A General Framework for One Database Private Information Retrieval

Arkady Yerukhimovich

Dept. of Computer Science, University of Maryland, College Park, MD, USA
arkady@cs.umd.edu

Abstract. In this paper we present and analyze a general scheme for one database computationally Private Information Retrieval. This scheme was first presented by Ostrovsky and Skeith in [13]. This general framework shows such a PIR scheme based on any secure homomorphic public key encryption scheme. We complete the analysis of this framework, present a proof of security, and do a careful analysis of the communication and computation complexity.

1 Introduction

In Private Information Retrieval(PIR) we view the database as an n-bit string x where the user wants to retrieve the bit x_i while keeping i private from the database. The traditional, information theoretic, model of PIR was formulated by Chor, et. al. in [3]. The authors show that n bits of communication are necessary to achieve information theoretic privacy if only one database is available. In order to circumvent this bound, the authors propose using multiple non-interacting databases. Much work in this area has produced upper and lower bounds for information theoretic PIR schemes with different numbers of databases.

A different model of PIR called Computationally Private Information Retrieval(cPIR) was proposed by Kushilevitz and Ostrovsky in [10] to get around this issue of multiple databases. In this model, the authors showed an $O(n^{\epsilon})$ communication complexity, one database cPIR protocol where the security of the user's query is based on a cryptographic assumption. In a further result [12], Mann generalized this scheme to use any bitwise encryption scheme with certain homomorphism properties. For a list of results on both information theoretic and computational PIR see [5, 6].

All these results deal with optimizing the communication complexity of cPIR protocols. However, a problem that has largely been ignored by all of them is the computational complexity of such algorithms. This is in part due to an obvious lower bound that the server must perform $\Omega(n)$ computation, because if the database does not touch a bit in generating its response it knows the user did not request that bit thus violating the security. It is this computational cost that caused people to claim that cPIR is not practical because the trivial solution of sending the entire database is faster [17]. However, by reducing the cost of computation per bit it may be possible to overcome this issue since computation is much cheaper than communication. Another common complaint about the PIR model is that a database is not an n-bit string. In an effort to resolve this, we present a PIR scheme that allows the database to contain integers instead of bits.

In this paper we present a general framework for PIR introduced by [13]. This scheme further generalizes the scheme from [12] and presents a construction of cPIR from any homomorphic, secure public key encryption scheme. We describe this scheme in detail and complete the analysis, showing its computation complexity and proof of security. We also discuss several concrete implementations of the general scheme and suggest possible ways for further improvements.

This paper is organized as follows. In section 2, we present some necessary definitions. In section 3, we explain the cPIR framework as follows. In section 3.1, we present a basic protocol which does not accomplish the promised communication complexity but is a building block used in the main construction. In section 3.2, we present the full construction building on the previous protocol. Then, in section 3.3 we prove the security of the presented protocol and in section 3.4 we show and analyze two concrete implementations of the presented scheme.

2 Definitions

2.1 CPA Secure Public Key Encryption

The definition presented below is based on Definition 10.3 from [9]. This notion of security was originally defined by Goldwasser and Micali in [8]. For a more rigorous treatment see Chapter 5 of [7].

A public key encryption scheme is a triple of algorithms $(\mathcal{G}, \mathcal{E}, \mathcal{D})$. The key generation algorithm, $\mathcal{G}$, takes as input 1^k, a string of k ones, and outputs (pk, sk) where pk is the public key of the encryption scheme and sk is the secret key. The encryption algorithm, $\mathcal{E}_{pk}(m)$, uses the public key to compute the encryption of a message m. The decryption algorithm, $\mathcal{D}_{sk}(c)$, computes the decryption of a ciphertext c using the secret key. We require that for every k, every (pk, sk) output by $\mathcal{G}(1^k)$, and every message m it holds that $m = \mathcal{D}_{sk}(\mathcal{E}_{pk}(m))$.

A public key encryption scheme is said to be secure against chosen plaintext attacks(CPA-secure) if no polynomial time bounded adversary can succeed in the following experiment with probability greater than $1/2 + negl(k)$ where $negl()$ is a negligible function, k is the security parameter, and Π is the environment running the experiment.

Eavesdropping Indistinguishability Experiment $PubK^{eav}_{A,\Pi}(k)$

1. $\mathcal{G}(1^k)$ is run to output (pk, sk)
2. Adversary A is given pk as an input.
3. A outputs two messages m_0, m_1 with $|m_0| = |m_1|$.
4. Random bit $b \leftarrow \{0, 1\}$ is chosen.
5. The ciphertext $c \leftarrow \mathcal{E}_{pk}(m_b)$ is computed.
6. A is given c and outputs a bit b'.
7. A succeeds if $b' = b$. We say that in this case $PubK^{eav}_{A,\Pi}(k) = 1$

Note that any adversary can succeed with probability 1/2 by just outputting a random bit b'. Also note that since A knows pk, he can compute the encryption of any message m'. Therefore, any encryption scheme that satisfies this notion must be probabilistic. Otherwise A can just compute $\mathcal{E}_{pk}(m_0)$ and $\mathcal{E}_{pk}(m_1)$ and see which of them is equal to c.

2.2 Group-Homomorphic Encryption [13]

An encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is called group-homomorphic if it has plaintext set G_p and ciphertext set G_c, where G_p and G_c are abelian groups and the following equation holds

$$\mathcal{D}(\mathcal{E}(a) \star \mathcal{E}(b)) = a * b \tag{1}$$

where $a, b \in G_p$, $\star$ is the group operation over G_c and $*$ is the group operation over G_p. Many of the known secure encryption schemes have these homomorphic properties. We will see two examples of such schemes in Section 3.4.

3 PIR Based on Homomorphic Encryption [13]

Theorem 1. *There exists a one database computational PIR scheme based on any CPA-secure homomorphic-encryption scheme with $O(n^\epsilon)$ communication complexity.*

In this section, we present a very general construction of the cPIR protocol of Theorem 1 from any such encryption scheme. This construction is a secure PIR as long as the underlying encryption scheme is CPA-secure. For clarity, we start with an $O(n)$ communication complexity protocol to demonstrate the techniques we will use. We then show how to extend this to an $O(n^\epsilon)$ protocol. Then we show several examples of concrete schemes that fit within our general framework.

From now on we will use $\mathcal{U}$ to represent the user in the PIR scheme and $\mathcal{DB}$ to represent the database.

3.1 The Basic Protocol

Definition Let, $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a CPA-secure group-homomorphic encryption scheme as defined in Section 2.2. Let G_p, G_c be the corresponding groups over the plaintext and ciphertext. Let ID_{G_p}, ID_{G_c} represent the identity elements in the corresponding groups. Let $g \in G_p$ be an element in G_p s.t. $g \neq ID_{G_p}$. We use $ord(g)$ to mean the order of g in G_p. Note that g is not necessarily a generator of G_p, although choosing g to be a generator will be useful as it maximizes $ord(g)$.

We view the database as an array $X = \{x_i\}_{i=1}^n$ where $x_i \in \mathbb{Z}_{ord(g)}$. Therefore, if we choose an element g such that $ord(g) > N$ the database can contain numbers in $\mathbb{Z}_N$. However, if the plain text space is $\mathbb{Z}_2$, as is the case in any bit-wise encryption scheme, then we have to view the database as an n-bit string, as it is in the traditional definition of PIR.

A 1-dimensional protocol We present this protocol as an introduction to the class of PIR protocols presented in this paper. While it does not accomplish any improvement in communication complexity over the trivial solution of sending the entire database it serves as a good example of the technique used in later protocols. Here $\mathcal{U}$ views the database as an n-element array $X = \{x_i\}_{i=1}^n$ and he wants to learn the value x_{i^*}.

1. $\mathcal{U}$ forms a query $Q = \{q_i\}_{i=1}^n$ where $q_i \in G_c$ and

$$q_i = \begin{cases} \mathcal{E}(g) & \text{if } i = i^* \\ \mathcal{E}(ID_{G_p}) & \text{otherwise} \end{cases}$$

2. $\mathcal{U}$ sends Q to $\mathcal{DB}$
3. $\mathcal{DB}$ sends back

$$R = \sum_{i=1}^{n} x_i \cdot q_i$$

Note that summation here represents the group operation of G_c and $\cdot$ represents the $\mathbb{Z}$-module action, this means that we repeat the group operation x_i times on q_i before summing.

4. $\mathcal{U}$ computes

$$\mathcal{D}(R) = \mathcal{D}\left(\sum_{i=1}^{n} x_i \cdot q_i\right) = \sum_{i=1}^{n} x_i \cdot \mathcal{D}(q_i) = x_{i^*} \cdot g$$

5. $\mathcal{U}$ sees that $x_{i^*} = 1$ if and only if $\mathcal{D}(R) = g$. If the discrete log problem is easy over G_p then $\mathcal{U}$ can instead compute $x_{i^*} = \log_g \mathcal{D}(R)$ and so we can use integers instead of bits as entries in the database allowing us to overcome the complaint about the database being an n-bit string.

Note that in this context discrete log means the following problem. Given $y = x \cdot g = g^x$ and g find $x \in \mathbb{Z}_{ord(g)}$ where the exponentiation is done over the group G_p. We write this as $x = \log_g y$. In some groups, such as the additive group $\mathbb{Z}_N$, this is easy because exponentiation is just equivalent to multiplication and a logarithm is just equivalent to division.

In this scheme $\mathcal{U}$ sends n elements of G_c and $\mathcal{DB}$ responds with 1 element of G_c. So $\mathcal{U}$'s communication complexity is $O(n)$. However, we can reduce this by balancing the communication between $\mathcal{U}$ and $\mathcal{DB}$ as shown in the next protocol.

A 2-dimensional protocol $\mathcal{U}$ and $\mathcal{DB}$, view the database as a $(\sqrt{n} \times \sqrt{n})$ array, $X = \{x_{(i,j)}\}_{i,j=1}^{\sqrt{n}}$. $\mathcal{U}$ wants to learn $x_{(i^*,j^*)}$ and does not want $\mathcal{DB}$ to learn anything about the values of i^* or j^*.

1. $\mathcal{U}$ forms a query $Q = \{q_i\}_{i=1}^{\sqrt{n}}$ where $q_i \in G_c$ and

$$q_i = \begin{cases} \mathcal{E}(g) & \text{if } i = i^* \\ \mathcal{E}(ID_{G_p}) & \text{otherwise} \end{cases}$$

2. $\mathcal{U}$ sends Q to $\mathcal{DB}$
3. For each column j, $\mathcal{DB}$ computes

$$R_j = \sum_{i=1}^{\sqrt{n}} x_{(i,j)} \cdot q_i$$

Note, that summation here represents the group operation of G_c and $\cdot$ represents the $\mathbb{Z}$-module action.

4. $\mathcal{DB}$ sends back $\{R_j\}_{j=1}^{\sqrt{n}}$ to $\mathcal{U}$
5. $\mathcal{U}$ computes

$$\mathcal{D}(R_{j^*}) = \mathcal{D}\left(\sum_{i=1}^{\sqrt{n}} x_{(i,j^*)} \cdot q_{(i,j^*)}\right) = \sum_{i=1}^{\sqrt{n}} x_{(i,j^*)} \cdot \mathcal{D}(q_{(i,j^*)}) = x_{(i^*,j^*)} \cdot g$$

6. $\mathcal{U}$ sees that $x_{(i^*,j^*)} = 1$ if and only if $\mathcal{D}(R_{i^*}) = g$. If, the discrete log problem is easy over G_p then $\mathcal{U}$ can instead compute $x_{(i^*,j^*)} = \log_g \mathcal{D}(R_{i^*})$.

Communication Complexity $\mathcal{U}$ sends $\sqrt{n}$ elements of G_c as his query for a total communication complexity of $\sqrt{n} \log |G_c|$. Setting the security parameter $k = \log |G_c|$ we get communication complexity $k\sqrt{n}$. $\mathcal{DB}$ also sends $\sqrt{n}$ elements of G_c. Therefore, the total communication complexity is $O(kn^{1/2})$ taking $k = n^\epsilon$ gives $CC = O(n^{1/2+\epsilon})$.

Computation $\mathcal{U}$ computes $\sqrt{n}$ encryption operations to generate a query and 1 decryption to interpret the result. $\mathcal{DB}$ computes $\sqrt{n}$ group operations over G_c. If the elements of the DB are not bits then $\mathcal{DB}$ must also compute $\sqrt{n}$ $\mathbb{Z}$-module actions and 1 discrete log computation.

3.2 A Further Improvement

In this section we show how to improve on the previous protocols to achieve $O(n^{\epsilon})$ communication complexity. We build up to the final protocol by presenting simpler protocols that take advantage of the same techniques. We then analyze the communication and computation complexity of the final protocol.

Definition Define $\phi : G_c \rightarrow \mathbb{Z}^l_{ord(g)}$ to be an injective mapping from the ciphertexts to an l-dimensional vector over $\mathbb{Z}_{ord(g)}$ where $ord(g)$ is the order of g in G_p. Here l can be any integer sufficiently large that $|\mathbb{Z}^l_{ord(g)}| = ord(g)^l > |G_c|$. ϕ can be any injective mapping between these sets. We only require that both ϕ and ϕ^{-1} be efficiently computable given the public information. Note that $l \geq 2$ for any such scheme if the underlying encryption scheme is CPA secure. This is because $ord(g) \leq |G_p|$ and $|G_p| < |G_c|$. The second inequality is due to the fact that the encryption scheme must be probabilistic to be CPA secure. There must be at least one element in $|G_c|$ that decrypts to each element of $|G_p|$ to ensure correct decryption. If we have randomized encryption then G_c must contain extra elements on top of this to account for the randomness, so $|G_c| > |G_p|$.

Another 2-dimensional protocol We present this algorithm to build an understanding of this class of PIR protocols. It achieves the same communication complexity as the basic 2-dimensional protocol presented earlier but demonstrates the new techniques.

1. $\mathcal{U}$ and $\mathcal{DB}$, view the database as a $(\sqrt{n} \times \sqrt{n})$ array, $X = \{x_{(i,j)}\}_{i,j=1}^{\sqrt{n}}$
2. $\mathcal{U}$ forms a query $Q = \left[\{q_i\}_{i=1}^{\sqrt{n}}, \{p_j\}_{j=1}^{\sqrt{n}}\right]$ where $q_i, p_j \in G_c$, q_{i^*} and p_{j^*} are set to encryptions of g and all other values are set to encryptions of ID_{G_p}.
3. $\mathcal{U}$ sends Q to $\mathcal{DB}$
4. For each row i, for each $t \in [l]$, $\mathcal{DB}$ computes

$$R_j = \sum_{i=1}^{\sqrt{n}} x_{(i,j)} \cdot q_i$$

$$\overline{R}_t = \sum_{j=1}^{\sqrt{n}} \phi(R_j)_t \cdot p_j$$

   where $\phi(R_j)_t$ is the t-th component of $\phi(R_j)$. Note that, as before, summation here represents the group operation of G_c and $\cdot$ represents $\mathbb{Z}$-module action.
5. $\mathcal{DB}$ sends back all the $\overline{R}_t$s to $\mathcal{U}$
6. $\mathcal{U}$ computes for every $t \in [l]$

$$\mathcal{D}(\overline{R}_t) = \mathcal{D}\left(\sum_{j=1}^{n} \phi(R_j)_t \cdot p_j\right) = \phi(R_{j^*})_t \cdot g$$

7. $\mathcal{U}$ calculates $\phi(R_{j^*})_t = \log_g \mathcal{D}(\overline{R}_t)$ if working over a group where discrete log is easy or checks if $\mathcal{D}(\overline{R}_t) = g$ otherwise.
8. $\mathcal{U}$ uses these values to reconstruct $\phi(R_{j^*})$ and computes $R_{j^*} = \phi^{-1}(\phi(R_{j^*})) = \sum_{i=1}^{\sqrt{n}} x_{(i,j^*)} \cdot q_i$.
9. $\mathcal{U}$ recovers $x_{(i^*,j^*)}$ from these values just as in the basic scheme.

The above protocol achieves the following communication complexity which will be shown in a more general setting later

$$CC_{\mathcal{U}} = 2kn^{1/2} \qquad CC_{\mathcal{DB}} = lk$$

The 3-dimensional protocol We present this protocol to show how to reduce the communication complexity of the previous protocol by viewing the database as a higher dimension array. This is used to show how to extend the previous scheme to one that views the database as a d-dimensional array and has significantly lower communication complexity.

1. $\mathcal{U}$ and $\mathcal{DB}$, view the database as a $(n^{1/3} \times n^{1/3} \times n^{1/3})$ array, $X = \{x_{(i,j,k)}\}_{i,j,k=1}^{n^{1/3}}$
2. $\mathcal{U}$ forms a query $Q = \left[\{q_i\}_{i=1}^{n^{1/3}}, \{p_j\}_{j=1}^{n^{1/3}}, \{o_k\}_{k=1}^{n^{1/3}}\right]$ where $q_i, p_j, o_k \in G_c$ s.t. q_{i^*}, p_{j^*}, o_{k^*} are set to encryptions of g and all other values are set to encryptions of ID_{G_p}.
3. $\mathcal{U}$ sends Q to $\mathcal{DB}$
4. For each row i, for each $t, u \in [l]$, $\mathcal{DB}$ computes

$$R_{jk} = \sum_{i=1}^{n^{1/3}} x_{(i,j,k)} \cdot q_i$$

$$\overline{R}_{kt} = \sum_{j=1}^{n^{1/3}} \phi(R_{jk})_t \cdot p_j$$

$$\overline{\overline{R}}_{tu} = \sum_{k=1}^{n^{1/3}} \phi(\overline{R}_{kt})_u \cdot o_k$$

5. $\mathcal{DB}$ sends back all the $\{\overline{\overline{R}}_{tu}\}$ to $\mathcal{U}$
6. $\mathcal{U}$ performs the following computations
   – For each $t \in [l]$
   –– For each $u \in [l]$ compute

$$\mathcal{D}(\overline{\overline{R}}_{tu}) = \mathcal{D}\left(\sum_{k=1}^{n^{1/3}} \phi(\overline{R}_{kt})_u \cdot o_k\right) = \phi(\overline{R}_{k^*t})_u \cdot g$$

   –– Use these l values to get $\overline{R}_{k^*t} = \sum_{j=1}^{n^{1/3}} \phi(R_{jk})_t \cdot p_j$
   – Get the l values $\overline{R}_{k^*t}$ for all $t \in [l]$
7. $\mathcal{U}$ gets $x_{(i^*,j^*,k^*)}$ from this the same way as in the 2-dimensional scheme.

This protocol achieves the following communication complexity.

$$CC_{\mathcal{U}} = 3kn^{1/3} \qquad CC_{\mathcal{DB}} = l^2k$$

The d-dimensional protocol Using the same techniques as in the two protocols presented earlier we can extend the protocol to a d-dimensional scheme where $\mathcal{U}$ and $\mathcal{DB}$ view the database as a d-dimensional array.

Communication Complexity The communication complexity of the d-dimensional PIR scheme presented above is as follows. Let $k = \log |G_c|$ be the security parameter of the encryption scheme.

– $\mathcal{U}$'s query consists of d sets of $n^{1/d}$ elements of G_c each.
– $\mathcal{DB}$'s response consists of l^{d-1} elements of G_c. This is because for every extra dimension l new elements are created for each element at the previous dimension.

Summarizing, we get the following communication complexity.

$$CC_{\mathcal{U}} = O(kdn^{1/d}) \qquad CC_{\mathcal{DB}} = O(l^{d-1}k) \tag{2}$$

Setting d to be a large constant, but keeping $d << n$, and $l, k = O(\log n)$ we get the promised result.

$$CC_{total} = O(n^{\epsilon}) \text{ for any } \epsilon > 0 \tag{3}$$

Computation The total computation that is performed per query by $\mathcal{U}$ and $\mathcal{DB}$ in the d-dimensional PIR scheme presented above is as follows.

– $\mathcal{U}$ computes the $dn^{1/d}$ encryptions necessary to generate his query.

$$\# \text{ encryptions} = dn^{1/d} \tag{4}$$

– $\mathcal{DB}$ performs the following computations to answer a query. We can rewrite the 3 equations describing $\mathcal{DB}$'s response for the 3-dimensional protocol as d equations for the d-dimensional protocol. We number these equations in the order presented above from 1 to d. In equation i, $\mathcal{DB}$ performs $n^{1/d}$ group operations for each of $l^{i-1}n^{(d-i)/d}$ elements for a total of $l^{i-1}n^{(d-i+1)/d}$ group operations in level i. Starting at level 2, $\mathcal{DB}$ also computes ϕ once for each of the $l^{i-1}n^{(d-i)/d}$ elements for a total of $l^{i-1}n^{(d-i)/d}$ computations of ϕ per level when $i \geq 2$. Writing this as a summation over the d equations we get.

$$\# \text{ group ops over } G_c = \sum_{i=1}^{d} l^{i-1}n^{(d-i+1)/d} = O(n) \qquad \text{if } l << n \tag{5}$$

$$\# \ \mathbb{Z}\text{-module actions} = \sum_{i=1}^{d} l^{i-1}n^{(d-i+1)/d} = O(n) \qquad \text{if } l << n \tag{6}$$

$$\# \text{ computations of } \phi = \sum_{i=2}^{d} l^{i-1}n^{(d-i)/d} = O(n^{(d-2)/d}) \qquad \text{if } l << n \tag{7}$$

– $\mathcal{U}$ performs the following computations to interpret the response. For the 1-dimension scheme $\mathcal{U}$ computes one decryption. In the i-dimension scheme $\mathcal{U}$ computes l^{i-1} decryptions and l^{i-2} computations of ϕ^{-1} to get to the situation in the $(i-1)$-dimension scheme. If working over a group where discrete log is easy and using a database containing numbers instead of bits, it also computes a discrete log for each decryption. Therefore we get,

$$\# \text{ decryptions} = \sum_{i=0}^{d-1} l^i = \frac{l^d - 1}{l - 1} \tag{8}$$

$$\# \text{ discrete logs} = \sum_{i=0}^{d-1} l^i = \frac{l^d - 1}{l - 1} \tag{9}$$

$$\# \text{ computations of } \phi^{-1} = \sum_{i=0}^{d-2} l^i = \frac{l^{d-1} - 1}{l - 1} \tag{10}$$

3.3 Security

We now prove the security of the 1-dimensional PIR scheme of Section 3.1. This proof can be trivially modified to work for any of the other schemes presented here by just including additional hybrid steps to deal with multiple encryptions of g and showing that we can not distinguish a query with multiple encryptions of g from a query with 1 encryption of g.

Definition Let Q_i to be a vector of length n of encryptions of ID_{G_p} except at position i which contains an encryption of g. Let Q_j be the same for index j. Let Q_0 be a vector of n encryptions of ID_{G_p}. Note that Q_i and Q_j are exactly the distributions for queries for indices i and j. Let k be a security parameter for the underlying encryption scheme and let $negl(k)$ be a negligible function.

Claim (1). For any probabilistic polynomial time(PPT) adversary A, $|Pr[A(Q_i) = 1] - Pr[A(Q_0) = 1]| \leq negl(k)$

Proof. Assume that for some PPT adversary A, $|Pr[A(Q_i) = 1] - Pr[A(Q_0) = 1]| = \epsilon$. We use this adversary to construct an adversary A' that breaks the CPA-security of the underlying encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ as follows.

A' - Given pk from $\mathcal{G}$ and playing the CPA indistinguishability game

1. Compute $n-1$ values $\mathcal{E}_{pk}(ID_{G_p})$ and place them in all places of the query, Q, except at index i
2. Set $m_0 = ID_{G_p}$, $m_1 = g$
3. Receive $c \leftarrow \mathcal{E}_{pk}(m_b)$ where $b \leftarrow \{0, 1\}$ is a uniformly random bit.
4. Place c into position i of Q.
5. Run $A(Q)$ and output what it outputs.

Note, that when $b = 0$ Q has the same distribution as Q_0 and when $b = 1$ Q has the same distribution as Q_i. So,

$$\begin{aligned} Pr[A' \text{ succeeds in CPA game}] &= Pr[A(Q) = 0 | b = 0] \cdot Pr[b = 0] + Pr[A(Q) = 1 | b = 1] \cdot Pr[b = 1] \\ &= (1/2)(1 - Pr[A(Q_0) = 1]) + (1/2) Pr[A(Q_i) = 1] \\ &= 1/2 + \frac{\epsilon}{2} \end{aligned}$$

Since we assumed that the underlying Encryption scheme was CPA-secure and so no A' can succeed except with probability negligibly greater than 1/2, this implies the $\epsilon \leq negl(k)$ and so no PPT adversary A can distinguish between Q_i and Q_0 with non-negligible probability.

Claim (2). For any PPT adversary A, $|Pr[A(Q_j) = 1] - Pr[A(Q_0) = 1]| \leq negl(n)$

Proof. The proof for this is the same as for claim 1 just replacing index i with index j.

Lemma 1. *For any PPT adversary A $|Pr[A(Q_i) = 1] - Pr[A(Q_j) = 1]| \leq negl(n)$*

Proof. Using claims 1 and 2 we get that

$$\begin{aligned} |Pr[A(Q_i) = 1] - Pr[A(Q_j) = 1]| &= |(Pr[A(Q_i) = 1] - Pr[A(Q_0) = 1]) - \\ &\quad (Pr[A(Q_j) = 1] - Pr[A(Q_0) = 1])| \\ &= |negl_1(k) - negl_2(k)| = negl(k) \end{aligned}$$

where $negl_1(), negl_2(), negl()$ are all negligible functions.

This lemma shows that no polynomially bounded adversary can distinguish between the distributions of queries for index i and for index j as long as the underlying encryption scheme satisfies CPA-security. The same proof can be extended to work for higher dimensional schemes by just modifying the argument to deal with multiple places containing encryptions of g.

3.4 Implementations

Historically several implementations of the general scheme presented above have appeared in literature. We briefly present two of these schemes and explain how to set up the general scheme to instantiate them.

Bit-Wise cPIR Schemes [10, 12] The original formulation of PIR viewed the database as an n-bit string. For this reason the earliest computational PIR protocols only dealt with $G_p = \mathbb{Z}_2$. The first such one-database scheme proposed is the PIR protocol presented by Kushilevitz and Ostrovsky in [10]. This was the first one-database PIR scheme to accomplish sub-linear communication complexity. The protocol is based on the Goldwasser-Micali public key encryption scheme of [8]. This is a homomorphic encryption scheme with $G_p = \mathbb{Z}_2$ and $G_c = \{x \in \mathbb{Z}_N^*\}$ s.t $J(x) = +1$, where N is the product of two large primes and $J(x)$ is the Jacobi symbol. An encryption of 0 is a random quadratic residue modulo N and an encryption of 1 is a random non-quadratic residue in $\mathbb{Z}_N$. The homomorphism property comes from the fact that the product of two quadratic residues is a quadratic residue and the product of a quadratic residue with a non-quadratic residue is a non-quadratic residue. The element $g = 1 \neq ID_{\mathbb{Z}_2}$ is chosen as a non-identity element. Notice that $ord(g) = 2$. Therefore a natural function to use as $\phi : G_c \rightarrow \mathbb{Z}_2^l$ is the function that maps an integer in $\mathbb{Z}_{\mathbb{N}}^*$ to its binary representation. This function is clearly publicly computable and invertible.

Setting G_c, G_p, g, and ϕ as above the recursive scheme from [10] with $d - 2$ levels of recursion is just the above scheme on d-dimensions with each level of the recursion viewed as looking for certain bits out of the answer generated at the previous level.

In [12], Mann generalized this scheme to use any other homomorphic encryption scheme that uses $\mathbb{Z}_2$ as its plaintext (bit-wise encryption). His scheme is just the same instantiation of the general protocol as [10] except using different encryption and decryption protocols and making the corresponding changes in G_c and ϕ

One thing to notice about any such scheme is that since $\log |G_c| = k$, the security parameter, and for any $g \in G_p = \mathbb{Z}_2$, $ord(g) \leq 2$ we need to set $l = k$ so that $|\mathbb{Z}_2^l| = 2^l = 2^k = |G_c|$ and ϕ can be injective. We can compute the communication and computation complexity for any such scheme using equations 2, 4 - 10. Doing so for communication complexity we get

$$CC_{\mathcal{U}} = O(kdn^{1/d}) \qquad CC_{\mathcal{DB}} = O(k^d)$$

This unfortunately causes a problem because to achieve total communication complexity $O(n^\epsilon)$ we need $n = \Omega(k^{d^2})$ where $\epsilon = 1/d$, otherwise the k^d term dominates in the total communication complexity. Since we need k to be sufficiently large to make the underlying encryption scheme secure this forces the database to be unrealistically large. Another shortcoming of these schemes is that since $G_p = \mathbb{Z}_2$ the database can only contain bits.

Going Beyond the n-bit String [2] A newer cPIR scheme with some new features is the cPIR scheme from [2]. In this scheme the database is an array of n integers instead of bits. In some people's eyes this makes this scheme much more realistic. In this scheme, the encryption used is the Paillier public key encryption scheme of [14]. In this encryption scheme $G_p = \mathbb{Z}_N$ and $G_c = \mathbb{Z}^*_{N^2}$ for $N = pq$ the product of two large primes. This scheme also chooses $g = 1$ as the non-identity element of G_p. Note that unlike the scheme from [10], here $ord(g) = N$. Now since $|\mathbb{Z}^*_{N^2}| < N^2$ and $ord(g) = N$ we only need $l \geq 2$ since $|\mathbb{Z}^2_{ord(g)}| = N^2 > |\mathbb{Z}^*_{N^2}|$, which is minimal for any secure encryption scheme, as mentioned earlier. The authors suggest using a map $\phi(x) = (\lfloor \frac{x}{N} \rfloor, x \bmod N)$ where $x \in \mathbb{Z}^*_{N^2}$. This provides the necessary mapping ϕ and such that it is easily computable and invertible. In fact we can compute ϕ by the division algorithm and computing ϕ^{-1} is just a multiplication and an addition. Also, since $G_c = \mathbb{Z}^*_{N^2}$ we need $k = \log |G_c| = 2 \log N$

Setting G_p, G_c, g, and ϕ as above into this general scheme we obtain the cPIR scheme from [2]. Again, we can plug these values into equations 2, 4 - 10 to find the communication and computation complexity of this scheme. Doing this for communication complexity we get

$$CC_{\mathcal{U}} = O(kdn^{1/d}) \qquad CC_{\mathcal{DB}} = O(2^{d-1}k)$$

Note that we do not have a k^d term in either of these equations unlike the bit-wise scheme. This makes $\mathcal{DB}$'s communication complexity much lower. In order to achieve total communication complexity $O(n^\epsilon)$ in this scheme we need to set $n = \Omega(2^{d^2})$ where $\epsilon = 1/d$. This still requires the database to be very large, however it is much better than the bitwise schemes. Another big bonus of this scheme is that since the discrete log in G_p is easy, it is just equivalent to division since Z_N is an additive group, and $ord(g) = N$ we can allow the database to contain elements of Z_N instead of just bits. We can recover the value of the entry we want by solving the discrete log problem in Z_N. The downsides of this scheme are as follows. Since $G_c = \mathbb{Z}_{N^2}$ the group operations are on numbers of length $2 \log N$ instead of $\log N$ as in the other scheme. Also, since the database now contains elements of $\mathbb{Z}_N$ instead of bits, the $\mathbb{Z}$-module action performed by $\mathcal{DB}$ in each step becomes much more costly.

4 A Note on Computation

In the general scheme presented above the computation for the user is dominated by $O(n^{1/d})$ calls to $\mathcal{E}$. The database's computation time is dominated by $O(n)$ group operations over the ciphertext group G_c and $\mathbb{Z}$-module operations. Therefore, there are several potential ways to reduce the computation performed.

One way to reduce the computation cost is to reduce the cost of the group operation over G_c. This significantly speeds up the computation that must be performed by the database as it also speeds up the $\mathbb{Z}$-module operations. One way to do this is to use an encryption scheme where G_c is an additive group. It may be possible to accomplish this by using the lattice based encryption schemes presented by Regev in [15, 16].

A way to reduce the computation that must be performed by both parties is to reduce the size of the security parameter, k. This reduces the size of the ciphertext group since $k = \log |G_c|$ and therefore makes all operations over G_c cheaper. Also this would most likely lower the cost of encryption and decryption as both of those operations run in time polynomial in k. However, simply reducing the size of k without changing the encryption scheme reduces the security of the cPIR scheme. Since the security guarantee is a function of k we can not reduce k too much without making the scheme insecure. Therefore, the way to use this to significantly improve the computation is to find an encryption scheme that is secure for shorter keys.

5 Other Constructions

The scheme presented in this paper achieves $O(n^\epsilon)$ communication complexity. However, it can be further extended to achieve polylog communication complexity as explained in [13] by using a "length-flexible" cryptosystem such as the one from [4]. A concrete example of such a PIR scheme, presented in [11], achieves $O(\log^2 n)$ communication complexity.

Another approach that has been used to construct polylog communication complexity PIR schemes is by using a new computational hardness assumption called the Φ-Hiding assumption. This assumption and the first cPIR scheme using it were developed by Cachin, Micali, and Stadler in [1]. This cPIR construction can not be expressed as a special case of the general framework presented in this paper and is thus outside the scope of this paper. We refer the reader to the cited papers for details on these schemes.

6 Conclusion

In this paper we presented and analyzed a very general one-database PIR scheme based on homomorphic public key encryption. The presented scheme allows one to plug in various encryption schemes to optimize computation or security. We hope that this paper serves as a guide for future research in the area, helping people find ways to improve the computational cost of computationally secure private information retrieval.

7 Acknowledgements

I would like to thank William Gasarch for suggesting this problem to me. I also thank William Gasarch and S. Dov Gordon for insightful comments that aided me in the writing of this paper.

References

1. C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. *Lecture Notes in Computer Science*, 1592:402–414, 1999. `citeseer.ist.psu.edu/cachin99computationally.html`.

2. Y. Chang. Single database private information retrieval with logarithmic communication, 2004. `citeseer.ist.psu.edu/chang04single.html`.
3. 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. `http://citeseer.ist.psu.edu/485350.html`.
4. I. Damgard, M. Jurik, and J. Nielsen. A generalization of paillier's public-key system with applications to electronic voting, 2003. `citeseer.ist.psu.edu/damgard03generalization.html`.
5. W. Gasarch. PIR website. `http://www.cs.umd.edu/~gasarch/pir/pir.html`.
6. W. Gasarch. A survey on private information retrieval, 2004. `citeseer.ist.psu.edu/gasarch04survey.html`.
7. O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge University Press, New York, NY, USA, 2004.
8. S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. `http://theory.lcs.mit.edu/~cis/pubs/shafi/1984-jcss.pdf`.
9. J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* Chapman & Hall/CRC Press, 2007.
10. E. Kushilevitz and R. Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 364–373, 1997. `citeseer.ist.psu.edu/kushilevitz97replication.html`.
11. H. Lipmaa. An oblivious transfer protocol with log-squared communication, 2004. `citeseer.ist.psu.edu/lipmaa05oblivious.html`.
12. E. Mann. Private access to distributed information, 1998. `http://eprint.iacr.org/`.
13. R. Ostrovsky and W. E. Skeith. A survey of single database pir: Techniques and applications. Cryptology ePrint Archive, Report 2007/059, 2007. `http://eprint.iacr.org/2007/059`.
14. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Lecture Notes in Computer Science*, 1592:223–??, 1999.
15. O. Regev. New lattice based cryptographic constructions. *Journal of the ACM*, 51(6):899–942, 2004. Preliminary version in STOC'03.
16. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93, New York, NY, USA, 2005. ACM Press. `http://www.cs.tau.ac.il/~odedr/`.
17. R. Sion and B. Carbunar. On the practicality of private information retrieval. In *NDSS '07: Proceedings of the 14th Annual Network and Distributed System Security Symposium (NDSS'07)*, 2007. `http://www.carbunar.com/pir.pdf`.