PAPER Special Section on Trust, Security and Privacy in Computing and Communication Systems

Privacy-Enhancing Queries in Personalized Search with Untrusted Service Providers

Yunsang OH[†a)], **Hyoungshick KIM**[††b)], ***Nonmembers***, ***and*** **Takashi OBI**[†c)], ***Member***

SUMMARY For personalized search, a user must provide her personal information. However, this sometimes includes the user's sensitive information about individuals such as health condition and private lifestyle. It is not sufficient just to protect the communication channel between user and service provider. Unfortunately, the collected personal data can potentially be misused for the service providers' commercial advantage (e.g. for advertising methods to target potential consumers). Our aim here is to protect user privacy by filtering out the sensitive information exposed from a user's query input at the system level. We propose a framework by introducing the concept of *query generalizer*. *Query generalizer* is a middleware that takes a query for personalized search, modifies the query to hide user's sensitive personal information adaptively depending on the user's privacy policy, and then forwards the modified query to the service provider. Our experimental results show that the best-performing query generalization method is capable of achieving a low *traffic overhead* within a reasonable range of *user privacy*. The increased *traffic overhead* varied from 1.0 to 3.3 times compared to the original query.

key words: *privacy, personalized search, privacy-enhancing query, query generalizer*

1. Introduction

With explosive growth in number of information sources, personalization is becoming a key requirement for users. It seems quite profitable for service providers so that they can selectively offer services or products to the end-users who are more interested and involved in them. For example, Amazon recommends some personalized products for a user based on the user's purchase history [1].

Intuitively, for personalization, a user must expose her specific information. However, this sometimes includes the user's sensitive information such as education attainment, gender, lifestyle and preferences or be clue to infer them. In particular, among users who seek electronic medical information, personal privacy was ranked as their most important concern [2]. Therefore personalized services cannot be popularly deployed without considering privacy concerns.

In this paper, we particularly examine how a privacy-enhancing personalized search can be designed and how effective it can be in mitigating the possible privacy threats. Unfortunately, service providers can sometimes misuse the user data for their commercial advantages. For instance, popular social network services such as Facebook and MySpace have intentionally exposed users' personal data to increase the number of their subscriber members [3]. Thus we cannot assume that the service provides are trustworthy. Privacy concerns about identifying a querier from search query have recently been discussed in detail by Dolin [4].

A possible approach is to use anonymous communication such as Tor network [5] to hide who wants to ask a query. At the first glance, it seems enough to provide a reasonable anonymity since the querier cannot be linked with the real user. In practice, however, the use of only anonymous communication is not sufficient; an adversary can infer the identity of the querier from her partial information exposed in the query. This attack can be achieved by linking the partial information, called *quasi-identifier* [6], to the auxiliary information 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 [7]. This result was recently updated by Golle. He showed that 63% of the population in the United States can be uniquely identified [8].

For this problem, the current mainstream research trend is to publish anonymized datasets through randomized or cryptographic techniques [9]–[11] to add noises to data records to achieve the privacy goals such as k-anonymity so that a record cannot be distinguished from at least $k-1$ other records. Although this approach was studied intensively in the context of public data releases, there are three significant limitations. First, the performance cost of the anonymization process may be very huge, especially for large and sparse databases. Second, the expensive update cost is also required to maintain anonymized datasets even if only a few records are newly inserted, deleted, or modified. Third, much more critical drawback is that it can drastically reduce the quality of the released datasets since anonymization is typically enforced through generalization [12].

It is in this spirit that we propose a framework that minimizes the data anonymization overheads at the time of database release. Instead of putting all the burden of anonymization on the preprocessing step before the release of database, this cost is spread to each query processing step: we introduce the concept of *query generalizer* to add random noises at the time of query processing. The main

Manuscript received March 15, 2011.
Manuscript revised June 28, 2011.

[†]The authors are with the Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology, Yokohama-shi, 226–8502 Japan.

[††]The author is with the Computer Laboratory, University of Cambridge, United Kingdom.

a) E-mail: oh.y.ab@m.titech.ac.jp
b) E-mail: hk331@cl.cam.ac.uk (Corresponding Author)
c) E-mail: obi@ip.titech.ac.jp

DOI: 10.1587/transinf.E95.D.143

idea here is to generalize users' queries including *quasi-identifiers* simply so that an adversary cannot obtain the querier's sensitive information from the queries; when a user tries to access a service with her personal information, a *query generalizer* should disallow this query since it may include highly sensitive information. Instead, her query is modified by removing some sensitive information selectively and the modified query is delivered to the service provider. Our framework makes this process transparent to the end-user. This paper contributes in the following areas:

1. We propose the architecture of our privacy preserving system using *query generalizer*. We discuss the pros and cons of two possible deployments for our framework and recommend three-tier architecture where *query generalizer* is independently deployed on the trusted third party. We explain why three-tier architecture is more proper for the proposed framework (read Sect. 3).
2. We discuss how to design the simple and efficient query generalization algorithm to modify the user's query which mitigates the exposure of private user information. We consider four reasonable schemes: Brute-Force, Best-Fit, Worst-Fit and Random-Fit strategies (read Sect. 4) and experimentally evaluate the proposed query generalization algorithms with real datasets. Simulation results show these schemes generally provide a low *traffic overhead* within a reasonable range of *user privacy*. We recommend the Worst-Fit strategy; it can provide privacy-preserving queries with the *traffic overhead* comparable to that of the original query (read Sect. 5).

2. Threat Model

We assume that an untrusted service provider has unlimited computational power (i.e. the information theoretic notion of security). After observing the query from a user without any computational restriction, the service provider can try to identify the user's personal information uniquely by combining the query with auxiliary information (also called external knowledge, background knowledge, or side information) that the service provider gleans from other channels such as the web, public records, or domain knowledge. Here our goal is to secure users' private information from compromised service providers.

Formally, auxiliary information can be modeled as database with its standard notation. 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, in Table 1, ("$100k", "$50k", "Asian", "Male", "Barnstable") is a tuple t. `[Loan]`, `[Income]`, `[Race]`, `[Gender]`, and `[County]` are attributes of the tuple. The attribute a_4, `[Gender]`, is a mapping to a domain D_4 = {"Female", "Male"} and $t.a_4$ is "Male".

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

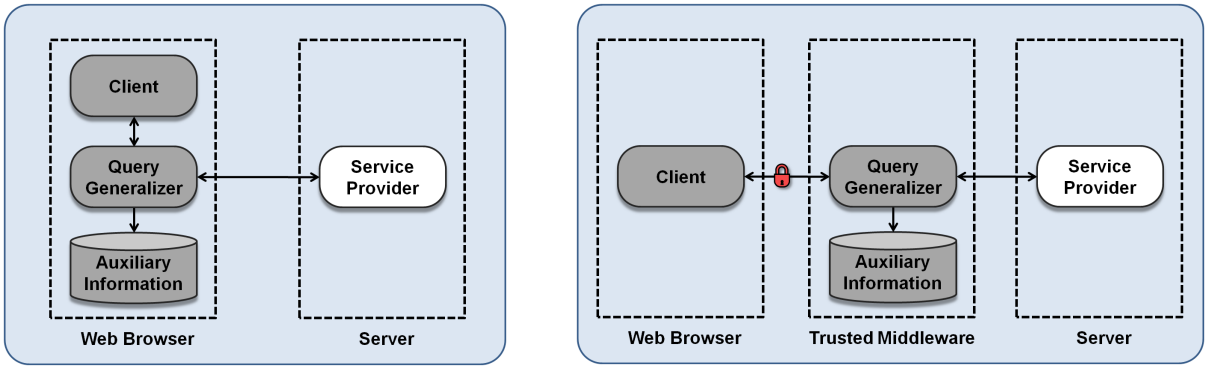
In the view of the adversary with the auxiliary information, we define a query Q from a user as an ordered list of m attribute values, $(v_1, v_2, \cdots, v_m)$, for $v_i \in D_i \cup \{*\}$ where $*$ means "don't care" value and m is the number of attributes in the auxiliary information. The inclusion of $*$ is used to represent the attributes that are not used in a query. For example, the real query "*Find jobs for pregnant black women with the $20k amount of loan.*" can be formally interpreted as Q = (`[Loan]` = "$20k", `[Income]` = $*$, `[Race]` = "Black", `[Gender]` = "Female", `[County]` = $*$) with Table 1. We note that in the view of the adversary, the information about "pregnancy" is ignored since it is not included in this auxiliary information. The information about "loan", "race" and "gender" is only used for guessing attacks although the information about "pregnancy" is exposed.

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 $P = \{p_1, p_2, \cdots, p_n\}$ which is known to the adversary and probability $p_i = P(X = x_i)$ for each possible answer x_i for $i \in [1, n]$.

The *Shannon entropy* is a traditional estimator of measuring the information of X as follows (here, log is to the base 2):

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i \tag{1}$$

Probably, this measure is generally good to show the amount of information leaked. However, it is not enough depending on what the adversary is able to do. When the adversary has an oracle which can confirm or refute her guess, a better metric is *the average number of guesses* required to guess the secret. The adversary's goal is to find the secret using as few guesses as possible. In practice, many popular web services provide functionalities (oracle) to answer adversaries' personal questions [13]. 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 [14]. The guessing



(a) Two-Tier Architecture (b) Three-Tier Architecture

Fig. 1 Two different deployments. The *gray* components represent newly added roles (client, query generalizer) and a database (auxiliary information).

entropy is simply the expected number of trials needed to correctly guess the value of a random variable X using the best strategy. It can be calculated as follows: Given N tuples corresponding to a query Q, the entropy G to guess the value of an attribute a is formally defined as follows:

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

where p_i is defined as $|x_i|/N$ for all n distinct domain values $\{x_1, x_2, \cdots, x_n\}$ of the attribute a that appear in the tuples corresponding to the query Q and $|x_i|$ denotes the number of the attribute value x_i appeared in the corresponding tuples. Without loss of generality, we assume that the probabilities are arranged as a monotonically decreasing distribution with $p_1 \geq p_2 \geq \cdots \geq p_n$.

For example, given a query Q = ([Loan] = "$20k", [Income] = $*$, [Race] = "Black", [Gender] = "Female", [County] = $*$), we found 3 tuples (i.e. the 3rd, 4th, and 12th) corresponding to the query Q in Table 1. Suppose that the adversary's goal is to guess the querier's income information. There are two possible values for the [Income] attribute: $40k (3rd and 4th) and $100k (12th). Thus the *guessing entropy* for the query Q and the [Income] can be computed as follows:

$$G(Q, \texttt{[Income]}) = \frac{2}{3} \cdot 1 + \frac{1}{3} \cdot 2 = \frac{4}{3} \tag{3}$$

In other words, the expected number of trials to guess the querier's the value of [Income] corresponding to the query Q is 4/3.

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 the service provider answers customized search results for the user using her personal information. We extend this to a framework for privacy-enhancing personalized search by introducing two functional components: Query Generalizer and Client.

The *query generalizer*'s role is to generate a *privacy-preserving* query $\widehat{Q}$ from the original query Q by removing some highly sensitive personal information depending on the privacy requirement of the user, and then forwards the modified query $\widehat{Q}$ to a service provider so that the service provider cannot guess the user's private information from 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 the user to the *query generalizer* against an adversary.

The *query generalizer* can access the auxiliary information (or statistics of the information) to estimate the amount of information included in queries. The service provider performs the search service with the modified query $\widehat{Q}$ and answers them to the *query generalizer*. After receiving the query response, the *query generalizer* selectively retrieves the exact search results for the original query Q from the query response for $\widehat{Q}$ and then relays them to the user. Thus the search results in our framework are exactly the same as the search results for the original query Q since $\widehat{Q}$ is a generalization of Q. In practice, *query generalizer* can be deployed as a physical entity together with other functional components or an independent physical entity.

Client serves as a proxy between a user and *query generalizer*. This role is generally integrated with other client-side technology (e.g. web browser) and may also include a user interface component. The primary purpose of *client* is to simply relay the user's search query to *query generalizer* instead of the service provider that the user originally wishes to interact with.

We here discuss the pros and cons of two possible deployments with *query generalizer* and *client* (see Fig. 1). Figure 1 (a) shows a deployment of two-tier architecture. Since most existing search services are based on a standard client-server architecture, two-tier architecture can be basically integrated into a general server-client architecture.

The *query generalizer* can be implemented as a thin-client software (e.g. a Web browser plug-in). This approach could provide a highly secure personalized search service without the assumption of secure channels between users and *query generalizer*. However, it has several drawbacks. First, it is not acceptable for a client to compute the entropy values in real time to generate a *privacy-preserving* query when there are too many tuples in the auxiliary information. This is because the computation overhead of entropy values is proportional to the number of tuples. Therefore we need to consider how to efficiently calculate entropy values. Second, it is not trivial to maintain the auxiliary information at the client since its size is often too huge and updated too much. Third, the communication cost is greatly increased between a client and a server compared to the original query for scenarios that require a strong privacy.

Alternatively, we recommend a deployment of three-tier architecture (see Fig. 1 (b)). This approach can solve the problems of two-tier architecture. The physical entity of "Trusted Middleware" allows personalized search environment without any concerns about the performance of *query generalizer*. In practice, secure channels between "User System" and "Trusted Middleware" can be easily established without extra efforts since Secure Sockets Layer/Transport Layer Security (SSL/TLS) [15] is already available and supported by mainstream web browsers (e.g. Internet Explorer, Safari, Firefox, Opera and Chrome).

Who's in charge of managing "Trusted Middleware" for three-tier architecture? Surely, the first candidate is a government agency. The government can support a regulated infrastructure to collect the auxiliary information and to guarantee the interoperability of all transactions processed for search services. There were already practical discussions [16], [17] to support three-tier architecture with a trusted government agency to oversee private matters. In this case, however, many users may resist using the proposed system due to the concern about government surveillance. In fact, a number of governments have been tempted to intervene in the technical aspects of security design, usually with effects that damage industry and lead government agencies to lose face. So we suggest that the proper role of government is to collect and to disseminate *auxiliary information* and/or its dependable statistics rather than to manage the technical mechanism directly. Under such circumstances, several (non-profit) standard bodies or organizations such as Tor Project (http://www.torproject.org/) can support free software and open infrastructure to provide truthful query generalization services.

In the next sections, we present how the user's original query is adaptively modified by using *query generalizer* and evaluate the performance of the presented algorithms.

4. Adaptive Query Generalization

The query generalization process 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, the *query generalizer* adaptively updates by removing some sensitive attribute values from the query. This process is repeated until the generalized query satisfies the user's privacy requirement.

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. With this definition, we also define that a query Q is *privacy-preserving* for an attribute a, if the guessing entropy $G(Q, a)$ from the query Q is greater than or equal to G_{min} that a user requires. This implies that given the query $\widehat{Q}$, the adversary must try to guess the user u's private value for a more than G_{min} times on average. In order to provide *privacy-preserving* queries, the query Q is modified to the query $\widehat{Q}$ by removing some private information included in Q until $G(\widehat{Q}, a) \geq G_{min}$ if the guessing entropy $G(Q, a)$ is less than G_{min}.

Surely, it is important to choose which attributes being removed to generalize a query since the *traffic overhead* of the query response is changed depending on how the query is modified. How can the *query generalizer* generalize the query Q with a little sacrifice of *traffic overhead*? Since the *query generalizer* cannot gain the exact information about the data maintained by the service provider, an appropriate strategy to minimize the *traffic overhead* may depend on the *guessing entropy* of the query. Probably, there exists a correlation between *guessing entropy* and *traffic overhead*; under the assumption of uniform distribution, the *traffic overhead* becomes larger as the *guessing entropy* increases.

So we now focus on finding the query to minimize the *guessing entropy* of the query while staying over G_{min}. This optimization problem, however, is also not trivial. The objective of this problem is to find a query $Q = (v_1, v_2, \cdots, v_m)$ and $v_i \in D_i \cup \{*\}$ with an attribute a to be protected and the minimum guessing entropy G_{min} such that the computed guessing entropy $G(Q, a)$ is at least G_{min} while it is minimized. For any attribute a, this problem can be formulated as follows:

$$\begin{aligned} &\text{minimize } f(x_1, x_2, \cdots, x_m) \\ &\text{subject to } f(x_1, x_2, \cdots, x_m) \geq G_{min} \end{aligned} \tag{4}$$

where $f(x_1, x_2, \cdots, x_m) = G(\widehat{v_1}, \widehat{v_2}, \cdots, \widehat{v_m}), a)$ such that $\widehat{v_i} = v_i$ if $x_i = 1$; otherwise, $\widehat{v_i} = *$.

Since f is a nonlinear objective function, this problem can be expressed as a Binary Integer Programming problem with a nonlinear objective function $f(x_1, x_2, \cdots, x_m)$, which is a well-known NP-hard problem [18]. This means that the *query generalizer* must use heuristics to find the attribute values being removed. Given a query $Q = (v_1, v_2, \cdots, v_m)$ and the guessing entropy constraint G_{min}, we consider the following four reasonable strategies to construct $\widehat{Q}$:

1. **Brute-Force**: This strategy examines all possible combinations of v_i and $*$ for each attribute in Q and select a combination $\widehat{Q}$ whose entropy $G(\widehat{Q}, a)$ is the mini-

mum where $G(\widehat{Q}, a) \geq G_{min}$. If more than two combinations have an identical entropy value, one of them is randomly selected.
2. **Best-Fit**: This strategy starts with a candidate query $\widehat{Q}$ that is the same as the original query Q. Then it iteratively updates $\widehat{Q}$ by substituting v_i with $*$ at a time to minimize the $G(\widehat{Q}, a)$ locally where $v_i \neq *$ for $i \in [1, m]$ until $G(\widehat{Q}, a) \geq G_{min}$.
3. **Worst-Fit**: This strategy starts with a candidate query $\widehat{Q}$ that is the same as the original query Q. Then it iteratively updates $\widehat{Q}$ by substituting v_i with $*$ at a time to maximize the $G(\widehat{Q}, a)$ locally where $v_i \neq *$ for $i \in [1, m]$ until $G(\widehat{Q}, a) \geq G_{min}$.
4. **Random-Fit**: This strategy starts with a candidate query $\widehat{Q}$ that is the same as the original query Q. Then it iteratively updates $\widehat{Q}$ by substituting v_i with $*$ at a time where v_i is randomly selected and $v_i \neq *$ for $i \in [1, m]$ until $G(\widehat{Q}, a) \geq G_{min}$.

These strategies can be processed at the *query generalizer* with the auxiliary information. The efficiency of the strategies can be represented with the time complexity. Basically, the computation of each strategy consists of two steps: (1) selection of candidate attributes and (2) computation of guessing entropy with the selected attributes.

In the step (2), the attribute values are sorted in decreasing order of their frequencies in order to compute guessing entropy. That is, the step (2) requires $O(n \log n)$ time where n is the number of distinct domain values in the target attribute a. The time complexity of this step is common to all strategies.

Unlike the step (2), the efficiency of the step (1) is greatly varied between the strategies: In the Brute-Force strategy, all 2^m possible combinations of m attributes always have to be considered as candidate attribute sets. Thus the total running time of Brute-Force is $O(2^m \cdot n \log n)$.

The Best-Fit and Worst-Fit strategies are greedy algorithms which select an attribute to be removed as the "best" choice at each iteration until the modified query $\widehat{Q}$ satisfies the privacy requirement G_{min}. The "best" choice can be selected in $O(m)$ at each iteration. The number of iterations in these strategies is the same as the number of attributes removed from the original query; that is, the time complexity of these strategies is *output-sensitive*. Let r be the number of removed attribute values after generating $\widehat{Q}$. Thus the total running time of the Best-Fit and Worst-Fit is $O(rm \cdot n \log n)$. In the worst case, r is at most $m - 1$. In practical situations, however, m is too large compared to the required privacy requirement G_{min}, so r is much smaller than $O(m)$. The Best-Fit and Worst-Fit strategies are faster in such situations.

The time complexity of the Random-Fit can be analyzed in the same manner except that the "best" choice is selected in $O(1)$ at each iteration. In other words, the time complexity of the Random-Fit strategy is $O(r \cdot n \log n)$. In Sect. 5, we will discuss what strategies are recommendable through the experiments on real datasets.

5. Simulation

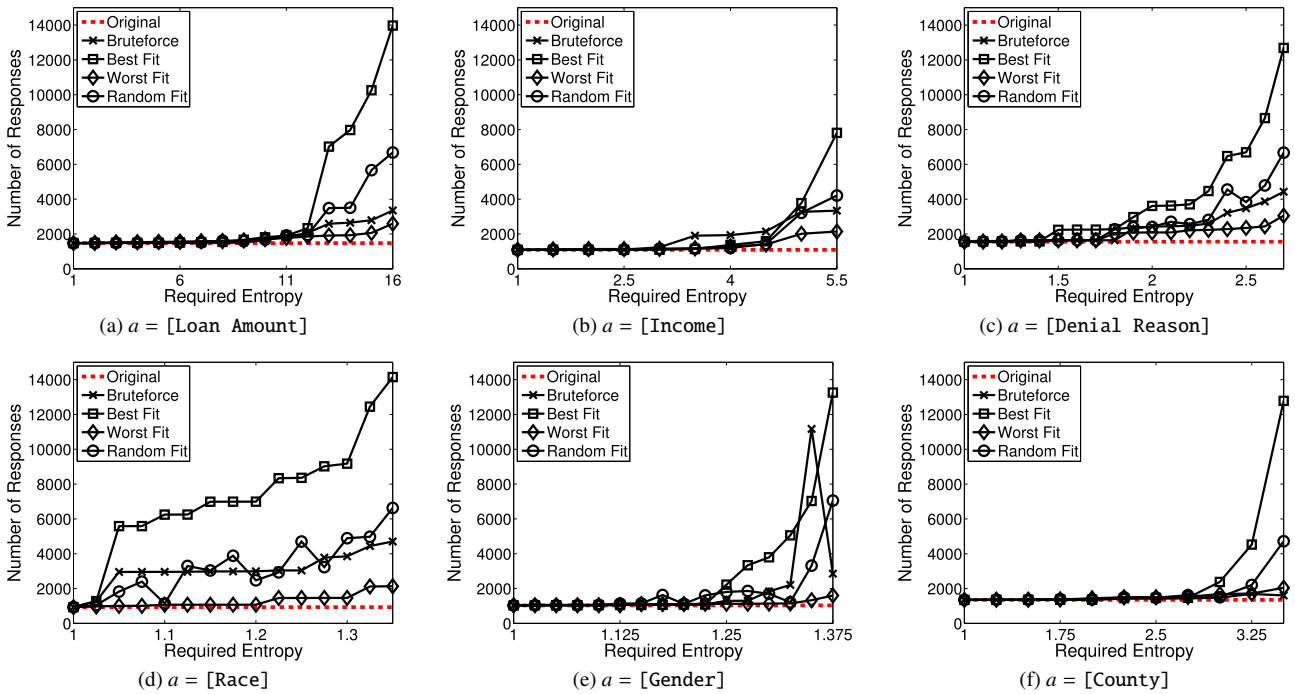
We evaluate our framework by measuring the *traffic overhead* between a *query generalizer* and a service provider. The *traffic overhead* is measured in terms of the number of records included in a query response from the service provider to the *query generalizer*, i.e. the number of tuples in service 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 (http://www.census.gov/) for both of an auxiliary information and a service provider's database. 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]` (Numeric values categorized by 236 groups)', '`[Income]` (Numeric values categorized by 206 groups)', '`[Denial Reason]` (9 categorizations including "Debt to income ratio", "Employment history", "Credit history", etc)', '`[County]` (14 counties in Massachusetts)', '`[Race]` (7 categorizations including "American Indian" or "Alaskan Native", "Asian" or "Pacific Islander", "Black", etc)', and '`[Gender]` (4 categorizations including "Male", "Female" and two others for information not provided by applicant)'.

From this database, a user's query is simulated with randomly selected attribute values in the given domain including the "don't care" value. In order to prevent generation of queries with too many "don't care" values, a random query is carefully generated by selecting either a domain value with the probability 2/3 or the "don't care" value with the probability 1/3 for an attribute in the query. For each target attribute test, its corresponding value is omitted. With 100 random queries, we evaluated the performance of the proposed framework. We assume that a service provider's database is identical to the public auxiliary information to maximize the effect of the auxiliary information although they are generally different in real-world systems.

For each query, we tested the four strategies discussed in Sect. 4. Figure 2 demonstrates the average *traffic overheads* of these strategies for selected six target attributes, respectively. For comparison, we also plotted the *original query* (the dashed line in Fig. 2) as an absolute lower bound. Note that the units of the G_{min} values have been adjusted to compare the outcomes effectively.

For each target attribute, the results have similar trends and some reasonable G_{min} values can be determined. For example, for $a =$ `[Income]` (see Fig. 2 (b)), while the slope increases rapidly when $G_{min} > 4.5$, our framework incurs a slight increase in *traffic overhead* when $G_{min} \leq 4.5$. It means that our framework can achieve a reasonable *user privacy* without a huge loss of traffic bandwidth in some privacy levels ($G_{min} \leq 4.5$).



(a) a = [Loan Amount] (b) a = [Income] (c) a = [Denial Reason]

(d) a = [Race] (e) a = [Gender] (f) a = [County]

Fig. 2 Traffic overhead.

Table 2 Traffic overhead ratio of Worst-Fit and Original.

Loan Amount		Income		Denial Reason		Race		Gender		County	
Entropy	Ratio	Entropy	Ratio	Entropy	Ratio	Entropy	Ratio	Entropy	Ratio	Entropy	Ratio
1	**1.0000**	1.0	**1.0000**	1.0	**1.0000**	1.00	**1.0000**	1.00	**1.0000**	1.00	**1.0000**
3	**1.0286**	1.5	**1.0199**	1.2	**1.0056**	1.05	**1.0811**	1.05	**1.0082**	1.25	**1.0040**
5	**1.0460**	2.0	**1.0397**	1.4	**1.0480**	1.10	**1.1477**	1.10	**1.0100**	1.50	**1.0081**
7	**1.0739**	2.5	**1.0404**	1.6	**1.0501**	1.15	**1.1501**	1.15	**1.0726**	1.75	**1.0164**
9	**1.1020**	3.0	**1.0540**	1.8	**1.2978**	1.20	**1.1502**	1.20	**1.0728**	2.00	**1.0644**
11	**1.2032**	3.5	**1.0776**	2.0	**1.3400**	1.25	**1.5614**	1.25	**1.0886**	2.25	**1.1092**
13	**1.2949**	4.0	**1.1839**	2.2	**1.4195**	1.30	**1.5668**	1.30	**1.1035**	2.50	**1.1112**
15	**1.3957**	4.5	**1.2763**	2.4	**1.4619**	1.35	**2.2858**	1.35	**1.2875**	2.75	**1.1252**
17	**2.6448**	5.0	**1.8444**	2.6	**1.5629**	1.40	**2.3662**	1.40	**1.5578**	3.00	**1.2455**
19	**3.3003**	5.5	**1.9672**	2.8	**2.5791**	1.45	**2.6099**	1.45	**1.7649**	3.25	**1.2658**

Overall, the Worst-Fit heuristic showed the best performance. Worst-Fit provides privacy-preserving queries without a significant increase in the *traffic overhead* compared to the original query (see Table 2). For example, the *traffic overhead* of Worst-Fit is greater than roughly 1.4 times than that of the original query even if $G_{min} = 15$ for [Loan Amount]. Considering the recent growth in networking technology, it is not a big penalty. In the dataset used, the size of a record is 9 bytes composed of 6 attributes (Loan Amount:Integer, Income:Integer, Denial Reason:Byte, County:Integer, Race:Byte, Gender:Byte). We assume that the size of integer type is 2 bytes. Under this setting, the average response size using Worst-Fit is about 17.6k only ($\approx \frac{2,000\times 9}{1,024}$) even if $G_{min} = 15$.

Unlike our expectations, Brute-Force, which requires exponential time in the number of attributes of the query, did not outperform the other heuristics. If the given values of the targeted attribute are uniformly distributed, the *traffic overhead* should be minimized when the query is provided with the minimally allowed entropy as we anticipated. However, our dataset rarely shows the uniform distribution. As an example, we plot the histogram of [Loan Amount] values (see Fig. 3 (a)), and the histogram shows that the distribution is skewed to the left. In order to quantitatively investigate the correlation between *guessing entropy* and *traffic overhead*, we also calculated the Pearson correlation coefficient. We plotted the *guessing entropy* for [Loan Amount] and the corresponding *traffic overhead* of the queries (see Fig. 3 (b)), and then observed a low correlation (0.369) unlike the ideal uniform distribution. Hence, we can imagine that the Brute-Force heuristic can sometimes generate undesirable results (e.g. particularly when $G_{min} \geq 20$).

In practice, Brute-Force tends to select a modified query with a large number of corresponding tuples in auxiliary information to reduce the probabilities p_i for $i = 1, \cdots, n$ of Eq. (2) in Sect. 2. Unfortunately, this tendency may incur exceptionally high network costs even though its calculated entropy is low. We found that a few exceptional

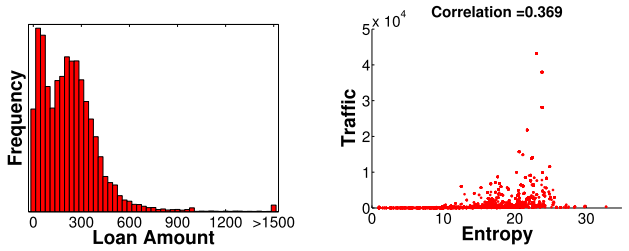


Fig. 3 Two observations from `[Loan Amount]`. (a) Histogram of `[Loan Amount]` values. We can see that the observed values are not uniformly distributed in practice. (b) The scatter plot depicting the correlation between *guessing entropy* for `[Loan Amount]` (x-axis) and *traffic overhead* (y-axis). For the 5,000 randomly generated queries, the corresponding *guessing entropy* and *traffic overhead* values are plotted. The Pearson correlation coefficient (0.369) indicates that there exists a weak positive correlation between *guessing entropy* and *traffic overhead*.

cases are encountered in our simulation. For example, when $G_{min} = 1.35$ for $a =$ `[Gender]`, the generalized query with the smallest entropy always consists of `[Race]` = "White" alone if the user's query contains the attribute value, `[Race]` = "White". However, this specific attribute value dominates 65.1% of the overall records in the database. Thus these queries incur huge costs. This is the reason why a fluctuation is observed in Brute-Force at $G_{min} = 1.35$ in Fig. 2 (e).

These results indicate that strategies to minimize *guessing entropy* alone are not enough to effectively reduce the *traffic overhead* for the queries when the distribution is skewed. We can see that Random-Fit sometimes provides comparable performance to Brute-Force. This is because Random-Fit is not forced to follow the Brute-Force's behavior to minimize the entropy of selected queries and then can avoid the exceptional cases which require significant additional network traffic.

In general, Worst-Fit tends to reduce not only the *guessing entropy* but also the number of the removed attributes from the original query. This property seems helpful in reducing the *traffic overhead*. Random-Fit also shows a similar trend. On the contrary, Best-Fit tends to increase the number of removed attributes and then shows disappointed results. Considering that the time complexity of Worst-Fit is $O(rm \cdot n \log n)$ (m: the number of attributes in the query, n: the number of distinct domain values in the target attribute, r: the number of removed attributes from the original query) and the outstanding performance in the entire privacy level G_{min}, we strongly recommend using Worst-Fit.

6. A Privacy Control Way for End-Users

As we mentioned in Sect. 4, it seems difficult for users to set the required entropy value themselves in practice since the purpose of entropy estimation is not generally intuitive. Therefore we suggest a reasonable method to help users control their privacy settings. Instead of setting G_{min} directly, a user can define l, derived from the concept of l-diversity [19] where (a, l) indicates that an adversary, trying to determine the user u's specific value for a, can only narrow down the possible values to a set containing at least l distinguishable sensitive attribute values. *Query generalizer* internally translates (a, l) into $(a, (l+1)/2)$ under the guessing entropy model. For example, if a user wants high diversity of $a =$ `[Loan Amount]` in Fig. 2 (a) with $l = 25$ (i.e. in order to find loan amount of the user, an adversary must choose one out of more than 25 values), the *query generalizer* sets $G_{min} = (l+1)/2 = 13$. Theorem 6.1 states that the size of the attribute value set for $\widehat{Q}$ is always at least l when $(l+1)/2$ is used as the required minimum guessing entropy to generate $\widehat{Q}$.

Theorem 6.1: When the query Q is modified into the query $\widehat{Q}$ with the guessing entropy value, which is greater than or equal to $(l+1)/2$, the size of the attribute value set for $\widehat{Q}$ is at least l.

Proof Let $l_{\widehat{Q}}$ and $G_{\widehat{Q}}$ be the size and the guessing entropy of the possible attribute values for $\widehat{Q}$, respectively. In other words, there are $l_{\widehat{Q}}$ possible values $\{x_1, \cdots, x_{l_{\widehat{Q}}}\}$ where $p_i = P(X = x_i)$ is the probability for each possible value x_i for $i \in [1, l_{\widehat{Q}}]$. Assume that $l_{\widehat{Q}}$ is strictly less than l. Since the uniform distribution has the maximum entropy, the following inequality is derived.

$$G_{\widehat{Q}} = \sum_{i=1}^{l_{\widehat{Q}}} p_i \cdot i \leq \frac{l_{\widehat{Q}} + 1}{2} < \frac{l+1}{2}$$

This inequality contradicts since the guessing entropy value of the query $\widehat{Q}$ is greater than or equal to $(l+1)/2$. ∎

7. Related Work

For privacy protection on public datasets, Samarati and Sweeney introduced the concept of "k-anonymity" [12], [20]. In this model, privacy is guaranteed by ensuring that any record in a released dataset be indistinguishable with respect to a set of attributes, called *quasi-identifier* [6] from at least $k-1$ other records in the dataset; thereby the risk of re-identification is kept under $1/k$. Many techniques [21]–[24] have been proposed to achieve k-anonymity requirement. Optimal k-anonymity problems were proved to be *NP-hard* for $k \geq 3$ [25]. Machanavajjhala et al. introduced the model of "l-diversity" [19] where every group of indistinguishable records contains at least l distinct sensitive attribute values; thereby the risk of attribute disclosure is kept under $1/l$. There are also many approaches [9]–[11] to publish anonymized datasets through randomized or cryptographic techniques for privacy preserving data mining.

Query restriction is a classical approach which restricts queries that may result in inference [26]. In this approach, queries are restricted by various criteria such as the size of query result [26], the overlap amongst successive queries [26], and partitions [27]. However, these techniques are significantly different from ours. While these techniques have tried to control the query results against a "malicious

querier", our technique is designed to minimize the chances to expose *quasi-identifiers* from the query inputs against a "malicious party with public datasets".

In information retrieval, the notion of private information retrieval (PIR) was introduced by Chor et al. [28]. PIR provides a strong privacy theoretically but it also incurs significant communication and computational overheads based on cryptographic operations. We note that their threat models are different from ours. These studies focused on hiding which records are retrieved by a user, while our goal is to protect users' private attribute values from their queries.

There were some investigations on location privacy [29]–[32]. For location query, the users' query input (i.e. the user's location) is blurred into a cloaked region depending on the privacy level. However, their anonymization techniques can be applied for location data only.

Yabo et al. [33] proposed a method that users can control the degree of the personalized service by avoiding the construction of a highly private user profile. Imada et al. [34] also proposed a method to control users' personal data by measuring the entropy from the gathered user data. However, these approaches are different from ours. We focus on how to construct a user's query inputs depending on the trade-off between user privacy and service utility; not the roles of functional components.

8. Conclusion

We proposed a framework based on the concept of *query generalizer* which effectively controls highly private information from users' queries. We discussed the pros and cons of two possible deployments of the framework and recommend three-tier architecture due to its practicality.

For efficient implementation of *query generalizer* we presented four schemes, the Brute-Force, Best-Fit, Worst-Fit and Random-Fit. Through the simulation on real datasets we compared and evaluated them. The Worst-Fit heuristic generally provides the best results. With this heuristic, *query generalizer* can satisfy user's privacy requirement to protect sensitive user information from being easily guessed without a significant increase in the *traffic overhead*. The increased overhead varied from 1.0 to 3.3 times compared to the original. We do emphasize however that these are initial implementations of a new framework idea; and we expect that our framework will adapt easily to other privacy models or query generalization schemes.

The computational overhead of entropy may greatly increase with the amount of auxiliary information. 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 for future work.

We could also extend this work to a real application such as the personalized healthcare information infrastructure [35] 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 *privacy* requirements in order for users to take control over their own health information privately. More specifically, this work could readily be adopted to protect sensitive information about patients by controlling the level of detail of their medical data in the transactions.

Acknowledgements

The authors would like to thank Wei Ming Khoo, Jun Ho Huh, Kazuma Shinoda, Yoshiro Muraji, Hiroyuki Suzuki and Joong Sun Lee for their careful attention and comments.

References

[1] P. Maglio and R. Barrett, "Intermediaries personalize information streams," Commun. ACM, vol.43, no.8, pp.96–101, 2000.
[2] M.A. Winker, A. Flanagin, B. Chi-Lum, J. White, K. Andrews, R.L. Kennett, C.D. DeAngelis, and R.A. Musacchio, "Guidelines for medical and health information sites on the Internet: principles governing AMA Web sites," J. American Medical Association, vol.283, no.12, pp.1600–1606, 2000.
[3] E. Steel and J.E. Vascellaro, "Facebook, MySpace confront privacy loophole," May 2010. http://online.wsj.com/article/SB10001424052748704513104575256701215465596.html, accessed Feb. 1, 2011.
[4] R.A. Dolin, "Search query privacy: the problem of anonymization," Hastings Science and Technology Law Journal, vol.2, no.2, p.137, April 2010.
[5] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," Proc. 13th USENIX Security Symposium, pp.303–320, 2004.
[6] 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.
[7] L. Sweeney, "Uniqueness of simple demographics in the U.S. population," LIDAP-WP4 Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, 2000.
[8] P. Golle, "Revisiting the uniqueness of simple demographics in the US population," WPES '06: Proc. 5th ACM Workshop on Privacy in Electronic Society, pp.77–80, ACM, 2006.
[9] R. Agrawal and R. Srikant, "Privacy-preserving data mining," Proceedings of the 2000 ACM SIGMOD international conference on Management of data, SIGMOD '00, New York, NY, USA, pp.439–450, ACM, 2000.
[10] B. Pinkas, "Cryptographic techniques for privacy-preserving data mining," ACM SIGKDD Explorations Newsletter, vol.4, pp.12–19, Dec. 2002.
[11] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan, "Incognito: efficient full-domain K-anonymity," Proc. 2005 ACM SIGMOD international conference on Management of data, SIGMOD '05, pp.49–60, New York, NY, USA, 2005.
[12] 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.
[13] S. Schechter, A.J.B. Brush, and S. Egelman, "It's no secret. measuring the security and reliability of authentication via "secret" questions," Proc. 2009 30th IEEE Symposium on Security and Privacy, pp.375–390, IEEE Computer Society, 2009.
[14] J.L. Massey, "Guessing and entropy," Proc. IEEE International Symposium on Information Theory, p.204, 1994.
[15] T. Dierks and C. Allen, "The TLS protocol version 1.0." RFC 2246 (Informational), 1999.
[16] A. Etzioni, The limits of privacy, Basic Books, 1999.

[17] M. Langheinrich, "Privacy by design - principles of privacy-aware ubiquitous systems," Proc. 3rd international conference on Ubiquitous Computing, UbiComp '01, pp.273–291, London, UK, 2001.
[18] R. Hemmecke, M. Kppe, J. Lee, and R. Weismantel, "Nonlinear integer programming," in 50 Years of Integer Programming 1958-2008, pp.561–618, Springer Berlin Heidelberg, 2010.
[19] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: privacy beyond *k*-anonymity," ACM Transactions on Knowledge Discovery from Data, vol.1, no.1, p.3, 2007.
[20] P. Samarati, "Protecting respondents' identities in microdata release," IEEE Trans. Knowl. Data Eng., vol.13, no.6, pp.1010–1027, 2001.
[21] V.S. Iyengar, "Transforming data to satisfy privacy constraints," KDD '02: Proc. Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.279–288, New York, NY, USA, 2002.
[22] R.J. Bayardo and R. Agrawal, "Data privacy through optimal *k*-anonymization," ICDE '05: Proc. 21st International Conference on Data Engineering, pp.217–228, Washington, DC, USA, 2005.
[23] B.C.M. Fung, K. Wang, and P.S. Yu, "Top-down specialization for information and privacy preservation," ICDE '05: Proc. 21st International Conference on Data Engineering, pp.205–216, 2005.
[24] K. LeFevre, D.J. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional *k*-anonymity," Data Engineering, 2006. ICDE '06. Proc. 22nd International Conference on Data Engineering, ed. L. Liu, A. Reuter, K.Y. Whang, and J. Zhang, p.25, Washington, DC, USA, April 2006.
[25] A. Meyerson and R. Williams, "On the complexity of optimal *K*-anonymity," PODS '04: Proc. Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp.223–228, New York, NY, USA, ACM, 2004.
[26] D. Dobkin, A.K. Jones, and R.J. Lipton, "Secure databases: protection against user influence," ACM Trans. Database Syst., vol.4, no.1, pp.97–106, 1979.
[27] C.T. Yu and F.Y. Chin, "A study on the protection of statistical data bases," SIGMOD '77: Proc. ACM SIGMOD International Conference on Management of Data, pp.169–181, New York, NY, USA, ACM, 1977.
[28] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," J. ACM, vol.45, no.6, pp.965–981, 1998.
[29] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," MobiSys '03: Proc. 1st International Conference on Mobile Systems, Applications and Services, pp.31–42, New York, NY, USA, 2003.
[30] M. Mokbel, C.Y. Chow, and W. Aref, "The new casper: query processing for location services without compromising privacy," VLDB '06: Proc. 32nd International Conference on Very Large Data Bases, pp.763–774, New York, NY, USA, 2006.
[31] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing location-based identity inference in anonymous spatial queries," IEEE Trans. Knowl. Data Eng., vol.19, no.12, pp.1719–1733, 2007.
[32] H. Kim, "A spatial cloaking framework based on range search for nearest neighbor search," DPM '09: 4th Workshop on Data Privacy Management, pp.93–105, Berlin Heidelberg, 2010.
[33] Y. Xu, K. Wang, B. Zhang, and Z. Chen, "Privacy-enhancing personalized web search," WWW '07: Proc. 16th international conference on World Wide Web, pp.591–600, New York, NY, USA, 2007.
[34] M. Imada, M. Ohta, M. Teramoto, and M. Yamaguchi, "A flexible personal data disclosure method based on anonymity quantification," IEICE Trans. Commun., vol.90-B, no.12, pp.3460–3469, Dec. 2007.
[35] 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.

Yunsang Oh received the B.S. and M.S. degrees in Computer Science and Engineering from Sogang University, Korea in 1997 and 2002, respectively. During 2003–2009, he worked in Samsung Electronics. He also served a member of OMA for DRM standardization in mobile networks. He is currently studying in Tokyo Institute of Technology, Japan as a Ph.D. student. His research interests include security and privacy in public Web services.

Hyoungshick Kim received the B.S. degree in information engineering at the SungKyunKwan University, Korea. He obtained the M.S. degree in department of computer science, KAIST, Korea in 2001. During 2004–2008, he worked in Samsung Electronics. He also served a member of DLNA and Coral standardization for DRM interoperability. He is currently studying in the University of Cambridge as a Ph.D. student. His research interests include privacy and anonymity in distributed systems.

Takashi Obi received his M.E. and Ph.D. degree in information processing from Tokyo Inst. of Tech, Tokyo, Japan, in 1992 and 1997, respectively. He was a Research Associate (1995–2002) at Imaging Science and Engineering Lab., Tokyo Inst. of Tech. and a Visiting Researcher (1998, 2000) at Univ. of Pennsylvania. He has been an Associate Professor at Dept. of Information Processing, Tokyo Inst. of Tech. since 2002. His current interest includes e-government, system security management and medical image informatics. He is a member of the IEEE, JAMIT and JSAP.