Frequency-based Constraint Relaxation for Private Query Processing in Cloud Databases

Kawamoto, Junpei
Kyushu University

Gillett, Patricia L.
École Polytechnique de Montréal

https://hdl.handle.net/2324/1445771

出版情報：Proceedings of the 27th Annual IEEE Canadian Conference on Electrical and Computer Engineering, pp.1275-1280, 2014-05. IEEE
バージョン：
権利関係：

Frequency-based Constraint Relaxation for Private Query Processing in Cloud Databases

Junpei Kawamoto
Kyushu University
744 Motooka, Nishiku
Fukuoka, Japan
Email: kawamoto@inf.kyushu-u.ac.jp

Patricia L. Gillett
École Polytechnique de Montréal
C.P. 6079, Succ. Centre-ville
Montréal, Québec, Canada
Email: patricia-lynn.gillett@polymtl.ca

***Abstract*—We introduce a new definition of privacy based on query frequencies, as well as a frequency-based constraint relaxation methodology for *private queries*. Private queries undergo processing so that users may obtain data from a database in such a way that the user's *search intentions*, i.e. the data which the user is interested in, will be protected against exposure. Most existing protocols for private querying rely on the following two constraints to achieve privacy: i) queries are encoded so that the database server can handle query processes but cannot actually decode queries; ii) the server is forced to check all data in the server when computing query results. Because of these constraints, even database servers cannot distinguish which data are selected from the database. However, this second constraint compels servers to spend $O(n)$ computational cost for each query processed, where n is the number of data entries on the server. We introduce a weaker privacy condition which ensures that search intentions are hidden within a portion of the database, as opposed to ordinary private queries which hide search intentions among all data in the database, and we argue that this definition of privacy is sufficient to combat attacks based on query frequencies. Our relaxation methodology relaxes the second constraint above and allows private querying while only examining a portion of the data in most cases. Our methodology is also flexible and applies not only to exact match queries in one dimensional data but also to range queries in one dimensional data and exact match queries in two dimensional data.**

I. Introduction

Private queries undergo processing to allow users to obtain data from a database without exposing which data the user is interested in (their *search intentions*), and they are a necessary technology in order for people to safely use web services such as databases as a service. This is because a user ’s search intentions are used enterprisingly, despite potentially containing sensitive information about the user. For example, let us imagine a magazine website where people can access articles of interest. In this case, a malicious person could sneak a peek at a reader ’s search intentions, i.e. which articles the reader accesses, and possibly determine the hobbies, political views, or religion of the reader. Some people would wish to keep such information private. To give a more serious example, let us imagine a website where users search a database for information about prescription medicines. In this case, malicious parties who steal the search intentions of a user might learn that the user is suffering from a serious illness.

We introduce three ideas for our discussion of private queries; search intention, query, and handled set. *Search intentions* are what users are really interested in and hope to obtain from information services. *Queries* are requests which the users send to servers to obtain information associated with their search intentions. These queries typically conform to a format defined in protocols which the users and the servers employ. The *Handled set* is the data set which servers must check to compute the results of a given query from the user. Figure 1 illustrates these ideas. A user of some magazine site is now interested in an article x, so the search intention of the user is x. If we assume that the user and the magazine site simply use the HTTP protocol, then the user sends a query “GET x HTTP/1.1” to obtain article x. A server of the site receives the query, checks part of the website ’s database, and finds article x. Finally, the server returns article x to the user.

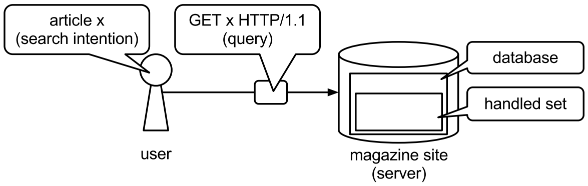


Fig. 1: Search intention, query, and handled set.

Most existing protocols for private querying [1], [2] achieve privacy by imposing the following two constraints: i) queries are encoded in such a way that the server can handle query processes but cannot actually decode queries; ii) the server is made to check all data in the database when computing any query result. To satisfy the first constraint, protocols employ secure encoding techniques such as encryption. By using the second constraint, the full database is taken as the handled set and even servers cannot distinguish which data have been selected from the entire database. On the other hand, the second constraint compels servers to spend $O(n)$ computational cost executing each query, where the database has n entries. This means that these protocols have a high computing cost to process each query, and do not seem to be suitable for large databases.

On the other hand, we might suppose that there are cases in which we do not need to retrieve the entire database to sufficiently obscure search intentions. For instance, let us recall the example of the magazine website. In that case, we may be able to sufficiently obscure political and religious views by retrieving data from a carefully chosen subset of the database rather than retrieving the entire database. This would allow us to relax the second constraint and achieve sufficient privacy of search intentions using a subset of the database. Another protocol which relaxes the second constraint is bbPIR [3], a naïve protocol which relaxes the second constraint and only ensures that the search intention x is hidden among $k-1$ neighbouring data entries, i.e. handled sets in bbPIR have only k items. However, this approach may not be sufficient. We can imagine a case for the magazine site in which x is a popular article and the other $k-1$ neighbouring articles are less popular and infrequently requested. If this fact is known, the user's search intention can be guessed from among bbPIR's relaxed handled set with high probability.

Agrawal et al. introduce a methodology which hides frequencies of queries [4] based on an order preserving encryption [5], [6]. This work achieves private queries in a different way than our work. By Agrawal et al.'s order preserving encryption scheme, encrypted items retain the same order as plain items, so they are comparable after encryption. Their schema can also modify the distribution of items; for example, we may have plain items from a zipf distribution which become uniformly distributed after encryption. However, they focus not only on the distribution of items but also on the frequencies of queries, and so adversaries who know the frequencies of queries are able to match items with search intentions even if encrypted items have uniform distributions. Lu also introduces a query processing methodology over encrypted items using order-preserving encryption and making tree structures [7]. It achieves $O(\log n)$ computational cost of servers, but it is also pointed out in the paper that the proposed methodology is not secure against frequency-based attacks. In contrast, our methodology gives a frequency-based relaxation approach which satisfies some requirements for privacy.

In this paper, we introduce a secure relaxation methodology which uses the frequencies of search intentions to guide the relaxation of the second constraint. The core idea of our methodology is that popular items can be guessed with higher probability, so the users search intentions should be better hidden by using a large handled set, while less popular items can be sufficiently hidden using a small handled set. Our proposal methodology reduces the handled sets in most cases, while also proving that the worst case risk of having ones search intentions exposed is no worse than when using unrelaxed protocols, which we refer to as *complete protocols*. We assume that complete protocols provide sufficient privacy, and, under this assumption, our approach does as well. We begin by introducing a relaxation methodology for exact match queries in one dimensional data. After that, we extend it to range queries in one dimensional data and exact match querying in two dimensional data. We discuss how our methodology can be applied to cPIR, a well known private querying protocol, and compare the efficiency of this with ordinary cPIR.

II. Basic notions

We assume that a database D consists of n items, i.e. $D = \{t_1, t_2, \cdots, t_n\}$, and a user wants to obtain the x-th item t_x from the database D[1]. In this case, the *search intention* of the user is x. Recalling the magazine website we introduced in section I, for example, this site has n articles and provides a table of contents (TOC) for those articles. A reader of this site, ie. a user, chooses an article x from the TOC and its associated number is sent to the server as a request. Yoshida et al. have proposed a keyword search protocol over this setting [8], so that we can extend this to some cases in which TOCs and catalogs are not provided and users need to search for items by keyword.

In most private querying protocols including cPIR [1] and IPP method [2], plain search intentions x are not sent to servers directly. Instead, search intentions are encoded and servers must check all data to compute query results, meaning that $H_{complete}(x) = D \quad \forall x \in D$. On the other hand, in bbPIR [3], the naïve relaxed protocol, the size of a handled set is k for any query, ie. $|H_{bbPIR}(x)| = k \quad \forall x \in D$.

We denote the frequency of a search intention $x \in D$ to be $\text{Freq}(x)$. We assume frequencies are normalized so that $\sum_{x \in D} \text{Freq}(x) = 1$.

Definition 1: Using this frequency function, we define *query risk* to be the conditional probability that the search intention is x given that we know the handled set H used. For the purposes of this paper, we will consider query risk a measure of exposure risk for private queries. Query risk is then formulated as

$$\text{Risk}\left(x | H(x)\right) = \frac{\text{Freq}(x)}{\sum_{y \in H(x)} \text{Freq}(y)}. \quad (1)$$

We assume that potential attackers know the frequencies of search intentions. Such an adversary is called a full-knowledge adversary. We suppose adversaries cannot see the user's search intention x directly but can learn the handled set $\forall x \in D : H(x)$. We also assume that servers may be adversaries, and that full-knowledge adversaries wish to obtain search intention x using handled set $H(x)$ and frequencies $\text{Freq}(x)$.

Definition 2: We say that a relaxed handled set $H_{relaxed}$ is secure if, for any search intention x, the query risk is at most the worst case query risk of the complete protocol. In other words, for all $x \in D$;

$$\text{Risk}\left(x | H_{relaxed}(x)\right) \leq \max_{y \in D} \text{Risk}\left(y | H_{complete}(y)\right). \quad (2)$$

We assume that the level of security provided by complete protocols is sufficient, and therefore that our approach will also be considered secure if it satisfies this condition.

Finally, we define the query processing costs for servers. We denote the query processing cost of a search intention x to be $C(x)$, which is the cardinality of the handled set $H(x)$. We also define the expected query processing cost C to be $C = \sum_{x \in D} \text{Freq}(x) C(x)$. In the complete protocol, servers must check all data for each query associated with a search intention x, so that $\forall x \in D$: $H_{complete}(x) = D$ and $C_{complete}(x) = n$. Therefore, $C_{complete} = \sum_{x \in D} \text{Freq}(x) C_{complete}(x) = n$.

III. Relaxation

Our constraint relaxation methodology lets servers process queries by only checking items in a subset of database D.

[1] For range queries, the user wants to obtain data in a sequence $[x, y]$, i.e. $\{t_x, t_{x+1}, \cdots, t_y\}$. We discuss this case further in section III-B.

```
Require: Database $D$, frequency function Freq.
  $bestsofar \leftarrow |D|+1,\ best \leftarrow [0, |D|-1]$
  $bestsum \leftarrow \sum_{y \in D} \text{Freq}(y),\ i \leftarrow x,\ j \leftarrow x$
  $vsum \leftarrow \text{Freq}(x),\ \mu \leftarrow \text{Freq}(x) / \max_{y \in D} \text{Risk}(y)$
  while $vsum < \mu$ and $i \geq 1$ do
    $i \leftarrow i-1,\ vsum \leftarrow vsum + \text{Freq}(i)$
  end while
  if $vsum \geq \mu$ then
    $bestsofar \leftarrow j-i+1,\ best \leftarrow [i,j],\ bestsum \leftarrow vsum$
  end if
  while $i \leq x$ do
    while $vsum \geq \mu$ do
      $j \leftarrow j+1,\ vsum \leftarrow vsum + \text{Freq}(j)$
    end while
    if $vsum \geq \mu$ then
      if $j-i+1 < bestsofar$ then
        $bestsofar \leftarrow j-i+1,\ best \leftarrow (i,j)$
        $bestsum \leftarrow vsum$
      else if $j-i+1 = bestsofar$ and $vsum > bestsum$
      then
        $bestsofar \leftarrow j-i+1,\ best \leftarrow (i,j)$
        $bestsum \leftarrow vsum$
      end if
    end if
    $i \leftarrow i+1,\ vsum \leftarrow vsum - \text{Freq}(i-1)$
  end while
  return $best$
```

Fig. 2: Find $H_r(x)$.

For simplicity, we first introduce the relaxation algorithm in one dimensional data. We only consider relaxed handled sets which are continuous, ie. $H_{relaxed} = \{t_i, t_{i+1}, \cdots, t_j\}$ where $i \leq x \leq j$. Because $H_{relaxed}(x) \subseteq H_{complete}(x)$, our methodology can reduce the computational costs of servers. Generally, any constraint relaxation holds negative aspects. In this case, it causes an increase in query risks. However, our approach bounds query risks for each search intention to complete protocols' worst case query risk.the maximum query risk of the complete protocols, ie. relaxed handled sets satisfy (2). Therefore, relaxed handled sets still protect privacy against full-knowledge adversaries. The following sections will explain how relaxed handled sets are calculated.

A. Exact match queries in 1D data

Problem 1: Find handled set H_r such that, for a search intention x:

1) $x \in H_r(x)$,
2) $\text{Risk}(x|H_r(x)) \leq \max_{y \in D} \text{Risk}(y)$,
3) the size of the set, $j-i+1$, is minimized among sets which meet the above two criteria,
4) if multiple solutions have equal size, the one which maximizes $\sum_{y \in H_r(x)} \text{Freq}(y)$ is chosen.

This handled set will be the smallest possible set which contains x and has sufficiently low query risk.

Figure 2 shows the $O(n)$ algorithm which solves this problem for 1D exact queries. However, we can compute $H_r(x)$ before querying time and use it repeatedly for the search intention x, so we presume that the computation of the handled set is not included in query processing time. The expected cost of the handled set H_r computed by Algorithm 2 is evaluated as $C_{relaxed} = \sum_{x \in D} \text{Freq}(x)|H_r(x)| \leq \sum_{x \in D} \text{Freq}(x) \times |D| = n$, and this will be smaller than that of a complete protocol.

B. Range queries in 1D data

We extend the previous algorithm to range queries in one dimensional data. Search intentions for 1D range queries are continuous subsets $\{t_{x_L}, t_{x_L+1}, \cdots, t_{x_R}\}$ instead of single values x. We denote these search intentions as $X \subseteq D$. For this X, we extend our definitions of frequencies and risks as follows:

$$\begin{aligned} \text{Freq}(X) &= \frac{\sum_{x \in X} \text{Freq}(x)}{\sum_{y \in 2^D} \text{Freq}(y)}, \\ \text{Risk}(X|H(X)) &= \frac{\text{Freq}(X)}{\sum_{y \in H(X)} \text{Freq}(y)}. \end{aligned}$$

We also extend query costs and expectations:

$$C(X) = |H(X)|, \quad C = \sum_{X \in 2^D} \text{Freq}(X) \times C(X).$$

In complete protocols, handled sets for a ranged search intention X are again the full database D. On the other hand, our relaxed handled sets will be continuous subsets of database D, ie. $H_r(X) = \{t_i, t_{i+1}, \cdots t_j\}$. Therefore, we extend Problem 1 for range queries in one dimensional data as follows.

Problem 2: Find a continuous subset $H_{rr} = \{t_i, t_{i+1}, \cdots, t_j\} \subseteq D$ which satisfies the following conditions for a given search intention X;

1) $X \subset H_{rr}$,
2) $Risk(X|H_{rr}(X)) \leq \max_{y \in D} \text{Risk}(y)$,
3) minimize $|H_{rr}(X)|$, and
4) maximize $\sum_{y \in H_{rr}(X)} \text{Freq}(y)$ under the condition 3).

This problem is solved by making a small modification to the algorithm shown in figure 2. Letting the range for range search intention X be $[x_L, x_R]$, we can initialize the algorithm as in problem 1 with the following alterations: when initializing the algorithm, let $x \leftarrow x_L$, $i \leftarrow x_L, j \leftarrow x_R, vsum \leftarrow Freq(X), \mu \leftarrow Freq(X)/max_{y \in D} Risk(y)$. As in algorithm 2, (i,j) will give the indices of the first and last members of the handled set at termination. As before, the complexity of the algorithm to find the handled set is $O(n)$.

C. Exact match queries in 2D data

Finally, we extend our relaxation algorithm to exact match queries in two dimensional databases. We consider a 2D database to be a matrix consisting of l rows and m columns, where $l \times m = n$. A user has search intention $x = (e, g)$. In complete protocols, $\forall x \in D$ the handled set is $H_{complete}(x) = D$, so $C_{complete}(x) = |H_{complete}(x)| = n$ regardless of x, with the servers accessing every item in the database.

We relax this constraint and choose as our handled set a submatrix which minimizes querying costs while satisfying condition (2), ie. sufficient security against full-knowledge adversaries. We will denote a submatrix which consists of rows i_a to i_b and columns j_a to j_b of D as $D[i_a : i_b, j_a : j_b]$. We define the problem of identifying a sub matrix for a search intention x as follows.

Problem 3: Find a submatrix of D, $H_{rm} = D[[top, top + height - 1]; [left, left + width - 1]]$ which satisfies the following conditions for a given search intention $x = (e, g)$;

1) x must be included, ie. $top \le e < top + height$ and $left \le g < left + width$,
2) $\text{Risk}(x|H_{rm}(x)) \le \max_{y\in D} \text{Risk}(y)$,
3) minimize $size = height \times width$, and
4) secondary to 3), maximize $\sum_{y\in H_{rm}(x)} \text{Freq}(y)$ among submatrices of the same size.

We can find this optimal submatrix using the algorithm shown in figure3. The initial best solution is the full matrix D. We observe that a submatrix can be defined uniquely by 4 variable: the indices (i, j) of its upper-leftmost element (the matrix's anchor position) and the submatrix's height and width. We solve the problem by iterating on height and width to consider different matrix shapes (while ignoring shapes which imply a larger size than the best known solution), and then by considering, for each shape, all anchor positions such that the associated submatrix would include element (e, g)). The best known solution is updated if a new matrix has fewer elements and a sum greater than the threshold, or if it has the same number of elements and a larger sum than the best known solution.

For convenience, we denote $\mathbb{1}^{h\times w}$ to be an h by w matrix in which every element is 1.

In order to reduce computation, we maintain two matrices C and M which store sums of parts of D. C is called the column sum matrix, and it stores sums of submatrices of D with dimensions $h \times 1$, ie, partial columns. $C_{i,j}$ stores the value of the column of height h which is anchored at $D_{i,j}$. ie, $C_{i,j} = \sum_{k=i}^{i+height-1} D_{k,j}$. Rather than compute all column sumns, we will keep values for only the submatrix of C given by $rangeC[0] \le i \le rangeC[1]$, $rangeC[2] \le j \le rangeC[3]$. When h increases, $updateC$ lengthens each column sum by one element. The rows of $rangeC$ will also be contracted below if necessary and expanded above if possible. The columns of $rangeC$, however, are extended as needed and contracted only if the column is guaranteed not to be needed again.

M is called the submatrix sum matrix, and it stores sums of submatrices of D with dimensions $h \times w$. Like C sums elements of D to produce partial column sums, M sums elements of C to produce submatrix sums. As with C, we only maintain data for the relevant portion of D, with this active range given by $rangeM$. For relevant (i, j), $M_{i,j}$ gives exactly the sum of the $h \times w$ matrix anchored at (i, j). Unlike C, M is updated with each increase in w, and must be reset with every increase in h. When w increases, $updateM$ finds the new sum matrix M for $h \times w$ matrices by adding a partial column sum (already calculated and stored in C) to each of the $h \times w - 1$ submatrix sums stored in M previously.

Functions $updateC$ and $updateM$ are omitted here, but each cost only $O(n)$ (where $n = lm$) because previously calculated partial sums are retained and exploited. When C and M are up to date for the current h, w, the scores of all relevant $h \times w$ submatrices are readily accessible. We can simply identify the largest element of M and update the best solution if appropriate. Since h loops over m values, w loops over l, and the tasks within the w loop have $O(n)$ complexity, the algorithm as a whole has $O(n^2)$ complexity.

```
Require: Database $D \in \mathbb{R}^{l\times m}$, frequency function Freq, $x = (e, g)$.
  $\mu \leftarrow \text{Freq}(x)/\max_{y\in D}\text{Risk}(y)$
  $vsum \leftarrow \sum_{i,j} D,\ size \leftarrow l \times m,\ best \leftarrow (0, 0, l, m)$
  $C \leftarrow -\mathbb{1}^{h\times w},\ C_{e,g} \leftarrow D_{e,g}$
  $rangeC \leftarrow (e, e, g, g)$
  $h \leftarrow 1,\ w \leftarrow 1,\ maxw \leftarrow m$
  while $h \le min(l, size)$ do
    $M \leftarrow -\mathbb{1}^{h\times w}$
    $M_{i,g} \leftarrow C_{i,g} \quad \forall \quad rangeC[0] \le i \le rangeC[1]$
    $rangeM \leftarrow (rangeC[0], rangeC[1], g, g)$
    while $w \le maxw$ do
      $(i^{new}, j^{new}) \leftarrow argmax(M), v^{new} \leftarrow M[i^{new}, j^{new}]$
      if $(v^{new} > vsum)$ or $(v^{new} \ge \mu$ and $h \times w < size)$
      then
        $vsum \leftarrow v^{new}, size \leftarrow h \times w$
        $best \leftarrow (i^{new}, j^{new}, h, w)$
        $maxw \leftarrow min(m, \lfloor \frac{size}{h} \rfloor)$
      end if
      $w \leftarrow w + 1$
      if $w \le maxw$ then
        $(C, rangeC, M, rangeM)$
        $\leftarrow updateM(h, w, C, rangeC, M, rangeM, D)$
      end if
    end while
    $h \leftarrow h + 1, w \leftarrow 1$
    if $h \le min(l, size)$ then
      $maxw \leftarrow min(m, \lfloor \frac{size}{h} \rfloor)$
      $rangeC[2] \leftarrow max(rangeC[2], g - maxw + 1)$
      $rangeC[3] \leftarrow min(rangeC[3], g + maxw - 1)$
      $(C, rangeC) \leftarrow updateC(h, C, rangeC, D)$
    end if
  end while
  return $best$
```

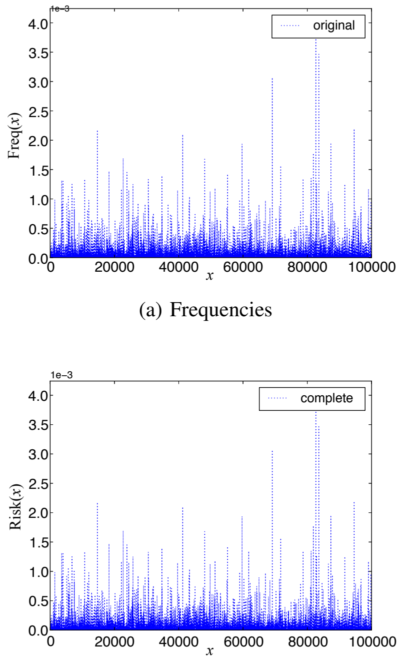
Fig. 3: Computation of $H_{rm}(x)$.

As with the 1D problems, this calculation can be computed infrequently and retained for future uses. For the 2D exact match case, further efficiency would be gained by adapting the method to process a batch of search intentions at once rather than finding handled sets for one search intention at a time.

IV. Evaluation

We evaluate relaxed frequencies (observed frequencies by servers), query risks, and costs over a real data set. We use the *Last.fm 1K data set*[2] which is a query log from the Last.fm music service. The dataset contains 992 users, 1 084 871 songs, and 19 150 868 queries. We sampled 100 000 songs and 1 800 145 queries from the original data set.

[2] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html



(a) Frequencies

(b) Query risks of complete protocols

Fig. 4: Overview of a sampled Last.fm data set.

The frequencies of search intentions in the sampled Last.fm data set are shown in figure 4a. Items in the database are sorted alphabetically and by song title. The x-axis in this figure represents search intentions x, and the y-axis shows the number of times users requested item t_x, which is associated with the search intention x. Figure 4b shows query risks of complete protocols. High frequency items have more risks than low frequency items.

Figure 5 shows observed frequencies, query risks, and costs when users employ our relaxation methodologies, and Table I shows maximum, minimum, and average query risks and costs. Figure 5a describes that observed frequencies are skewed, Again, the x-axis in this figure represents search intentions x, and the y-axis shows the number of times users requested item t_x, which is associated with the search intention x. Figure 5b shows query risks from relaxed handled sets. Because of relaxation, in most cases, those risks are bigger than those of the complete protocols shown in figure 4b. From table Ia, minimum and average query risks are in fact bigger than those of complete protocols. However, any risks from the relaxed handled set do not exceed the maximum risks of the complete protocol, ie. that the relaxed handled set is frequentistically private. Figure 5c shows query processing costs for servers. These plots illustrate that our relaxation methodologies reduce costs in most cases. From table Ib, the maximum cost is equal to that of complete protocols, but in other cases, the costs are much smaller. The average cost is 6.5% that of complete protocols.

We also have evaluations on a sorted Last.fm data set. The data set is constructed by sorting the sampled Last.fm data in order of frequency, so that t_1 is the most requested song and t_n is one of the most infrequent songs. The frequencies in this data set are shown in figure 6a. In this graph, x-axis means search intentions x and the y-axis means frequencies of those search intentions. Note that the y-axis is plotted by log scale.

Figure 6 and table II compare query risks and costs between complete protocols and our relaxed protocol. In all figures of figure 6, x-axes represent search intentions and y-axes are log scale. Query risks in our relaxed protocol stay below the maximum acceptable risk, ie. the maximum query risk of a complete protocol. Table IIa also supports this result, and the difference between the minimum and maximum risk is quite small. On the other hand, query processing costs in the relaxed protocol are reduced from those in the complete protocols as we can see in figure 6c and table IIb. For these tests, the average cost of the relaxed protocol is only 1.1% that of a complete protocol. Comparing alphabetically ordered search intentions and frequency ordered search intentions, we see that sorting by frequency improves query processing costs of servers much more. In this evaluation, we see improvement by a factor of over 5. Therefore, sorting all items by frequency order before deploying databases is a better strategy. We assume the frequencies of all items are public, so that users can convert alphabetically ordered search intentions to frequency ordered search intentions easily.

V. Conclusion

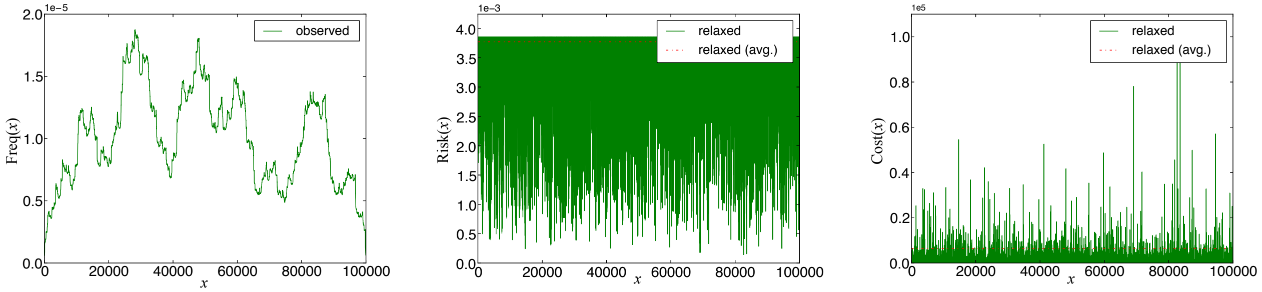
We introduced a frequency-based constraint relaxation methodology for private queries. This methodology was motivated by the knowledge that most existing private querying protocols provide a high level of privacy but impose an extreme computational cost on servers. We relaxed constraint (ii) of the complete protocols so that only a subset of the database is retrieved for each query. Although using a smaller set will increase the query risk, query risks with our method are not worse than in the worse case for the complete protocol.

We presented dynamic programming algorithms for three scenarios: 1D exact match, 1D range, and 2D exact match queries. In each case, we demonstrated how partial sums can be stored and reused to efficiently calculate the handled set for a given search intention. We leave an open problem for future work, which is that an attacker may be able to use knowledge about the size of the handled set to better guess a users' search intention. One simple way to reduce this vulnerability might be to add noise to the choice of handled set by randomly increasing the size of the handled set.

We evaluated our methodology using a real dataset from Last.fm. The results confirm that our protocol can reduce computational costs in servers in most cases, and that the risk of a query being exposed is not bigger than the maximum risk in complete protocols.

Acknowledgment

This work is partly supported by The Nakajima Foundation, Artificial Intelligence Research Promotion Foundation, and Grant-in-Aid for Young Scientists (B) (26730065), Japan Society for the Promotion of Science (JSPS).



(a) Frequency

(b) Query risk

(c) Cost

Fig. 5: Observed frequency, query risk, and cost of a sampled Last.fm data set.

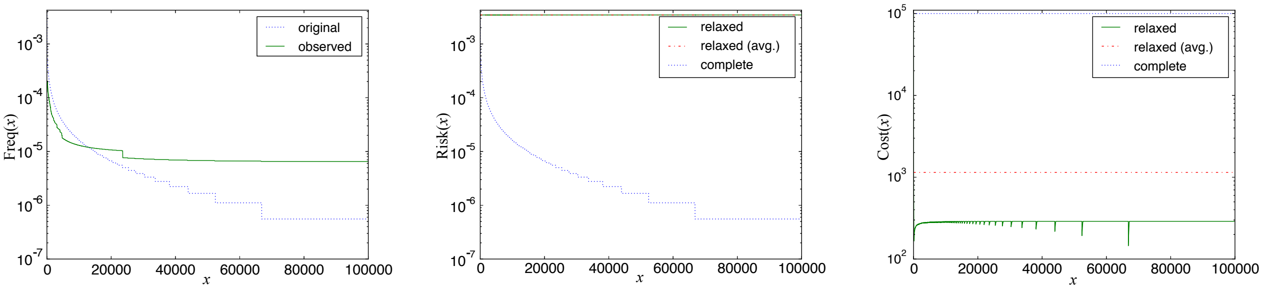
TABLE I: Comparison of query risk and cost (dictionary order).

(a) Query risk

	min.	max.	avg.
complete	5.5551×10^{-7}	3.8564×10^{-3}	1.0000×10^{-5}
relaxed	1.4217×10^{-4}	3.8564×10^{-3}	3.7710×10^{-3}

(b) Cost

	min.	max.	avg.
complete	1.0×10^5	1.0×10^5	1.0×10^5
relaxed	2.0	1.0×10^5	6.4176×100^3



(a) Frequency

(b) Query risk

(c) Cost

Fig. 6: Observed frequency, query risk, and cost of a sorted Last.fm data set (sorted by frequencies).

TABLE II: Comparison of query risk and cost (frequent order).

(a) Risk

	min.	max.	avg.
complete	5.5551×10^{-7}	3.8564×10^{-3}	1.0000×10^{-5}
relaxed	3.3769×10^{-3}	3.8564×10^{-3}	3.4590×10^{-3}

(b) Cost

	min.	max.	avg.
complete	1.0×10^5	1.0×10^5	1.0×10^5
relaxed	1.45×10^2	1.0×10^5	1.1466×10^3

References

[1] E. Kushilevitz and R. Ostrovsky, “Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval,” in *Proc. of the 38th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1997, pp. 364–373.

[2] J. Kawamoto and M. Yoshikawa, “Private Range Query by Perturbation and Matrix Based Encryption,” in *Proc. of the Sixth IEEE International Conference on Digital Information Management*. Melbourne, Australia: IEEE Computer Society, 2011, pp. 211–216.

[3] S. Wang, D. Agrawal, and A. E. Abbadi, “Generalizing PIR for Practical Private Retrieval of Public Data,” in *Proc. of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy*. Rome, Italy: Springer, 2010, pp. 1–16.

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, “Order Preserving Encryption for Numeric Data,” in *Proc. of the 23rd ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM Press, 2004, pp. 563–574.

[5] A. Boldyreva, N. Chenette, Y. Lee, and A. O’Neill, “Order-Preserving Symmetric Encryption,” in *Proc. of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Cologne, Germany: Springer-Verlag, 2009, pp. 224–241.

[6] A. Boldyreva, N. Chenette, and A. O. Neill, “Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions,” in *Proc. of the 31st International Cryptology Conference*, Santa Barbara, CA, USA, 2011, pp. 578–595.

[7] Y. Lu, “Privacy-Preserving Logarithmic-time Search on Encrypted Data in Cloud,” in *Proc. of the 19th Annual Network & Distributed System Security Symposium*, San Diego, CA, USA, 2012.

[8] R. Yoshida, Y. Cui, R. Shigetomi, and H. Imai, “The practicality of the keyword search using PIR,” in *Proc. of the International Symposium on Information Theory and its Applications*. Auckland, New Zealand: IEEE Computer Society, 2008, pp. 1–6.