Generalizing PIR for Practical Private Retrieval of Public Data

Shiyuan Wang, Divyakant Agrawal, and Amr El Abbadi

Department of Computer Science, UC Santa Barbara
{sywang, agrawal, amr}@cs.ucsb.edu

Abstract. Private retrieval of public data is useful when a client wants to query a public data service without revealing the specific query data to the server. Computational Private Information Retrieval (cPIR) is able to achieve complete privacy for a client, but is deemed impractical since it involves expensive computation on all the data on the server. Besides, it is inflexible if the server wants to charge the client based on the service data that is exposed. k-Anonymity, on the other hand, is flexible and cheap for anonymizing the querying process, but is vulnerable to privacy and security threats. In this paper, we propose a practical and flexible approach for the private retrieval of public data called *Bounding-Box PIR* (bbPIR). Using bbPIR, a client specifies both privacy requirement and service charge budget. The server satisfies the client's requirements, and at the same time achieves overall good performance in terms of computation and communication costs. bbPIR generalizes cPIR and k-Anonymity in that the bounding box can include as much as all the data on the server or as little as just k data items. The effectiveness of bbPIR compared to cPIR and k-Anonymity is verified using extensive experimental evaluation.

1 Introduction

We consider a special query problem called *private retrieval of public data*, in which a client retrieves data from a public server by providing its private data as the filtering condition, while not revealing the exact values of the private data in the query. A typical example is privacy-preserving location based services [17, 8], in which the private data is a single geographic location point, and the public data contains all possible points of interests within its neighborhood. A more general and promising use is in personalized search and recommendation services through big internet information service providers such as Google, Yahoo, and Microsoft. Users need these public services in their daily lives, but they are concerned that their personal information might be disclosed or compromised through analysis or inferences, even by the server. For example, a researcher with a potentially new idea does not want to reveal her idea to Google when she is searching for "prior art".

Currently, query privacy is not properly provided by service providers, mainly because there are no strong business incentives for the service providers to pay for the potentially expensive costs brought by enhancing the client privacy. Therefore, we consider a service model, in which the server can charge the client based on the size of the public data exposed to the client as a result of the private retrieval. The size of the exposed data depends on the private retrieval protocol for privacy purposes and is generally larger than the size of the answer to the query.

To enable practical query privacy in the service model, we have the following desiderata for a private retrieval solution:

1. *Practical*. The solution should try to minimize the *communication* overhead between a client and the server and the *computation* overhead on the server and clients. Given that queries may be issued from client devices with limited capabilities, the solution should not impose sophisticated requirements on the client side.
2. *Flexible privacy and reasonable charge*. A client can specify the required degree of privacy and the desired charge limit. The server can charge the client per query according to the private retrieval protocol which they agree on. The solution should make sure that the server satisfies a client's privacy specification and does not overcharge.

The two closest studies which could be adapted for developing a possible solution to this problem are k-*Anonymity* [21] and *Computational Private Information Retrieval* (cPIR) [18]. k-Anonymity has been applied in privacy-preserving location based services [17], where the location point of a user is blurred into a cloaked region consisting of at least k near by user points and the server returns the nearest points of interests to the cloaked region. k serves as a configurable degree of privacy. Similarly in the more general setting of private retrieval, the original private query is injected into some random data that is close to the private data in the query, such that a private data item cannot be identified from at least k data items. Then the server returns all the public data that matches the anonymized private data, which is exposed to the client and thus chargeable. However, a potential security threat with k-Anonymity is that both the client query and the server answer, although anonymized for protecting the client's privacy, are in plain texts that can be seen by a third party. The privacy of k-Anonymity for numeric data has also been questioned in a number of proposals [23, 13, 14] for potential *proximity breach*: The real private data and the blurred data could be so close that the server can conclude with probability $1/k$ that the private data is in a narrow range.

Computational Private Information Retrieval (cPIR) [11] retrieves a bit from a public bit string on a server without revealing to the server the position of the desired bit under some intractability assumption. To achieve the most balanced performance for both communication and computation costs, cPIR protocol requires the public data to be organized as a matrix. It achieves computationally complete privacy by incurring expensive computations over all public data on the server, and keeps the data communication secure by transmitting random information hiding vectors. The exposed, chargeable data is only a column of the public data matrix. However, because of its expensive computation costs on the server, even the cPIR technique with the least expensive operation, modular multiplication [11], is criticized as up to two orders of magnitude less efficient than simply transferring the entire data from the server to the client [19].

To achieve the above mentioned desiderata and seek a trade-off between the cost of retrieval and the degree of privacy, we propose a generalized private retrieval approach called *Bounding-Box PIR* (bbPIR) that unifies both k-Anonymity and cPIR. The public service data is organized as a matrix as in cPIR. A client anonymizes her private query data in a matrix called *bounding box*, whose range corresponds to a sub matrix of the public data matrix. The size of the bounding box is decided by the client's privacy requirement and desired charge limit. The area of the bounding box determines the privacy that the client can achieve, the larger the area, the higher the privacy obtained, but with higher computation and communication cost, and vice versa.

Compared to k-Anonymity, bbPIR is secure in data communication between a client and the server, because it transfers the information hidden in a bounding box instead of plain text data. Moreover, bbPIR does not suffer from proximity breach as much as k-Anonymity, because the bounding box not only includes item values that are close to the query item, but also includes item values that are not close to the query item. Compared to cPIR, bbPIR is more practical for lowering the computation costs. At one extreme, bbPIR degenerates into k-Anonymity if the range of the bounding box is a single column on the public data matrix. At the other extreme, bbPIR becomes cPIR if the range of the bounding box is the entire matrix.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 briefly explains cPIR. Section 4 describes our data model. Section 5 presents the proposed bbPIR approach. Section 6 experimentally evaluates bbPIR. Section 7 concludes the paper.

2 Related Work

Our work on private retrieval of public data is inspired by the research on Private Information Retrieval, k-Anonymity and privacy-preserving query processing.

In general, Private Information Retrieval (PIR) models private retrieval of public data as a theoretical problem as follows: Given a database which stores a binary string $x = x_1...x_n$ of length n, a client wants to retrieve x_i privately such that the database does not learn i. Chor et al. [6] first introduced the PIR problem and gave solutions on two and more database servers. Since in practice a database server is often restricted to perform only polynomial-time computations (e.g. as assumed in cryptographic applications [4]), Kushilevitz and Ostrovsky proposed a single database, computational PIR solution [11], which we refer to as cPIR in the paper. Follow-up single database computational PIR proposals improve the communication overhead, as surveyed in [18], but they use even more expensive operations than modular multiplications as pointed out by Sion and Carbunar [19], thus are not feasible for practical applications. Williams and Sion [22] attempt to make cPIR practical by using oblivious RAM. However, their approach is designed for the problem settings where client data is outsourced to the server, thus is not applicable in our context.

k-Anonymity is a widely adopted privacy policy. It generalizes or suppresses the values of data records such that each record is indistinguishable among at least k records with close values in the released private data [21]. In some contexts, the basic principle of k-Anonymity is not sufficient to protect data privacy, for example in a group of data with little diversity or high similarity. Therefore, a number of proposals have proposed new privacy principles to enhance the privacy of k-Anonymity [16, 15, 23, 13, 14]. However, they are not easy to apply in private retrieval of public data applications. Besides, they share the same security threat as k-Anonymity: all the data communication contents can be seen by a third party. Finally, the potential charge due to the k extra revealed data items from the server has never been quantified before. Our framework explicitly addresses this extra cost.

A related problem to private retrieval of public data is privacy-preserving query processing. One of the most representative applications is privacy-preserving location based service [17, 8], where our proposal of bbPIR can be applied as well. Privacy-preserving query processing has also been studied in more general applications [7, 1, 2]. Nevertheless,

these works do not address the practical service model in this paper, where the server has huge amounts of data of business value, and the client wants to retrieve a small part of these data without revealing the query to the server.

3 Background on *c*PIR

*c*PIR is designed to retrieve the value of a single bit in a large matrix [11]. It relies on the computational intractability of *Quadratic Residuosity*. Let N be a natural number, and $Z_N^* = \{x | 1 \leq x \leq N, gcd(N, x) = 1\}$. x is a *quadratic residue* (QR) mod N if

$$\exists\, y \in Z_N^*\ s.t.\ y^2 = x\ mod\ N \tag{1}$$

Otherwise, x is a *quadratic nonresidue* (QNR) mod N [9]. For example, given N = 17, 8 is QR because 5^2 = 8 mod 17, and 3 is QNR because y^2 = 3 mod 17 has no solution. The problem is considered computationally most difficult if $N = p_1 \cdot p_2$, where p_1 and p_2 are distinct large primes with equal number of bits, $m/2$. Let $Z_N^{+1} = \{x \in Z_N^* | (\frac{x}{N}) = 1\}$. The *Quadratic Residuosity Assumption* (QRA) says that for $x \in Z_N^{+1}$, without knowing p_1 and p_2 in advance, the probability of distinguishing x between a QR and a QNR is negligible for large enough number of bits, m [11].

Determining whether the number x is QR or QNR is much easier if p_1 and p_2 are known. Based on *Euler's theorem* [9], x is QR if and only if

$$x^{(p_1-1)/2}\ mod\ p_1 = 1\ \wedge\ x^{(p_2-1)/2}\ mod\ p_2 = 1 \tag{2}$$

and QNR otherwise.

Let n be the total number of public data items (bits in this case). The public data is organized into an $s \times t$ binary matrix M (choose $s = t = \lceil \sqrt{n} \rceil$ for balanced communication costs between client and server). Let (e, g) be the two dimensional address of the bit entry queried by the client (Please refer to Table 1 for a complete summary of the notations used in this paper). The *c*PIR protocol is as follows:

1. Initially, the client sends to the server an m-bit number N which is the product of two random $m/2$-bit primes p_1 and p_2.
2. To retrieve entry (e, g) in M, the client generates a vector of t m-bit random numbers in Z_N^{+1}, $y = [y_1, ..., y_t]$, s.t. y_g is a QNR and all other y_i ($i \neq g$) are QR. It sends the vector y to the server.
3. The server computes for each row i of M a modular product $z_i = \Pi_{j=1}^{t} w_{i,j}$, where $w_{i,j} = y_j^2$ if $M_{i,j}$ = 0, and $w_{i,j} = y_j$ if $M_{i,j}$ = 1.
4. The server sends to the client $z_1, ..., z_s$.
5. The client determines that $M_{e,g}$ = 0 if z_e is QR, and $M_{e,g}$ = 1 if z_e is QNR.

For example in Fig. 1, the client sends $N = 7 \times 5 = 35$ to the server initially. When the client wants to retrieve the bit at $M_{2,3}$, she generates a vector y for the second row of the matrix and sends it to the server, where y_3 is a QNR 17, and $y_1 = 4$, $y_2 = 16$, $y_4 = 11$ are QR for $N = 35$. Upon receiving y, the server computes for each row of the matrix a modular product z_i (27, 17, 33, 17), e.g. $z_2 = (4^2 \times 16 \times 17 \times 11^2)$ mod 35 = 17.

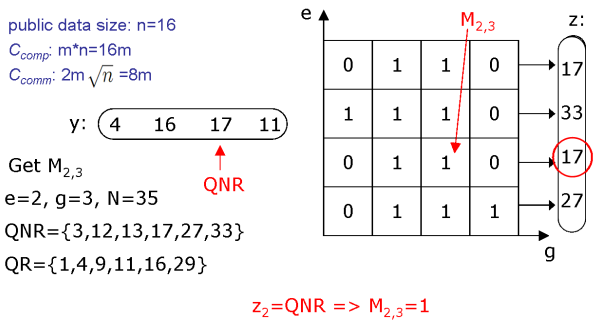


Fig. 1. cPIR Example

Since $z_2 = 17$ is a QNR, when the client receives the vector z from the server, she obtains $M_{2,3} = 1$.

Note that the server can not figure out if a y_i or z_i is QR or QNR, because the server does not know p_1 and p_2, but the client can. In step 5, the client is able to interpret $M_{i,g}$ $(1 \leq i \leq s)$ by analyzing z_i. By running one round of cPIR, all s bits in the column of the requested bit entry are exposed to the client and become chargeable.

Table 1. Summary of Notations

Notation	Description
k	for k-Anonymity
n	total number of public data items
m	modulus bit size
p_1, p_2	$m/2$-bit primes
$N = p_1 \cdot p_2$	modulus, product of 2 primes p_1 and p_2
s, t	number of rows, columns in public data matrix
$M_{s \times t}$	public data matrix
(e, g)	address of the client request entry on M
y	client query vector of m-bit random numbers
z	server reply vector of m-bit random numbers
b	number of bits in each data item
ρ	upper bound of privacy breach probability
μ	upper bound of server charge
P_{brh}	privacy breach probability
C_{srv}	server charge
C_{comm}, C_{comp}	communication and server computation cost
r, c	number of rows, columns in a query bounding box
w	minimum number of keys in a bin of a histogram

4 Data Model

We generalize the standard cPIR model in several ways. We consider a (key, address, value) data store, where each value is a b-bit data item. The public data of size n is organized in

an $s \times t$ matrix M ($s = t = \lceil \sqrt{n} \rceil$ by default). Each public data item x has a numeric key KA that determines the two dimensional address of x in M. x is only accessible through its address. Given the public data sorted by KA in ascending order, they are put in M, columnwise from the leftmost column to the rightmost column. Two query types are supported for the retrieval of x: *query by address* and *query by key*. The latter is translated to *query by address* in the retrieving process.

The client is free to specify the privacy requirement and the desired charge budget (ρ, μ), where ρ is a privacy breach limit (the upper bound probability that a requested item can be identified by the server), and μ is a server charge limit (the upper bound of the number of items that are exposed to the client for one requested item). For example, in the case of k-Anonymity, $\rho = 1/k$, $\mu \geq k$. For a public data set of size n, the best achievable privacy (the minimum ρ) is $1/n$. Similarly, the maximum charge that a client can incur is n ($\mu \leq n$), when the entire data is communicated to the client.

Based on the desiderata in Section 1, we keep track of four important metrics: (1) *Communication Cost* C_{comm}, the cost of data communication between the client and the server in terms of number of bits, including the client query and the server answer. (2) *Computation Cost* C_{comp}, the computation cost of private retrieval on the server in terms of the number of involved public data bits. The computation cost on the client is not considered here, because it is generally much smaller than the computation cost on the server, as later shown in our experiment results. (3) *Privacy Breach Probability* P_{brh}, the probability that the server can figure out a requested item, $P_{brh} \leq \rho$. (4) *Server Charge* C_{srv}, the number of interpretable public data items retrieved from the server, $C_{srv} \leq \mu$. We refer to the first two metrics as the *performance* metrics, and the last two metrics as the *quality of service* metrics.

In the case of k-Anonymity, given that we transmit k bits for anonymizing one requested bit, $C_{comm} = 2 \cdot k$ (the client query and the server answer), $C_{comp} = k$ (server matching k bits), $P_{brh} = 1/k$ and $C_{srv} = k$. k-Anonymity can satisfy any privacy requirement and charge budget (ρ, μ) s.t. $\rho \cdot \mu \geq 1$. In the case of cPIR for retrieving one bit, the client query (row vector y) and the server answer (column vector z) are both vectors of $\lceil \sqrt{n} \rceil$ m-bit numbers, and m-bit modular multiplication is applied on all the data in M. Therefore, all the above metric values are fixed: $C_{comm} = m \cdot (t + s) = 2 \cdot m \cdot \lceil \sqrt{n} \rceil$, $C_{comp} = m \cdot n$, $P_{brh} = 1/(s \cdot t) \leq 1/n$ and $C_{srv} = s = \lceil \sqrt{n} \rceil$. As an example, the top left of Fig. 1 shows the calculated C_{comm} and C_{comp} for private retrieval of one bit (e.g. $M_{2,3}$) on a 4×4 matrix with $n = 16$ bits. cPIR can satisfy any privacy requirement and charge budget (ρ, μ) s.t. $\rho \geq 1/n$, $\mu \geq \lceil \sqrt{n} \rceil$.

5 Bounding-Box PIR

From the above analysis on cPIR and k-Anonymity, we can see that they are not flexible enough to satisfy any user desired quality of service, and they do not achieve overall good performance on all the metrics. A new practical approach for private retrieval of public data which achieves both user desired flexibility and overall good performance is thus needed. To design such an approach, we need the security and privacy of cPIR, as well as the flexibility and computation performance of k-Anonymity. On the other hand, we should reduce the

impractical costs of cPIR, and mitigate the security and proximity privacy threats of k-Anonymity. So we propose the following private retrieval approach called *Bounding-Box PIR* (bbPIR), which unifies and seeks a practical tradeoff between cPIR and k-Anonymity.

The basic idea of bbPIR is to use a bounding box BB (an $r \times c$ rectangle corresponding to a sub-matrix of M) as an anonymized range around the address of item x requested by the client, and then apply cPIR on the bounding box. bbPIR will find an appropriately sized bounding box that satisfies the privacy request ρ, and achieves overall good performance in terms of Communication and Computation Costs without exceeding the Server Charge limit μ for each retrieved item.

Since bbPIR operates on an $r \times c$ sub-matrix of M instead of the entire matrix M as in cPIR, its client query (row vector y) is a vector of c m-bit numbers, its server answer (column vector z) is a vector of r m-bit numbers, and m-bit modular multiplication is applied on all the data in the sub-matrix. Therefore, C_{comm}(bbPIR) is related to $m \cdot c$ and $m \cdot r$. C_{comp}(bbPIR) is proportional to the area of the bounding box $m \cdot r \cdot c$. P_{brh}(bbPIR) is equal to the ratio of one entry out of the bounding box $1/(r \cdot c)$. C_{srv}(bbPIR) is the number of rows in the sub-matrix r, because similar to cPIR, a client can interpret the data within the same column.

We proceed from the problem of private retrieval of one bit to private retrieval of one item. We start by only supporting *query by address* in Section 5.1 and Section 5.2, assuming that the client knows the exact address of the entry on M to be retrieved. Then for practical purposes, in Section 5.3 we relax this assumption and support *query by key* by using a public data histogram published by the server. In the following, we focus on private retrieval of one item, based on which more complex private queries can be supported.

5.1 Private Retrieval of One Bit

In private retrieval of one bit, M is a bit matrix, and we need an $r \times c$ bounding box BB around the queried bit entry (e, g). The client query is a single row vector y, and the server answer is a single column vector z. So the Communication Cost in this case is $C_{comm} = m \cdot (c + r)$. Since m-bit modular multiplication has to be applied to $r \cdot c$ bits in BB, the Computation Cost in this case is $C_{comp} = m \cdot r \cdot c$.

We have two constraints according to the client's requirement (ρ, μ): Privacy Breach Probability $P_{brh} \leq \rho$, and Server Charge $C_{srv} = r \leq \mu$. Choose BB to be the minimum bounding box that is sufficient to satisfy the privacy breach limit ρ. It is easy to see that its area $|BB| = r \cdot c = \lceil 1/\rho \rceil$, so the minimum Computation Cost is $C_{comp} = m \cdot r \cdot c = m \cdot \lceil 1/\rho \rceil$. Then the goal is to minimize the Communication Cost $C_{comm} = m \cdot (c + r)$ without exceeding the charge limit μ, which is equivalent to minimizing $c+r$. $min\,(c+r)$ is theoretically achieved when $c = r$. Given that $r \cdot c = \lceil 1/\rho \rceil$, $r = c = \lceil \sqrt{r \cdot c}\,\rceil = \lceil \sqrt{1/\rho}\,\rceil$. In our case because $r \leq \mu$ must hold, $min\,(c + r)$ depends on whether $\mu \geq \lceil \sqrt{1/\rho}\,\rceil$.

The protocol for private retrieval of one bit, which we call *bbPIR-1bit*, is described as follows:

1. Initially, the client sends to the server the size of the bounding box BB with area $\lceil 1/\rho \rceil$. The number of rows r and the number of columns s in the corresponding sub-matrix, BB, are decided as follows:

If $\mu \geq \lceil \sqrt{1/\rho} \rceil$, set

$$r = c = \lceil \sqrt{1/\rho} \rceil \tag{3}$$

Otherwise, set

$$r = min(\mu, \lceil 1/\rho \rceil, s),\ c = min(\lceil 1/(\rho \cdot r) \rceil, t) \tag{4}$$

2. To retrieve entry (e, g) in M, the client first places BB on M with the above defined dimensions r, c, s.t. BB covers (e, g), and BB is within the address space of M.
3. Then, the client generates a vector of c m-bit random numbers in Z_N^{+1}, $y = [y_1, ..., y_c]$, s.t. y_g is QNR and all other y_i ($i \neq g$) are QR. It sends the coordinates of BB and vector y to the server.
4. The server computes for each row i of the sub-matrix BB a modular product $z_i = \Pi_{j=1}^{c} w_{i,j}$, where $w_{i,j} = y_j^2$ if $M_{i,j}$ = 0, and $w_{i,j} = y_j$ if $M_{i,j}$ = 1.
5. The server sends to the client $z_1, ..., z_r$.
6. The client determines that $M_{e,g}$ = 0 if z_e is QR, and $M_{e,g}$ = 1 if z_e is QNR.

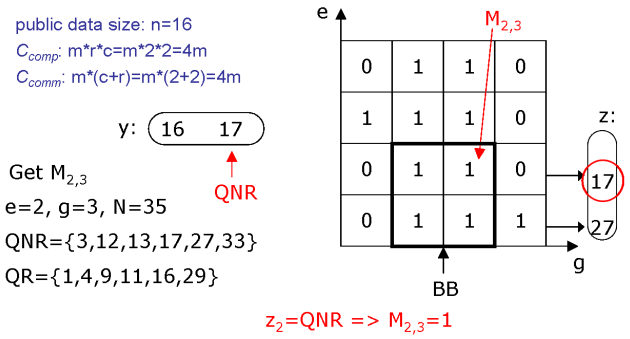


Fig. 2. *bb*PIR Example: Private Retrieval of One Bit

Fig. 2 illustrates the same example query as in Fig. 1. Suppose the client specifies $\rho = 1/4, \mu = 2$, a 2×2 bounding box is sufficient. Note that the placement of the bounding box BB has some flexibility. It does not need to be in the same position as in Fig. 2, as long as it covers $M_{2,3}$. Compared to Fig. 1, now the client only needs to generate a vector y of size 2, and the server only needs to do computations on 2 rows with one modular multiplication operation for each row. Because the sizes of vector y and vector z are reduced, the communication cost is also reduced proportionally.

By determining the dimensions r, c of BB in step 1 of *bbPIR-1bit*, *bb*PIR is able to satisfy any privacy requirement $\rho \geq 1/n$ and charge limit $\mu < s$. This is much more flexible than in the cases of k-Anonymity and *c*PIR. In step 2, we assume the client knows the coordinates of the boundary points on M, so is able to make sure BB is within the address space of M.

The comparisons of k-Anonymity, *c*PIR and *bb*PIR for private retrieval of one bit on the four metrics that we defined in Section 4 are shown in Table 2. We omit the constant cost of sending the size and coordinates of the bounding box in step 1 and step 2. Note that these

metric values, especially C_{comm} and C_{comp} are not exact, because different large constants could be involved in different data operations. The effects of these constants are captured in the experimental evaluation presented in Section 6.

Compared to k-Anonymity, *bb*PIR is able to achieve better privacy for the same charge or a lower charge for the same privacy. Compared to *c*PIR, generally if $\rho < 1/n$, $r \cdot c < n$, $c + r < 2 \cdot \lceil \sqrt{n} \rceil$, the communication cost, computation cost and charge of *bb*PIR are all lower than those of *c*PIR. However, if we make the bounding box a single column, i.e. $r = k$ and $c = 1$, there is no point in using the m-bit random number to hide the column g, and *bb*PIR degenerates into k-Anonymity. At the other extreme, if we set $r = c = \lceil \sqrt{n} \rceil$, *bb*PIR degenerates into *c*PIR.

Table 2. Comparisons on Private Retrieval of One Bit

Method	k-Anonymity	*c*PIR	*bb*PIR
C_{comm}	$2 \cdot k$	$2 \cdot m \cdot \lceil \sqrt{n} \rceil$	$m \cdot (c + r)$
C_{comp}	k	$m \cdot n$	$m \cdot r \cdot c$
P_{brh}	$1/k$	$1/n$	$1/(r \cdot c)$
C_{srv}	k	$\lceil \sqrt{n} \rceil$	r

5.2 Private Retrieval of One Item

Private retrieval of one item is similar to private retrieval of one bit. The difference is that the public data matrix M now is a binary matrix and the item in each entry has b bits. *bb*PIR works in a similar way as *c*PIR in that it retrieves one bit at one time. In order to retrieve a b bit item x, we can repeat *bb*PIR b times to get one bit at one time. However, the client query, row vector y, can be reused b times on b bits of x. Only the server answer, column vector z, needs to be re-calculated for each of the b bits. Therefore, the Communication Cost is $C_{comm} = m \cdot c + m \cdot b \cdot r$. Similarly, m-bit modular multiplication will be applied on each bit of the $r \cdot c$ items in BB, so the Computation Cost is $C_{comp} = m \cdot b \cdot r \cdot c$.

The two constraints from the client's requirement are same: Privacy Breach Probability $P_{brh} \leq \rho$, and Charge $C_{srv} = r \leq \mu$. Choose BB to be the minimum bounding box that is sufficient to satisfy the privacy breach limit ρ. So its area $|BB| = r \cdot c = \lceil 1/\rho \rceil$, and the minimum Computation Cost is $C_{comp} = m \cdot b \cdot r \cdot c = m \cdot b \cdot \lceil 1/\rho \rceil$. Then the goal is to minimize the Communication Cost $C_{comm} = m \cdot c + m \cdot b \cdot r$ without exceeding the charge limit μ, which is equivalent to minimizing $c + b \cdot r$. $min\,(c + b \cdot r)$ is achieved when $c = b \cdot r$. Given that $r \cdot c = \lceil 1/\rho \rceil$, $r = \lceil \sqrt{1/(\rho \cdot b)} \rceil$, $c = \lceil \sqrt{b/\rho} \rceil$. In our case because $r \leq \mu$ must hold, $min\,(c + b \cdot r)$ depends on whether $\mu \geq \lceil \sqrt{1/(\rho \cdot b)} \rceil$.

The protocol for private retrieval of one item, which we call *bbPIR-1item*, is described as follows:

1. Initially, the client sends to the server an m-bit number N which is the product of two random $m/2$-bit primes p_1 and p_2, and the size of the bounding box BB with area $\lceil 1/\rho \rceil$. The number of rows and columns, r and s in the bounding box BB are decided

as follows:
If $\mu \geq \lceil \sqrt{1/(\rho \cdot b)} \rceil$, set

$$r = \lceil \sqrt{1/(\rho \cdot b)} \rceil,\ c = \lceil \sqrt{b/\rho} \rceil \tag{5}$$

Otherwise, set

$$r = min(\mu, \lceil 1/\rho \rceil, s),\ c = min(\lceil 1/(\rho \cdot r) \rceil, t) \tag{6}$$

2. To retrieve entry (e, g) in M, the client first places BB on M with the above defined dimensions r, c, s.t. BB covers (e, g), and BB is within the address space of M.
3. Then, the client generates a vector of c m-bit random numbers in Z_N^{+1}, $y = [y_1, ..., y_c]$, s.t. y_g is QNR and all other y_i ($i \neq g$) are QR. It sends the coordinates of BB and vector y to the server.
4. The server computes for each row i of the sub-matrix BB a modular product $z_i = \Pi_{j=1}^{c} w_{i,j}$, where $w_{i,j} = y_j^2$ if $M_{i,j}$ = 0, and $w_{i,j} = y_j$ if $M_{i,j}$ = 1.
5. The server sends to the client $z_1, ..., z_r$.
6. The client determines that $M_{e,g}$ = 0 if z_e is QR, and $M_{e,g}$ = 1 if z_e is QNR.
7. Repeat step 4-6 to obtain the remaining $b - 1$ bits of the requested item in (e, g).

The comparisons of k-Anonymity, cPIR and bbPIR for private retrieval of one item on the performance and the quality of service metrics are shown in Table 3.

Table 3. Comparisons on Private Retrieval of One Item

Method	k-Anonymity	cPIR	bbPIR
C_{comm}	$2 \cdot b \cdot k$	$m \cdot \lceil \sqrt{n} \rceil + m \cdot b \cdot \lceil \sqrt{n} \rceil$	$m \cdot (c + b \cdot r)$
C_{comp}	$b \cdot k$	$m \cdot b \cdot n$	$m \cdot b \cdot r \cdot c$
P_{brh}	$1/k$	$1/n$	$1/(r \cdot c)$
C_{srv}	k	$\lceil \sqrt{n} \rceil$	r

5.3 Practical Considerations

In the above formulation, we assume that the client knows the exact address of the requested entry (e, g). However in practice, *query by key* is more common. In this case, the exact knowledge of how the public data is organized on the server is not available to the client. The client has to figure out the address of the requested item, (e, g), from the requested key.

In fact, the same problem also exists in cPIR. Unfortunately, it has largely been ignored by the PIR community. One proposal enabling *query by key*, builds an index structure for mapping a keyword to a physical address on the server [5]. Query privacy is achieved by an oblivious walk on the index structure. This oblivious walk requires the client to run as many as $O(b \cdot logn)$ rounds of PIR on the data bits dispersed in the matrix, and consequently incurs much higher communication and computation costs.

Although extra communication and computation costs are not avoidable for translating query keys to addresses, we would like an efficient and privacy-aware way of translation. One simple solution could be relying on a trusted third party who has access to both private

and public data, such as a trusted anonymizer in privacy-preserving location based services [17]. However, the client's query is disclosed to the third party, and her privacy is subject to the security of the third party. So we want to avoid such solution.

The basic idea of our solution is that we let the server publish a one-dimensional histogram, H, on the key field KA and the dimensions of the public data matrix M, s (number of rows) and t (number of columns). The histogram is only published to authorized clients. The publishing process, which occurs infrequently, could be encrypted for security. Then when a client issues a query, she calculates an address range for the queried entry by searching in which bin of H the key of the query data falls.

Assume a predefined threshold w, which is the minimum number of keys in each bin of the histogram. Consecutive keys are allocated in a bin. To make the address translation easier, we should match the bins with the partitioned address space of M. Right now we require $w \leq s$ for simplicity.

Once the bins are matched in H onto M, H can be transformed into an $\lfloor s/w \rfloor \times t$ matrix HM. An entry (e', g') on HM contains w keys if it corresponds to one of the first $\lfloor s/w \rfloor - 1$ bins in column g', or $s - w * (\lfloor s/w \rfloor - 1)$ keys if it corresponds to the last bin in column g'. Thus, H can be built greedily by checking whether $w * (\lfloor s/w \rfloor - 1)$ keys have been scanned in the current column on M. If it is true, assign the next $s - w * (\lfloor s/w \rfloor - 1)$ keys to a new bin and proceed to a new column. Otherwise, assign the next w keys to a new bin. For example, if we have 25 keys and a 5×5 matrix M, for $w = 2$, we assign the first two keys in one bin and the last three keys in another bin for each column from left to right, so the resulting HM is a 2×5 matrix with 10 bins, as illustrated in Fig. 3. This approach is similar to building an equi-depth histogram [10] (if $s \ mod \ w = 0$, H becomes an equi-depth histogram).

Knowing the organization of HM, the client is able to calculate the address of the requested entry on HM, (e', g'). The address range of the corresponding entries on M is $[(e' - 1) \cdot w + 1, e' \cdot w] \times [g', g']$ for $e' < \lfloor s/w \rfloor$, or $[(e' - 1) \cdot w + 1, s] \times [g', g']$ for $e' = \lfloor s/w \rfloor$. One advantage of $w \leq s$ is that we only need to run *bb*PIR once to obtain all the entries in the address range of the requested bin. As an example of query by key through the histogram in Fig. 3, if a client requests the item with key 53, she first finds 53 is in the 5th bin of H, corresponding to entry $HM_{1,3}$ on HM. Then she runs *bb*PIR once to obtain the entries in $HM_{1,3}$, where she finds the answer $M_{2,3}$.

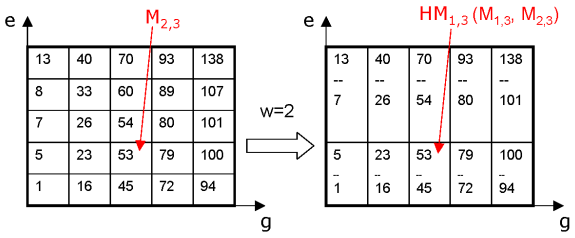


Fig. 3. Example of Locating By Histogram

Note that the number of rows in a bounding box, r, should be no less than w. *bb*PIR in query by key can satisfy any privacy and charge specification (ρ, μ) s.t. $\rho \geq 1/n$ and $\mu \geq w$.

6 Experimental Evaluation

Our experiments evaluate the performance and the quality of service of *bb*PIR against *c*PIR and k-Anonymity for private retrieval queries. Maintaining the overall proximity philosophy of privacy-preserving location based services [17], as well as the generalization approach of k-Anonymity based data publishing [20, 12], the k-Anonymity private retrieval method is implemented by sending a consecutive range of data items that covers the original private query item. This consecutive range has a concise and efficient representation, as it is specified by the lower end of the range and k. The *performance* and *quality of service* metrics are defined in Section 4. The results in different aspects (including different public data matrix shapes, different (ρ, μ) specifications, *query by address* and *query by key*) demonstrate that *bb*PIR is practical for safeguarding the client's query privacy while safeguarding the server's business revenue.

We implemented the three private retrieval methods in C++. For *bb*PIR, we use *bbPIR-1item* protocol. We ran a majority of the experiments on an extended data set generated from a real data set Adult [3] which is commonly used in the literature. The Adult database has 32561 records with 15 attributes of categorical or numeric data types. We take the first 3 attributes and generate 10^6 records by randomly picking attribute values from the attributes of the original 32561 records. Then the total number of data items, n, is 10^6, and the number of bits for each data item, b, is 208. Only for the experiment on proximity privacy of numeric data (Section 6.4), we use a synthetic data set with 10^6 numeric keys and values. All the default values in the experiments are listed in Table 4. For each value or range of a tested variant, we run 100 random queries and report the average metrics. The queries are *query by address* by default. In Section 6.5 we specifically study *query by key*. The testbed is a Linux server with Intel 2.40GHz CPU and 3GB memory, running Federal Core 8 OS.

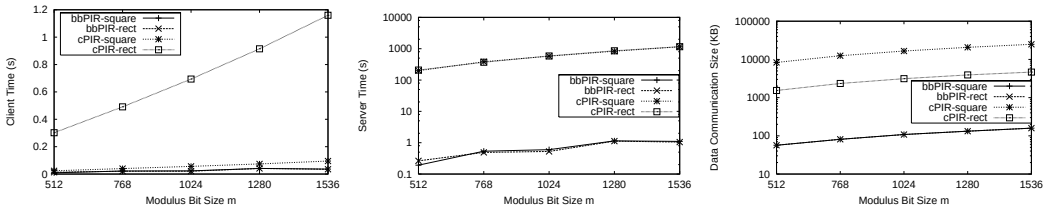
Table 4. Default Values in Experiments

Variant	Default Value
n	10^6
b	208
s, t, C_{srv}(*c*PIR)	10^3
k, C_{srv}(k-Anonymity)	10^3
P_{brh}(*c*PIR)	10^{-6}
ρ, P_{brh}(*bb*PIR), P_{brh}(k-Anonymity)	10^{-3}
μ, C_{srv}(*bb*PIR)	50
m	1024

6.1 Effects of Square Matrix vs. Rectangular Matrix

In previous sections, we assume that the public data matrix M is square, s.t. $s = t = \sqrt{n}$. It achieves the smallest communication cost $min(m \cdot (s + t)) = 2 \cdot m \cdot \sqrt{n}$ for *c*PIR when M is a bit matrix with $b = 1$. But when M is a binary data matrix with $b > 1$, $s = t = \sqrt{n}$ are not optimal values for minimizing the communication cost of *c*PIR $min(m \cdot (b \cdot s + t))$. The optimal values become

$$s = \lceil \sqrt{n/b} \rceil,\ t = \lceil \sqrt{b \cdot n} \rceil \tag{7}$$



(a) Comparison of Client Time (b) Comparison of Server Time (c) Comparison of Data Communication Size

Fig. 4. Effect of Data Matrix Shape, Vary Modulus Bit Size m, Fix $\rho = 10^{-3}$, $\mu = 50$

in which case M becomes a rectangular matrix with fewer rows and more columns.

To give cPIR that advantage and evaluate the effects of the data matrix shape on bbPIR, we compare the performance of cPIR and bbPIR on square and rectangular matrices (denoted as cPIR-square, cPIR-rect, bbPIR-square and bbPIR-rect in Fig. 4) while fixing $\rho = 10^{-3}$, $\mu = 50$ and varying the modulus bit size m from 512 to 1536 according to [19]. Fig. 4 shows the comparison results of client computation time, server computation time and data communication size. Privacy Breach Probabilities for cPIR and bbPIR in both matrix shapes are fixed to the default values 10^{-6} and 10^{-3} respectively as listed in Table 4. Server Charge for cPIR is equal to the number of rows in the matrix, s, which is 10^3 on square matrix and 70 on rectangular matrix. Server Charge for bbPIR is equal to the number of rows in the bounding box, r, which is 3 on both square matrix and on rectangular matrix. We can see from Fig. 4 that bbPIR is basicly not affected by the shape of the data matrix. cPIR takes the advantage of the rectangular matrix to reduce data communication size. However, it pays off by taking more time to generate the query vector y on the client side (see Fig. 4(a)). However, client time is almost negligible when compared against server time. This is the reason that we only count server time in the previous computation cost analysis.

Compared to cPIR, bbPIR achieves much better performance with a lower but acceptable privacy level. Especially the server computation time of bbPIR drops down dramatically and becomes promising for practical use (it is less than 1.15s in both matrix shapes).

6.2 Effects of Modulus Bit Size

After examining the effects of the data matrix shape on cPIR and bbPIR, we study the effects of the modulus bit size m on the performances of bbPIR, cPIR and k-Anonymity. From now on, we use a square matrix, because there is little difference between square matrix and rectangular matrix for bbPIR.

Recall that a potential security threat with k-Anonymity is that the data communication between the client and the server is in plain text and can be seen by a third party eavesdropper. In k-Anonymity, the client query, in plain text, is an address range of the anonymized entries, which is similar to the dimensions and coordinates of the bounding box on M in the case of bbPIR. This does not give a third party any useful information if the third party does not know M. But if the server answer is also in plain text, a third party will know

the exact result contents from the server to the client. To provide k-Anonymity the same security level as *bb*PIR, we apply a popular public key encryption algorithm, RSA, on the server answer of k-Anonymity.

We calculate the server computation and communication costs of the security enhanced k-Anonymity, which we denote as k-Anonymity (RSA) and abbreviate as k-A (RSA) in Fig. 5, by a theoretical analysis according to [4]. First, since RSA encryption and decryption is the dominant cost, we simplify the client computation as decryption of the server answer, and simplify the server computation as encryption of its answer. Note that RSA encryption and decryption are done by modular exponentiation functions $Ciphertext = Data^{public\_key} (mod\, N)$ and $Data = Ciphertext^{private\_key} (mod\, N)$ respectively, where $public\_key \cdot private\_key = O(N)$. A modular exponentiation can be transformed to a number of modular multiplications. To estimate RSA encryption and decryption time based on *bb*PIR server computation (modular multiplication) time, we use the same m-bit modulus N, and estimate both $public\_key$ and $private\_key$ as $m/2$-bit numbers for k-Anonymity (RSA). We take the minimum number of modular multiplications required for a modular exponentiation, $m/2 - 1$, in the binary method [4]. Since *bb*PIR needs $\lceil 1/\rho \rceil$ modular multiplications, we can calculate RSA encryption and decryption time as

$$T_{server}(bbPIR)/(\lceil 1/\rho \rceil) \cdot (m/2 - 1) \tag{8}$$

in which T_{server} (*bb*PIR) is the server computation time in *bb*PIR. Second, the cipher text size can be estimated as

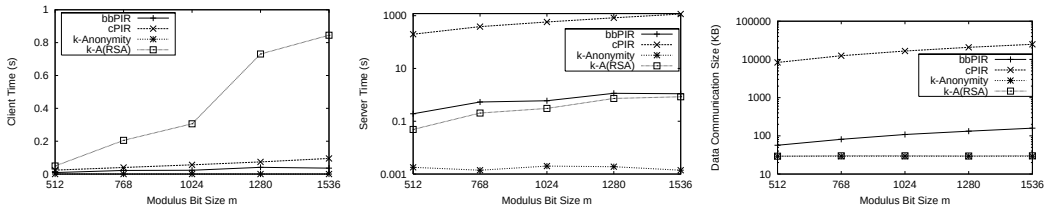
$$|Ciphertext| = |Data| + m - (Data\ mod\ m) \tag{9}$$

so the data communication size of k-Anonymity (RSA) can be estimated as

$$C_{comm}(k-Anonymity)/|Data| \cdot |Ciphertext| \tag{10}$$

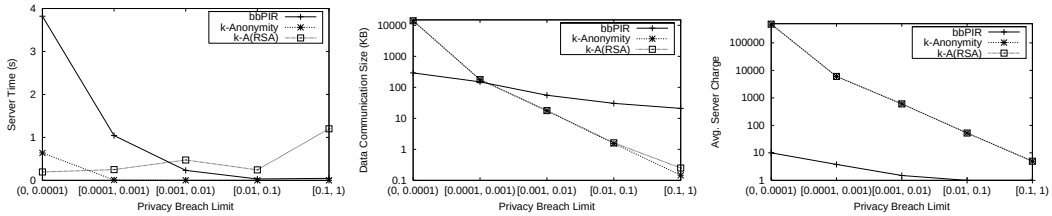
in which the operator $|\,|$ refers to the number of bits, and m is used as the block size of RSA.

We fix $\rho = 10^{-3}$, $\mu = 50$, and vary the modulus bit size m from 512 to 1536 according to [19]. Fig. 5 illustrates that a larger key size increases the computation time for *c*PIR, *bb*PIR and k-Anonymity (RSA), and the data communication size for *c*PIR and *bb*PIR. k-Anonymity, without RSA encryption, is definitely not affected by key size, thus its performance is clearly better than the other three methods at the loss of data communication security. k-Anonymity (RSA), with RSA encryption on the server and decryption on the client, takes more computation time on the client than *bb*PIR as seen in Fig. 5(a), and similar time on the server to *bb*PIR as seen in Fig. 5(b). Compared to the original k-Anonymity, we can conclude that if security is needed, additional costs are not avoidable and should be expected. But the cipher text in k-Anonymity (RSA) does not incur much additional data communication costs, as seen in Fig. 5(c). In contrast, *c*PIR and *bb*PIR generate a large random number for each bit, thus incur larger data communication costs. Note that the performances of *c*PIR and *bb*PIR can be further improved using the optimization in [8] to avoid redundant modular multiplications.



(a) Client Time (b) Server Time (c) Data Communication Size

Fig. 5. Comparison of *bb*PIR, *c*PIR, k-Anonymity and k-Anonymity(RSA), Vary Modulus Bit Size m, Fix $\rho = 10^{-3}$, $\mu = 50$, $k = 10^3$



(a) Server Time (b) Data Communication Size (c) Server Charge

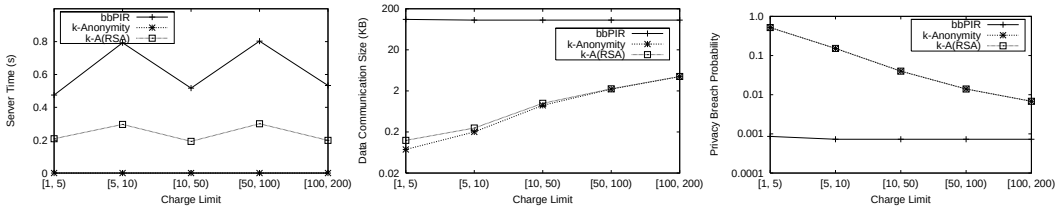
Fig. 6. Comparison of *bb*PIR, k-Anonymity and k-Anonymity(RSA), Vary ρ $(1/k)$, Fix $\mu = 50$

6.3 Effects of Privacy and Charge Specification

In the following two experiments, we study the effects of privacy breach limit ρ $(1/k$ in k-Anonymity) and charge limit μ (k in k-Anonymity) on *bb*PIR and k-Anonymity. We fix the modulus bit size $m = 1024$. We do not show the client computation time here, because it is almost negligible compared to server computation time.

We first fix $\mu = 50$, and vary ρ in 5 ranges, $(0, 10^{-4})$, $[10^{-4}, 10^{-3})$, $[10^{-3}, 10^{-2})$, $[10^{-2}, 10^{-1})$ and $[10^{-1}, 1)$. For each range, we mimic the requests from different clients by randomly generating 100 values of ρ in the corresponding range, and running 100 queries. We then take the average metric results. Fig. 6 demonstrates a general trend that a lower privacy requirement (higher ρ values) reduces both computation and communication costs for *bb*PIR and k-Anonymity. However, we can see in Fig. 6(a) that higher ρ values do not reduce the server computation time of k-Anonymity (RSA), because the values of ρ (corresponding to values of $k = 1/\rho$) do not impact the computational complexity of RSA encryption. For the same privacy breach limit, k-Anonymity usually needs more server charge than *bb*PIR as seen in Fig. 6(c), which is not appealing to most internet users even if it takes less computation and communication cost. k-Anonymity (RSA) has the exact same server charge as k-Anonymity, and almost does not incur additional communication cost, so the two curves of k-Anonymity (RSA) and k-Anonymity overlap in Fig. 6(b) and Fig. 6(c).

We then fix $\rho = 10^{-3}$, and vary μ in 5 ranges, $(1, 5)$, $[5, 10)$, $[10, 50)$, $[50, 100)$ and $[100, 200)$. Similarly to above, we generate 100 values of μ and run 100 queries in each range. In contrast to the effects of ρ, larger values of μ do not result in better performance



(a) Server Time (b) Data Communication Size (c) Privacy Breach Probability

Fig. 7. Comparison of *bb*PIR, k-Anonymity and k-Anonymity(RSA), Vary μ (k), Fix $\rho = 10^{-3}$

as seen in Fig. 7. The performance of *bb*PIR is quite stable in Fig. 7(b) and Fig. 7(c), because ρ is fixed, and according to formula 5 in Section 5.2, the bounding box is stable regardless of different charges. For the same charge limit, we can see in Fig. 7(c) that both k-Anonymity and k-Anonymity (RSA) cannot reach the same privacy as in *bb*PIR, and their real privacy breach probabilities are always larger than 10^{-3}. Similarly as the previous experiment on ρ, k-Anonymity (RSA) achieves the same privacy as k-Anonymity, and almost does not incur additional communication cost, so the two curves of k-Anonymity (RSA) and k-Anonymity overlap in Fig. 7(b) and Fig. 7(c).

6.4 Proximity Privacy of Numeric Data

In this experiment, we specifically study the proximity privacy of *bb*PIR and k-Anonymity on numeric data. As pointed out in [23, 13, 14], there should be enough difference between the data items in an anonymized range (in a bounding box in the case of *bb*PIR) under a privacy breach probability P_{brh}, which we call *neighborhood difference*, otherwise it can be concluded that the private data is within a very narrow range with probability P_{brh}.

Instead of using the Adult data set which is non-numeric, we generate a synthetic data set with 10^6 numeric data keys and values, both of which follow a Zipf distribution and are in the range of [0.0, 1.0]. We measure the neighborhood difference as the absolute difference between the maximum and minimum values in an anonymized range (or in a bounding box) following [23]. We then fix $\mu = 50$, and vary ρ from 10^{-4} to 10^{-1} (correspondingly set $k = 1/\rho$ and vary k from 10^4 to 10 in k-Anonymity). Since the bounding box used in *bb*PIR contains both the data items whose values are close to each other in a column and the data items whose values are far away in different columns of the matrix, the neighborhood difference in *bb*PIR is almost more than 100 times of the difference in k-Anonymity for $\rho < 0.1$ as seen in Fig. 8. The results suggest that k-Anonymity is vulnerable to proximity inference attack on numeric data.

An alternative k-Anonymity implementation by sending $k-1$ dummy retrieval requests is less likely to have the proximity privacy breach problem. However, it has two drawbacks compared to the consecutive range k-Anonymity implementation. First, it incurs more communication costs by sending the addresses of all k items in the case of *query by address*, and sending multiple separated address ranges for the dummy requests in the case of *query by key*. Second, it is not suitable for applications that do not want to lose all the proximity

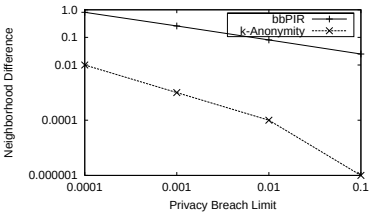


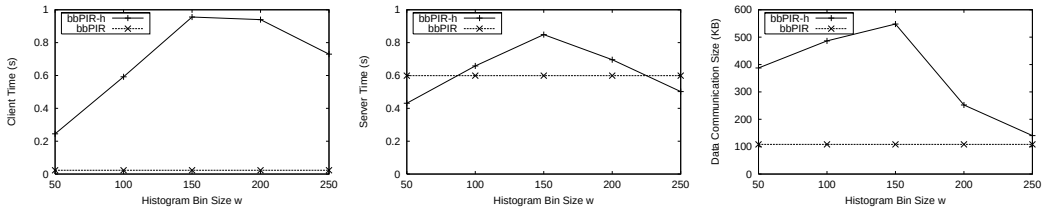
Fig. 8. Comparison of *bb*PIR and k-Anonymity, Vary Privacy Breach Limit ρ, Fix $\mu = 50$

information, e.g. in privacy-preserving location based services. In contrast, *bb*PIR does not suffer from proximity privacy breach as much as as k-Anonymity without losing all the proximity information.

6.5 Effects of Query by Key

Finally we study the costs of *query by key* compared to the costs of *query by address* in *bb*PIR. As we discuss in Section 5.3, clients have to calculate an address range of the requested data key internally, and then retrieve all the data items whose keys fall in that range. Note that interpreting all these items could lead to additional costs on the client, so we need to specifically study the client computation time, which was almost negligible before, but can not be ignored here.

The size of the requested address range is determined by the granularity of the server published histogram, the minimum number of keys in each bin, w. We vary w from 50 to 250, set $\mu = w$ (since $\mu \geq w$ must hold) for *query by key*, fix $\mu = 50$ for *query by address*, and fix $\rho = 10^{-3}$ for both *query by key* and *query by address*. We use the default Adult data set as the public data and generate numeric keys for each record. The comparison result of *query by key* and *query by address*, denoted as *bb*PIR-h and *bb*PIR respectively, is shown in Fig. 9. It is interesting to note that the computation and communication costs of query by key are not monotone functions of w, as we can see in Fig. 9. The reason is the following. Since ρ is fixed, the area of the bounding box is fixed. As w increases, the number of rows in the bounding box increases, and in contrast, the number of columns in the bounding box decreases. Recall that client computation includes both one time query vector generation and b times result interpretation. As w increases, the client has to pay more charges for the private retrieval service, and consequently the client has to interpret more result items. Since this interpretation is for each of b bits, the client time of *bb*PIR-h increases first in Fig. 9(a). However, the client only needs to generate a smaller size of query vector, and this advantage seems to counteract the larger amounts of work on result interpretation for $w > 150$ in Fig. 9(a). Similarly in Fig. 9(b), the server time of *bb*PIR is also affected by two contrary factors. With the increase of w, the server needs to do less modular multiplications in one row, but has to compute on a larger number of rows. To explain Fig. 9(c), the communication cost consists of one client query vector and b server answer vectors, and a larger w leads to a smaller size of client query, but larger



(a) Client Time (b) Server Time (c) Data Communication Size

Fig. 9. Comparison of query-by-key (*bb*PIR-h) and query-by-address (*bb*PIR), Vary Histogram Granularity w, Fix $\rho = 10^{-3}$

sizes of server answers. Server answers account for larger amounts of data communication at first, but then the smaller size of client query counteracts them and brings down the data communication for $w > 150$. However, consistently query by key increases the computation and communication costs. These overheads are still reasonable, as query by key provides a practical solution to the impractical assumption of cPIR, i.e., that the client knows a priori the exact location of the requested data item.

7 Conclusion

Enabling practical private retrieval of public data is useful for privacy aware internet services, but has not received much attention in the database community. Computational Private Information Retrieval (cPIR) achieves complete privacy for a client, but is impractical because of its expensive computations involving the entire public service data. On the other hand, k-Anonymity based private retrieval approach achieves cheap computation and communication, but is subject to the threats of proximity breach and insecure communication, as well as inflexibility between privacy and charge constraints.

In designing a practical approach for private retrieval of public data on single server settings, we follow the cPIR approach to achieve privacy and security, and adopt the principle of flexible privacy from k-Anonymity. We call our proposed approach *Bounding-Box PIR* (*bb*PIR). *bb*PIR generalizes cPIR by adjusting a bounding box which trades complete privacy for flexible partial privacy, but bounds computation and communication costs. Given an internet business service model where clients can specify their privacy requirements and service charge budgets (ρ, μ), *bb*PIR is able to achieve lower charge or higher privacy compared to k-Anonymity. We also design a practical low cost solution for enabling the retrieval by keys instead of retrieval by addresses of the matrix. The experimental results confirm the efficiency and effectiveness of our proposals.

8 Acknowledgement

We would like to thank Gabriel Ghinita for providing us his basic PIR implementation on location based service.

References

[1] R. Agrawal, A. V. Evfimievski, and R. Srikant. Information sharing across private databases. In *SIGMOD Conference*, pages 86–97, 2003.

[2] N. Anciaux, M. Benzine, L. Bouganim, P. Pucheral, and D. Shasha. Ghostdb: querying visible and hidden data without leaks. In *SIGMOD Conference*, pages 677–688, 2007.

[3] A. Asuncion and D. Newman. UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.

[4] Çetin Kaya Koç. High-speed RSA implementation. Technical Report TR 201, RSA Laboratories, 1994.

[5] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. Technical Report TRCS 0917, Department of Computer Science, Technian, 1997.

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

[7] C. Clifton, M. Kantarcioglu, A. Doan, G. Schadow, J. Vaidya, A. K. Elmagarmid, and D. Suciu. Privacy-preserving data integration and sharing. In *DMKD*, pages 19–26, 2004.

[8] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD Conference*, pages 121–132, 2008.

[9] http://marauder.millersville.edu/∼bikenaga/numbertheory/numbertheorynotes.html. Number theory notes.

[10] Y. Ioannidis. The history of histograms (abridged). In *VLDB*, pages 19–30, 2003.

[11] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997.

[12] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *SIGMOD Conference*, pages 49–60, 2005.

[13] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization. In *KDD*, pages 277–286, 2006.

[14] J. Li, Y. Tao, and X. Xiao. Preservation of proximity privacy in publishing numerical sensitive data. In *SIGMOD Conference*, pages 473–486, 2008.

[15] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, pages 106–115, 2007.

[16] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.

[17] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: A privacy-aware location-based database server. In *ICDE*, pages 1499–1500, 2007.

[18] R. Ostrovsky and W. E. S. III. A survey of single-database private information retrieval: Techniques and applications. In *Public Key Cryptography*, pages 393–411, 2007.

[19] R. Sion and B. Carbunar. On the computational practicality of private information retrieval. In *Network and Distributed System Security Symposium*, 2007.

[20] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):571–588, 2002.

[21] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[22] P. Williams and R. Sion. Usable private information retrieval. In *Network and Distributed System Security Symposium*, 2008.

[23] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. Aggregate query answering on anonymized tables. In *ICDE*, pages 116–125, 2007.