

Generalizing PIR for Practical Private Retrieval of Public Data

Shiyuan Wang
Department of Computer
Science, UC Santa Barbara
sywang@cs.ucsb.edu

Divyakant Agrawal
Department of Computer
Science, UC Santa Barbara
agrawal@cs.ucsb.edu

Amr El Abbadi
Department of Computer
Science, UC Santa Barbara
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 used 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 executes queries on the data of a public server with its private data as the filtering condition, while not revealing the exact values of the private data in the query. One example of such queries is selection on public data based on a condition such as a keyword or an index provided by the client. We make three reasonable assumptions in this context:

1. The size of the private data is several orders of magnitude smaller than the size of the public data.
2. The server is honest but curious, meaning that it may guess the values of the private data from the query access patterns.
3. We assume 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 retrieved data depends on the private retrieval protocol for privacy purposes and is generally larger than the size of the answer to the query.

From Assumptions (1) and (3), we can infer that a solution in which the client has to retrieve the entire public data is

expensive and infeasible. From Assumption (2), we contend that the query privacy of the client should be protected by minimizing the probability of correct guesses made by the server.

A typical example of private retrieval of public data is privacy-preserving location based services [16, 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 subscriptions through big internet information service providers such as Google, Yahoo, and Microsoft. On one hand, users need these public services in their daily lives. On the other hand, users are concerned that their private profile data or their personal tastes might be disclosed or compromised through analysis or inferences.

To apply private retrieval of public data in more general applications, we have the following desiderata for a solution:

1. *Practical*. In order for the server to support a large number of clients, the solution should have good performance, in that it 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 data 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.

In contrast to the extensive studies on privacy-preserving location based services, few studies have addressed the general problem of practical private retrieval of public data in single server settings. Furthermore, this is the first paper, to our knowledge, that incorporates the realistic assumption of charging clients for exposed data, even if they didn't request it. This assumption imposes a realistic penalty on the approaches that hide the requested data among noisy data.

The two closest studies which could be adapted for developing a possible solution to this problem are *k*-Anonymity [19]

and *Computational Private Information Retrieval* (cPIR) [17]. k -Anonymity has been applied in privacy-preserving location based services [16], where the location point of a user is blurred into a cloaked region consisting of at least k user points and the server returns the nearest points of interests based on the cloaked region. k serves as a configurable degree of privacy. Similarly in our problem setting, the client could insert some related data into the original private data, 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 (indexes), 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, such as an index or a matching key in our case, has also been questioned in a number of proposals [21, 12, 13]: The real private data and the blurred data could be so close that the server can conclude with the probability $1/k$ that the private data is in a very narrow range. This is called *proximity breach* [13].

Computational Private Information Retrieval (cPIR) [11] retrieves a bit from a public bit string 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 cPIR protocol requires the public data to be organized in a matrix. A client interacts with the server using a random information hiding vector, and retrieves the desired bit from the result of the server’s computation on the random vector and the public data matrix. cPIR achieves computationally complete privacy by using expensive operations over all public data on the server, and automatically keeps the data communication secure by transmitting large random numbers generated in a similar way as RSA encryption and decryption [4]. 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 [18]. An alternative solution which attempts to bypass the expensive computation uses Oblivious RAM to hide the data access pattern [20]. Nevertheless, this approach is designed for the problem settings where client data is outsourced to the server, thus is not applicable in our context.

To achieve the above mentioned desiderata and seek a trade-off between the cost of answering a query and the quality 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 in 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 regulated by the client’s privacy requirement and desired charge limit (ρ, μ) , where ρ is the upper bound probability that a requested item can be identified by the server, and μ is the upper bound of the number of items that are exposed to the client because of one re-

quested item. Clients are free to specify their privacy and charge specifications (ρ, μ) . 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. If privacy is not needed at all, the bounding box shrinks to the requested item, and the cost is the lowest. If complete privacy is required, the bounding box covers the entire public data matrix, and consequently the cost is the highest.

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 the original data. Moreover, the bounding box in bbPIR not only includes the item values that are close to the query item, but also includes the 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.

Our contributions are summarized as follows.

1. We propose a generalized approach for private retrieval of public data called *Bounding-Box PIR* (bbPIR), which is a flexible and practical variant of cPIR. It also extends to k -Anonymity based private querying method.
2. We present a practical low cost solution for enabling retrieval by matching data keys instead of addresses of the matrix.
3. We demonstrate experimentally that bbPIR has the potential to be applied in large privacy aware internet applications: bbPIR largely outperforms cPIR in terms of performance, and achieves better quality of service than k -Anonymity in terms of high privacy and low charge.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 briefly explains the background knowledge of cPIR. Section 4 describes the data model that we work with. 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 $x = x_1 \dots x_n$ 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 no less than two database servers. They also showed that single database PIR does not exist in the information-theoretic sense. Nevertheless, given that in practice we can assume

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 limit of privacy breach probability
μ	upper limit of server charge
P_{brh}	privacy breach probability
C_{srv}	server charge
C_{comm}, C_{comp}	communication, 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

a database server is 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 c PIR in the paper. Although there have been follow-up single database computational PIR proposals that focus on improving the communication overhead of c PIR, as surveyed in [17], we do not consider them in this paper because they use even more expensive operations than modular multiplications as pointed out by Sion and Carbunar [18], thus are not feasible for practical applications.

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 in the released private data [19]. 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 improve the privacy of k -Anonymity [15, 14, 21, 12, 13]. 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. Note that generalization or suppression, which works for private data release, does not work for private retrieval of public data, because clients want accurate information, not blurred information. Finally, the potential charge due to the k extra revealed data items from the server was never quantified before. Our framework explicitly addresses this extra cost.

A closely related problem to private retrieval of public data is privacy-preserving query processing. One of the most rep-

resentative applications is privacy-preserving location based service. k -Anonymity was first applied as a flexible privacy preserving method in location based service [16]. However, since it needs a third party anonymizer, PIR was adopted later for removing that need [8]. Our proposal of bb PIR can also be applied in privacy-preserving location based services.

The problem of privacy-preserving query processing has also been studied in more general applications. Nevertheless, majority of the work is on querying across private sources [7], which does not consider the characteristics of client server settings in our case. For example, Agrawal et al.'s proposal for private data sharing and computation [1] ships the entire encrypted data between the two parties and does not exploit the fact that only a small part of the public data needs to be disclosed to the client as the answer of a private query. GhostDB [2] is a dedicated system for private querying on public data. It prevents the disclosure of the private data by keeping them in a secure USB key, and only allows the public data to flow to the USB key. However, it does not protect against the privacy breach of the query, and it transfers a major part of the computation load to the client.

3. BACKGROUND ON c PIR

c PIR relies on the computational intractability of the *Quadratic Residuosity* problem. 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

$$\exists y \in Z_N^* \text{ s.t. } y^2 = x \text{ mod } N \quad (1)$$

Otherwise, x is a *quadratic nonresidue* (QNR) mod N [9]. For example, given $N = 17$, 8 is QR because $5^2 = 8 \text{ mod } 17$, and 3 is QNR because $y^2 = 3 \text{ mod } 17$ has no solution. The problem is considered computationally most difficult if $N = p_1 \cdot p_2$, where p_1 and p_2 are two large distinct primes with equal number of bits, $m/2$. Let

$$Z_N^{+1} = \{y \in Z_N^* | (\frac{y}{N}) = 1\} \quad (2)$$

The *Quadratic Residuosity Assumption* (QRA) says that for $y \in Z_N^{+1}$, without knowing p_1 and p_2 in advance, the probability of distinguishing y between a QR and a QNR is negligible for large enough number of bits m [11].

The problem is much easier if p_1 and p_2 are known. Based on *Euler's theorem* [9], x is QR if and only if

$$x^{(p_1-1)/2} \text{ mod } p_1 = 1 \wedge x^{(p_2-1)/2} \text{ mod } p_2 = 1 \quad (3)$$

and QNR otherwise.

c PIR is designed to retrieve the value of a single bit in a large matrix [11]. It utilizes the computational difference in determining whether a number is QR or QNR if p_1 and p_2 are known or not. 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 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

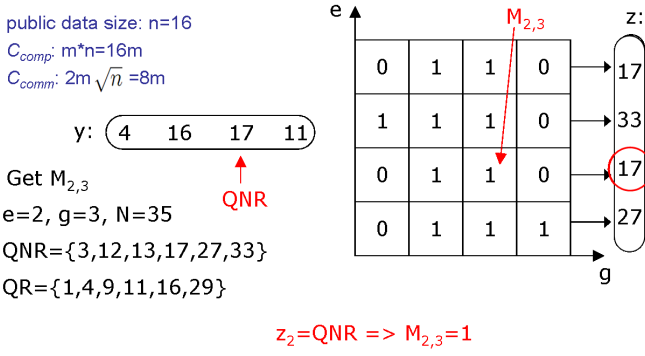


Figure 1: cPIR Example

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} , $y = [y_1, \dots, y_t]$, 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 $z_i = \prod_{j=1}^t w_{i,j}$, 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, \dots, 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 before issuing any queries. Then 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. $z_2 = (4^2 \times 16 \times 17 \times 11^2) \bmod 35 = 17$. Since $z_2 = 17$ is a QNR, when the client receives the vector z from the server, she obtains $M_{2,3} = 1$.

Recall 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. Although y_g is set to be QNR, y_g^2 is QR for $M_{e,g} = 0$ (step 3). Note that in step 5, the client is able to interpret $M_{i,g}$ ($1 \leq i \leq s$) by analyzing z_i . So by running one round cPIR, all s bits in the column of the requested bit entry are exposed to the client and hence could be charged to the client.

The above cPIR protocol for private retrieval of one bit can be extended to support more realistic private data retrievals. To retrieve a data item with b bits, repeat cPIR from the third step to the last step for b times to retrieve the one bit index of the data item at one time. Vector y can be reused b times, while vector z must be calculated on different bits of M b times. To retrieve multiple data items that are distributed in a different columns of M , execute the extended cPIR procedure for retrieving one item a rounds from the second step.

4. DATA MODEL

We work with a (key, address, value) data store. The public data of size n is organized in an $s \times t$ matrix M ($s = t = \lceil \sqrt{n} \rceil$ by default), where each entry is a b -bit public data item. Each public data item x (value) has a numeric key KA which identifies the address of x in M . x is only accessible by its address. Sorting the public data by KA in ascending order, the public data items 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 has to be translated to *query by address* in the retrieving process.

The client specifies both the privacy requirement and the charge budget as (ρ, μ) , where ρ is the privacy breach limit (the highest allowed probability that a requested item can be identified by the server), and μ is the charge limit for one item (the upper bound of the number of items that are exposed to the client because of the requested item). For example in the case of k -Anonymity, $\rho = 1/k, \mu \geq k$. We note that for a public data set of size n , the best achieved 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.

For analyzing a private retrieval method, we focus on practical communication and computation performance as well as the quality of the privacy service. Thus 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. We are not concerned about the computation cost on the client here, because it is generally much smaller than the computation cost on the server and is not a bottleneck, as later shown in our experiment results. (3) *Privacy Breach Probability* P_{brh} , the probability that the server can figure out a requested item. It must hold that $P_{brh} \leq \rho$. (4) *Server Charge* C_{srv} , the number of interpretable public data items retrieved from the server. It must hold that $C_{srv} \leq \mu$. All related variables are listed in Table 1. 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 the private retrieval of 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 lists the calculated C_{comm} and C_{comp} for private retrieval of a 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 metric analysis on c PIR 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. In designing such an approach, we desire the security and privacy of c PIR, as well as the flexibility and computation performance of k -Anonymity. On the other hand, we should avoid or reduce the impractical performance cost of c PIR, as well as the security and proximity privacy threat of k -Anonymity. So we propose the following private retrieval approach called *Bounding-Box PIR* (bb PIR), which unifies and seeks a practical tradeoff between c PIR and k -Anonymity.

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

Since bb PIR operates on an $r \times c$ sub-matrix of M instead of the $s \times t$ matrix M as in c PIR, 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 whole bounding box $1/(r \cdot c)$. $C_{srv}(bbPIR)$ is the number of rows in the sub-matrix r , because similar to c PIR, a client using bb PIR can only 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, and assume 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 the public data histogram distribution published by the server. In the following private query problems, we use the same N (m bits), p_1 and p_2 ($m/2$ bits respectively) for running bb PIR on different queries. So we omit the constant cost of sending N .

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

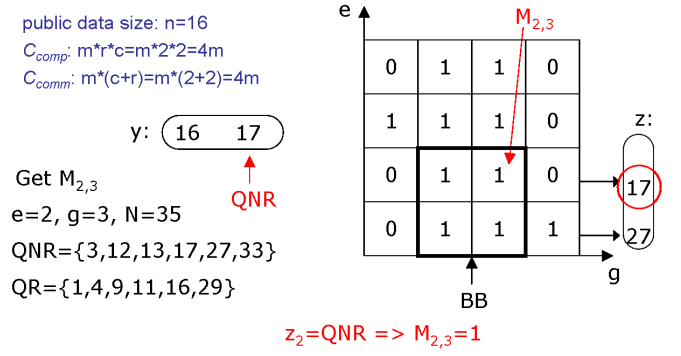


Figure 2: bb PIR Example: Private Retrieval of One Bit

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:

- 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

$$r = c = \lceil \sqrt{1/\rho} \rceil \quad (4)$$

Otherwise, set

$$r = \min(\mu, s), \quad c = \min(\lceil 1/(\rho \cdot r) \rceil, t) \quad (5)$$

- To retrieve entry (e, g) in M , the client first locates the coordinates of BB on M with the above defined dimensions r, c , s.t. (e, g) is as near to the center of BB as possible, and BB is within the address space of M .
- Then, the client generates a vector of c m -bit random numbers in Z_N^{+1} , $y = [y_1, \dots, y_c]$, 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.
- The server computes for each row i of the sub-matrix BB a modular product $z_i = \prod_{j=1}^c w_{i,j}$, where $w_{i,j} = y_j^2$ if $M_{i,j} = 0$, and $w_{i,j} = y_j$ if $M_{i,j} = 1$.
- The server sends to the client z_1, \dots, z_r .
- 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 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. 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*, *bbPIR* is able to satisfy any privacy requirement and charge limit (ρ, μ) that is within the size boundary of M . This is much more flexible than in the cases of k -Anonymity and *cPIR*. 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 . However, the coordinates of BB are not unique, and generally only the odd number size r and c can have (e, g) as the center of BB . For example in Fig. 2, the queried bit $M_{2,3}$ is not at the center of a 2×2 BB .

The comparisons of k -Anonymity, *cPIR* and *bbPIR* 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, *bbPIR* 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 *bbPIR* are all lower than those of *cPIR*. 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 *bbPIR* degenerates into k -Anonymity. At the other extreme, if we set $r = c = \lceil \sqrt{n} \rceil$, *bbPIR* degenerates into *cPIR*.

Table 2: Comparisons on Private Retrieval of One Bit

Method	k -Anonymity	<i>cPIR</i>	<i>bbPIR</i>
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. The client query in a single row vector y can be applied on b bits of the public data, but to retrieve each of the b bits of the requested item, the server answer has to include b column vectors z . So the Communication Cost in this case is $C_{comm} = m \cdot c + m \cdot b \cdot r$. Similarly, m -bit modular multiplication has to be applied to each bit of the $r \cdot c$ items in BB , so the Computation Cost in this case 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 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 to BB are decided as follows:
If $\mu \geq \lceil \sqrt{1/(\rho \cdot b)} \rceil$, set

$$r = \lceil \sqrt{1/(\rho \cdot b)} \rceil, c = \lceil \sqrt{b/\rho} \rceil \quad (6)$$

Otherwise, set

$$r = \min(\mu, s), c = \min(\lceil 1/(\rho \cdot r) \rceil, t) \quad (7)$$

2. To retrieve entry (e, g) in M , the client first locates the coordinates of BB on M with the above defined dimensions r, c , s.t. (e, g) is as near to the center of BB as possible, 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} , $y = [y_1, \dots, y_c]$, 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 $z_i = \prod_{j=1}^c w_{i,j}$, 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, \dots, 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) .

Note that in practical applications, since b could be bigger than $1/\rho$, e.g. the public data items are movie introductions, the value $\lceil \sqrt{1/(\rho \cdot b)} \rceil$ could be equal to 1, and $\mu \geq \lceil \sqrt{1/(\rho \cdot b)} \rceil$ holds. Then the bounding box could be a highly skewed rectangle with a single row but many columns, which puts large computation load on the client side for generating and determining QR and QNR for each column. Thus to provide better quality of service for the client in these cases, even if the condition $\mu \geq \lceil \sqrt{1/(\rho \cdot b)} \rceil$ holds, we use formula 7 instead of formula 6.

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 Section 5.1 and Section 5.2, we assume a client knows the exact address of the requested entry (e, g) . However in practice *query by key* is very common, where a client issues a query with its private data as a condition and expects the matching public data returned as the answer. 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) in order to apply *bbPIR*.

The same problem also exists in cPIR. In fact, 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]. Then the client issuing a query with a keyword condition achieves the query privacy by an oblivious walk on the index structure. But unfortunately, this oblivious walk requires the client to run as many as $O(b \cdot \log n)$ rounds of PIR on 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 by 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 [16]. It could be efficient, but 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 introducing another party into the communication between the client and the server.

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.

We assume a predefined threshold w which is the minimum number of keys in each bin. Recall in Section 4 KA can be sorted in ascending order. 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 . Thus we require $w \leq s$ for simplicity ($w > s$ is also possible if we enforce each bin to contain multiple columns of M).

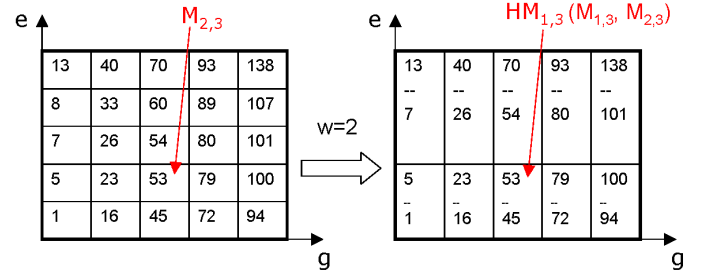


Figure 3: Example of Locating By Histogram

Once the bins are matched in H onto M , H can be transformed to be 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' . So the histogram H can be built greedily by checking the condition whether $w * (\lfloor s/w \rfloor - 1)$ keys have been scanned in the current column on M . Assign the next $s - w * (\lfloor s/w \rfloor - 1)$ keys to a new bin if the condition is true and then proceed to a new column, or assign the next w keys to a new bin if it is false. For example, if we have 25 keys and a 5×5 matrix M , for $w = 2$, we assign the first two keys in a bin and the last three keys in another bin for each column from the leftmost column to the rightmost column, 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 $s \bmod w = 0$, 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

$$[(e' - 1) \cdot w + 1, e' \cdot w] \times [g', g'] \quad (8)$$

for $e' < \lfloor s/w \rfloor$, or

$$[(e' - 1) \cdot w + 1, s] \times [g', g'] \quad (9)$$

for $e' = \lfloor s/w \rfloor$. One advantage of $w \leq s$ is that we only need to run *bbPIR-item* once to obtain all the entries in the address range of the requested bin, because these entries are in the same column of the bounding box, g , thus the client can interpret $M_{i,g}$ by analyzing z_i in the server answer. Therefore, the communication and server computation costs are not sensitive to increasing w . Instead, the client computation cost is expected to increase with w . As an example of query by key through 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 *bbPIR-item* once to obtain the entries in $HM_{1,3}$, where she finds the answer $M_{2,3}$.

The number of rows in a bounding box, r should be no less than w . We can satisfy any privacy and charge specification (ρ, μ) s.t. $\rho \leq 1/w$ and $\mu \geq w$.

6. EXPERIMENTAL EVALUATION

Our experiments evaluate the performance and the quality of service of *bbPIR* against cPIR and k -Anonymity for private data retrieval queries. The *performance* metrics include

Communication Cost (data communication size) and Computation Cost (server/client computation time). The *quality of service* metrics include Privacy Breach Probability and Server Charge. The results in different settings (including different public data matrix shapes, different (ρ, μ) specifications, *query by address* and *query by key*) demonstrate that *bbPIR* is practical for safeguarding the client’s query privacy while protecting the server’s business profits.

We implemented the three private retrieval methods in C++. For *bbPIR*, we use *bbPIR-Item* protocol and formula 7. We run majority of the experiments on 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. To load them in a binary data matrix, the total number of data items $n = 32561$, and the total bits of each data item $b = 1120$. For one experiment on proximity privacy (Section 6.4), since pure numeric data is easy to measure proximity, we use a synthetic data set with 100K numeric data items. Without explicitly mentioning, we use default values in the experiments as 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	32561
b	1120
s, t	181 (square matrix)
ρ	0.01
μ	10
k	100
m	1024
$P_{brh}(cPIR)$	0.000031
$P_{brh}(bbPIR), P_{brh}(k\text{-Anonymity})$	0.01
$C_{srv}(cPIR)$	181 (square matrix)
$C_{srv}(bbPIR)$	10
$C_{srv}(k\text{-Anonymity})$	100

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 *cPIR* 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 *cPIR* $\min(m \cdot (b \cdot s + t))$. The optimal values become

$$s = \lceil \sqrt{n/b} \rceil, t = \lceil \sqrt{b \cdot n} \rceil \quad (10)$$

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 $\rho = 0.01$, $\mu = 10$ and varying the modulus bit size m from 512 to 1536 according to [18]. 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 $1/32561=0.000031$ and 0.01 respectively as listed in Table 4. Server Charge for *cPIR* is equal to the number of rows in the matrix, s , which is 181 on square matrix and 6 on rectangular matrix. Server Charge for *bbPIR* is equal to the number of rows in the bounding box, r , which is 10 on square matrix and 6 on rectangular matrix. We can see from Fig. 4 that *bbPIR* is basically 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.

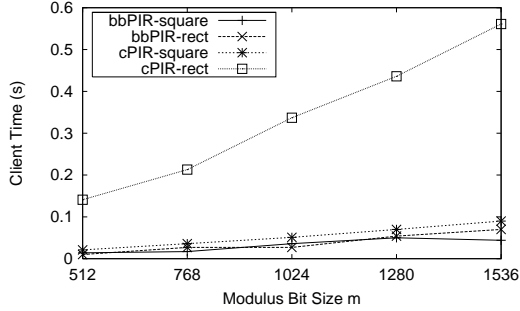
Compared to *cPIR*, *bbPIR* achieves much better performance with a lower but acceptable privacy level. Especially the server computation time of *bbPIR* drops down dramatically and becomes promising for practical use (it is less than 0.75s in both matrix shapes).

6.2 Effects of Modulus Bit Size

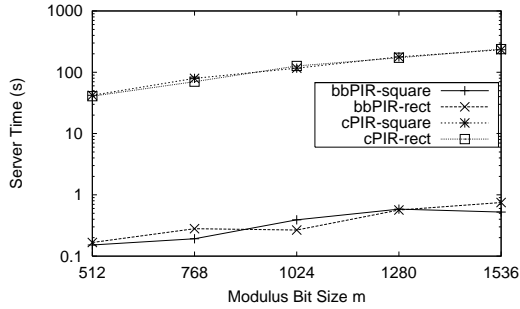
After examining the effects of the data matrix shape on *cPIR* and *bbPIR*, we specifically study the effects of the modulus bit size m on the performances of *bbPIR*, *cPIR* and k -Anonymity.

Recall 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 can be an address range of the anonymized entries similar to the dimensions and coordinates of bounding box on M in the case of *bbPIR*, and does not provide a third party 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 communication contents between the client and the server. To provide k -Anonymity the same security level as *bbPIR*, we can apply a popular encryption algorithm RSA on the server answer in k -Anonymity.

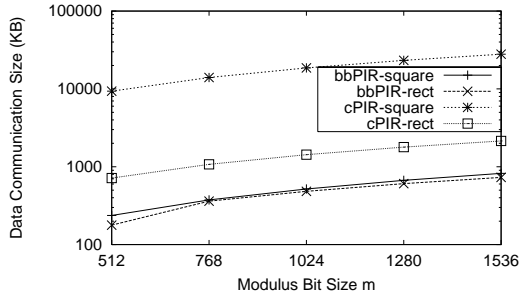
We estimate the computation and communication costs of the security enhanced k -Anonymity, which we denote as k -Anonymity (RSA) and abbreviate as k -A (RSA) in Fig. 5, by the following theoretical analysis according to [4]. First, since RSA encryption and decryption is the dominant cost, we simplify 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 $Ciphertext = Data^{public_key} \pmod{N}$ and $Data = Ciphertext^{private_key} \pmod{N}$ 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 *bbPIR* server computation (modular multiplication) time, we use the same m -bit modulus N , and estimate both $public_key$ and $private_key$ as $m/2$ -



(a) Comparison of Client Time



(b) Comparison of Server Time



(c) Comparison of Data Communication Size

Figure 4: Effect of Data Matrix Shape, Vary Modulus Bit Size m , Fix $\rho = 0.01$, $\mu = 10$

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 $bbPIR$ needs $\lceil 1/\rho \rceil$ modular multiplications, we can calculate RSA encryption and decryption time as

$$T_{server}(bbPIR)/(\lceil 1/\rho \rceil) \cdot (m/2 - 1) \quad (11)$$

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

$$|Ciphertext| = |Data| + m - (Data \bmod m) \quad (12)$$

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

be estimated as

$$C_{comm}(k-Anonymity)/|Data| \cdot |Ciphertext| \quad (13)$$

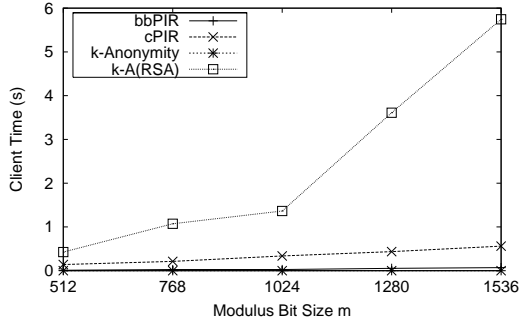
in which the operator $||$ refers to the number of bits, and m is used as the block size of RSA.

We use a rectangular matrix in favor of $cPIR$, and fix $\rho = 0.01$, $k = 100$, $\mu = 10$. Fig. 5 illustrates that larger key size adversely impacts the computation time for $cPIR$, $bbPIR$ and k -Anonymity (RSA), and data communication size for $cPIR$ and $bbPIR$. 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. The computation time of k -Anonymity is close to 0 in Fig. 5(a) and Fig. 5(b). k -Anonymity (RSA), with RSA encryption on the server and decryption on the client, takes more computation time on both the server and the client than $bbPIR$, as seen in Fig. 5(a) and 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), which is obtained by encrypting blocks instead of bits, does not incur much additional data communication costs, as seen in Fig. 5(c). In contrast, $cPIR$ and $bbPIR$ generate a large random number for each bit, thus incur larger data communication costs. Note that the performances of $cPIR$ and $bbPIR$ can be further improved using the optimization in [8] to avoid redundant modular multiplications.

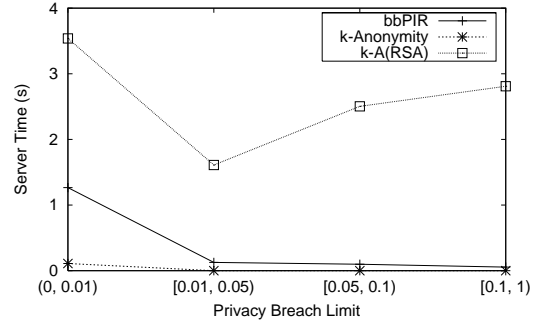
6.3 Effects of Privacy and Charge Specification

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 $bbPIR$ and k -Anonymity. From now on, we use a square matrix, because there is little difference between square matrix and rectangular matrix for $bbPIR$. 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.

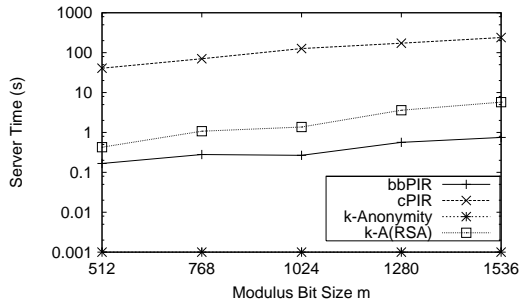
We first fix $\mu = 10$, and vary ρ in four ranges, $(0, 0.01)$, $[0.01, 0.05)$, $[0.05, 0.1)$ and $[0.1, 1)$. For each range, we mimic the requests from different clients by randomly generating 100 values of ρ in that corresponding range and running 100 queries. We then take the average metric results. Fig. 6 demonstrates a general trend that a lower privacy requirement (higher ρ values) reduces both computation and communication costs for $bbPIR$ 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 values of $k = 1/\rho$) do not impact on the computational complexity of RSA encryption. For the same privacy breach limit, k -Anonymity usually needs more server charge than $bbPIR$ as seen in Fig. 6(c), which is not appealing to most internet users even if it takes less computation and communication cost. Only under very weak privacy constraints is k -Anonymity comparable in charge to $bbPIR$, in which cases the bounding box of $bbPIR$ almost degenerates into a single column on the public data matrix. 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



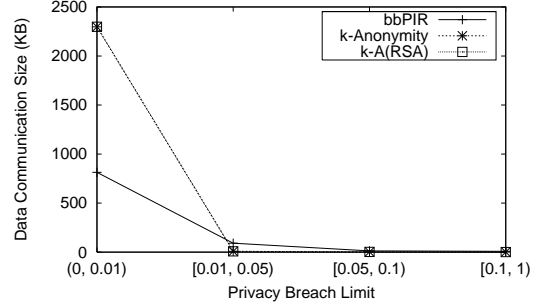
(a) Comparison of Client Time



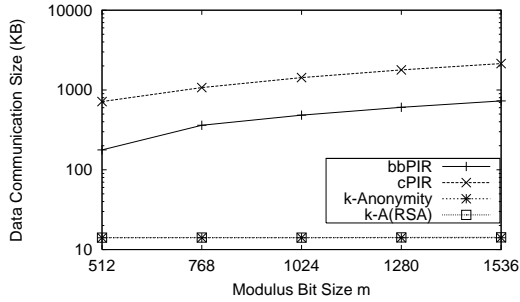
(a) Comparison of Server Time



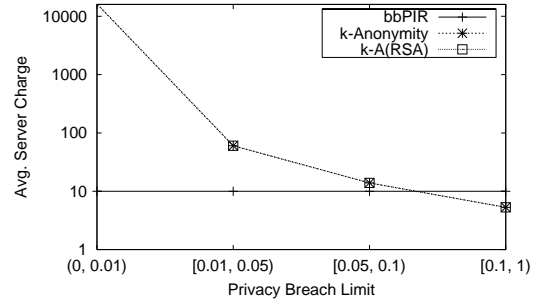
(b) Comparison of Server Time



(b) Comparison of Data Communication Size



(c) Comparison of Data Communication Size



(c) Comparison of Server Charge

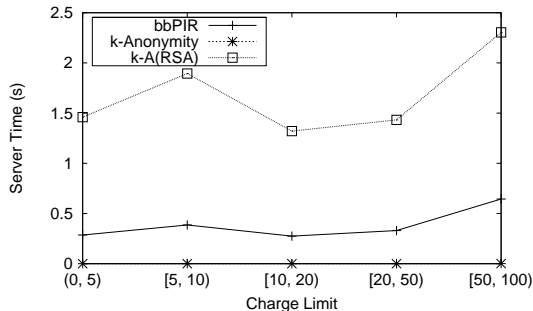
Figure 5: Comparison of $bbPIR$, $cPIR$, k -Anonymity and k -Anonymity(RSA), Vary Modulus Bit Size m , Fix $\rho = 0.01$, $\mu = 10$, $k = 100$

Figure 6: Comparison of $bbPIR$, k -Anonymity and k -Anonymity(RSA), Vary Privacy Breach Limit ρ ($1/k$), Fix $\mu = 10$

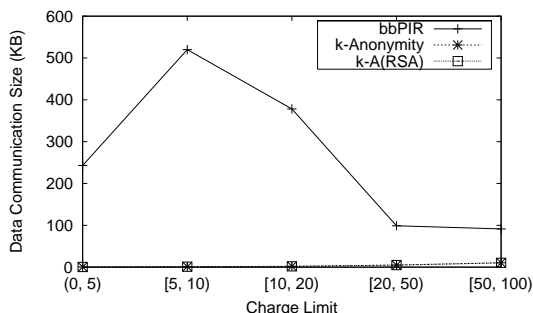
k -Anonymity (RSA) and k -Anonymity overlap in Fig. 6(c) and Fig. 6(b).

We then fix $\rho = 0.01$, and vary μ in five ranges, (0, 5), [5, 10), [10, 20), [20, 50) and [50, 100). Similarly as 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 as seen in Fig. 7. Theoretically the server computation cost of $bbPIR$ $m \cdot b \cdot r \cdot c$ should not change, because ρ is fixed. But the computation for different bounding boxes could be different because of the slight difference of modular multiplication between a 0 bit and a 1 bit. A

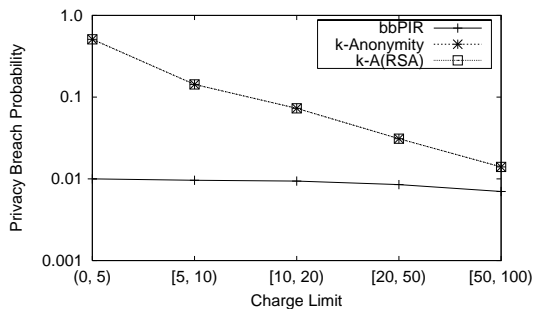
proper value of μ which gives a good bounding box shape could achieve the best performance on both computation and communication costs. For the same charge limit, We can see in Fig. 7(c) that k -Anonymity cannot reach the same privacy as in $bbPIR$, and its real privacy breach probability is always larger than 0.01, so does k -Anonymity (RSA). 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(c) and Fig. 7(b).



(a) Comparison of Server Time



(b) Comparison of Data Communication Size



(c) Comparison of Privacy Breach Probability

Figure 7: Comparison of *bbPIR*, *k*-Anonymity and *k*-Anonymity(RSA), Vary Charge Limit μ (k), Fix $\rho = 0.01$

To summarize the above two experiments, *k*-Anonymity could not achieve the same high privacy, low charge as *bbPIR*, thus its quality of service is not as good as in *bbPIR*.

6.4 Proximity Privacy of Numeric Data

In this experiment, we specifically study the proximity privacy of *bbPIR* and *k*-Anonymity on numeric data. As pointed out in [21, 12, 13], there should be enough difference between the data items in an anonymized range (in a bounding box in the case of *bbPIR*) under a privacy breach probability P_{brh} , which we call *neighborhood difference*, otherwise one can conclude that the private data is within a very narrow

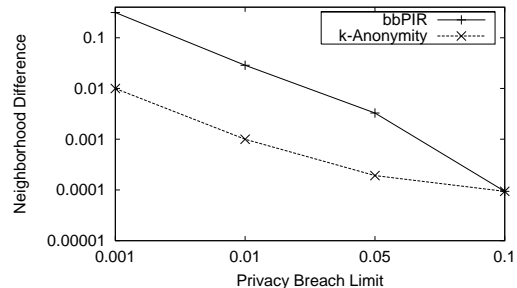


Figure 8: Comparison of *bbPIR* and *k*-Anonymity, Vary Privacy Breach Limit ρ , Fix $\mu = 10$

range with probability P_{brh} .

Since the Adult data set is not pure numeric and thus is not appropriate for measuring proximity, we generate a synthetic data set with 100K numeric data items that are uniformly distributed within $[0.0, 0.1]$. We measure the neighborhood difference as the absolute difference between the maximum value and minimum value in an anonymized range (or in a bounding box) following [21]. We then fix $\mu = 10$, and vary ρ from 0.001 to 0.1 (correspondingly set $k = 1/\rho$ and vary k from 1000 to 10 in *k*-Anonymity). Since the bounding box used in *bbPIR* 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 *bbPIR* is more than 30 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. Note that the neighborhood differences in *bbPIR* and *k*-Anonymity are the same for $\rho = 0.1$, because we fix $\mu = 10$ and the bounding box becomes a single column with 10 rows.

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 *bbPIR*. 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. Although these items are in the same column and do not incur additional computation cost on the server, interpreting all these items could lead to considerable additional cost to the client. So we need to specifically study the client computation time, which is almost negligible before but can not be ignored here.

The size of the requested address range is determined by the granularity of server published histogram, the minimum number of keys in each bin, w . We vary w from 10 to 100, set $\mu = w$ (since $\mu \geq w$ must hold) and fix $\rho = 0.01$. 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 *bbPIR-h* and *bbPIR* respectively, is shown in Fig. 9. We can see that larger values of w increases the client computation time. However, this overhead is still reasonable as it provides a practical solution to the impractical assumption of *cPIR*, i.e., that the client

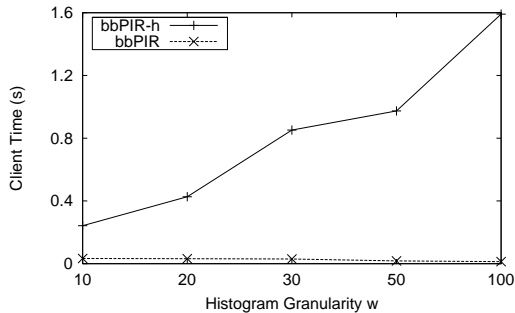


Figure 9: Comparison of query-by-key (*bbPIR-h*) and query-by-address (*bbPIR*), Vary Histogram Granularity w , Fix $\rho = 0.01$

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 and flexible approach of private retrieval of public data on single server settings, we follow *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* (*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 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.

9. 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/~mllearn/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, Technion, 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. Suci. 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. Workload-aware anonymization. In *KDD*, pages 277–286, 2006.
- [13] J. Li, Y. Tao, and X. Xiao. Preservation of proximity privacy in publishing numerical sensitive data. In *SIGMOD Conference*, pages 473–486, 2008.
- [14] N. Li, T. Li, and S. Venkatasubramanian. t -closeness: Privacy beyond k -anonymity and l -diversity. In *ICDE*, pages 106–115, 2007.
- [15] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkatasubramanian. l -diversity: Privacy beyond k -anonymity. In *ICDE*, page 24, 2006.
- [16] 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.
- [17] 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.
- [18] R. Sion and B. Carbunar. On the computational practicality of private information retrieval. In *Network and Distributed System Security Symposium*, 2007.
- [19] L. Sweeney. k -anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.
- [20] P. Williams and R. Sion. Usable private information retrieval. In *Network and Distributed System Security Symposium*, 2008.
- [21] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. Aggregate query answering on anonymized tables. In *ICDE*, pages 116–125, 2007.