{U}$ cannot reveal what are the items he needs, as this will violate the privacy constraint. It is therefore natural to see the $2kR$-bit string $(u_1, \ldots, u_R, v_1, \ldots, v_R)$ as two new databases $(u_1, \ldots, u_R)$ and $(v_1, \ldots, v_R)$ and $\mathcal{U}$ wants only one item from both of them.

Remark that until here, we have ignored the fact that the function $[[\cdot]]_y$ has values in $\mathbb{Z}_N$, since we have used it just to retrieve a bit. We now strongly use this fact: The two strings $(u_1, \ldots, u_R)$ and $(v_1, \ldots, v_R)$ have entries in $\mathbb{Z}_N$ and since the function $[[\cdot]]_y$ has values in $\mathbb{Z}_N$, it is possible to consider them as new databases. In this way we need only two new invocations of the cPIR scheme itself. Remark these two invocations of the cPIR scheme require just one new query to be sent, because the user wants the same item (the $r^{*\text{th}}$) from both the databases.

For clarity, we will not present the scheme as a triple of algorithm $(\mathcal{Q}, \mathcal{A}, \mathcal{R})$ but rather as a recursive scheme. However it is important to notice that the user can compute in advance all parts of the query he needs to send and send all of them at once. Hence the new protocol can still be implemented in a single round.

The protocol would consist of L level of recursion, we will denote the l^{th} level by Level_l (we will use the subscript l referring to Level_l, but when it is impossible we will use superscripts). Let Level_1 be the basic scheme $\mathcal{B}$ described above. We set:

$$\begin{cases} x^L = x \text{ viewed as a } R_L \times C_L \text{ matrix,} \\ n_L = |x^L| = n, \\ j_L \in \mathcal{I}_{n_L} \text{ a generic position in the database. It is associated with a pair} \\ \qquad (r_L, c_L) \in \mathcal{I}_{R_L} \times \mathcal{I}_{C_L} \text{ such that } (r_L - 1)C_L + c_L = j_L, \\ i_L = (r_L^*, c_L^*) \text{ the index of the desired item.} \end{cases}$$

For every $L \geq l \geq 1$ do Level_l:

- View the database x^l as a $R_l \times C_l$ matrix (thus $R_l C_l = n_l$). Let $i_l = (r_l^*, c_l^*)$ be the index of the item $\mathcal{U}$ wants to retrieve. $\mathcal{U}$ and $\mathcal{S}$ simulate protocol $\mathcal{B}$ with this setting.

- If $l > 1$, then $\mathcal{S}$ does not send his answer $(u_1^l, \ldots, u_{R_l}^l, v_1^l, \ldots, v_{R_l}^l)$ to $\mathcal{U}$ but he considers it as two new databases with entries in $\mathbb{Z}_N$ and of length R_l. That is $\mathcal{U}$ and $\mathcal{S}$ go to Level_{l-1} twice with:
  - As database $\begin{cases} x^{l-1} = (u_1^l, \ldots, u_{R_l}^l) \text{ for the } 1^{\text{st}} \text{ invocation of } \text{Level}_{l-1}, \\ x^{l-1} = (v_1^l, \ldots, v_{R_l}^l) \text{ for the } 2^{\text{nd}} \text{ invocation.} \end{cases}$
    Thus $n_{l-1} = |x^{l-1}| = R_l$,
  - As index $i_{l-1} = r_l^*$ (since $\mathcal{U}$ is only interested in $u_{r_l^*}^l$ and $v_{r_l^*}^l$).
- If $l = 1$, $\mathcal{S}$ sends his answer $(u_1^1, \ldots, u_{R_1}^1, v_1^1, \ldots, v_{R_1}^1)$ to $\mathcal{U}$.
- $\mathcal{U}$ uses $u_{r_l^*}^l$ and $v_{r_l^*}^l$ as in $\mathcal{B}$ to retrieve $x_{i_l}^l$.

Remark that $n_{l-1} = R_l$ and i_{l-1} are the same for each invocation of Level_{l-1}. This implies that the length of the database decreases at each step ($n_{l-1} = R_l = n_l/C_l$) and that $\mathcal{U}$ sends only one query for all the invocation of level Level_{l-1}.

Remark 5.2. In the original paper [6] the author proves that the communication complexity of $\mathcal{CR}$ is $\mathcal{O}(n^{1/c})$, for any integer constant $c > 1$. In this work we give a more precise estimation, proving that $\mathcal{CC}_{\mathcal{CR}}(n) = \mathcal{O}\big(2^{2\sqrt{\ln n}}\big)$.

Theorem 5.2.1. *The protocol $\mathcal{CR}$ defined above is a cPIR protocol such that $\mathcal{CC}_{\mathcal{CR}}(n) = \mathcal{O}\big(2^{2\sqrt{\log n}}\big)$.*

Proof. We have to prove that $\mathcal{CR}$ verifies the definition 2.0.2 and that it has the required communication complexity.

1. (Correctness) The correctness follows from the correctness of $\mathcal{B}$. Formally, we prove by induction that at the end of Level_l $\mathcal{U}$ retrieves the $i_l{}^{\text{th}}$ item of x^l, for every $1 \leq l \leq L$.

   For $l = 1$ it is trivial since Level_1 is $\mathcal{B}$.

   Suppose it is true for $l - 1$, then the 1^{st} invocation of Level_{l-1} allows the user to retrieve the $r_l^{*\text{th}}$ item of $(u_1^l, \ldots, u_{R_l}^l)$ that is the $u_{r_l^*}^l$. Similarly the 2^{nd} invocation of Level_{l-1} allows the user to retrieve $v_{r_l^*}^l$. $\mathcal{U}$ now use them as in $\mathcal{B}$ and he retrieves $x_{i_l}^l$.

2. (Privacy) It follows from CRA; we can prove it as we have done for the privacy of $\mathcal{P}$.

3. (Communication complexity) For all the executions of Level_l, $\mathcal{U}$ sends $(q_1^l \ldots, q_{C_l}^l)$, with $q_c^l \in \mathbb{Z}_{N^2}^*$ (we stress that he sends only one query valid for all the executions); so $\mathcal{U}$ sends $2kC_l$ bits.

   $\mathcal{S}$ replies sending $(u_1^l, \ldots, u_{R_l}^l, v_1^l, \ldots, v_{R_l}^l)$, with $u_r^l, v_r^l \in \mathbb{Z}_N$; so $\mathcal{S}$ sends $2kR_l$ bits.

   To compute the communication complexity we need to fix R_l and C_l. For every $l \in \mathcal{I}_L$, we set $C_l = n^{1/(L+1)}$, thus $R_l = n^{l/(L+1)}$. In this setting, $\mathcal{U}$ sends $2kn^{1/(L+1)}$ bit for each level, that is $\mathcal{U}$ sends $2kLn^{1/(L+1)}$ bits. $\mathcal{S}$ sends his answer only when he performs Level_1 and for each execution of Level_1 he sends $2kR_1 = 2kn^{1/(L+1)}$ bits. To conclude we have to calculate how many executions of Level_1 are needed.

   Remark that for each execution of Level_l, we need 2 executions of Level_{l-1}. It is easy to prove by induction that Level_l is executed 2^{L-l} times, for every $l \in \mathcal{I}_L$. Therefore we have 2^{L-1} executions of Level_1 and so $\mathcal{S}$ sends $2^{L-1}2kn^{1/(L+1)} = 2^L kn^{1/(L+1)}$ bits.

   Hence the total amount of communication is:

$$\mathcal{CC}_{\mathcal{CR}}(n) = k + 2kLn^{1/(L+1)} + 2^L kn^{1/(L+1)} = n^{1/(L+1)}(2kL + 2^L k) + k.$$

   The security parameter k is fixed and small (with respect to n). We have to choose the number of levels of recursion L in such a way that it minimizes the communication complexity. Assuming that the $\mathcal{CC}_{\mathcal{CR}}(n)$ is differentiable with respect to L and considering k and n as parameters, we can look for a minimum studying the first derivative of $\mathcal{CC}_{\mathcal{CR}}(n)$ with respect to L:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{CC}_{\mathcal{CR}}(n)}{\mathrm{d}L} &= -\frac{n^{1/(L+1)}\ln n}{(L+1)^2}(2kL + 2^L k) + n^{1/(L+1)}(2k + 2^L k \ln 2) = \\ &= 2kn^{1/(L+1)}\left(1 + 2^{L-1}\ln 2 - \frac{(L+2^{L-1})\ln n}{(L+1)^2}\right).\end{aligned}$$

We have:

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{CC}_{\mathcal{CR}}(n)}{\mathrm{d}L} > 0 \quad &\Longleftrightarrow \quad 1 + 2^{L-1}\ln 2 - \frac{(L+2^{L-1})\ln n}{(L+1)^2} > 0 \\
&\Longleftrightarrow \quad (L+1)^2(1+2^{L-1}\ln 2) > (L+2^{L-1})\ln n \\
&\Longleftrightarrow \quad |L+1| = L+1 > \sqrt{\frac{(L+2^{L-1})\ln n}{1+2^{L-1}\ln 2}} \approx \\
&\qquad\qquad \approx \sqrt{\frac{(L+2^{L-1})\ln n}{2^{L-1}\ln 2}} \approx \sqrt{\frac{\ln n}{\ln 2}} = \sqrt{\log n}.
\end{aligned}$$

Thus the best choice is $L \approx \sqrt{\log n} - 1$. We see that it does not depend on the security parameter k. With such a L we have:

$$\begin{aligned}
\mathcal{CC}_{\mathcal{CR}}(n) &= n^{1/(L+1)}(2kL + 2^L k) + k \approx \\
&\approx \left(2^{\log n}\right)^{1/\sqrt{\log n}} \left(2k\left(\sqrt{\log n} - 1\right) + 2^{\sqrt{\log n}-1}k\right) + k = \\
&= 2^{\sqrt{\log n}} 2k\left(\sqrt{\log n} - 1\right) + 2^{\sqrt{\log n}} 2^{\sqrt{\log n}-1}k + k = \\
&= 2^{\sqrt{\log n}+\log(\sqrt{\log n}-1)+1}k + 2^{2\sqrt{\log n}-1}k + k = \mathcal{O}\left(2^{2\sqrt{\log n}}\right).
\end{aligned}$$

□

At each level, $\mathcal{U}$ sends $n^{1/(L+1)}$ elements of $\mathbb{Z}^*_{N^2}$ and all but one are in $\mathscr{NR}_{N^2}$. Since $|\mathscr{NR}_N| = \phi(N)$, in order to avoid repetitions (which would reveal where the N^{th} non-residue is not), we must choose the security parameter k in such a way that $\phi(N) \geq n^{1/(L+1)}$. We have $k = \lceil \log N \rceil = 2\lceil \log p_1 \rceil = 2\lceil \log p_2 \rceil$ and $\phi(N) = N - p_1 - p_2 + 1$, thus we must have $2^k - 2^{k/2+1} + 1 \geq n^{1/(L+1)}$.

5.3 $\mathcal{HE}$: based on homomorphic encryption scheme

This scheme uses the *Discrete Logarithm Problem*, thus we start this section giving a short presentation of this well-known problem.

5.3.1 Discrete Logarithm Problem

The function $y = x^e \mod N$ is called *modular exponentiation*; N can be either a prime or a composite. We can invert the modular exponentiation in two ways: with respect to x or with respect to e. The fist way deals with the e^{th} residuosity of y modulo N, the second way is the discrete logarithm.

Definition 5.3.1 (Discrete Logarithm Problem (DLP)). *Let N be a positive integer. The* Discrete Logarithm Problem *(or* DLP*) is: Given $x, y \in \mathbb{Z}_N$, find an $e \in \mathbb{N}$ such that $x^e = y \mod N$. Such an e is called the* discrete logarithm with base x of y modulo N.

If N is a (large) composite, it is believed that solving the Discrete Logarithm Problem without knowing the factorization of N is computationally hard.

5.3.2 Homomorphic Encryption Scheme

This protocol is by Ostrovsky and Skeith III and it is based on particular cryptosystems. Let $(\mathfrak{K}, \mathfrak{E}, \mathfrak{D})$ be a *(public-key) cryptosystem* [12] with $\mathfrak{K}$ (resp. $\mathfrak{E}$, resp. $\mathfrak{D}$) the *key generator* (resp. *encryption*, resp. *decryption*) *algorithm.* To construct our cPIR protocol, we only need that the cryptosystem is:

Secure : The cryptosystem is secure against a *chosen-plaintext attack i.e.* the distributions of $\mathfrak{E}$'s outputs (called *ciphertexts*) are computationally indistinguishable by varying the input (called *plaintext*). It implies that $\mathfrak{E}$ must be probabilistic;

Homomorphic over an abelian group : The plaintext set and the ciphertext set are abelian groups; we denote them by $(G, *)$ and $(G', \star)$ respectively and from this point forward we use additive notation for the group operations.We also require that for every $a, b \in G$:

$$\mathfrak{D}(\mathfrak{E}(a) \star \mathfrak{E}(b)) = a * b.$$

¿From this point forward we denote by $\cdot$ the operations which make the abelian groups G and G' $\mathbb{Z}$-modules.

5.3.3 Protocol $\mathcal{HE}$: generic construction for any homomorphic encryption scheme

Let $(\mathfrak{K}, \mathfrak{E}, \mathfrak{D})$ be a secure homomorphic encryption scheme as above. In order to have a cryptosystem of any conceivable use, we must have that $|G| > 1$; so there exists at least one element $g \in G$ such that ord $(g) = m > 1$. If the DLP in G is hard, we consider the database x as a n-bit string $x = (x_1, \ldots, x_n) \in \mathbb{Z}_2^n$. Otherwise (for instance when G is an additive group of integers: in this case

the discrete logarithm is just a division) we view the database x not as a bit string, but as a sequence of n elements in $\mathbb{Z}_m$ (that is $x = (x_1, \ldots, x_n) \in \mathbb{Z}_m^n$).

We construct the protocol $\mathcal{HE}$ in two steps: the 1^{th} step is to present a basic cPIR scheme based on homomorphic encryption; the 2^{th} step is to improve it considering the database as a d-dimentional cube, for some $d \in \mathbb{N}_{>0}$.

Remark 5.4. In the original paper [21] the second step is done only for $d = 2$ and the generalization is left to the reader. Here we do it, describing precisely how the protocol works for any $d > 0$.

Basic scheme $\mathcal{B}$

Let G, G', g and m be as above. Let $i \in \mathcal{I}_n$ be the index of the item desired by $\mathcal{U}$.

$\mathcal{Q}$:
- $\texttt{INPUT}\leftarrow (1^n, i)$,
- For every $1 \leq j \leq n$, choose $q_j \in_R G'$ such that:
$$\begin{cases} \mathfrak{D}(q_i) = g, \\ \mathfrak{D}(q_j) = 0_G \quad \forall j \neq i, \end{cases}$$
- $\texttt{OUTPUT}\rightarrow Q = (q_1, \ldots, q_n)$;

$\mathcal{A}$:
- $\texttt{INPUT}\leftarrow (x, Q)$
with $\begin{cases} x = (x_1, \ldots, x_n) \text{ the database}, \\ Q \in (G')^n \text{ the query sent by } \mathcal{U}, \end{cases}$
- Let $a = \sum_{j=1}^{n} (x_j \cdot q_j) \ \in G'$,
- $\texttt{OUTPUT}\rightarrow (a)$;

$\mathcal{R}$:
- $\texttt{INPUT}\leftarrow (1^n, i, a)$
with a the answer sent by $\mathcal{S}$,
- Let $b' = \mathfrak{D}(a) \ \in G$,
- Let $b \in \mathbb{Z}$ be such that:
$$\begin{cases} b = \log_g b' & \text{if the DLP is easy (then } b \in \mathbb{Z}_m), \\ b = 1 \Leftrightarrow b' = g & \text{otherwise (then } b \in \mathbb{Z}_2), \end{cases}$$
- $\texttt{OUTPUT}\rightarrow (b)$.

Theorem 5.4.1. *The protocol $\mathcal{B}$ defined above is a cPIR protocol such that $\mathcal{CC}_{\mathcal{B}}(n) = \mathcal{O}(kn)$, with $k = \lceil \log |G'| \rceil$.*

Proof. We have to prove that $\mathcal{B}$ verify the definition 2.0.2 and that it has the required communication complexity.

1. (Correctness) Since $(\mathfrak{K}, \mathfrak{E}, \mathfrak{D})$ is a homomorphic cryptosystem and $\cdot$ is the $\mathbb{Z}$-module action on G and G', we have:

$$\begin{aligned} b' &= \mathfrak{D}(a) = \mathfrak{D}\left(\sum_{j=1}^{n} x_j \cdot q_j\right) = \sum_{j=1}^{n} \mathfrak{D}(x_j \cdot q_j) = \sum_{j=1}^{n} x_j \cdot \mathfrak{D}(q_j) = \\ &= \left(\sum_{j\neq i, j=1}^{n} x_j \cdot \mathfrak{D}(q_j)\right) * \big(x_i \cdot \mathfrak{D}(q_i)\big) = 0_G * (x_i \cdot g) = x_i \cdot g. \end{aligned}$$

   If DLP is easy, then $b = \log_g b' = \log_g (x_i \cdot g) = x_i$ (recall that we use additive notation for the group operations).

   Otherwise, we have set $x_i \in \mathbb{Z}_2$. If $x_i = 1$, then $b' = g \neq 0_G$ (the inequality follows from the fact that we have chosen g so that $\operatorname{ord}(g) > 1$). On the contrary if $x_i = 0$, then $b' = 0_G$. Thus $x_i = 1$ if and only it $b' = g$, so $b = x_i$.

2. (Privacy) By our assumption on the cryptosystem, the distribution of ciphertexts obtained ciphering g is computationally indistinguishable from the one obtained ciphering 0_G. Let $i, i' \in \mathcal{I}_n$ be some different indices and let D_i and $D_{i'}$ be the distributions of queries on i and i' respectively. If $Q = (q_1, \ldots, q_n) \in D_i$ (resp. $Q \in D_{i'}$), then it consists of n elements of G', all but the i^{th} (resp. i'^{th}) ciphering 0_G and q_i (resp. $q_{i'}$) ciphers g. If $\mathcal{S}$ can distinguish D_i and $D_{i'}$, then he can distinguish $\mathfrak{E}(g)$ and $\mathfrak{E}(0_G)$, in contradiction to our assumption. Thus $\mathcal{S}$ cannot efficiently gain any information about i.

   The formal proof is exactly as that one done to prove the privacy of protocol $\mathcal{B}$ in Chapter 3.

3. (Communication complexity) Let $k = \lceil \log |G'| \rceil$ be the security parameter. $\mathcal{U}$ sends $Q = (q_1, \ldots, q_n) \in (G')^n$ to $\mathcal{S}$, so $\mathcal{U}$ sends kn bits; $\mathcal{S}$ replies sending $a \in G'$, so $\mathcal{S}$ sends k bits. Thus $\mathcal{CC}_{\mathcal{B}}(n) = kn + k = \mathcal{O}(kn)$.

□

The communication complexity of $\mathcal{B}$ is proportional to n and so this protocol is not more efficient than the trivial solution. The next step is to modify $\mathcal{B}$ so as to obtain a protocol with less communication complexity.

Protocol $\mathcal{HE}$

Let G, G', g and m be as above (in particular $m = \text{ord}\,(g) > 1$). The key idea is to consider the database $x = (x_1, \ldots, x_n)$ as a d-dimensional cube, for some $d \in \mathbb{N}_{>0}$. That is $x = (x_{j_1,\ldots,j_d})_{j_s \in \mathcal{I}_{n^{1/d}}}$. Recall that if DLP is hard in G, then $x_{j_1,\ldots,j_d} \in \mathbb{Z}_2$; otherwise $x_{j_1,\ldots,j_d} \in \mathbb{Z}_m$. Let $i = (i_1, \ldots, i_d)$ be the index of the item desired by $\mathcal{U}$.

Let $\theta : G' \hookrightarrow \mathbb{Z}^l$ be an injective map such that θ and θ^{-1} are efficiently computable and for every $t \in \mathcal{I}_l$ and for every $y \in G'$:

$$\theta(y)_t < m = \text{ord}\,(g)$$

with $\theta(y)_t$ the t^{th} component of $\theta(y)$. That is $\theta : G' \hookrightarrow \mathbb{Z}_m^l$.

Being θ injective, we have $G' \leq |\mathbb{Z}_m^l| = m^l$. The decryption function $\mathfrak{D} : G' \to G$ is always surjective (since we can cipher and decipher any plaintext), but it is never injective (since the cryptosystem must be probabilistic and so any plaintext corresponds to many different ciphertexts). It follows that $|G'| > |G|$; thus $m = \text{ord}\,(g) \leq |G| < |G'| \leq m^l$. Therefore we always have $l > 1$.

The definition of θ may seem quite tricky, but actually it is not: we do not ask any algebraic conditions from θ, but it can be any easily computed injective map. For instance, since $\text{ord}\,(g) > 1$, θ can be the map which sends any element of G' into its binary expansion (it is clearly injective); in this case we have to choose $l \geq k$.

$\mathcal{Q}$: – $\texttt{INPUT}\leftarrow (1^n, i = (i_1, \ldots, i_d))$,

– For every $s \in \mathcal{I}_d$, for $1 \leq j \leq n^{1/d}$ choose $q_{j_s}^s \in_R G'$ such that:

$$\begin{cases} \mathfrak{D}(q_{i_s}^s) = g, \\ \mathfrak{D}(q_{j_s}^s) = 0_G \quad \forall j_s \neq i_s, \end{cases}$$

– $\texttt{OUTPUT}\rightarrow Q = (q_{j_s}^s)_{s \in \mathcal{I}_d, j_s \in \mathcal{I}_{n^{1/d}}}$;

$\mathcal{A}$: – $\texttt{INPUT}\leftarrow (x, Q)$

$$\text{with} \begin{cases} x = (x_{j_1,\ldots,j_d})_{j_s \in \mathcal{I}_{n^{1/d}}} \text{ the database,} \\ Q = (q_{j_s}^s) \in (G')^{dn^{1/d}} \text{ the query sent by } \mathcal{U}, \end{cases}$$

The answer algorithm is recursive and consists of d levels of recursion. We denote the r^{th} level by Level_r.

- Level_1. For every $(j_2, \dots, j_d) \in \mathcal{I}_{n^{1/d}}^{d-1}$ let:
$$a^1_{j_2,\dots,j_d} = \sum_{j_1=1}^{n^{1/d}} \left(x_{j_1,j_2,\dots,j_d} \cdot q^1_{j_1}\right),$$
- Level_2. For every $(j_3, \dots, j_d) \in \mathcal{I}_{n^{1/d}}^{d-2}$, for $1 \le t_2 \le l$ let:
$$(a^2_{j_3,\dots,j_d})_{t_2} = \sum_{j_2=1}^{n^{1/d}} \left[\theta(a^1_{j_2,j_3,\dots,j_d})_{t_2} \cdot q^2_{j_2}\right]$$
with $\theta(a^1_{j_2,\dots,j_d})_{t_2}$ the t_2^{th} component of $\theta(a^1_{j_2,\dots,j_d}) \in \mathbb{Z}_m^l$,
- Level_r $(2 < r < d)$. For every $(j_{r+1}, \dots, j_d) \in \mathcal{I}_{n^{1/d}}^{d-r}$ and for every $(t_2, \dots, t_{r-1}) \in \mathcal{I}_l^{r-2}$, for $1 \le t_r \le l$ let:
$$(a^r_{j_{r+1},\dots,j_d})_{t_2\dots t_{r-1}t_r} = \sum_{j_r=1}^{n^{1/d}} \left[\theta\left((a^{r-1}_{j_r,\dots,j_d})_{t_2\dots t_{r-1}}\right)_{t_r} \cdot q^r_{j_r}\right],$$
This scheme can be applied also for $r = 2, d$ paying attention to the existence of indices.
- Level_d. For every $(t_2, \dots, t_{d-1}) \in \mathcal{I}_l^{d-2}$, for $1 \le t_d \le l$ let:
$$(a^d)_{t_2\dots t_{d-1}t_d} = \sum_{j_d=1}^{n^{1/d}} \left[\theta\left((a^{d-1}_{j_d})_{t_2\dots t_{d-1}}\right)_{t_d} \cdot q^d_{j_d}\right],$$
- $\texttt{OUTPUT}\rightarrow a = \left((a^d)_{t_2\dots t_d}\right)_{t_s \in \mathcal{I}_l} \in (G')^{l^{d-1}}$;

$\mathcal{R}$: - $\texttt{INPUT}\leftarrow \left(1^n, i = (i_1, \dots, i_d), a = \left((a^d)_{t_2\dots t_d}\right)_{t_s \in \mathcal{I}_l}\right)$,
The reconstruction algorithm is also recursive and it consists of d levels of recursion. As usual we denote the r^{th} level by Level_r.

- Level_d. For every $(t_2, \dots, t_{d-1}) \in \mathcal{I}_l^{d-2}$, for $1 \le t_d \le l$ do:
  - Let $e^d_{t_2\dots t_{d-1}t_d} = \mathfrak{D}(a^d_{t_2\dots t_{d-1}t_d}) \in G$,
  - Let $f^d_{t_2\dots t_{d-1}t_d} \in \mathbb{Z}$ be such that:
$$\begin{cases} f^d_{t_2\dots t_{d-1}t_d} = \log_g\left(e^d_{t-2\dots t_{d-1}t_d}\right) & \text{if DLP is easy} \\ & \text{(then } f^d_{t_2\dots t_{d-1}t_d} \in \mathbb{Z}_m\text{)}, \\ f^d_{t_2\dots t_{d-1}t_d} = 1 \Leftrightarrow e^d_{t-2\dots t_{d-1}t_d} = g & \text{otherwise} \\ & \text{(then } f^d_{t_2\dots t_{d-1}t_d} \in \mathbb{Z}_2\text{)}, \end{cases}$$
  - Let $f^d_{t_2\dots t_{d-1}} = (f^d_{t_2\dots t_{d-1}1}, \dots, f^d_{t_2\dots t_{d-1}l}) \in \mathbb{Z}_m^l$,
  - Let $\overline{a}^{d-1}_{t_2\dots t_{d-1}} = \theta^{-1}(f^d_{t_2\dots t_{d-1}}) \in G'$,
- Level_r $(d > r > 2)$. For every $(t_2, \dots, t_{r-1}) \in \mathcal{I}_l^{r-2}$, for $1 \le t_r \le l$ do:
  - Let $e^r_{t_2\dots t_{r-1}t_r} = \mathfrak{D}(\overline{a}^r_{t_2\dots t_{r-1}t_r}) \in G$,

* Let $f^r_{t_2\dots t_{r-1}t_r} \in \mathbb{Z}$ be such that:

$$\begin{cases} f^r_{t_2\dots t_{r-1}t_r} = \log_g\left(e^r_{t-2\dots t_{r-1}t_r}\right) & \text{if DLP is easy} \\ & \text{(then } f^r_{t_2\dots t_{r-1}t_r} \in \mathbb{Z}_m\text{)}, \\ f^r_{t_2\dots t_{r-1}t_r} = 1 \Leftrightarrow e^r_{t-2\dots t_{r-1}t_r} = g & \text{otherwise} \\ & \text{(then } f^d_{t_2\dots t_{d-1}t_d} \in \mathbb{Z}_2\text{)}, \end{cases}$$

* Let $f^r_{t_2\dots t_{r-1}} = (f^r_{t_2\dots t_{r-1}1}, \dots, f^r_{t_2\dots t_{r-1}l}) \in \mathbb{Z}^l_m$,
* Let $\overline{a}^{r-1}_{t_2\dots t_{r-1}} = \theta^{-1}(f^r_{t_2\dots t_{r-1}}) \in G'$,

– Level$_2$. For $1 \leq t_2 \leq l$ do:

* Let $e^2_{t_2} = \mathfrak{D}(\overline{a}^2_{t_2}) \in G$,
* Let $f^2_{t_2} \in \mathbb{Z}$ be such that:

$$\begin{cases} f^2_{t_2} = \log_g\left(e^2_{t-2}\right) & \text{if DLP is easy (then } f^2_{t_2} \in \mathbb{Z}_m\text{)}, \\ f^2_{t_2} = 1 \Leftrightarrow e^2_{t-2} = g & \text{otherwise (then } f^d_{t_2\dots t_{d-1}t_d} \in \mathbb{Z}_2\text{)}, \end{cases}$$

* Let $f^2 = (f^2_1, \dots, f^2_l) \in \mathbb{Z}^l_m$,
* Let $\overline{a}^1 = \theta^{-1}(f^2) \in G'$,

– Level$_1$:

* Let $e^1 = \mathfrak{D}(\overline{a}^1) \in G$,
* Let $f^1 \in \mathbb{Z}$ be such that:

$$\begin{cases} f^1 = \log_g\left(e^1\right) & \text{if DLP is easy (then } f^1 \in \mathbb{Z}_m\text{)}, \\ f^1 = 1 \Leftrightarrow e^1 = g & \text{othewise (then } f^1 \in \mathbb{Z}_2\text{)}, \end{cases}$$

– OUTPUT$\rightarrow (f^1)$.

Theorem 5.4.2. *The protocol $\mathcal{HE}$ defined above is a cPIR protocol such that $\mathcal{CC}_{\mathcal{HE}}(n) = \mathcal{O}(n^{1/d})$, for any $d \in \mathbb{N}_{>0}$.*

Proof. We have to prove that $\mathcal{HE}$ verifies the definition 2.0.2 and that it has the required communication complexity.

1. (Correctness) We prove by induction that, for every $(t_2, \dots, t_d) \in \mathcal{I}^{d-2}_l$, $\overline{a}^r_{t_2\dots t_r} = (a^r_{i_{r+1},\dots,i_d})_{t_2\dots t_r}$, for every $d > r \geq 1$ (paying attention to the existence of the indices).

If $r = d-1$, then:

$$e^d_{t_2\dots t_{d-1}t_d} = \mathfrak{D}(a^d_{t_2\dots t_{d-1}t_d}) = \mathfrak{D}\left(\sum_{j_d=1}^{n^{1/d}}\left[\theta\big((a^{d-1}_{j_d})_{t_2\dots t_{d-1}}\big)_{t_d}\cdot q^d_{j_d}\right]\right) =$$

$$= \sum_{j_d=1}^{n^{1/d}}\left[\theta\big((a^{d-1}_{j_d})_{t_2\dots t_{d-1}}\big)_{t_d}\cdot \mathfrak{D}(q^d_{j_d})\right] = \theta\big((a^{d-1}_{i_d})_{t_2\dots t_{d-1}}\big)_{t_d}\cdot g.$$

If DLP is easy, then $f^d_{t_2\dots t_{d-1}t_d} = \log_g\left(e^d_{t_2\dots t_{d-1}t_d}\right) = \theta\big((a^{d-1}_{i_d})_{t_2\dots t_{d-1}}\big)_{t_d}$.
Otherwise $e^d_{t_2\dots t_{d-1}t_d} = g \Leftrightarrow \theta\big((a^{d-1}_{i_d})_{t_2\dots t_{d-1}}\big)_{t_d} = 1$.
Thus $f^d_{t_2\dots t_{d-1}t_d} = \theta\big((a^{d-1}_{i_d})_{t_2\dots t_{d-1}}\big)_{t_d}$ and so $f^d_{t_2\dots t_{d-1}} = \theta\big((a^{d-1}_{i_d})_{t_2\dots t_{d-1}}\big)$. Therefore $\overline{a}^{d-1}_{t_2\dots t_{d-1}} = \theta^{-1}(f^d_{t_2\dots t_{d-1}}) = (a^{d-1}_{i_d})_{t_2\dots t_{d-1}}$.
Suppose it is true for r, then:

$$e^r_{t_2\dots t_{r-1}t_r} = \mathfrak{D}(\overline{a}^r_{t_2\dots t_{r-1}t_r}) = \mathfrak{D}((a^r_{i_{r+1},\dots,i_d})_{t_2\dots t_{r-1}t_r}) =$$

$$= \mathfrak{D}\left(\sum_{j_r=1}^{n^{1/d}}\left[\theta\big((a^{r-1}_{j_r,i_{r+1},\dots,i_d})_{t_2\dots t_{r-1}}\big)_{t_r}\cdot q^r_{j_r}\right]\right) =$$

$$= \sum_{j_r=1}^{n^{1/d}}\left[\theta\big((a^{r-1}_{j_r,i_{r+1},\dots,i_d})_{t_2\dots t_{r-1}}\big)_{t_r}\cdot \mathfrak{D}(q^r_{j_r})\right] =$$

$$= \theta\big((a^{r-1}_{i_r,i_{r+1},\dots,i_d})_{t_2\dots t_{r-1}}\big)_{t_r}\cdot g.$$

If DLP is easy, then $f^r_{t_2\dots t_{r-1}t_r} = \log_g\left(e^r_{t_2\dots t_{r-1}t_r}\right) = \theta\big((a^{r-1}_{i_r,i_{r+1},\dots,i_d})_{t_2\dots t_{r-1}}\big)_{t_r}$.
Otherwise $e^r_{t_2\dots t_{r-1}t_r} = g \Leftrightarrow \theta\big((a^{r-1}_{i_r,i_{r+1},\dots,i_d})_{t_2\dots t_{r-1}}\big)_{t_r} = 1$. Thus $f^r_{t_2\dots t_{r-1}t_r} = \theta\big((a^{r-1}_{i_r,i_{r+1},\dots,i_d})_{t_2\dots t_{r-1}}\big)_{t_r}$.
So $f^r_{t_2\dots t_{r-1}} = \theta\big((a^{r-1}_{i_r,i_{r+1},\dots,i_d})_{t_2\dots t_{r-1}}\big)$. Therefore $\overline{a}^{r-1}_{t_2\dots t_{r-1}} = \theta^{-1}(f^r_{t_2\dots t_{r-1}}) = (a^{r-1}_{i_r,i_{r+1},\dots,i_d})_{t_2\dots t_{r-1}}$.
Hence $\overline{a}^1 = a^1_{i_2,i_3,\dots,i_d}$ and it follows that:

$$e^1 = \mathfrak{D}(\overline{a}^1) = \mathfrak{D}\big((a^1_{i_2,\dots,i_d})_{t_2}\big) = \mathfrak{D}\left(\sum_{j_1=1}^{n^{1/d}}\left[x_{j_1,i_2,\dots,i_d}\cdot q^1_{j_1}\right]\right) =$$

$$= \sum_{j_1=1}^{n^{1/d}}\left[x_{j_1,i_2,\dots,i_d}\cdot \mathfrak{D}(q^1_{j_1})\right] = x_{i_1,i_2,\dots,i_d}\cdot g.$$

If DLP is easy, then: $f^1 = \log_g(e^1) = x_{i_1,\ldots,i_d}$.
Otherwise $e^1 = g \Leftrightarrow x_{i_1,\ldots,i_d} = 1$; Thus $f^1 = x_{i_1,\ldots,i_d}$.

2. (Privacy) It follows from the privacy of $\mathcal{B}$.

3. (Communication complexity) The security parameter is $k = \lceil \log |G'| \rceil$. $\mathcal{U}$ sends $Q \in (G')^{dn^{1/d}}$, that is $kdn^{1/d}$ bits, to $\mathcal{S}$ who replies sending $a \in (G')^{l^{d-1}}$, that is kl^{d-1} bits. Thus the total amount of communication is:

$$\mathcal{CC}_{\mathcal{HE}}(n) = kdn^{1/d} + kl^{d-1} = \mathcal{O}\big(n^{1/d}\big)$$

Note that for every $\epsilon > 0$, if we choose $d = \lceil 1/\epsilon \rceil$, than we have that $\mathcal{CC}_{\mathcal{HE}}(n) = \mathcal{O}\big(n^{\epsilon}\big)$.

□

Remark 5.5. This generic method actually captures protocols $\mathcal{P}$ and $\mathcal{CR}$ as long as the appropriate encryption schemes are in place. Our aim now is to show it precisely. In the original paper [21] the authors claim it without giving details.

5.5.1 Protocol $\mathcal{P}$: special case of $\mathcal{HE}$ for encryption scheme based on QRA

This cyptosystem is by Goldwasser and Micali [13]. In this subsection we use the notation of Chapter 3.

Cryptosystem $(\mathfrak{K}, \mathfrak{E}, \mathfrak{D})$ based on QRA: Alice=receiver, Bob=sender

Security parameter $k \in \mathbb{N}$.

$\mathfrak{K}$: It is run by Alice.

- Choose at random $p_1 \neq p_2$ prime numbers such that $|p_1| = |p_2| = k/2$,
- Let $N = p_1 p_2$,
- *Plaintext set* $G = (\mathbb{Z}_2, +)$ additive group.
  Thus $0_G = 0$ and $g = 1$,
- *Ciphertext set* $G' = (\mathbb{Z}_N^*, \times)$ multiplicative group,
- Choose $y \in_R \mathscr{PQR}_N$,

– *Public key*$=(N, y)$ held by both Alice and Bob,

– *Private key*$=(p_1, p_2)$ held only by Alice;

$\mathfrak{E}$: It is run by Bob to cipher $x \in \mathbb{Z}_2$ as $c \in \mathbb{Z}_N^*$.

– Let $c = \mathfrak{E}(x) = y^x r^2 \mod N$, with $r \in_R \mathbb{Z}_N^*$;

$\mathfrak{D}$: It is run by Alice to reconstruct x from c.

– Let $\mathfrak{D}(c) \in \mathbb{Z}_2$ be such that: $\mathfrak{D}(c) = 0 \Leftrightarrow c \in \mathscr{QR}_N$.

As Alice knows the factorization of N, she can compute the quadratic residuosity of c.

It is easy to see that the cryptosystem is correct (clearly $c \in \mathscr{QR}_N$ if and only if $x = 0$) and also secure under QRA. Moreover the cryptosystem is *homomorphic*: of course $(\mathbb{Z}_2, +)$ and $(\mathbb{Z}_N^*, \times)$ are abelian groups and we also have:

$$\mathfrak{D}(\mathfrak{E}(x)\mathfrak{E}(x') \mod N) = x + x' \mod 2$$

for every $x, x' \in \mathbb{Z}_2$. Infact $\mathfrak{E}(x)\mathfrak{E}(x') = y^x r^2 y^{x'} r'^2 = y^{x+x'}(rr')^2 \mod N$. As $y \in \mathscr{PQR}_N$, $\mathfrak{E}(x)\mathfrak{E}(x') \in \mathscr{QR}_N \Leftrightarrow x + x' = 0 \mod 2$, as wanted.

Protocol $\mathcal{P}$

Let $N = p_1 p_2$, $G = (\mathbb{Z}_2, +)$, $G' = (\mathbb{Z}_N^*, \times)$, $g = 1$ be as above. Thus $m = \text{ord}\,(g) = 2$. For some $d \in \mathbb{N}_{>0}$, let the database be $x = (x_{j_1,\dots,j_d})_{j_s \in \mathcal{I}_{n^{1/d}}}$ with $x_{j_1,\dots,j_d} \in \mathbb{Z}_2$. Let $i = (i_1, \dots, i_d)$ be the index of the bit $\mathcal{U}$ wants to retrieve.

Let $\theta : G' \hookrightarrow \mathbb{Z}_m^l$ be the map which sends any element of $G' = \mathbb{Z}_N^*$ into its binary expansion, that is $\theta : \mathbb{Z}_N^* \hookrightarrow \mathbb{Z}_2^k$, with $k = \lceil \log |G'| \rceil = \lceil \log N \rceil$ security parameter.

$\mathcal{Q}$: – $\texttt{INPUT}\leftarrow (1^n, i = (i_1, \dots, i_d))$,

– For every $s \in \mathcal{I}_d$, for $1 \le j_s \le n^{1/d}$ choose $q_{j_s}^s \in_R \mathbb{Z}_N^*$ such that:

$$\begin{cases} q_{i_s}^s = \mathfrak{E}(1) \quad (i.e.\ q_{i_s}^s \in \mathscr{PQR}_N), \\ q_{j_s}^s = \mathfrak{E}(0) \quad (i.e.\ q_{j_s}^s \in \mathscr{QR}_N) \qquad \forall j_s \neq i_s, \end{cases}$$

– $\texttt{OUTPUT}\rightarrow Q = (q_{j_s}^s)_{s \in \mathcal{I}_d, j_s \in \mathcal{I}_{n^{1/d}}}$;

$\mathcal{A}$: $(\mathbb{Z}_N^*, \times)$ is a multiplicative group, so we use the multiplicative notation for G', instead of the additive one used in $\mathcal{HE}$:

– $\texttt{INPUT}\leftarrow (x, Q)$,

– Level_1. For every $(j_2, \ldots, j_d) \in \mathcal{I}_{n^{1/d}}^{d-1}$, let:

$$a^1_{j_2,\ldots,j_d} = \prod_{j_1=1}^{n^{1/d}} (q^1_{j_1})^{x_{j_1,j_2,\ldots,j_d}} \mod N,$$

– Level_r $(2 \leq r \leq d)$. For every $(j_{r+1}, \ldots, j_d) \in \mathcal{I}_{n^{1/d}}^{d-r}$ and for every $(t_2, \ldots, t_{r-1}) \in \mathcal{I}_k^{r-2}$, for $1 \leq t_r \leq k$ let:

$$(a^r_{j_{r+1},\ldots,j_d})_{t_2\ldots t_{r-1}t_r} = \prod_{j_r=1}^{n^{1/d}} (q^r_{j_r})^{\left((a^{r-1}_{j_r,\ldots,j_d})_{t_2\ldots t_{r-1}}\right)_{t_r}} \mod N,$$

with $\left((a^{r-1}_{j_r,\ldots,j_d})_{t_2\ldots t_{r-1}}\right)_{t_r}$ the $t_r{}^{\text{th}}$ bit of the binary expansion in $\mathbb{Z}_2^k$ of $(a^{r-1}_{j_r,\ldots,j_d})_{t_2\ldots t_{r-1}}$,

– $\texttt{OUTPUT}\rightarrow a = \left((a^d)_{t_2\ldots t_d}\right)_{t_s\in\mathcal{I}_k} \in (\mathbb{Z}_N^*)^{k^{d-1}}$;

$\mathcal{R}$: – $\texttt{INPUT}\leftarrow \left(1^n, i = (i_1, \ldots, i_d), a = \left((a^d)_{t_2\ldots t_d}\right)_{t_s\in\mathcal{I}_k}\right)$,

– Level_d. For every $(t_2, \ldots, t_{d-1}) \in \mathcal{I}_k^{d-2}$, for $1 \leq t_d \leq k$ do:

* Let $f^d_{t_2\ldots t_{d-1}t_d} = \mathfrak{D}(a^d_{t_2\ldots t_{d-1}t_d}) \in \mathbb{Z}_2$
(Recall $g = 1$, therefore $f^d_{t_2\ldots t_{d-1}t_d} = 1 \Leftrightarrow e^d_{t_2\ldots t_{d-1}t_d} = g$ means $f^d_{t_2\ldots t_{d-1}t_d} = e^d_{t-2\ldots t_{d-1}t_d}$),
* Let $f^d_{t_2\ldots t_{d-1}} = (f^d_{t_2\ldots t_{d-1}1}, \ldots, f^d_{t_2\ldots t_{d-1}k}) \in \mathbb{Z}_2^k$,
* Let $\overline{a}^{d-1}_{t_2\ldots t_{d-1}} = \theta^{-1}(f^d_{t_2\ldots t_{d-1}}) = f^d_{t_2\ldots t_{d-1}1} \ldots f^d_{t_2\ldots t_{d-1}k}$ viewed as the binary expansion of an element of $\mathbb{Z}_N^*$,

– Level_r $(d > r > 2)$. For every $(t_2, \ldots, t_{r-1}) \in \mathcal{I}_k^{r-2}$, for $1 \leq t_r \leq k$ do:

* Let $f^r_{t_2\ldots t_{r-1}t_r} = \mathfrak{D}(\overline{a}^r_{t_2\ldots t_{r-1}t_r}) \in \mathbb{Z}_2$,
* Let $\overline{a}^{r-1}_{t_2\ldots t_{r-1}} = \theta^{-1}(f^r_{t_2\ldots t_{r-1}1}, \ldots, f^r_{t_2\ldots t_{r-1}k}) = f^r_{t_2\ldots t_{r-1}1} \ldots f^r_{t_2\ldots t_{r-1}k}$ viewed as the binary expansion of an element of $\mathbb{Z}_N^*$,

– Level_1. Let $f^1 = \mathfrak{D}(\overline{a}^1) \in \mathbb{Z}_2$,

– $\texttt{OUTPUT}\rightarrow (f^1)$.

5.5.2 Protocol $\mathcal{CR}$: special case of $\mathcal{HE}$ for encryption scheme based on CRA

This cyptosystem is by Paillier [22] and infact it is called *Paillier's cryptosystem.* In this subsection we use the notation of section 5.1.

Cryptosystem $(\mathfrak{K}, \mathfrak{E}, \mathfrak{D})$ based on CRA: Alice=receiver, Bob=sender

Security parameter $k \in \mathbb{N}$.

$\mathfrak{K}$: It is run by Alice.

- Choose at random $p_1 \neq p_2$ prime numbers such that $|p_1| = |p_2| = k/2$,
- Let $N = p_1 p_2$,
- *Plaintext set* $G = (\mathbb{Z}_N, +)$ additive group.
  Thus $0_G = 0$,
- *Ciphertext set* $G' = (\mathbb{Z}^*_{N^2}, \times)$ multiplicative group,
- Choose $y \in_R \mathcal{V}$.
  By Corollary 5.1.9 this can be done efficiently by checking whether $\gcd\left(L(y^{\lambda(N)} \mod N^2), N\right) = 1$,
- *Public key*$=(N, y)$ held by both Alice and Bob,
- *Private key*$=(p_1, p_2)$ held only by Alice;

$\mathfrak{E}$: It is run by Bob to cipher $x \in \mathbb{Z}_N$ as $c \in \mathbb{Z}^*_{N^2}$.

- Let $c = \mathfrak{E}(x) = \mathfrak{E}_y(x, r) = y^x r^N \mod N^2$, with $r \in_R \mathbb{Z}^*_N$;

$\mathfrak{D}$: It is run by Alice to reconstruct x from c.

- Let $\mathfrak{D}(c) \in \mathbb{Z}_N$ be such that: $\mathfrak{D}(c) = \frac{L(c^{\lambda(N)} \mod N^2)}{L(y^{\lambda(N)} \mod N^2)} \mod N$.

As Alice knows the factorization of N, she can compute $\mathfrak{D}(c)$.

Clearly under CRA the cryptosystem is secure and its correctness follows from Theorem 5.1.10: $\mathfrak{D}(c) = [[c]]_y = [[y^x r^N \mod N^2]]_y = x$. Moreover the cryptosystem is *homomorphic*: of course $(\mathbb{Z}_N, +)$ and $(\mathbb{Z}^*_{N^2}, \times)$ are abelian groups and we also have:

$$\mathfrak{D}(\mathfrak{E}(x)\mathfrak{E}(x') \mod N^2) = x + x' \mod N$$

for every $x, x' \in \mathbb{Z}_N$. Infact $\mathfrak{E}(x)\mathfrak{E}(x') = y^x r^N y^{x'} r'^N = y^{x+x'}(rr')^N \mod N^2$ and $\mathfrak{D}(y^{x+x'}(rr')^N \mod N^2) = [[y^{x+x'} rr'^N \mod N^2]]_y = x + x' \mod N$, as wanted.

Protocol $\mathcal{CR}$:

Let $N = p_1p_2$, $G = (\mathbb{Z}_N, +)$, $G' = (\mathbb{Z}^*_{N^2}, \times)$ be as above. Let $g = 1$, thus $m = \text{ord}\,(g) = N$. For some $d \in \mathbb{N}_{>0}$, let the database be $x = (x_{j_1,\ldots,j_d})_{j_s \in \mathcal{I}_{n^{1/d}}}$, with $x_{j_1,\ldots,j_d} \in \mathbb{Z}_N$. Let $i = (i_1, \ldots, i_d)$ be the index of the item $\mathcal{U}$ wants to retrieve.

Let $\theta : G' \hookrightarrow \mathbb{Z}^l_m$ be the map defined as follows:

$$\begin{array}{rccc} \theta: & \mathbb{Z}^*_{N^2} & \hookrightarrow & \mathbb{Z}^2_N \\ & x & \mapsto & (\left\lfloor \frac{x}{N} \right\rfloor, x - N \left\lfloor \frac{x}{N} \right\rfloor). \end{array}$$

That is $\theta(x) = (u, v) \in \mathbb{Z}^2_N$ with u and u respectively the quotient and the remainder of the division of x by N; that is $x = uN + v$.

$\mathcal{Q}$: – $\texttt{INPUT}\leftarrow (1^n, i = (i_1, \ldots, i_d))$,

– For every $s \in \mathcal{I}_d$, for every $1 \le j_s \le n^{1/d}$ choose $q^s_{j_s} \in_R \mathbb{Z}^*_{N^2}$ such that:

$$\begin{cases} q^s_{i_s} = \mathfrak{E}(1) \quad (\text{since } g = 1), \\ q^s_{j_s} = \mathfrak{E}(0) \quad \forall j_s \neq i_s. \end{cases}$$

Thus $q^s_{i_s} = y(r^s_{i_s})^N \mod N^2$ and $q^s_{j_s} = (r^s_{j_s})^N \mod N^2$, for some $r^s_{i_s}, r^s_{j_s} \in_R \mathbb{Z}^*_N$. That is $q^s_{j_s} \in NR \Leftrightarrow j_s \neq i_s$,

– $\texttt{OUTPUT}\rightarrow Q = (q^s_{j_s})_{s \in \mathcal{I}_d, j_s \in \mathcal{I}_{n^{1/d}}}$;

$\mathcal{A}$: $(\mathbb{Z}^*_{N^2}, \times)$ is a multiplicative group, so we use the multiplicative notation for G', instead of the additive one used in $\mathcal{HE}$:

– $\texttt{INPUT}\leftarrow (x, Q)$,

– Level_1. For every $(j_2, \ldots, j_d) \in \mathcal{I}^{d-1}_{n^{1/d}}$, let:

$$a^1_{j_2,\ldots,j_d} = \textstyle\prod_{j_1=1}^{n^{1/d}} (q^1_{j_1})^{x_{j_1,j_2,\ldots,j_d}} \mod N^2,$$

– Level_r $(2 \le r \le d)$. For every $(j_{r+1}, \ldots, j_d) \in \mathcal{I}^{d-r}_{n^{1/d}}$ and for every $(t_2, \ldots, t_{r-1}) \in \{1, 2\}^{r-2}$, for $t_r = 1, 2$ let:

$$(a^r_{j_{r+1},\ldots,j_d})_{t_2\ldots t_{r-1}t_r} = \textstyle\prod_{j_r=1}^{n^{1/d}} (q^r_{j_r})^{\theta\left((a^{r-1}_{j_r,\ldots,j_d})_{t_2\ldots t_{r-1}}\right)_{t_r}} \mod N^2,$$

with $\theta\left((a^{r-1}_{j_r,\ldots,j_d})_{t_2\ldots t_{r-1}}\right)_{t_r}$ the $t_r{}^{\text{th}}$ component of $\theta\left((a^{r-1}_{j_r,\ldots,j_d})_{t_2\ldots t_{r-1}}\right)$. That is the quotient if $t_r = 1$ and the remainder if $t_r = 2$;

– $\texttt{OUTPUT}\rightarrow a = \left((a^d)_{t_2\ldots t_d}\right)_{t_s \in \{1,2\}} \in (\mathbb{Z}^*_{N^2})^{2^{d-1}}$;

$\mathcal{R}$: – $\texttt{INPUT}\leftarrow \left(1^n, i = (i_1, \ldots, i_d), a = \left((a^d)_{t_2\ldots t_d}\right)_{t_s \in \{1,2\}}\right)$,

- Level_d. For every $(t_2, \ldots, t_{d-1}) \in \{1,2\}^{d-2}$, for $t_d = 1, 2$ do:
  - Let $f^d_{t_2 \ldots t_{d-1} t_d} = \mathfrak{D}(a^d_{t_2 \ldots t_{d-1} t_d}) \in \mathbb{Z}_N$,
    (recall $g = 1$ and $\mathbb{Z}_N$ is an additive group, therefore $f^d_{t_2 \ldots t_{d-1} t_d} = \log_g (e^d_{t-2 \ldots t_{d-1} t_d})$ is exactly $e^d_{t-2 \ldots t_{d-1} t_d}$),
  - Let $\overline{a}^{d-1}_{t_2 \ldots t_{d-1}} = \theta^{-1}(f^d_{t_2 \ldots t_{d-1} 1}, f^d_{t_2 \ldots t_{d-1} 2}) = f^d_{t_2 \ldots t_{d-1} 1} N + f^d_{t_2 \ldots t_{d-1} 2} \in \mathbb{Z}^*_{N^2}$,
- Level_r $(d > r \geq 2)$. For every $(t_2, \ldots, t_{r-1}) \in \{1,2\}^{r-2}$, for $t_r = 1, 2$ do:
  - Let $f^r_{t_2 \ldots t_{r-1} t_r} = \mathfrak{D}(\overline{a}^r_{t_2 \ldots t_{r-1} t_r}) \in \mathbb{Z}_N$,
  - Let $\overline{a}^{r-1}_{t_2 \ldots t_{r-1}} = \theta^{-1}(f^r_{t_2 \ldots t_{r-1} 1}, f^r_{t_2 \ldots t_{r-1} 2}) = f^r_{t_2 \ldots t_{r-1} 1} N + f^r_{t_2 \ldots t_{r-1} 2} \in \mathbb{Z}^*_{N^2}$
- Level_1. Let $f^1 = \mathfrak{D}(\overline{a}^1) \in \mathbb{Z}_N$,
- $\texttt{OUTPUT} \rightarrow (f^1)$.

Chapter 6

Oblivious Transfer

6.1 Oblivious Transfer

The *Oblivious Transfer* is a family of cryptographic schemes which was presented by Brassard, Crépeau and Sántha in [4]. These schemes all take place between two parties: Alice (or $\mathcal{A}$) and Bob (or $\mathcal{B}$). $\mathcal{A}$ holds some secrets and she is willing to disclose exactly one of them and nothing more to $\mathcal{B}$, at his choice; conversely $\mathcal{B}$ does not want $\mathcal{A}$ to learn which secret he chose to learn. An Oblivious Transfer protocol has to satisfy these condition, regardless of its communication complexity.

According to the number of secrets owned by $\mathcal{A}$ and their length, the Oblivious Transfer schemes are divided into three main classes:

$\binom{2}{1}$–$\mathbf{OT}_2$ The simplest situation is *One-out-of-two Bit Oblivious Transfer*, denoted by $\binom{2}{1}$–OT_2: $\mathcal{A}$ owns two single-bit secrets, generally called b_0 and b_1. The protocol enables $\mathcal{A}$ to transfer one of b_0 or b_1 to $\mathcal{B}$ who chooses secretly which bit b_c he gets. This is done in *all-or-nothing* fashion: It means that $\mathcal{B}$ cannot get (even partial) information about b_0 and b_1 at the same time, however malicious or (computationally) powerful he is, and that $\mathcal{A}$ finds out nothing about c.

$\binom{2}{1}$–$\mathbf{OT}_2^k$ A generalization of the previous situation is *One-out-of-two String Oblivious Transfer*, denoted by $\binom{2}{1}$–OT_2^k: This time $\mathcal{A}$ owns two secret k-bit strings, generally denoted by w_0 and w_1 and $\mathcal{B}$ wants to learn w_c for a secret $c \in \mathbb{Z}_2$ of his choice. $\mathcal{A}$ is willing to collaborate provided that $\mathcal{B}$ does not learn any information about w_{c+1} (we mean $c+1 \mod 2$), but $\mathcal{B}$ will not participate if $\mathcal{A}$ can obtain information about c.

$\binom{t}{1}$–$\mathbf{OT}_2^k$ The final extension is *One-out-of-t String Oblivious Transfer*, denoted by $\binom{t}{1}$–OT_2^k: Here $\mathcal{A}$ has t secrets k-bit strings $w_0, w_1, \ldots, w_{t-1}$ and $\mathcal{B}$ wants to learn w_c for a secret integer $0 \leq c < t$ of his choice. As usual it must be impossible for $\mathcal{B}$ to obtain information on more than one w_i and for $\mathcal{A}$ to obtain information about which secret $\mathcal{B}$ learns.

One of the most important result about Oblivious Transfer is that the more general $\binom{t}{1}$–OT_2^k indeed can be reduced to the simpler $\binom{2}{1}$–OT_2 [3].

It is clear that, even if with some differences, the Oblivious Transfer Problem is similar to our Private Information Retrieval Problem. In the next section we present a new class of PIR schemes that links the two problems.

6.2 Symmetrically PIR schemes

As we have seen PIR schemes allows a user to retrieve information from a database, replicated in k servers (with $k \geq 2$ in the information-theoretic setting and $k \geq 1$ in the computational setting), while maintaining his interest private; the main cost measure for such a protocol is its communication complexity. We stress that the main goal of PIR protocols is to guarantee the user's privacy regardless of the servers' privacy: Indeed PIR schemes can allow the user to obtain additional information, as we have seen in the previous chapters.

Actually the servers' privacy is a natural and crucial requirement in many settings. For example, consider a commercial server who sells information to users, charging by the amount of data that a user retrieved: In this scenario both user's privacy and server's privacy are essential. The question of preventing the user from learning more than what he asks was first considered by Gertner, Ishai, Kushilevitz and Malkin [11] who introduced the stronger model of *Symmetrically Private Information Retrieval* (or SPIR) where the privacy of the servers, as well as of the user, is guaranteed.

A SPIR protocol can be realized both in information-theoretic and computational setting, but it involves a modification to the standard model. This is necessary because information-theoretic SPIR, regardless of their complexity, cannot be achieved in the original PIR setting, in which the servers do not interact with each other at all [11]. Here we continue to disallow direct interaction between the servers, but we allow them to share a common random string,

unknown to the user. We denote this random string by ρ and we add it to servers' secret input.

Definition 6.2.1 (SPIR). *A (k-server)* Symmetrical PIR *(or* SPIR*) scheme is a triple of algorithm $(\mathcal{Q}, \mathcal{A}, \mathcal{R})$ as in definition 2.0.1 or 2.0.2 satisfying in addiction the following third condition:*

3. *(Server's privacy) Let $i \in \mathcal{I}_n$ be an index. For every n-bit strings x, y such that $x_i = y_i$, we must have:*

$$\Pr\left[\left(\mathcal{A}\big(x, \rho, \mathcal{Q}(1^n, i', r)^1\big), \ldots, \mathcal{A}\big(x, \rho, \mathcal{Q}(1^n, i', r)^k\big)\right) = (a_1, \ldots, a_k)\right] =$$
$$= \Pr\left[\left(\mathcal{A}\big(y, \rho, \mathcal{Q}(1^n, i', r)^1\big), \ldots, \mathcal{A}\big(y, \rho, \mathcal{Q}(1^n, i', r)^k\big)\right) = (a_1, \ldots, a_k)\right]$$

for every $i' \in \mathcal{I}_n$ and $i' \neq i$, for every possible answer $(a_1, \ldots, a_k)$ (with a_j from server $\mathcal{S}_j$) and for every random strings r and ρ.

This definition means that $\mathcal{U}$ cannot learn any information about the database other than a single *physical* bit. It is quite complicated, involving two indices i, i', and one may be tempted to simplify it requiring that, for any index i, the answers $\mathcal{U}$ receives are independent of the database x given x_i. However, this (stronger) condition cannot be satisfied: it is impossible that, when $\mathcal{U}$ asks for the i^{th} bit, the answers he receives depend only on x_i!

Remark that, according to the above definition, a SPIR protocol enable the servers to achieve *information-theoretic privacy*. As for the user's privacy, we can relax the constrain requiring the two probability to be only computationally indistinguishable: We obtain SPIR schemes that guarantee *computational privacy* to the servers. In this setting, we can also consider a slight variation of the model, by replacing the shared random string with a pseudo-random one. So the servers share a short random seed and they generate from it a longer pseudo-random string (the same for all the servers) [12]. This allows the servers to save storage space.

In PIR setting we want to protect user's privacy against the servers. Since (in our assumption) these schemes are 1-round, the servers cannot cheat, but they can only try to infer information from the queries. In SPIR setting the situation is different: $\mathcal{U}$ receives and sends messages and so he can both try to infer extra information from the answer he receives and try to cheat sending illegal queries. Therefore there are two types of SPIR:

Honest-but-curious user Those that protect servers against a honest-but-curious user, that is a user that follows the protocol but he tries to use the information gathered to find out more information.

Dishonest user Those that protect servers against a possibly dishonest user, that is a user that may chooses to not follow the protocol in order to obtain some information.

A protocol which satisfies the second (stronger) requirement is actually an Oblivious Transfer scheme. In particular when we require information-theoretic privacy for the user (and in general when there are more than one server), the SPIR protocol is a distributed version of $\binom{n}{1}$–OT_2^1 (recall that Oblivious Transfer involves two single parties); while the SPIR counterpart of a 1-server cPIR is an implementation of $\binom{n}{1}$–OT_2^1.

Since the Oblivious Transfer is a well-known family of cryptographic schemes, it is in the habit of calling SPIR protocols against dishonest user just Oblivious Transfer.

A classical problem is to transform PIR schemes into OT schemes: for instance in [11] Gertner et al. present a way to construction OT protocols from itPIR schemes and [10] Di Crescenzo, Malkin and Ostrovsky show how we can transform cPIR into OT. In the next section we analyze protocol $\mathcal{P}$ from Oblivious Transfer point of view.

6.3 Protocol $\mathcal{P}$ in Oblivious Transfer setting

The cPIR protocol $\mathcal{P}$ is not a SPIR scheme neither against dishonest users nor against curious-but-honest users. In fact, as we have seen, one invocation of the scheme enables a curious user to obtain $n^{1/(L+1)}$ bits of the database. On the other hand, if $\mathcal{U}$ is dishonest, he can cheat sending more than one pseudo-quadratic residue modulo N in the query of some level of recursion and in this way he can obtain the XOR of two bits.

6.3.1 $\mathcal{SP}$: Against Honest-But-Curious User

Scheme $\mathcal{P}$ can be transformed into a SPIR $\mathcal{SP}$ protocol secure against honest-but-curious users just adding a final level in the recursion, say Level_0. In fact at the end each of Level_1, $\mathcal{S}$ sends his answer $(a_1^1, \ldots, a_{R_1}^1)$, with $R_1 = n^{1/(L+1)}$.

However $\mathcal{U}$ is interested only in $a^1_{{r_1}^*}$, thus we can consider the answer as k new databases x^0 as usual (the j^{th} database is the ordered concatenation of the j^{th} bit of each a^1_r) and iterate the protocol once more, using $i_0 = {r_1}^*$.

Following exactly the protocol for Level_0 as for a generic level, we view each new database as a bit-matrix of dimension $R_0 \times C_0$, with $C_0 = n^{1/(L+1)}$ and $R_0 = 1$ (that is as a string); therefore i_0 is associated with the pair $(1, r_1^*)$, that is $c_0^* = r_1^*$. The user has to send one more query $(q^0_1, \ldots, q^0_{C_0})$ with only $q^0_{c_0^*} \in \mathscr{PQR}_N$ as usual, but this time $\mathcal{S}$ computes only a^0_1 and sends it to $\mathcal{U}$. $\mathcal{U}$ then uses all the answers he receives exactly as in the original scheme.

Theorem 6.3.1. *The protocol $\mathcal{SP}$ described above is a SPIR protocol secure against honest-but-curious users such that $\mathcal{CC}_{\mathcal{SP}}(n) = \mathcal{O}\big(e^{c\sqrt{\ln n}}\big)$, for some $c > 0$.*

Proof. We have to prove that $\mathcal{SP}$ satisfies the definition and it has the required communication complexity.

1. (Correctness) The correctness is trivial: Level_0 is correct because $a^0_1 = \prod_{c=1}^{C_0} (q^0_c)^{x^0_{1,c}}$ and so, computing the quadratic residuosity of a^0_1, $\mathcal{U}$ obtain x^0_{1,c^*_0}. Since $c^*_0 = r^*_1$, $\mathcal{U}$ can reconstruct $a^1_{r^*_1}$ As we have seen, it recursively implies that $\mathcal{U}$ can reconstruct the desired bit of the original database.

2. (User's privacy) As for $\mathcal{P}$.

3. (Communication complexity) $\mathcal{U}$ sends just one more query for Level_0, that is $kn^{1/(L+1)}$ bits. Thus $\mathcal{U}$ sends $kLn^{1/(L+1)} + kn^{1/(L+1)} + k = (L+1)kn^{1/(L+1)} + k$ bits.

   For each execution of Level_0, $\mathcal{S}$ sends one elements of $\mathbb{Z}^*_N$, that is k bits. We have to calculate how many executions of Level_0 are needed. Notice that each execution of Level_1 invokes k executions of Level_0; since we have proved that there are k^{L-1} executions of Level_1, it follows that Level_0 is executed k^L times. Hence $\mathcal{S}$ sends $k^L k = k^{L+1}$ bits.

   Thus the total amount of communication is:

$$\mathcal{CC}_{\mathcal{SP}}(n) = (L+1)kn^{1/(L+1)} + k^{L+1} + k = \mathcal{O}\big(e^{c\sqrt{\ln n}}\big)$$

   with the usual choice of $L \approx \sqrt{\frac{\ln n}{\ln k}}$.

4. (Server's privacy against honest-but-curious user) We have seen that the knowledge of the whole answer $(a_1^1, \ldots, a_{n^{1/(L+1)}}^1)$ allows $\mathcal{U}$ to obtain $n^{1/(L+1)}$ bits of the database. The new method allows the user to retrieve only one element form that answer: in fact $\mathcal{U}$ receives only a_1^0 and, using it, he can just retrieve (all the bits of the binary expansion of) $a_{r_1^*}^1$ (recall $c_0^* = r_1^*$). Therefore $\mathcal{U}$ can learn only the desired bit of the original database and nothing more.

□

Comparing $\mathcal{CC}_{\mathcal{SP}}(n)$ with the communication complexity of $\mathcal{P}$, we see that $\mathcal{SP}$ is more efficient. In fact:

$$\begin{aligned}
\mathcal{CC}_{\mathcal{SP}}(n) < \mathcal{CC}_{\mathcal{P}}(n) &\iff (L+1)kn^{1/(L+1)} + k^{L+1} + k < \\
&\quad < Lkn^{1/(L+1)} + k^L n^{1/(L+1)} + k \iff \\
&\iff n^{1/(L+1)} + k^L < k^{L-1} n^{1/(L+1)} \iff \\
&\iff n^{1/(L+1)}(k^{L-1} - 1) > k^L \iff \\
&\iff n^{1/(L+1)} > \frac{k^L}{(k^{L-1} - 1)}.
\end{aligned}$$

This is true because we have seen that, choosing $L \approx \sqrt{\frac{\ln n}{\ln k}}$, we have that $n^{1/(L+1)} \approx k^L$. Thus this variation of $\mathcal{P}$ is a SPIR protocol and it also has less communication complexity, thus it is better in any case (unless it is explicitly required that the user retrieves several bits). Actually the general construction of Ostrovsky and Skeith III applied to the quadratic residuosity encryption scheme is exactly $\mathcal{SP}$.

To well understand how $\mathcal{SP}$ works consider the following example.

Example 6.3.2. *Consider example 3.5.2. At Level$_0$ we view each $\mathcal{S}$'s answer coming from Level$_1$ as k 1×3 bit-matrices and $\mathcal{U}$ wants their 2^{th} element (because the knowledge of the 1^{st} element from each $\mathcal{S}$'s answer allows the user to retrieve x_2, the knowledge of the 2^{nd} x_{11} and the knowledge of the 3^{rd} x_{20}). Thus $\mathcal{U}$ must add to his query $(q_1^0, q_2^0, q_3^0) \in \mathscr{QR}_N \times \mathscr{PQR}_N \times \mathscr{QR}_N$. For each new 1×3 matrix $\mathcal{S}$ sends his answer which consists of only one element of $\mathbb{Z}_N^1$. Computing its*

quadratic residuosity, $\mathcal{U}$ can retrieve (only) x_{11}.

$$\left(\begin{array}{c|c|c} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \\ x_{10} & \underline{\boldsymbol{x_{11}}} & x_{12} \\ x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} \\ x_{19} & x_{20} & x_{21} \\ x_{22} & x_{23} & x_{24} \\ x_{25} & x_{26} & x_{27} \end{array}\right) \qquad \left(\begin{array}{|c|cc} \hline x_2 & x_5 & x_8 \\ \underline{\boldsymbol{x_{11}}} & x_{14} & x_{17} \\ x_{20} & x_{23} & x_{26} \\ \hline \end{array}\right) \qquad \left(\begin{array}{ccc} x_2 & \boxed{\boldsymbol{x_{11}}} & x_{20} \end{array} \right)$$

6.3.2 $\mathcal{SP}'$: Against Dishonest User

If the user is dishonest, the only way he can cheat is sending more than one pseudo-quadratic residue modulo N in the query of some level l of recursion. Suppose $\mathcal{U}$ sends $(q_1^l, \ldots, q_C^l)$ (where $C = C_l = n^{1/(L+1)}$), with $q_{c^*}^l, q_{c'^*}^l \in \mathscr{P}\mathscr{Q}\mathscr{R}_N$ and $q_c^l \in \mathscr{Q}\mathscr{R}_N$ for $c \neq c^*, c'^*$. Then:

$$a_{r^*} = {q_{c^*}^l}^{x_{r^*,c^*}^l} {q_{c'^*}^l}^{x_{r^*,c'^*}^l} y^2$$

for some $y \in \mathbb{Z}_N^*$. Therefore $a_{r^*} \in \mathscr{Q}\mathscr{R}_N$ if and only if $x_{r^*,c^*}^l \oplus x_{r^*,c'^*}^l = 0$. Thus the user can know the XOR of two bits of x^l.

If $l \neq L$, it implies that $\mathcal{U}$ can know the bitwise XOR of two elements of $\mathbb{Z}_N^*$ (that is $a_i^{l+1} \oplus a_{i'}^{l+1}$). It does not allows the user to learn any information about a_i^{l+1} or $a_{i'}^{l+1}$ only under a supplementary assumption, called *XOR assumption*:

Conjecture 6.3.3 (XOR assumption). *Let $p_1 \neq p_2$ be prime numbers such that $|p_1| = |p_2|$. Let $N = p_1 p_2$. For every $z \in \mathbb{Z}_N$:*

$$\begin{cases} \Pr\left[(x \in \mathscr{Q}\mathscr{R}_N) \wedge (y \in \mathscr{Q}\mathscr{R}_N) | x \oplus y = z\right] = 1/4 \\ \Pr\left[(x \in \mathscr{Q}\mathscr{R}_N) \wedge (y \in \mathscr{P}\mathscr{Q}\mathscr{R}_N) | x \oplus y = z\right] = 1/4 \\ \Pr\left[(x \in \mathscr{P}\mathscr{Q}\mathscr{R}_N) \wedge (y \in \mathscr{Q}\mathscr{R}_N) | x \oplus y = z\right] = 1/4 \\ \Pr\left[(x \in \mathscr{P}\mathscr{Q}\mathscr{R}_N) \wedge (y \in \mathscr{P}\mathscr{Q}\mathscr{R}_N) | x \oplus y = z\right] = 1/4 \end{cases}$$

for any random $x, y \in \mathbb{Z}_N^1$.

Thus if we assume the XOR assumption too, we have that if $\mathcal{U}$ tries to cheat in any level different from Level_L, he wastes is chances to learn any bit.

On the contrary, if the user sends two pseudo-quadratic residue modulo N in the query of the Level$_L$ (*i.e.* $l = L$), he obtains $x_i^L \oplus x_{i'}^L$ that is the XOR of two bits of the database. It violates the privacy constrain since we ask that the user learns nothing more than a *physical* bit of the database. If we assume the XOR conjecture, we can eliminate this problem just adding a step before the protocol starts. We call the new scheme $\mathcal{SP}'$.

Initial phase

- $\mathcal{U}$ sends to $\mathcal{S}$ a random $y \in \mathscr{PQR}_N$;
- $\mathcal{S}$ ciphers the database in the following way: for every $j \in \mathcal{I}_n$, he chooses $z_j \in \mathbb{Z}_N^1$ such that

$$\begin{cases} \text{if } x_j = 0 \Rightarrow z_j = r_j^2, & \text{for some } r_j \in_R \mathbb{Z}_N^*, \\ \text{if } x_j = 1 \Rightarrow z_j = y r_j^2, & \text{for some } r_j \in_R \mathbb{Z}_N^*; \end{cases}$$

- $\mathcal{S}$ constructs k new databases such that the t^{th} database is the ordered concatenation of the t^{th} bit of every z_j;
- $\mathcal{U}$ and $\mathcal{S}$ run in parallel k executions of $\mathcal{SP}$ (one for each new database).

Remark that $\mathcal{U}$ does not need to send k different queries to run the k executions of $\mathcal{SP}$, but only one query is needed. In fact, to retrieve x_i for some $i \in \mathcal{I}_n$, $\mathcal{U}$ must learn z_i and so he has to ask the i^{th} element in each execution of $\mathcal{SP}$. Actually $\mathcal{U}$'s query in $\mathcal{SP}'$ is exactly the same of the one needed to perform $\mathcal{SP}$.

Theorem 6.3.4. *The protocol $\mathcal{SP}'$ described above is an OT scheme such that $\mathcal{CC}_{\mathcal{SP}'}(n) = \mathcal{O}\big(e^{c\sqrt{\ln n}}\big)$, for some $c > 0$.*

Proof. Correctness, user's privacy and security against honest-but-curious users follow immediately from the ones of $\mathcal{SP}$. It remains to prove that $\mathcal{SP}'$ is secure against dishonest users and to study its communication complexity.

1. (Server's privacy against dishonest user) If $\mathcal{U}$ is dishonest and he sends two pseudo-quadratic residue in the query of Level$_L$, he obtain the XOR of two bit of the new databases that is the bitwise XOR of two z_j. Under XOR assumption, in this way $\mathcal{U}$ wastes his chances to learn any bit. Therefore under QRA and XOR assumption $\mathcal{SP}'$ is an OT protocol.

2. (Communication complexity) As we have already stressed, $\mathcal{U}$'s query is exactly as in $\mathcal{SP}$, thus $\mathcal{U}$ sends $(L+1)kn^{1/(L+1)}+k$ bits.

   $\mathcal{S}$ sends k^{L+1} bits performing $\mathcal{SP}$, since k executions of $\mathcal{SP}$ are needed, $\mathcal{S}$ sends $kk^{L+1}=k^{L+2}$ bits.

   Finally, we have to compute the communication complexity of the initial phase: here the only exchange of information is the sending of $y \in \mathscr{PQR}_N$ from $\mathcal{U}$ to $\mathcal{S}$. Thus the initial phase has a complexity cost of k bits.

   Therefore the total amount of communication is:

$$\mathcal{CC}_{\mathcal{SP}'}(n) = (L+1)kn^{1/(L+1)} + 2k + k^{L+2} = \mathcal{O}\big(e^{c\sqrt{\ln n}}\big)$$

   with the usual choice of $L \approx \sqrt{\frac{\ln n}{\ln k}}$.

□

Comparing the communication complexities of $\mathcal{SP}$ and $\mathcal{SP}'$ we have:

$$\begin{aligned}\frac{\mathcal{CC}_{SP'}(n)}{\mathcal{CC}_{SP}(n)} &= \frac{(L+1)kn^{1/(L+1)} + k^{L+2} + 2k}{(L+1)kn^{1/(L+1)} + k^{L+1} + k} = \\ &= 1 + \frac{1 + k^L(k-1)}{(L+1)n^{1/(L+1)} + k^L + 1} > 1.\end{aligned}$$

Therefore $\mathcal{SP}$ is always more efficient, even if the two protocols have the same asymptotic behavior by increasing n.

Chapter 7

Conclusions

The main part of this work was devoted in finding useful variations of the cPIR scheme by Kushilevitz and Ostrovsky [17]. The result are summarized in the following table, where n is the number of bits the database needs to be stored, $k = \lceil \log N \rceil$ the security parameter, $c = \sqrt{\ln k}$ and L the number of levels of recursion (recall that all these protocols are recursive):

Prot. (Ass.)	Nr. bits $\mathcal{U}$ gets	Block	SPIR (h-b-c)	OT	Communication complexity		
					User's side	Server's side	Tot.
$\mathcal{P}$ (QRA)	1	No	No	No	$Lkn^{1/(L+1)} + k$	$k^L n^{1/(L+1)}$	$\mathcal{O}\left(e^{c\sqrt{\ln n}}\right)$
$\mathcal{P}'$ (QRA)	$n^{1/(L+1)}$	Yes	Yes	No	$Lkn^{1/(L+1)} + k$	$k^L n^{1/(L+1)}$	$\mathcal{O}\left(e^{c\sqrt{\ln n}}\right)$
$\mathcal{N}_1$ (QRA)	2	No	No	No	$Lkn^{1/(L+1)} + k$	$k^L n^{1/(L+1)}$	$\mathcal{O}\left(e^{c\sqrt{\ln n}}\right)$
$\mathcal{N}_2$ (QRA)	2	Yes	No	No	$Lk(\frac{n}{2})^{1/(L+1)} + k$	$k^L(\frac{n}{2})^{1/(L+1)}$	$\mathcal{O}\left(e^{c\sqrt{\ln(\frac{n}{2})}}\right)$
$\mathcal{M}_1$ (QRA)	m	No	No	No	$Lkn^{1/(L+1)} + k$	$k^L n^{1/(L+1)}$	$\mathcal{O}\left(e^{c\sqrt{\ln n}}\right)$
$\mathcal{M}_2$ (QRA)	m	Yes	No	No	$Lk(\frac{n}{m})^{1/(L+1)} + k$	$k^L(\frac{n}{m})^{1/(L+1)}$	$\mathcal{O}\left(e^{c\sqrt{\ln(\frac{n}{m})}}\right)$
$\mathcal{SP}$ (QRA)	1	No	Yes	No	$(L+1)kn^{1/(L+1)} + k$	k^{L+1}	$\mathcal{O}\left(e^{c\sqrt{\ln n}}\right)$
$\mathcal{SP}'$ (QRA, XOR)	1	No	Yes	Yes	$(L+1)kn^{1/(L+1)} + 2k$	k^{L+2}	$\mathcal{O}\left(e^{c\sqrt{\ln n}}\right)$

Table 7.1: Results

In the 2$^{\text{th}}$ column we list the number of bits that a honest-not-curious user

obtains with one execution of the protocol. In the 4^{th} column, with '(h-b-c)' we mean SPIR schemes secure against honest-but-curious users.

In the standard situation, that is when the user wants one bit regardless to anything else, the best protocol is $\mathcal{SP}$ because it has less communication complexity then $\mathcal{P}$, as we can see comparing the server's side communication. $\mathcal{M}_2$, used to retrieve the block containing the desired bit and so it, is also better than $\mathcal{P}$; but, since m cannot be too large (otherwise $2^m\mathscr{R}_N$ has not enough elements), $\mathcal{SP}$ is still better.

Of course $\mathcal{M}_1$ has to be preferred when we want to allow the user to possibly retrieve more than one bit in any position. Remark that the most interesting characteristic of $\mathcal{M}_1$ is that it enables the user to learn any number at his choice (in $\mathcal{I}_m$) of bits and that the server cannot learn not only *which* bits he obtains, but also *how many*. As we have already remarked, the problem of this protocol is that we must have $2^{k-m-1} - 2^{k/2-m} + 2^{-m-1} \geq n^{1/(L+1)}$, therefore the user has to pay attention when he chooses m and eventually it has to pick a bigger security parameter k.

On the other hand, when we want to enable the user to retrieve a block of bits we can use $\mathcal{P}'$ or $\mathcal{M}_2$. Actually the best solution is combine the two protocols, in this way we obtain a scheme that allows the user to retrieve with only one execution $n^{1/(L+1)}$ consecutive blocks of m bits each, that is $mn^{1/(L+1)}$ consecutive bits. In this way we have more freedom in fixing the length of the blocks. Remark that, as the variation $\mathcal{P}'$ has the same communication complexity as the original $\mathcal{P}$, combining $\mathcal{M}_2$ with $\mathcal{P}'$ does not change the communication complexity of $\mathcal{M}_2$, which is less than the one of $\mathcal{P}'$.

When we deal with server's privacy we can only use $\mathcal{SP}$ or $\mathcal{SP}'$: the former when we want to protect the server only against honest-but-curious users, the latter when we want also to protect him against dishonest users. Notice that $\mathcal{P}'$, used to retrieve a block of $n^{1/(L+1)}$ bits, can be considered as a SPIR scheme secure against honest-but-curious users because it allows an (honest) user to retrieve a block of bit and nothing more. It follows that, combining $\mathcal{P}'$ and $\mathcal{M}_2$, we obtain a scheme with the same property.

Finally we stress that $\mathcal{SP}'$ requires a stronger assumption. An open problem is to find a better solution, that is a way to transform protocol $\mathcal{P}$ into an OT protocol that requires only QRA.

Appendix A

Information-Theoretic PIR Schemes

Our work only deals with computational PIR schemes. For completeness, here we shortly describe some information-theoretic PIR schemes.

As we have seen in Chapter 2, the itPIR problem is a non-trivial issue only if there are more than one servers, so in this appendix we assume $k \geq 2$. The main results obtained in this field are summarized in the following table:

Prot.	Author	Ref.	Communication complexity for k servers
$\mathcal{IT}_1$	Chor et al. (95)	[7]	$k \log k n^{1/\log k} = \mathcal{O}\big(n^{1/\log k}\big)$
$\mathcal{IT}_2$	Chor et al. (95)	[7]	$k \log k n^{1/\log k + \log\log k} = \mathcal{O}\big(n^{1/\log k + \log\log k}\big)$
$\mathcal{IT}_3$	Chor et al. (98)	[8]	$k^2 \log k n^{1/k} = \mathcal{O}\big(n^{1/k}\big)$
$\mathcal{IT}_4$	Ambainis (97)	[1]	$2^{k^2} n^{1/(2k-1)} = \mathcal{O}\big(n^{1/(2k-1)}\big)$
$\mathcal{IT}_5$	Itoh (99)	[16]	$k! n^{1/(2k-1)} = \mathcal{O}\big(n^{1/(2k-1)}\big)$
$\mathcal{IT}_6$	Ishai, Kushilevitz (99)	[15]	$k^3 n^{1/(2k-1)} = \mathcal{O}\big(n^{1/(2k-1)}\big)$
$\mathcal{IT}_7$	Beilmel et al. (02)	[2]	$n^{\mathcal{O}(n^{\log\log k/k \log k})}$

Table A.1: Main results on itPIR

In [7, 8] Chor et al. introduced the PIR problem and presented the protocols $\mathcal{IT}_1$, $\mathcal{IT}_2$ and $\mathcal{IT}_3$; in [1] Ambainis presented $\mathcal{IT}_4$ which has communication complexity significantly smaller. Subsequently, there were several attempts to improve Ambainis' upper bound; while these attempts resulted in finding new and very different PIR schemes, they all failed in breaking the $\mathcal{O}(n^{1/(2k-1)})$ bound: The constants, which depend only on k, were significantly improved, but the number of servers k is usually considered to be *small* and in particular independent of the length n of the database. Therefore $\mathcal{IT}_5$ and $\mathcal{IT}_6$ have the same asymptotic behavior as $\mathcal{IT}_4$. The $\mathcal{O}(n^{1/(2k-1)})$ barrier was finally broken by Beimel et al. [2] who presented an itPIR with communication complexity $n^{\mathcal{O}(n^{\log\log k/k \log k})}$.

In this appendix we examine the most interesting itPIR schemes among those of Table A.1.

A.1 $\mathcal{IT}_1$: a k-server itPIR scheme with communication complexity $\mathcal{O}\big(n^{1/\log k}\big)$

Let $k = 2^d$, for any integer $d \geq 1$. We can assume, without loss of generality, that $n = l^d$, for some $l \in \mathbb{N}$.

Key idea: View the database $x \in \mathbb{Z}_2^n$ as a d-dimensional cube in $\big(\mathbb{Z}_2^l\big)^d$.

Thus we associate the string $x \in \mathbb{Z}_2^n$ with an array $x = (x_{j_1,\ldots,j_d})_{j_t \in \mathcal{I}_l}$ and each position $j \in \mathcal{I}_n$ with a d-uple $(j_1, \ldots, j_d) \in \mathcal{I}_l^d$ in a natural manner. In particular, the index i of the desired bit is associated with the d-uple $(i_1, \ldots, i_d)$.

It is also convenient to associate each server with a bit-string of length d: The s^{th} server $\mathcal{S}_s$ is associated with the binary expansion of $(s-1) \in \mathcal{J}_{k-1}$ which is a d-bit string (recall $k = 2^d$).

Protocol $\mathcal{IT}_1$

$\mathcal{Q}$: – $\texttt{INPUT}\leftarrow \big(1^n, i = (i_1, \ldots, i_d), r = (R_1^0, \ldots, R_d^0)\big)$
with $R_t^0 \subseteq_R \mathcal{I}_l$, for every $1 \leq t \leq d$,
– Let $R_t^1 = R_t^0 \oplus i_t$,
– $\texttt{OUTPUT}\rightarrow \big(q_1 = (R_1^0, R_2^0, \ldots, R_d^0), \ldots, q_k = (R_1^1, R_2^1, \ldots, R_d^1)\big)$
with $(R_1^{s_1}, \ldots, R_d^{s_d})$ the query intended for the server $\mathcal{S}_{s+1}$ when the concatenation $s_1 \ldots s_d$ is the binary expansion of s with $s \in \mathcal{J}_{k-1}$;

$\mathcal{A}$: – $\texttt{INPUT}\leftarrow (x, q = (R_1, \ldots, R_d))$
with $\begin{cases} x = (x_{j_1,\ldots,j_d})_{j_t \in \mathcal{I}_l}, \\ R_t \subseteq \mathcal{I}_l \quad \text{for every } 1 \leq t \leq d, \end{cases}$
– Let $a = \bigoplus_{\substack{j_1 \in R_1 \\ \ddots \\ j_d \in R_d}} x_{j_1,\ldots,j_d} \in \mathbb{Z}_2$,
– $\texttt{OUTPUT}\rightarrow (a)$;

$\mathcal{R}$: – $\texttt{INPUT}\leftarrow (1^n, i = (i_1, \ldots, i_d), r = (R_1^0, \ldots, R_d^0), a_1, \ldots, a_k)$
with a_s the answer sent by $\mathcal{S}_s$,

– Let $b = \bigoplus_{s=1}^{k} a_s \ \in \mathbb{Z}_2$,

– OUTPUT$\rightarrow (b)$;

Theorem A.1.1. *The protocol $\mathcal{IT}_1$ defined above is an itPIR protocol such that $\mathcal{CC}_{\mathcal{IT}_1}(n) = \mathcal{O}(n^{1/\log k})$, with $k \geq 2$ the number of servers.*

Proof. We have to prove that $\mathcal{IT}_1$ verifies the definition 2.0.1 and that it has the required communication complexity.

1. (Correctness) The contribution of each bit $x_{j_1,\ldots,j_d}$ of the database to the xor computing b depends on the number of subcubes $R_1^{s_1} \times \ldots \times R_d^{s_d}$ that contain the position $(j_1, \ldots, j_d)$. The position $(i_1, \ldots, i_d)$ appears in a single subcube because i_t appears in exactly one of the sets R_t^0 and $R_t^1 = R_t^0 \oplus i_t$, for every $t \in \mathcal{I}_d$. On the contrary, each other position $(j_1 \ldots, j_d)$ appears in an even number (possibly zero) of subcubes: If $j_t \neq i_t$, then $j_t \in R_t^0$ ifa and only if $j_t \in R_t^1$. It implies that, for every $(s_1, \ldots, s_d) \in \mathbb{Z}_2^d$:

$$\begin{gathered}(j_1, \ldots, j_d) \in R_1^{s_1} \times \ldots \times R_{t-1}^{s_{t-1}} \times R_t^0 \times R_{t+1}^{s_{t+1}} \times \ldots \times R_d^{s_d} \\ \text{if and only if} \\ (j_1, \ldots, j_d) \in R_1^{s_1} \times \ldots \times R_{t-1}^{s_{t-1}} \times R_t^1 \times R_{t+1}^{s_{t+1}} \times \ldots \times R_d^{s_d}.\end{gathered}$$

   Hence, in the exclusive-or computing b, the contribute of all these positions vanishes while that of position $(i_1, \ldots, i_d)$ remains. Therefore $b = x_{i_1,\ldots,i_d}$.

2. (Privacy) R_t^0 is a random subset of $\mathcal{I}_l$ for every $t \in \mathcal{I}_d$, hence so is each R_t^1. Therefore each server receives d randomly (and independently) chosen subset of $\mathcal{I}_l$, that is the queries each server receives are identically distributed by varying the index i.

3. (Communication complexity) The user sends d subset of $\mathcal{I}_l$ to each server who replies with 1 bit. Thus the total amount of communication is $k(dl+1)$ bits. Since $d = \log k$ and $l = n^{1/d}$, we have:

$$\mathcal{CC}_{\mathcal{IT}_1}(n) = k \log(k) n^{1/\log k} + k = \mathcal{O}(n^{1/\log k}).$$

□

A.2 $\mathcal{IT}_4$: a k-server itPIR scheme with communication complexity $\mathcal{O}\big(n^{1/(2k-1)}\big)$

The itPIR scheme of this section is by Ambainis [1]. His new idea is to combine a 2-server itPIR protocol with a $(k-1)$-server one using recursion, to obtain a k-server itPIR scheme with smaller communication complexity.

To apply Ambainis' idea, the 2-server itPIR protocol should satisfy the following constraints:

1. The most of communication goes from servers to user,
2. Only few bits from each answer are necessary,
3. The user knows in advance which bits are necessary (*i.e.* he knows their position in the answer strings).

We consider the the following 2-server itPIR scheme.

Protocol $\mathcal{B}$ (2-server)

The main idea is that $\mathcal{U}$, $\mathcal{S}_1$, $\mathcal{S}_2$ simulate protocol $\mathcal{IT}_1$ for 2^{2k-1} servers (we denote these *virtual* servers by $\mathcal{S}'_1, \ldots, \mathcal{S}'_{2^{2k-1}}$). So $d = 2k-1$ and let l be the integer such that $n^d = l$. We associate the database $x \in \mathbb{Z}_2^n$ with an array $x = (x_{j_1,\ldots,j_{2k-1}})_{j_t \in \mathcal{I}_l}$ and each position $j \in \mathcal{I}_n$ with a $(2k-1)$-uple $(j_1, \ldots, j_{2k-1}) \in \mathcal{I}_l^{2k-1}$ in a natural manner. We also associate each virtual server $\mathcal{S}'_s$ with a $(2k-1)$-bit string as before.

$\mathcal{Q}$:
- $\texttt{INPUT}\leftarrow \big(1^n, i = (i_1, \ldots, i_{2k-1}), r = (R_1^0, \ldots, R_{2k-1}^0)\big)$ with $R_t^0 \subseteq_R \mathcal{I}_l$, for each $1 \le t \le 2k-1$,
- Let $R_t^1 = R_t^0 \oplus i_t$,
- $\texttt{OUTPUT}\rightarrow \big(q_1 = (R_1^0, R_2^0, \ldots, R_{2k-1}^0), q_2 = (R_1^1, R_2^1, \ldots, R_{2k-1}^1)\big)$ with q_1 the query intended for $\mathcal{S}_1$ and q_2 for $\mathcal{S}_2$;

$\mathcal{A}$: Server $\mathcal{S}_1$:

- $\texttt{INPUT}\leftarrow (x, q_1)$ with $\begin{cases} x = (x_{j_1,\ldots,j_{2k-1}})_{j_t \in \mathcal{I}_l}, \\ q_1 = (R_1^0, \ldots, R_{2k-1}^0) \text{ query sent by } \mathcal{U}, \end{cases}$

– Let $a = \bigoplus_{\substack{j_1 \in R_1^0 \\ \dots \\ j_{2k-1} \in R_{2k-1}^0}} x_{j_1,\dots,j_{2k-1}} \in \mathbb{Z}_2$,
– $\texttt{OUTPUT} \rightarrow (a)$;
– For every $t \in \mathcal{I}_{2k-1}$, for $1 \leq u \leq l$ do:
  * Let $R'_t = R_t^0 \oplus u$,
  * Let $R'_{t'} = R_{t'}^0$, for every $1 \leq t' \leq 2k-1$ and $t' \neq t$,
  * Let $a = \bigoplus_{\substack{j_1 \in R'_1 \\ \dots \\ j_{2k-1} \in R'_{2k-1}}} x_{j_1,\dots,j_{2k-1}} \in \mathbb{Z}_2$,
  * $\texttt{OUTPUT} \rightarrow (a)$;

Server $\mathcal{S}_2$:

– $\texttt{INPUT} \leftarrow (x, q_2)$
  with $\begin{cases} x = (x_{j_1,\dots,j_{2k-1}})_{j_t \in \mathcal{I}_l}, \\ q_2 = (R_1^1, \dots, R_{2k-1}^1) \text{ query sent by } \mathcal{U}, \end{cases}$
– Let $a = \bigoplus_{\substack{j_1 \in R_1^1 \\ \dots \\ j_{2k-1} \in R_{2k-1}^1}} x_{j_1,\dots,j_{2k-1}} \in \mathbb{Z}_2$,
– $\texttt{OUTPUT} \rightarrow (a)$,
– For all the possible $(R'_1, \dots, R'_{2k-1})$ such that:
  1. For every $t \in \mathcal{I}_{2k-1}$, $R'_t = R_t^1$ or $R_t^1 \oplus u$, for some $u \in \mathcal{I}_l$,
  2. There exist at least two $t \in \mathcal{I}_{2k-1}$ such that $R'_t = R_t^1$,

  do:
  * Let $a = \bigoplus_{\substack{j_1 \in R'_1 \\ \dots \\ j_{2k-1} \in R'_{2k-1}}} x_{j_1,\dots,j_{2k-1}} \quad \mathbb{Z}_2$,
  * $\texttt{OUTPUT} \rightarrow (a)$;

$\mathcal{R}$: – $\texttt{INPUT} \leftarrow (1^n, i = (i_1, \dots, i_{2k-1}), r = (R_1^0, \dots, R_{2k-1}^0), (a_s)_s)$
  with $(a_s)_s$ all the answers sent by $\mathcal{S}_1$ and $\mathcal{S}_2$,
– Let $b = \bigoplus_{\substack{s \text{ corresponding} \\ \text{to correct guess of} \\ \mathcal{S}_1 \text{ and } \mathcal{S}_2}} a_s \in \mathbb{Z}_2$,
– $\texttt{OUTPUT} \rightarrow (b)$.

In short, $\mathcal{S}_1$ simulates $\mathcal{S}'_1$ (labeled with $(0, \dots, 0) \in \mathbb{Z}_2^{2k-1}$) and all the $2k-1$ servers having label with Hamming distance 1 from $(0, \dots, 0)$. Since $\mathcal{S}_1$ knows

only $(R_1^0, \ldots, R_{2k-1}^0)$, he does not know the exact query each such a server should receive. Therefore, starting from $(R_1^0, \ldots, R_{2k-1}^0)$. $\mathcal{S}_1$ computes all the possible queries of each of these virtual server and simulates him on all these virtual queries.

Remark that a server having label with Hamming distance 1 from $(0, \ldots, 0)$ should receive a query consisting of R_t^1 and $R_{t'}^0$ for every $t' \in \mathcal{I}_{2k-1}$ and $t' \neq t$. As $R_t^1 = R_t^0 \oplus i_t$ with i_t, knowing only R_t^0, there are l possibility for R_t^1. Hence $\mathcal{S}_1$ sends l bits for each server he simulates. So $\mathcal{U}$ receives $1+(2k-1)l = \mathcal{O}(n^{1/(2k-1)})$ bits from $\mathcal{S}_1$ ($l = n^{1/(2k-1)}$). Actually, $\mathcal{U}$ only needs the $2k$ bits corresponding to the correct guess of R_t^1 by R'_t.

At the same time $\mathcal{S}_2$ simulates $\mathcal{S}'_{2k-1}$ (labeled with $(1, \ldots, 1) \in \mathbb{Z}_2^{2k-1}$) and all the other servers having label with Hamming distance less than or equal to $2k-3$ from $(1, \ldots, 1)$. It means that $\mathcal{S}_2$ simulates the rest of the servers that $\mathcal{S}_1$ does not simulate, thus $\mathcal{S}_2$ simulates $\sum_{m=1}^{2k-3} \binom{2k-1}{m} = 2^{2k-1} - 2k$ virtual servers. Each server whose label has Hamming distance m from $(1, \ldots, 1)$ sends l^m bits to $\mathcal{U}$. In fact $\mathcal{S}_2$ knows only $(R_1^1, \ldots, R_{2k-1}^1)$; suppose he simulates such a server. Let $t_1, \ldots, t_m$ the positions of zeros in the label of the virtual server; then $\mathcal{S}_2$ has to consider all the possibility for $R_{t_1}^0, \ldots, R_{t_m}^0$. Since for each such a set there are l possibility, then S_2 computes l^m possible queries for the server he wants to simulate. Therefore he has to run l^m simulations obtaining l^m bits. So $\mathcal{U}$ receives $1 + \sum_{m=1}^{2k-3} \binom{2k-1}{m} l^m = \mathcal{O}(n^{(2k-3)/(2k-1)})$ bits from $\mathcal{S}_2$. Actually, $\mathcal{U}$ only needs the $2^{2k-1} - 2k$ bits corresponding to the correct guess of the queries.

The transmission of each $R_t^{s_t}$ needs $l = n^{1/(2k-1)}$ bits, thus $\mathcal{U}$ sends $(2k-1)l$ bits to each server. Hence the total amount of communication from $\mathcal{U}$ is $2(2k-1)n^{1/(2k-1)} = \mathcal{O}(n^{1/(2k-1)})$ bits.

Therefore $\mathcal{B}$ verifies the required constraints (the most of communication goes from servers to user, $\mathcal{U}$ needs a constant amount of bits from $\mathcal{S}_2$'s answer (that is the $2^{2k-1}-2k$ bits corresponding to the correct guess) and $\mathcal{U}$ knows their positions). Moreover it is easy to see that $\mathcal{B}$ is a 2-server itPIR (the proof of its correctness and privacy is the same as $\mathcal{IT}_1$). Therefore, according Ambainis' idea, we can use $\mathcal{B}$ as subroutine to construct a k-server itPIR scheme with smaller communication complexity.

Protocol $\mathcal{IT}_4$

This protocol uses $\mathcal{B}$ as subroutine; we denote with $\mathcal{U}^{\mathcal{B}}$ (resp. $\mathcal{S}_s^{\mathcal{B}}$, resp. $a_s^{\mathcal{B}}$) the user (resp. the s^{th} server, resp. the s^{th} answer) of protocol $\mathcal{B}$. The general idea is that one server simulates $\mathcal{S}_1^{\mathcal{B}}$ and sends his answer, while the others servers simulate $\mathcal{S}_2^{\mathcal{B}}$. Since the answer $a_2^{\mathcal{B}}$ is in general a very long bit-string and the user is interested only in some of its bits (as we have seen above), they do not send their answer, but they consider it as a new (shorter) database.

For clarity, we will not present the scheme as a triple of algorithms $(\mathcal{Q}, \mathcal{A}, \mathcal{R})$ but rather as a recursive protocol. However it is important to notice that the user can compute in advance all the queries he needs to send; so he can send all of them at one, resulting in a single-round scheme. The scheme would consist of $(k-1)$ level of recursion and we denote the l^{th}-level by Level$_l$. We set $k_1 = k$ and $x^1 = x$, so $n_1 = |x^1| = n$.

For $1 \leq l \leq k-1$ do Level$_l$:

- Let $k_l = k - l + 1 (= k_{l-1} - 1)$.

  $\mathcal{U}, \mathcal{S}_l, \ldots, \mathcal{S}_k$ perform $\mathcal{B}$, using k_l instead of k, as follows:

- $\mathcal{U}$ and $\mathcal{S}_l$ simulate $\mathcal{U}^{\mathcal{B}}$ and $\mathcal{S}_1^{\mathcal{B}}$ and $\mathcal{S}_l$ sends his answer $(a_1^{\mathcal{B}})_l$ to $\mathcal{U}$ (the subscript l refers to the level).

- For every $l+1 \leq s \leq k$, $\mathcal{U}$ and $\mathcal{S}_s$ simulate $\mathcal{U}^{\mathcal{B}}$ and $\mathcal{S}_2^{\mathcal{B}}$.

  If $l = k-1$, then $\mathcal{S}_s = \mathcal{S}_k$ sends his answer $(a_2^{\mathcal{B}})_{k-1}$ to $\mathcal{U}$.

  Otherwise $\mathcal{S}_s$ does not send his answer $(a_2^{\mathcal{B}})_l$ to $\mathcal{U}$, but he considers it as a new database:

  - Let $\begin{cases} x^{l+1} = (a_2^{\mathcal{B}})_l, \\ n_{l+1} = |x^{l+1}| = \mathcal{O}(n_l^{(2k_l-3)/(2k_l-1)}); \end{cases}$
  - $\mathcal{U}, \mathcal{S}_{l+1}, \ldots, \mathcal{S}_k$ go to Level$_{l+1}$, with x^{l+1} as database. Since $\mathcal{U}$ wants to retrieve $2^{k_l-1} - 2k_l$ bits from it (as we have seen above), $\mathcal{U}, \mathcal{S}_{l+1}, \ldots, \mathcal{S}_k$ perform $2^{2k_l-1} - 2k_l$ iterations of Level$_{l+1}$.

- $\mathcal{U}$ performs the algorithm $\mathcal{R}$ of protocol $\mathcal{B}$ using $(a_1^{\mathcal{B}})_l$ and the relevant bits of $(a_2^{\mathcal{B}})_l$ to reconstruct the desired bit of x^l.

Theorem A.2.1. *The protocol $\mathcal{IT}_4$ defined above is an itPIR protocol such that $\mathcal{CC}_{\mathcal{IT}_4}(n) = \mathcal{O}(n^{1/(2k-1)})$, with $k \geq 2$ the number of servers.*

Proof. We have to prove that $\mathcal{IT}_4$ verifies the definition 2.0.1 and that it has the required communication complexity.

1. (Correctness) $\mathcal{S}_l$ sends the whole $(a_1^{\mathcal{B}})_l$ to $\mathcal{U}$ who preforms $2^{2k_l-1} - 2k_l$ iteration of Level_{l+1} in order to retrieve the relevant bits of $(a_2^{\mathcal{B}})_l$. Thus, assuming the correctness of Level_{l+1}, $\mathcal{U}$ can reconstruct the desired bit of $x^l = (a_2^{\mathcal{B}})_{l-1}$. Since at Level_{k-1} $\mathcal{S}_k$ sends to $\mathcal{U}$ the whole $(a_2^{\mathcal{B}})_{k-1}$, we can recursively see that the protocol is correct.

2. (Privacy) It follows from the privacy of $\mathcal{B}$.

3. (Communication complexity) We prove by induction that $\mathcal{O}(n_l^{1/(2k_l-1)}) = \mathcal{O}(n^{1/(2k-1)})$, for every $1 \leq l \leq k-1$.

   For $l = 1$ it is trivial($n_1 = n$ and $k_1 = k$).

   Suppose it is true for l, then:

$$\begin{aligned} \mathcal{O}\big((n_{l+1})^{1/(2k_{l+1}-1)}\big) &= \mathcal{O}\Big(\big(n_l^{(2k_l-3)/(2k_l-1)}\big)^{1/(2(k_l-1)-1)}\Big) = \\ &= \mathcal{O}\big(n_l^{1/(2k_l-1)}\big) = \mathcal{O}\big(n^{1/(2k-1)}\big). \end{aligned}$$

   At Level_l $\mathcal{U}$ sends to $\mathcal{S}_l, \ldots, \mathcal{S}_k$ $\mathcal{O}(n_l^{1/(2k_l-1)}) = \mathcal{O}(n^{1/(2k-1)})$ bits and $\mathcal{S}_l$ replies with $\mathcal{O}(n_l^{1/(2k_l-1)}) = \mathcal{O}(n^{1/(2k-1)})$ bits. On the contrary, if $l \neq k-1$ then $\mathcal{S}_{l+1}, \ldots, \mathcal{S}_k$ do not send any bit. When $l = k-1$, $\mathcal{S}_k$ sends $\mathcal{O}(n_l^{(2k_l-3)/(2k_l-1)}) \overset{(k_l=2)}{=} \mathcal{O}(n_l^{1/(2k_l-1)}) = \mathcal{O}(n^{1/(2k-1)})$ bits.

   Since the number of levels and the number of iterations of each level depend only on k, which is constant and in particular independent of n, the total amount of communication is $\mathcal{CC}_{\mathcal{IT}_4}(n) = \mathcal{O}\big(n^{1/(2k-1)}\big)$.

□

Bibliography

[1] A. Ambainis. Upper bound on communication complexity of private information retrieval. In *ICALP '97: Proceedings of the 24th International Colloquium on Automata, Languages and Programming*, pages 401–407, London, UK, 1997. Springer-Verlag.

[2] A. Beimel, Y. Ishai, E. Kushilevitz, and J. F. Raymond. Breaking the o(n1/(2k-1)) barrier for information-theoretic private information retrieval. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 261–270, Washington, DC, USA, 2002. IEEE Computer Society.

[3] G. Brassard, C. Crépeau, and J. M. Robert. Information theoretic reductions among disclosure problems. In *FOCS*, pages 168–173, 1986.

[4] G. Brassard, C. Crépeau, and M. Sántha. Oblivious transfers and intersecting codes. *IEEETIT: IEEE Transactions on Information Theory*, 42, 1996.

[5] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. *Lecture Notes in Computer Science*, 1592:402–414, 1999.

[6] Y. Chang. Single database private information retrieval with logarithmic communication, 2004.

[7] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, page 41, Washington, DC, USA, 1995. IEEE Computer Society.

[8] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[9] J. A. Clark and J. L. Jacob. A survey of authentication protocol literature. Technical Report 1.0, 1997.

[10] G. Di Crescenzo, T. Malkin, and R. Ostrovsky. Single database private information retrieval implies oblivious transfer. *Lecture Notes in Computer Science*, 1807:122+, 2000.

[11] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *J. Comput. Syst. Sci.*, 60(3):592–629, 2000.

[12] O. Goldreich. *Foundations of Cryptography*, volume Basic Tools. Cambridge University Press, 2001.

[13] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[14] K. Ireland and M. Rosen. *A Classical Introduction to Modern Number Theory*. Number 84 in Graduate Texts in Mathematics. Springer, 2nd edition, 1998.

[15] Y. Ishai and E. Kushilevitz. Improved upper bounds on information-theoretic private information retrieval (extended abstract). In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 79–88, New York, NY, USA, 1999. ACM Press.

[16] T. Itoh. Efficient private information retrieval. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, 1999.

[17] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, page 364, Washington, DC, USA, 1997. IEEE Computer Society.

[18] E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. *Lecture Notes in Computer Science*, 1807:104–121, 2000.

[19] E. Mann. Private access to distributed information, 1998.

[20] Sanjeev Kumar Mishra and Palash Sarkar. Symmetrically private information retrieval. In *INDOCRYPT*, pages 225–236, 2000.

[21] R. Ostrovsky and W. E. Skeith III. A survey of single database pir: Techniques and applications. Cryptology ePrint Archive, Report 2007/059, 2007.

[22] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Lecture Notes in Computer Science*, 1592:223–238, 1999.