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 query to the server. Computational Private Information Retrieval (*c*PIR) achieves complete privacy for clients, 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. We propose a practical and flexible approach for the private retrieval of public data called *Bounding-Box* PIR (*bb*PIR). Using *bb*PIR, a client specifies both privacy requirements and a service charge budget. The server satisfies the client's requirements, and achieves overall good performance in computation and communication. *bb*PIR generalizes *c*PIR 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 efficiency of *bb*PIR compared to *c*PIR and the effectiveness of *bb*PIR compared to *k*-Anonymity are verified in extensive experimental evaluations.

1 Introduction

We consider a special query called *private retrieval of public data*, in which a client retrieves data from a public server using some of its private data as predicates, while not revealing the exact values of the private data in the query. A typical example is privacy-preserving location based services [1,2], 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. For example, a researcher with a potentially new idea does not want to reveal her exact idea to Google when she is searching for "prior art".

Currently, the privacy of queries is not properly protected 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 client privacy. Therefore, we consider a service model, in which the server can charge the client

S. Foresti and S. Jajodia (Eds.): Data and Applications Security XXIV, LNCS 6166, pp. 1–16, 2010.
© IFIP International Federation for Information Processing 2010

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 and is generally larger than the size of the answer to the query.

To enable practical query privacy in the above 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 as well as the *computation* overhead. Given that queries may be issued from client devices with limited capabilities, the solution should not impose sophisticated requirements on the client.
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* [3] and *Computational Private Information Retrieval* (*c*PIR) [4]. *k*-Anonymity has been used in privacy-preserving location based services [1], where the location point of a user is blurred into a cloaked region consisting of at least k nearby user locations and the server returns the nearest points of interests to the cloaked region. The parameter k serves as a configurable degree of privacy. Similarly in the more general setting of private retrieval, one could insert into the private query 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 text that can be seen by a third party. The privacy of *k*-Anonymity for numeric data has also been questioned by a number of proposals [5,6,7] 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 (*c*PIR) [8] 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, the *c*PIR 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. Due to its expensive computation costs on the server, even the *c*PIR technique with the least expensive operation, modular multiplication [8], is criticized as being up to two orders of magnitude less efficient than simply transferring the entire data from the server to the client [9].

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 (*bb*PIR) 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 rectangle called *bounding box*, whose range corresponds to a sub matrix of the public data matrix. The size of the bounding box is determined 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 costs, and vice versa.

Compared to k-Anonymity, *bb*PIR 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, *bb*PIR does not suffer from proximity breach as much as k-Anonymity, because the bounding box includes data values that are not close to the query value. Compared to cPIR, *bb*PIR is more practical because of its lower computation cost. At one extreme, *bb*PIR degenerates into k-Anonymity if the range of the bounding box is a single column on the public data matrix. At the other extreme, *bb*PIR becomes cPIR if the range of the bounding box is the entire public data 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 *bb*PIR approach. Section 6 experimentally evaluates *bb*PIR. 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 location based services.

Private Information Retrieval (PIR) derives from the following theoretical problem: 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. [10] first introduced the PIR problem and provided solutions for multiple database servers. Observing that a database server is often restricted to perform only polynomial-time computations, Kushilevitz and Ostrovsky proposed a single database, computational PIR solution [8], which we refer to as cPIR in the paper. Follow-up single database cPIR proposals improve the communication overhead, as surveyed in [4], but they use even more expensive operations than the modular multiplications used in [8], as pointed out by Sion and Carbunar [9], thus are not feasible for practical applications. Williams and Sion [11] attempt to make cPIR practical by using oblivious RAM. However, their approach is designed for 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 [3]. 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 designed new privacy principles to enhance the privacy of k-Anonymity [12,13,5,6,7]. However, they are not easy to apply in the private retrieval of public data applications. Besides, they share the same security threat as k-Anonymity: all the communication contents can be seen by a third party.

Mokbel et al. [1] use k-Anonymity to implement query privacy in location based services, in which the anonymization of user location points is done on a third party server. Ghinita et al. [2] argue that a third party anonymizer is not needed if cPIR is used. Our bbPIR can also be applied in privacy-preserving location based services. It incurs less costs than cPIR and does not need a third party anonymizer.

3 Background on *c*PIR

cPIR is designed to retrieve a single bit in a large matrix privately [8]. 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.* $y^2 = x \ mod \ N$. Otherwise, x is a *quadratic nonresidue* (QNR) mod N [14]. The problem becomes 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) states 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 [8].

However, determining whether the number x is a QR or a QNR is much easier if knowing p_1 and p_2. Based on *Euler's theorem* [14], x is a QR if and only if

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

and a 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 cost between the client and the server). Let (e, g) be the two dimensional address of the bit queried by the client (Refer to Table 1 for a summary of our notations). The cPIR 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 a QR, and $M_{e,g} = 1$ if z_e is a QNR.

For example in Fig. 1, the client sends $N = 35$ 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, where y_3 is a QNR 17, and $y_1 = 4$, $y_2 = 16$, $y_4 = 11$ are QR. Upon receiving y, the server computes for each row of the matrix a modular product z_i, e.g. $z_2 = (4^2 \times 16 \times 17 \times 11^2) \bmod 35 = 17$. Since $z_2 = 17$ is a QNR, when the client receives the vector z from the server, she obtains $M_{2,3} = 1$.

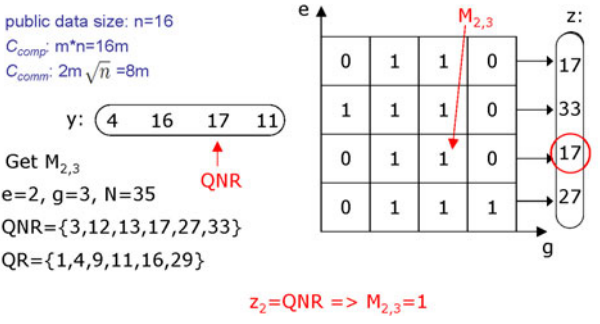


Fig. 1. cPIR Example

Note that the server can not figure out if a y_i or z_i is a QR or a 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 every $M_{i,g}$ $(1 \leq i \leq s)$ by analyzing the corresponding z_i. Thus 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
$N = p_1 \cdot p_2$	modulus, product of two $m/2$-bit primes p_1 and p_2
s, t	number of rows, columns in the 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 answer 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. The public data are sorted by KA in ascending order and then 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 retrieval.

A client can specify her privacy requirement and 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 Sect. 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), $C_{comp} = k$, $P_{brh} = 1/k$ 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 to retrieve 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 the 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 the 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 threats of plain text communication and proximity privacy breach of k-Anonymity. Hence, we propose a 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 finds 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, $C_{comm}(bb\text{PIR})$ is proportional to $m \cdot c$ and $m \cdot r$. $C_{comp}(bb\text{PIR})$ is proportional to the area of the bounding box, $m \cdot r \cdot c$. $P_{brh}(bb\text{PIR})$ equals the ratio of one entry out of the bounding box, $1/(r \cdot c)$. $C_{srv}(bb\text{PIR})$ 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 start by supporting *query by address* in Sect. 5.1, assuming that the client knows the exact address of the entry on M to retrieve. Then for practical purposes, in Sect. 5.2 we relax this assumption and support *query by key* by using a public data histogram published by the server. We focus on private retrieval of one item, based on which more complex private queries can be supported.

5.1 Query by Address

bbPIR is similar to cPIR in that it retrieves one bit at a time. In order to retrieve a b bit item x, bbPIR can be repeated b times. 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 $C_{comm} = m \cdot c + m \cdot b \cdot r$. Since m-bit modular multiplication will be applied on each bit of the $r \cdot c$ items in the bounding box, the Computation Cost $C_{comp} = m \cdot b \cdot r \cdot c$.

We have two constraints based on the client's requirement (ρ, μ): $P_{brh} \leq \rho$, and $C_{srv} = r \leq \mu$. Choose BB to be the minimum bounding box that satisfies the

privacy breach limit ρ. It is easy to see that its area $|BB| = r \cdot c = \lceil 1/\rho \rceil$, and the minimum Computation Cost $C_{comp} = m \cdot b \cdot r \cdot c = m \cdot b \cdot \lceil 1/\rho \rceil$. 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 $(c + b \cdot r)$. Because $min\,(c + b \cdot r)$ is achieved when $c = b \cdot r$, given that $r \cdot c = \lceil 1/\rho \rceil$, $r = \lceil \sqrt{1/(\rho \cdot b)}\, \rceil$, $c = \lceil \sqrt{b/\rho}\, \rceil$. Since $r \leq \mu$ must hold, $min\,(c + b \cdot r)$ depends on whether $\mu \geq \lceil \sqrt{1/(\rho \cdot b)}\, \rceil$. The *bb*PIR protocol 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 dimensions of the bounding box BB of 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
   $$r = \lceil \sqrt{1/(\rho \cdot b)}\, \rceil,\ c = \lceil \sqrt{b/\rho}\, \rceil \tag{2}$$
   Otherwise, set
   $$r = min(\mu, \lceil 1/\rho \rceil, s),\ c = min(\lceil 1/(\rho \cdot r) \rceil, t) \tag{3}$$
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. 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 a 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 a QR, and $M_{e,g} = 1$ if z_e is a QNR.
7. Repeat steps 4-6 to obtain the remaining bits of the requested item in (e, g).

Figure 2 illustrates the same example query as in Fig. 1, retrieving $M_{2,3}$ from a 4×4 bit matrix M. Suppose a client specifies $\rho = 1/4, \mu = 2$, then a 2×2 bounding box suffices to satisfy her requirements. The placement of the bounding box BB is flexible, as long as it covers $M_{2,3}$. Compared to Fig. 1, because the sizes of vectors y and z are reduced, the computation and communication costs are reduced proportionally.

A comparison of k-Anonymity, cPIR and *bb*PIR for the private retrieval of one item on the performance and the quality of service metrics are shown in Table 2. We omit the constant cost for sending N, the size and coordinates of the bounding box in step 1 and step 2.

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 cPIR, 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 cPIR. If we make the bounding

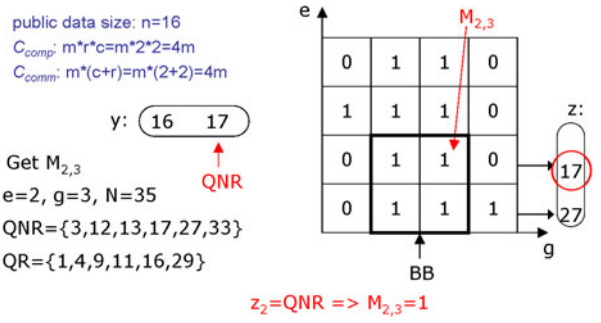


Fig. 2. *bb*PIR Example: Private Retrieval of One Bit

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. If we set $r = c = \lceil \sqrt{n} \rceil$, *bb*PIR degenerates into cPIR. By determining the dimensions r, c of BB in step 1, *bb*PIR is able to satisfy any privacy requirement $\rho \geq 1/n$ and charge limit $\mu < s$. It is thus much more flexible than k-Anonymity and cPIR.

Table 2. Comparisons on Private Retrieval of One Item

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

5.2 Query by Key

In the above formulation, we assume that clients know 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. Clients have 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 enables *query by key* by building an index structure for mapping a keyword to a physical address on the server and processing a query by an oblivious walk on the index [15]. This oblivious walk requires running as many as $O(b \cdot logn)$ rounds of PIR, and consequently incurs high 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. We also want to avoid trusting a third party.

The basic idea for our solution is that the server publishes a one-dimensional histogram, H, on the key field KA and the dimensions of the public data matrix M, s and t. The histogram is only published to authorized clients. The publishing process, which occurs infrequently, is encrypted for security. When a client issues a query, she calculates an address range for the queried entry by searching the bin of H where the query data falls.

Assume a predefined threshold w, which is the minimum number of keys in each bin of the histogram. To simplify address translation, we require $w \leq s$. Consecutive keys are allocated in the bins of H by scanning M columnwise from left to right. If $w \cdot (\lfloor s/w \rfloor - 1)$ keys have been scanned in the current column on M, assign the next $s - w \cdot (\lfloor s/w \rfloor - 1)$ keys to a new bin and proceed to a new column. Otherwise, assign the next w keys to a new bin. At the end of this process, the bins in H are matched onto M, and H is transformed into an $\lfloor s/w \rfloor \times t$ matrix, HM. 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. The result, HM, is a 2×5 matrix with 10 bins, as illustrated in Fig. 3.

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 bbPIR 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 in the 5th bin of H, corresponding to entry $HM_{1,3}$ on HM. Then she runs bbPIR once to obtain the entries in $HM_{1,3}$, where she finds the answer $M_{2,3}$.

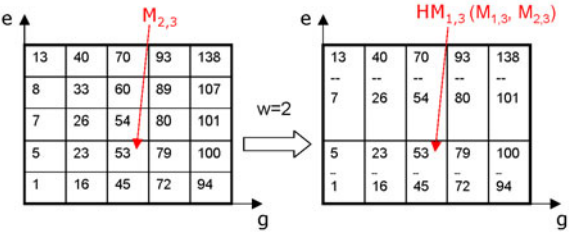


Fig. 3. Example of Locating By Histogram

Note that $r \geq w$ must hold. bbPIR 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* (metrics defined in Sect. 4) of bbPIR against cPIR and k-Anonymity for private retrieval queries. Maintaining the overall proximity philosophy of privacy-preserving location based services [1], as well as the generalization approach of k-Anonymity based data publishing [16,17], the k-Anonymity private retrieval method is implemented by sending a consecutive range of data items that covers the original private query item, which are specified by the lower end of the range and k.

We implemented the three private retrieval methods in C++. We ran a majority of the experiments on an extended data set generated from a real data set, Adult [18]. The Adult database has 32561 records with 15 attributes of categorical or numeric data types. We kept its first 3 attributes and generated 10^6 records by randomly picking attribute values from 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 (Sect. 6.2), we used a synthetic data set of size 10^6. All the default values in the experiments are listed in Table 3. For each value or range of a variant to test, we ran 100 random queries and reported the average results. The queries are *query by address* by default. In Sect. 6.3 we specifically study *query by key*. Our testbed is a Linux server with Intel 2.40GHz CPU and 3GB memory, running Federal Core 8 OS. The experiment results demonstrate that *bb*PIR is practical for safeguarding client query privacy as well as the server's business revenue.

Table 3. Default Values in Experiments

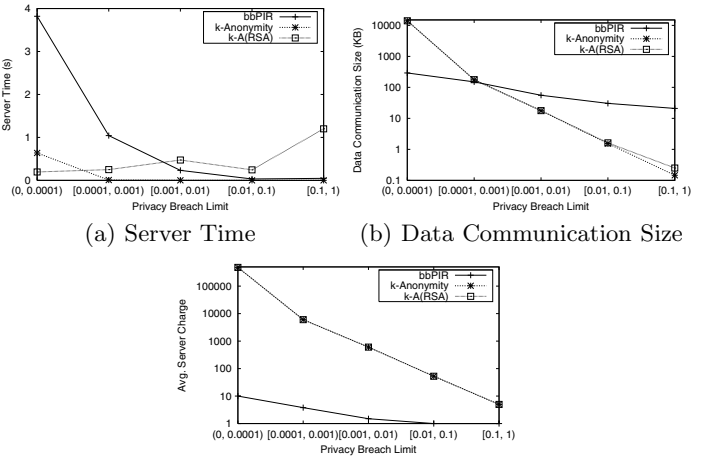
Variant	Default Value
n	10^6
b	208
s, t, C_{srv}(cPIR)	10^3
k, C_{srv}(k-Anonymity)	10^3
P_{brh}(cPIR)	10^{-6}
ρ, P_{brh}(*bb*PIR), P_{brh}(k-Anonymity)	10^{-3}
μ, C_{srv}(*bb*PIR)	50
m	1024

6.1 Effects of Privacy and Charge Specification

In the following two experiments, we study the effects of the privacy breach limit, ρ ($1/k$ in k-Anonymity), and the charge limit, μ (k in k-Anonymity), on *bb*PIR and k-Anonymity. We do not show the client computation times here, since they are almost negligible compared to server computation times.

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 does not give a third party any useful information without knowing M. But if the server answer is also in plain text, a third party will know the exact result contents sent by the server. Thus to provide k-Anonymity the same security level as *bb*PIR, we applied a popular public key encryption algorithm, RSA, on the server answer of k-Anonymity. We denote the security enhanced k-Anonymity as k-Anonymity (RSA) and abbreviate it as k-A (RSA). Please refer to our technical report [19] for more details on the analysis of its computation and communication costs.

In the first experiment, we fixed $\mu = 50$, and varied ρ ($1/k$) 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,



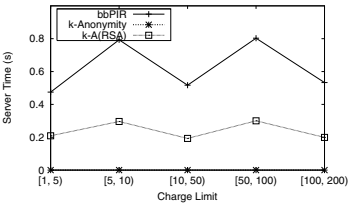
(a) Server Time (b) Data Communication Size

(c) Server Charge

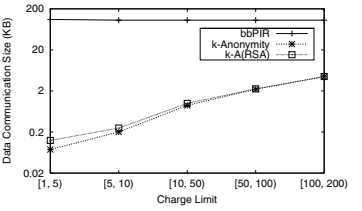
Fig. 4. Comparison of *bb*PIR, k-Anonymity and k-A (RSA). Vary ρ $(1/k)$, and Fix $\mu = 50$.

we mimiced requests from different clients by randomly generating 100 values of ρ in the range, and running 100 queries. We then took the average results. Figure 4 demonstrates a general trend that a lower privacy requirement (a higher ρ and a correspondingly lower k value) reduces both computation and communication costs for *bb*PIR and k-Anonymity. However, as demonstrated in Fig. 4(a), higher ρ (and corresponding lower k) values do not reduce the server computation time of k-A (RSA), because they do not impact the computational complexity of RSA encryption. For the same privacy breach limit, k-Anonymity usually incurs more server charges than *bb*PIR as seen in Fig. 4(c), which is not appealing to most internet users. 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-A (RSA) and k-Anonymity overlap in Fig. 4(b) and Fig. 4(c).

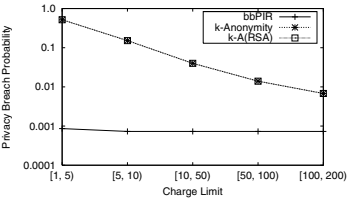
In the second experiment, we fixed $\rho = 10^{-3}$, and varied μ (k) in 5 ranges, $(1, 5)$, $[5, 10)$, $[10, 50)$, $[50, 100)$ and $[100, 200)$. Similar to the above, we generated 100 values of μ and ran 100 queries in each range. In contrast to the effects of ρ, larger values of μ do not result in better performance, as seen in Fig. 5. The performance of *bb*PIR remains constant in Fig. 5(b) and Fig. 5(c), because ρ is fixed, according to Formula (2) in Sect. 5.1, the dimensions of the bounding box are fixed regardless of different charges. For the same charge limit, we can see in Fig. 5(c) that both k-Anonymity and k-A (RSA) cannot reach the same privacy as in *bb*PIR, and their real privacy breach probabilities $P_{brh} > 10^{-3}$. Similar to the previous experiment on ρ, k-A (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. 5(b) and Fig. 5(c).



(a) Server Time



(b) Data Communication Size



(c) Privacy Breach Probability

Fig. 5. Comparison of *bb*PIR, k-Anonymity and k-A (RSA). Vary μ (k), and Fix $\rho = 10^{-3}$.

6.2 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 [5,6,7], 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 the private data can be determined in a narrow range with probability P_{brh}.

Instead of using the non-numeric Adult data set, we generated a synthetic data set with 10^6 numeric data keys and values, which follow a Zipf distribution and are in the range of [0.0, 1.0]. We measured the neighborhood difference as the absolute difference between the maximum and minimum values in an anonymized range (or in a bounding box) following [5]. We fixed $\mu = 50$, and varied ρ ($1/k$) from 10^{-4} to 10^{-1}. 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 the difference in k-Anonymity for $\rho < 0.1$ as seen in Fig. 6. The results suggest that *bb*PIR is much more resistant to proximity inference attack on numeric data than k-Anonymity. Our technical report [19] also discusses the tradeoff of an alternative k-Anonymity implementation by sending $k - 1$ dummy retrieval requests.

6.3 Effects of Query by Key

Finally we study the costs of *query by key* (QBK) compared to the costs of *query by address* (QBA) in *bb*PIR. As discussed in Sect. 5.2, clients have to calculate

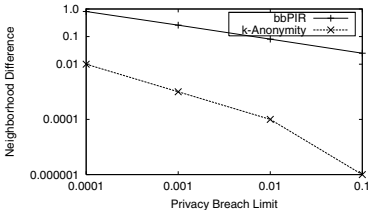
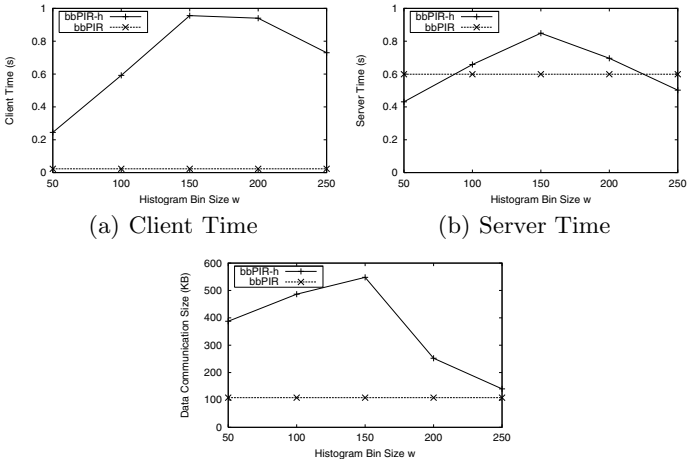


Fig. 6. Comparison of *bb*PIR and k-Anonymity. Vary ρ, and Fix $\mu = 50$.



(a) Client Time (b) Server Time

(c) Data Communication Size

Fig. 7. Comparison of QBK (*bb*PIR-h) and QBA (*bb*PIR). Vary Histogram Bin Size w, and Fix $\rho = 10^{-3}$.

an address range of the requested key, and retrieve all the data items whose keys fall in that range. Interpreting all these items could lead to additional costs on the client, so we need to specifically study the client computation times here.

The size of the requested address range is determined by the minimum number of keys in each bin of the server published histogram, w. We varied w from 50 to 250, set $\mu = w$ (since $\mu \geq w$ must hold) for QBK, fixed $\mu = 50$ for QBA, and fixed $\rho = 10^{-3}$ for both query types. We used the extended Adult data set as public data and generated numeric keys for each record. The comparison result of QBK and QBA, denoted as *bb*PIR-h and *bb*PIR respectively, is shown in Fig. 7. For Fig. 7, it is interesting to note that the computation and communication costs of QBK are not monotone functions of w. The reason for this behavior can be explained as follows. Since ρ is fixed, the area of the bounding box is fixed. As w increases, the number of rows in the bounding box, r, increases, and contrarily, the number of columns in the bounding box, c, decreases. The related computation and communication costs change with both r and c, so

they increase and then decrease as in Fig. 7. More details are explained in [19]. However, consistently QBK 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 items.

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 deemed impractical due to its expensive computations involving the entire public data. On the other hand, k-Anonymity based private retrieval 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.

To design a practical approach for private retrieval of public data on single server settings, we followed the cPIR approach to achieve privacy and security, and adopted the principle of flexible privacy from k-Anonymity. We proposed an approach called *Bounding-Box* PIR (bbPIR). bbPIR 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), bbPIR is able to achieve lower charges or higher privacy compared to k-Anonymity. We also designed a practical low cost solution for enabling retrieval by keys instead of retrieval by addresses of the matrix. The experimental results confirmed the efficiency and effectiveness of our proposals.

Acknowledgement

This work is partially supported by NSF Grant IIS-0847925. We wish to thank Gabriel Ghinita for providing us his cPIR implementation.

References

1. Mokbel, M.F., Chow, C.Y., Aref, W.G.: The new casper: A privacy-aware location-based database server. In: ICDE, pp. 1499–1500 (2007)
2. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., Tan, K.L.: Private queries in location based services: anonymizers are not necessary. In: SIGMOD Conference, pp. 121–132 (2008)
3. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5), 557–570 (2002)
4. Ostrovsky, R., Skeith III, W.E.: A survey of single-database private information retrieval: Techniques and applications. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 393–411. Springer, Heidelberg (2007)

5. Zhang, Q., Koudas, N., Srivastava, D., Yu, T.: Aggregate query answering on anonymized tables. In: ICDE, pp. 116–125 (2007)
6. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Workload-aware anonymization. In: KDD, pp. 277–286 (2006)
7. Li, J., Tao, Y., Xiao, X.: Preservation of proximity privacy in publishing numerical sensitive data. In: SIGMOD Conference, pp. 473–486 (2008)
8. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: FOCS, pp. 364–373 (1997)
9. Sion, R., Carbunar, B.: On the computational practicality of private information retrieval. In: Network and Distributed System Security Symposium (2007)
10. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM 45(6), 965–981 (1998)
11. Williams, P., Sion, R.: Usable private information retrieval. In: Network and Distributed System Security Symposium (2008)
12. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. In: ICDE 24 (2006)
13. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: ICDE, pp. 106–115 (2007)
14. `http://marauder.millersville.edu/~bikenaga/numbertheory/numbertheorynotes.html`: Number theory notes
15. Chor, B., Gilboa, N., Naor, M.: Private information retrieval by keywords. Technical Report TRCS 0917, Department of Computer Science, Technian (1997)
16. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5), 571–588 (2002)
17. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: SIGMOD Conference, pp. 49–60 (2005)
18. Asuncion, A., Newman, D.: UCI machine learning repository (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
19. Wang, S., Agrawal, D., Abbadi, A.E.: Generalizing pir for practical private retrieval of public data. Technical Report 2009-16, Department of Computer Science, UCSB (2009)