

# A New Privacy Enhancing Method for Personalized Search

Yunsang Oh†

Hyounghick Kim‡

Takashi Obi†

†Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology  
4259-G2-2 Nagatsuta, Midori-ku, Yokohama 226-8502, JAPAN  
oh.y.ab@m.titech.ac.jp, obi@ip.titech.ac.jp

‡Computer Laboratory, University of Cambridge  
JJ Thomson Avenue, Cambridge, CB3 0FD, United Kingdom  
hk331@c1.cam.ac.uk

**Abstract** For personalized search, a user must provide her personal information. However, this may include the user’s sensitive information about privacy. The most serious concern is that the service provider itself can be an adversary who misuses the sensitive information. Our aim is to protect user privacy by controlling the sensitive information exposed from a user’s query inputs. We propose a framework called query generalizer which is a middleware that takes a query for personalized search, modifies the query to protect against exposure of users’ sensitive personal information, and then forwards the modified query to the service provider. Our simulation results show that our framework is capable of achieving a low traffic overhead within a reasonable range of user privacy. Moreover, we examine that a simple randomized algorithm practically performs well for query generalization.

## 1 Introduction

With explosive growth in number of information sources, personalized search services are increasingly popular and promising in Web portals and search engine sites. For example, Amazon displays personalized product recommendations based on a user’s purchase history [1]. However, for personalized search, a user must expose her private information such as health conditions, education attainment, gender, lifestyle and preferences. Unfortunately, it’s not enough just to protect the communication channel between user and service provider since service providers are not completely trustworthy. The collected personal data can potentially be misused for the service providers’ commercial advantage. For example, some service providers can misuse users’ personal data for marketing purposes since they wish to advertise their product to targeted users well. So we should consider this potential privacy concern as an important line for personalized search services.

A possible approach is to use pseudonyms instead of a user’s real user identity to hide who wants to ask a query. However, it is not suffi-

cient to preserve user privacy effectively. The most significant issue is that an adversary can infer the identity of querier with the querier’s partial information exposed only. This attack can be achieved by linking the partial information, called *quasi-identifier* [2], to the auxiliary information (also called external knowledge) collected from other channels such as the Web or public records. For example, Sweeney showed that 87% (216 million of 248 million) of the population in the United States can be uniquely identified based only on zip code, gender and date of birth with the voter registration list, which is publicly available [3]. This result was recently updated by Golle. He showed that 63% of the population in the United States can be uniquely identified [4].

For this problem, the current mainstream solution is to publish anonymized datasets. Samarati and Sweeney introduced the concept of “*k*-anonymity” [5, 6] and many techniques [7, 8, 9, 10] have been proposed based on the similar concepts. However there are some limitations. First, the performance cost of the anonymization process may be very huge, es-

pecially for large and sparse databases. Second, the expensive update cost is also required to maintain anonymized data even if only a few records are newly inserted, deleted, or modified. Third, it can drastically reduce the quality of the released datasets because of generalization.

Alternatively, we propose a novel technique without applying anonymization techniques. The main idea here is to restrict users’ queries by filtering out *quasi-identifier* in a reasonable manner at the system level so that an adversary cannot infer the querier’s real identity with a high probability. Our framework can be flexibly extended to protect not only a user’s identity but also user’s characteristics such as race, ethnicity, gender and educational attainment. This paper contributes in the following areas:

1. we propose the architecture of our privacy preserving system using a middleware called **query generalizer**, which filters out sensitive query inputs (read Section 3).
2. we demonstrate the feasibility of our method by evaluating its performance with two intuitive query generalization algorithms (read Section 4), brute-force and random filtering out, to mitigate the exposure of user’s sensitive information properly (read Section 5).

## 2 Threat Model

We assume that a computationally unlimited adversary, who is able to observe the query transmitted from a user to a service provider, tries to identify the user’s personal information such as race, ethnicity, gender and educational attainment. For example, given a user’s gender and age in the captured query, an adversary can try to guess the querier’s race by linking with (public) auxiliary information such as Table 1. Here, we consider service provider itself as an adversary (see Figure 1).

For auxiliary information, standard database notations are used here and in the rest of the

Table 1: An Example of Auxiliary Information

Loan	Income	Race	Gender	County
\$100k	\$50k	Asian	Male	Barnstable
\$20k	\$20k	White	Male	Dukes
\$20k	\$40k	Black	Female	Essex
\$20k	\$40k	Black	Female	Dukes
\$20k	\$20k	Black	Male	Hampden
\$40k	\$30k	White	Male	Hampshire
\$10k	\$20k	Asian	Male	Essex
\$20k	\$100k	White	Female	Norfolk
\$100k	\$500k	White	Male	Bristol
\$10k	\$20k	White	Male	Suffolk
\$40k	\$30k	Asian	Female	Norfolk
\$20k	\$100k	Black	Female	Essex

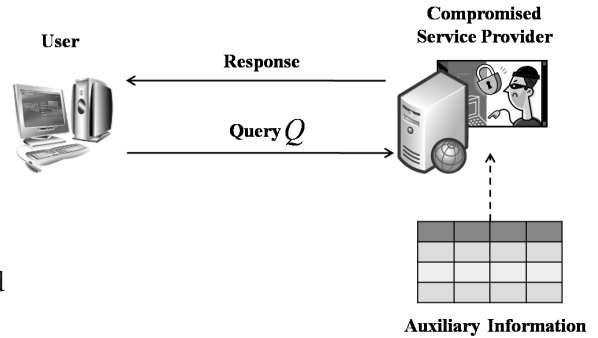


Figure 1: Threat Model

paper. A *domain* is a finite set of mutually exclusive and exhaustive values. Let  $t$  be a tuple consisting of  $m$  attributes,  $a_i$  be an attribute of  $t$  and  $D_i$  be a domain of  $a_i$  for  $i \in [1, m]$ . An attribute  $a_i$  is a mapping from a set of tuples to a domain  $D_i$  and  $t.a_i$  represents the mapped value in a domain  $D_i$ . For example, the attribute  $a_4$ , [Gender], is a mapping to a domain  $D_4 = \{“Female”, “Male”\}$  and  $t.a_4$  is “Male”.

In the view of the adversary with the auxiliary information, we define a query  $Q$  from a user as a set of  $m$  attribute values regarding the auxiliary information,  $\{t.a_i = v_i\}$  for  $i \in [1, m]$  and  $v_i \in D_i \cup \{\perp\}$  where  $\perp$  means “don’t care” value and  $m$  is the number of attributes in the auxiliary information. The inclusion of  $\perp$  is used to represent the attributes

that are not used in a query. For example, a query “*Find jobs for pregnant black women with the \$20k amount of loan.*” can be formally interpreted as  $Q = \{[\text{Loan}] = \text{“\$20k”}, [\text{Income}] = \perp, [\text{Race}] = \text{“Black”}, [\text{Gender}] = \text{“Female”}, [\text{County}] = \perp\}$  with Table 1.

We now turn to a measure to quantify how secure a personal query  $Q$  is against guessing attacks. This measure can be defined with *entropy*. We consider the value of the *target attribute* to be a random variable  $X$  drawn from a finite distribution  $P = \{p_1, p_2, \dots, p_n\}$  which is known to the adversary and probability  $p_i = P(X = x_i)$  for each possible answer  $x_i$  for  $i \in [1, n]$  ( $p_1 \geq p_2 \geq \dots \geq p_n$ ). We assume that the adversary’s goal is to find the secret using as few guesses as possible. Under this threat model, supposing that the adversary knows the distribution for  $X$ , the best strategy is obviously to start by guessing the most likely  $X$  and proceed in decreasing order of likelihood until the secret is determined. This entropy is known well as the *guessing entropy* introduced by Massey [11]. The guessing entropy is simply the expected number of trials needed to correctly guess the value of a random variable  $X$  using the best strategy. For example, given a query  $Q = \{[\text{Loan}] = \text{“\$20k”}, [\text{Income}] = \perp, [\text{Race}] = \text{“Black”}, [\text{Gender}] = \text{“Female”}, [\text{County}] = \perp\}$ , we found 3 tuples corresponding to the query  $Q$ , which means the 3rd, 4th, and 12th tuples in Table 1. Suppose that the adversary’s goal is to guess the querier’s value of  $[\text{Income}]$ . There are two possible values for the  $[\text{Income}]$  attribute: \$40k (3rd and 4th) and \$100k (12th). Thus the *guessing entropy* for the query  $Q$  and the  $[\text{Income}]$  can be computed as follows:

$$G(X_Q, [\text{Income}]) = \frac{2}{3} \cdot 1 + \frac{1}{3} \cdot 2 = \frac{4}{3} \quad (1)$$

In other words, the expected number of trials to guess the querier’s the value of  $[\text{Income}]$  corresponding to the query  $Q$  is  $4/3$ . For the tuples satisfying the conditions of a query  $Q$ , the entropy  $G$  to guess the value of an attribute  $a$  is formally defined as follows:

$$G(X_Q, a) = \sum_{i=1}^n p_i \cdot i \quad (2)$$

### 3 The Proposed Architecture

Conventional personalization services are generally based on standard client-server architecture: “user” and “service provider”. A user sends a query including her personal information to a service provider and receives customized search results. We extend this to a framework for privacy-enhancing personalized search by introducing a new component called **query generalizer**. **Query generalizer** generates a *privacy-preserving* query  $\hat{Q}$  against guessing attacks from an adversary with the auxiliary information and the captured query inputs. In practice, a user’s original query  $Q$  is likely to be generated without serious privacy consideration. Therefore we need to filter out some highly sensitive private information in the original query  $Q$  at the system level. A secure channel is necessarily required to securely deliver the original query  $Q$  from user to **query generalizer** against eavesdroppers. This requirement can be easily established without extra efforts since Secure Sockets Layer/Transport Layer Security (SSL/TLS) [12] is already available and supported by mainstream web browsers (e.g. Internet Explorer, Safari, Firefox, Opera and Chrome).

**Query generalizer** takes the original query  $Q$  from a user, modifies  $Q$  by removing some highly sensitive information depending on the privacy requirement of the user, and then forwards the modified query  $\hat{Q}$  to a service provider. **Query generalizer** must access the auxiliary information (or statistics of the information) to estimate the amount of information included in queries. The service provider computes the search results for the modified query  $\hat{Q}$  and answers them to **Query generalizer**. After receiving the query response, **query generalizer** selectively retrieves the exact search results for the original query  $Q$  from the query response for  $\hat{Q}$  and then relays them to the user. We note that the search results in our framework are exactly the same as the search

results for the original query  $Q$  since  $\hat{Q}$  is a generalization of  $Q$ .

## 4 Query Generalizer

The query generalization process at **query generalizer** is based on computation of the privacy level of a user query. Given a user’s query, **query generalizer** checks whether the query achieves the user’s desired privacy level. Otherwise, **query generalizer** incrementally updates by removing some sensitive attribute values from the query. This process is repeated until the generalized query satisfies the user’s privacy requirement.

An issue here is how to set the user’s privacy requirement. In this paper, we assume that a user  $u$ ’s privacy requirement is formally defined as a tuple  $(a, G_{min})$  where  $G_{min}$  indicates the required minimum guessing entropy to limit the information about the attribute  $a$ , which can be obtained from her query. We define that a query  $Q$  is *privacy-preserving* for an attribute  $a$ , if the guessing entropy  $G(X_Q, a)$  from the query  $Q$  is greater than or equal to  $G_{min}$  that a user requires. This implies that given the query  $\hat{Q}$ , the adversary must try to guess the user  $u$ ’s specific value for  $a$  more than  $G_{min}$  times on average.

It seems important to choose which attribute values being removed to generalize a user’s query since the size of *traffic overhead* changes depending the modified queries. Unfortunately, finding a *privacy-preserving* query to minimize the *traffic overhead* is infeasible since it has no idea about the service provider’s resources. In fact, finding the optimal *privacy-preserving* query is *NP-hard* even if the service provider’s resources is completely given. This problem can be reduced to the subset sum problem, which is a well-known *NP-hard* problem [13]. Therefore, we consider two following heuristics based on the belief that the *traffic overhead* becomes larger as the *guessing entropy* for a query increases.

1. **Brute-Force:** In this strategy, **query generalizer** examines all possible subsets of the attribute values in  $Q$  and se-

lects a subset  $C$  with the minimum guessing entropy  $G(X_C, a)$  where  $G(X_C, a) \geq G_{min}$ . If more than two subsets have an identical entropy value, one of them is randomly selected.

2. **Random-Fit:** In this strategy, **query generalizer** starts with a candidate query  $C$  that is the same as the original query  $Q$ . **Query generalizer** iteratively updates  $C$  by substituting  $\{C.a_i = v_i\}$  with  $\{C.a_i = \perp\}$  at a time where  $v_i$  is randomly selected and  $v_i \neq \perp$  for  $i \in [1, m]$  until  $G(X_C, a) \geq G_{min}$ .

In Section 5, we discuss which strategy is more recommendable using the simulation on a real dataset.

## 5 Simulation

We experimentally evaluate our framework by measuring the *traffic overhead* between **query generalizer** and a service provider. The *traffic overhead* is measured in terms of the number of records included in a query response from the service provider to **query generalizer**, i.e. the number of tuples in service provider’s database corresponding to a query. We note that the size of query inputs can be ignored since it is relatively too small compared to the size of a query response in practice.

We used the Home Mortgage Disclosure Act (HMDA) Loan Application Register Data in year 2007 from U.S. census bureau<sup>1</sup> for an auxiliary information. For a service provider’s database, we used the same database to maximize the effect of the auxiliary information. From the database, we selected 66,270 people who were denied the loan at least once and live in Massachusetts, United States. Among attributes in the database, we selected 6 attributes - [Loan Amount]<sup>2</sup>, [Income]<sup>3</sup>, [Denial

<sup>1</sup><http://www.census.gov/>

<sup>2</sup>Numeric values categorized by 236 groups

<sup>3</sup>Numeric values categorized by 206 groups

Reason]<sup>4</sup>, [Race]<sup>5</sup>, [Gender]<sup>6</sup> and [County]<sup>7</sup>. We assume that users want to hide the information about their [Income] values by varying  $G_{min}$  from 1 to 7.

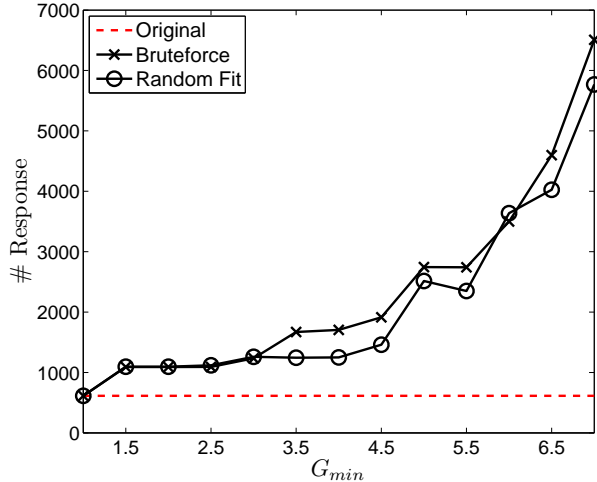


Figure 2: Traffic Overhead for [Income]

We simulated real users’s queries by generating 100 queries with randomly selected attribute values in the given domain including the “don’t care” value. For each input query, we tested the **Brute-Force** and **Random-Fit** strategies discussed in Section 4. Figure 2 demonstrates the average *traffic overheads* of these strategies for [Income] attribute. For comparison, we also plotted the *original query* (the dashed red line in Figure 2) as an absolute lower bound.

This result indicates that we might determine the optimal tradeoff between *user privacy* and *traffic overheads*. In Figure 2, while the slope increases rapidly when  $G_{min} > 4.5$ , the generalized query incurs a slight increase in *traffic overhead* when  $G_{min} \leq 4.5$  compared to the original query. In other words, our framework is capable of achieving a strong *user privacy* and a reasonable *traffic overhead* within some privacy levels ( $G_{min} \leq 4.5$ ). For example, the proposed framework provides com-

<sup>4</sup>9 values such as “Debt to income ratio”, “Credit history”, etc

<sup>5</sup>7 values such as “American Indian”, “Alaskan Native”, etc

<sup>6</sup>4 values including “M”, “F” and two others for data not open

<sup>7</sup>14 counties in Massachusetts, United States

parable performance with the original query when  $G_{min} \leq 4.5$  even if **Random-Fit** strategy is used. When  $G_{min} = 4.5$ , the **Random-Fit** strategy only increases 2.26 times than the *traffic overhead* of the original query. Considering the recent growth in networking technology, it is not a big penalty.

Another interesting observation is that the performance of **Brute-Force** is not better than that of **Random-Fit** in practice. More exactly, **Random-Fit** outperformed **Brute-Force** slightly except  $G_{min} = 6$ . It means that the simple **Random-Fit** may be enough for our purpose without incurring significant additional computational expense at **query generalizer**.

## 6 Conclusion

We proposed a framework based on the concept of **query generalizer** which effectively controls highly private information for personalized services. If a trustworthy **query generalizer** is provided by the third party such as the government, users can access privacy-enhancing personalized services by using mainstream web browsers alone without the installation of an extra software.

In this paper we presented two query generalization strategies, the **Brute-Force** and **Random-Fit**. Through the simulation, we empirically analyzed and compared the performances of two strategies. Interestingly, it is hard to say which is the better strategy as the performances are very close. Therefore we will search more advanced query generalization strategies to provide better results.

The computation of entropy values may greatly increase computational overheads as the amount of auxiliary information is increased. Therefore we need to consider how to efficiently calculate entropy values. We suggest two practical ideas: approximation using sampling of tuples and the use of pre-computation for frequently asked queries. We consider extending our work to reducing the computation overhead of entropy values as important lines for future work.

As part of the future work, we could extend this work to a real application such as

the personalized healthcare information infrastructure [14] considered to be built by Japanese government to improve the disease prevention, and the quality and efficiency of health care. This service is built with highly robust security requirements in order for users to take control over their own health information privately. We expect that our proposal will be also practically used for other applications such as “job recommendation” and “academic survey”.

## References

- [1] P. Maglio and R. Barrett, “Intermediaries personalize information streams,” *Communications of the ACM*, vol.43, no.8, pp.96–101, 2000.
- [2] T. Dalenius, “Finding a needle in a haystack or identifying anonymous census records,” *Journal of Official Statistics*, vol.2, no.3, pp.329–336, 1986.
- [3] L. Sweeney, “Uniqueness of simple demographics in the U.S. population,” LIDAP-WP4 Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, 2000.
- [4] P. Golle, “Revisiting the uniqueness of simple demographics in the US population,” *WPES ’06: Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, New York, NY, USA, pp.77–80, ACM, 2006.
- [5] P. Samarati, “Protecting respondents’ identities in microdata release,” *IEEE Transactions on Knowledge and Data Engineering*, vol.13, no.6, pp.1010–1027, 2001.
- [6] L. Sweeney, “ $k$ -anonymity: a model for protecting privacy,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol.10, no.5, pp.557–570, 2002.
- [7] V.S. Iyengar, “Transforming data to satisfy privacy constraints,” *KDD ’02: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, pp.279–288, ACM, 2002.
- [8] R.J. Bayardo and R. Agrawal, “Data privacy through optimal  $k$ -anonymization,” *ICDE ’05: Proceedings of the 21st International Conference on Data Engineering*, Washington, DC, USA, pp.217–228, IEEE Computer Society, 2005.
- [9] B.C.M. Fung, K. Wang, and P.S. Yu, “Top-down specialization for information and privacy preservation,” *ICDE ’05: Proceedings of the 21st International Conference on Data Engineering*, Washington, DC, USA, pp.205–216, IEEE Computer Society, 2005.
- [10] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan, “Mondrian multidimensional  $k$ -anonymity,” *Data Engineering, 2006. ICDE ’06. Proceedings of the 22nd International Conference on Data Engineering*, ed. L. Liu, A. Reuter, K.Y. Whang, and J. Zhang, Washington, DC, USA, p.25, IEEE Computer Society, April 2006.
- [11] J.L. Massey, “Guessing and entropy,” *Proceedings of the IEEE International Symposium on Information Theory*, p.204, 1994.
- [12] T. Dierks and C. Allen, “The TLS protocol version 1.0.” RFC 2246 (Informational), 1999.
- [13] M.R. Garey and D.S. Johnson, *Computers and intractability; a guide to the theory of NP-completeness*, W. H. Freeman & Co., New York, NY, USA, 1990.
- [14] K. Kita, J. Lee, H. Suzuki, N. Taira, M. Yachida, H. Yamamoto, Y. Homma, T. Obi, M. Yamaguchi, and N. Ohyama, “The personal health information reference system based on e-P.O.Box conception,” *Korean Society of Medical Informatics*, vol.14, no.3, pp.213–220, 2008.