Proxy-Terms Based Query Obfuscation Technique for Private Web Search

SHARIQ BASHIR1, DAPHNE TECK CHING LAI2, AND OWAIS AHMED MALIK2
1Institute of Applied Data Analytics (IADA), Universiti Brunei Darussalam (UBD), Bandar Seri Begawan BE1410, Brunei
2School of Digital Science (SDS), Universiti Brunei Darussalam (UBD), Bandar Seri Begawan BE1410, Brunei
Corresponding author: Shariq Bashir (shariq.bashir@ubd.edu.bn)

ABSTRACT
Search engines store users’ queries in a query log for performing personalized information retrieval. However, query logs cause privacy concerns and reveal a lot of information about individuals if used against them. Private web search (PWS) provides a privacy-preserving information retrieval (IR) facility which allows users to retrieve information from an IR system without revealing true search queries. Current PWS techniques that are explored in the domain of web search are query obfuscation-based private web search (OB-PWS). These techniques achieve web privacy by injecting cover queries into the user profiles. However, existing OB-PWS techniques submit true queries along with cover queries and achieve query obfuscation in an isolated manner without considering the similarity between consecutive queries. In this article, we propose a proxy-terms based query obfuscation technique that allows users to retrieve information from an IR system through proxy queries without submitting true queries. IR system automatically generates cover queries and true queries from the proxy queries and cannot differentiate whether the user is trying to retrieve information for the cover queries or true query. We analyzed the effectiveness of the proposed technique on test queries, and develop a similarity metric for testing the accuracy of the proposed technique. Promising results of experiments confirm the effectiveness of the proposed technique.

INDEX TERMS
Private web search, private information retrieval, query obfuscation, web search privacy, information retrieval.

I. INTRODUCTION
We use search engines for searching different kinds of information. Search queries contain a valuable source of information and reveal a lot of information about individuals related to their health issues [1], institutions they work for, political interests [2], [3], religious beliefs, interest in new products, employment search, news, etc., [4]–[7]. Modern search engines store user queries in the query log. The query log is frequently used in the advanced information retrieval (IR) algorithms for refining the retrieved information with the help of personalized information retrieval (IR) [8]–[10]. Although search engines use query log for personalized IR, query log may cause privacy concerns, revealing a lot of private information about individuals if used against them [6], [11], [12].

65% of users believed that it was unacceptable to record their searches. 73% web users say they feel uncomfortable with the search engines which log their search queries. The other concerns are query log can be hacked, or a search engine can use it for commercial purpose [13]. For example, the search engine can sell the query log to other organizations without the permission of users. Hackers can attack query log and can sell it to other organizations. The attackers can utilize the personal information of users stored in the query log for revealing information such as their interests in future products, employment, health issues, political or religious beliefs, etc., [7].

Imagine a scenario where a scientist, researcher, or inventor of a company is working on a new technology. Suppose the company discovers a new invention, which can improve the functionality of existing products. The company would be interested to file a patent application, however, before filing the patent application the company needs to search the prior-art using a search engine to find whether there is already any patent filed similar to the invention. In this situation,
suppose there is a competitor company that is interested in the future inventions of a company. If the competitor has access to the query search log of the company, the competitor can infer valuable information from the query including the company has invented which new invention. The query log can give the competitor an important advantage over the other company.

Another similar situation that can cause search privacy issues is companies utilizing search query logs during the hiring of candidates. Suppose there is a hiring company and a candidate has applied for a vacant position. In this situation, if the hiring company utilizes the search query log and found that in the past the candidate had searched information from a search engine related to treatment or symptoms of a disease, such information can put the candidate in a disadvantaged position. The company can infer that the person has either symptoms or looking for the treatment of this disease, and hiring this person can add extra health insurance burden on the company. Based on this information, the company can reject the candidate. In both situations above, the attacker (which can be an individual, company, organization, institution, or hacker) benefits from the lack of a privacy preservation mechanism between the search engine and the query log.

Private web search (PWS) provides a privacy preservation facility for information retrieval [13]–[15]. It allows users to search for information from a search engine without submitting true queries. A trivial approach for PWS can be to search the information from search engines with queries that contain only stop words. Unfortunately, this solution is not suitable for web search as the stop words virtually retrieve all documents from the server, and the client machines need to locally retrieve and rank the desired information. Another solution is to hide the user identities by anonymous web browsing approach [16]–[18]. This can be achieved by querying the search engine through proxy servers or dynamic IP addresses. However, anonymizing the client identities does not solve the privacy problem as the proxy servers can start logging the clients’ web searches. This approach moves the privacy threat from search engines to proxy servers.

PWS techniques that are recently explored in the domain of web search are query obfuscation-based private web search (OB-PWS) [19]–[25]. The OB-PWS based techniques achieve web privacy by inserting noise in the user profiles maintained by a search engine. Figure 1 shows the architecture of query obfuscation-based private web search. For each search query, the OB-PWS based technique generates cover queries related to different topics. These cover queries along with the true query are sent to the IR system for hiding the true intent of users [20], [22], [26]. As a result, the IR system cannot accurately classify whether the user is trying to retrieve information from the true query or cover queries. The IR system retrieves and sends the rank lists of true and cover queries to the client machine. The client machine receives the results of all queries and only keeps the rank lists of true query. Furthermore, if confronted with a sensitive or uncomfortable query, users may claim that it was generated by the OB-PWS tool and obtain plausible deniability about having issued the query [19].

The main limitation of existing OB-PWS based techniques is that these techniques perform query obfuscation for the consecutive user queries in an isolated manner without computing similarity between consecutive queries [20], [22], [27]. We list here the main limitations of existing OB-PWS based techniques which do not make them attractive solutions for PWS.

- Existing techniques submit true queries along with cover queries to the IR system. The true queries can still reveal important information about users’ search intents.
- Existing techniques generate cover queries in an isolated manner without considering the similarity between consecutive queries. This does not obfuscate queries if a user issues consecutive related queries from a single search session.
- Recently an OB-PWS based technique has been proposed to generate cover queries by considering the similarity between consecutive queries [22]. However, this technique requires indexing a big collection at client machines related to different topics which is not suitable for smart devices with limited memory. Furthermore, this technique does not achieve similarity between consecutive queries when the users issue consecutive queries from multiple sessions or from different devices.

A user can continue the current search task with multiple sessions for several days or from different devices.

In this article, we propose a new technique for OB-PWS which we call proxy-terms based private web search (ProxyTermsPWS). One of the obvious advantages of the proposed technique is that the client machines do not generate and issue cover queries to the IR system. Query obfuscation is achieved by proxy-terms based indexing technique. This provides a facility that the users can search the required information from an IR system through proxy queries.

The proxy-terms based OB-PWS technique works as follows. Given a collection of topics containing sensitive and non-sensitive information, the ProxyTermsPWS maps the terms of topics containing sensitive information (which we call proxy topics) with the terms of topics containing non-sensitive information (which we call proxy topic), and indexes the mapping in a proxy dictionary. The major challenge is how to generate optimal terms mapping between proxy and cover topics’ terms so that the ProxyTermsPWS generates relevant cover and proxy queries when the users issue consecutive related queries. Once the proposed technique achieves optimal proxy-terms mapping, the ProxyTermsPWS makes publicly available the proxy dictionary to all users. Given the proxy dictionary, when a user issues a search query to the IR system. The client machine transforms the terms of the true query with the proxy terms using the mapping available in the proxy dictionary. The client machine issues the proxy query to the IR system. The IR system assumes that the user is interested to retrieve the information for the proxy query. However, because of the ProxyTermsPWS...
FIGURE 1. The architecture of query obfuscation based private web search.

TABLE 1. Proxy Dictionary showing proxy-term mapping for the proxy topic Roasted Chicken and cover topics Changing Engine Oil and Depression Treatment.

| Proxy Terms  | Cover Terms         |
|--------------|---------------------|
| Chicken      | Engine, Depression  |
| Roast        | Oil, Treatment      |
| Cook         | Lubricant, Symptom  |
| Recipe       | Motor, Anxiety      |
| Vegetable    | Penzoin, Diagnose   |
| Delicious    | Motorcycle, Health  |
| Breast       | Synthetic, Mental   |
| Easy         | Change, Condition   |
| Salt         | Leak, Sleep         |
| Butter       | Filter, Bipolar     |
| Dinner       | Grade, Mood         |
| Bake         | Hot, Experience     |
| Juicy        | Fuel, Medication    |
| Oven         | Viscosity, Disorder |
| Lemon        | Performance, Therapy|

facility, the IR system allows users to retrieve information from cover topics through proxy queries. Therefore, the IR system cannot differentiate whether the user is retrieving information for the cover queries or the proxy query. The IR system generates cover queries and true query from the mapping available in the proxy dictionary. The IR system individually processes all queries and retrieves information of all queries. The IR system independently prepares rank list of each query and sends the rank lists to client machine. Once the search results are returned to the client machine, only the rank list related to the true query is provided to the user.

To further understand this technique, suppose the ProxyTermsPWS has achieved optimal proxy-terms mapping for all the terms of cover topics depression treatment and changing engine oil with the terms of proxy topic roasting chicken. Table 1 shows an example of proxy-terms mapping and proxy dictionary. Imagine there are two users and an IR system provides a ProxyTermsPWS facility. First user wants to retrieve information related to depression treatment, and the second user wants to retrieve information related to chicken roast. After both users issue search queries to the IR system, the client machine of first user transforms the terms depression and treatment with the proxy terms chicken and roast Figure 2. The client machine of second user also transforms the query. However, since the proxy terms and the true query terms are similar to each other, therefore, the ProxyTermsPWS generates proxy query chicken roast for the second user. The proxy queries chicken roast of both users are issued to the IR system. The IR system searches the terms chicken and roast from the proxy dictionary and founds that the term chicken is the proxy term of terms depression and engine, and the term roast is the proxy term of terms treatment and oil. Using the proxy dictionary the IR system generates two cover queries: depression treatment and engine oil along the proxy query: chicken roast. The IR system retrieves the documents for all queries and independently prepares a rank list of each query and sends them back to the client machines. Once the search results are returned to the client machines, the client machines only provide the rank lists of true queries to users and ignore the results of cover queries. As both client machines issue proxy queries to the IR system, therefore, IR system cannot classify whether the users are retrieving information related to depression treatment, chicken roast or engine oil. Figure 2 shows the architecture of private web search using proxy-terms based query obfuscation technique.

A. MAIN CONTRIBUTIONS
The main characteristics of ProxyTermsPWS are:
- **Storage Efficiency:** The client machines do not generate cover queries. It is not required to store and index big
collections of such cover queries at client machines. This technique is highly suitable for smart devices with limited memory. Since the ProxyTermsPWS transforms the cover terms with the proxy terms, therefore, IR system automatically generates and processes all cover queries.

- **Information Retrieval without True Query:** Using the ProxyTermsPWS facility, the users retrieve the information through proxy queries without submitting true queries. Thus, the true search intents of users are not revealed to the IR system.

- **Web Search Privacy for Related Queries:** The term mapping between the proxy and cover terms is achieved by the IR system by considering proximity relatedness between neighbors terms [28]. The proximity neighbor terms of cover topics are transformed with the proximity neighbor terms of proxy topics. Thus consecutive cover queries generated by the IR system always have high similarities between them. The similarity is automatically achieved when a user issues consecutive queries from a single search session, multiple search sessions, or from different devices. For instance, given the example presented above for the query *depression treatment*. Suppose the user issues another consecutive query *depression treatment medication* from the current search session. The client machine transforms the true query with the proxy query *chicken roast juicy* which is related to the initial query *chicken roast*. When the IR system receives the proxy query, it generates two cover queries: *depression treatment medication* and *engine oil fuel* which are related to the initial cover queries.

- **Query Obfuscation with High Computation:** Since the client machines do not generate cover queries, there is no need to execute high processing algorithms on client machines for calculating similarity between consecutive queries.

- **Retrieval Effectiveness:** The proposed solution does not impose extra indexing cost on the server. Whenever a user retrieves information from an IR system, the ProxyTermsPWS provides similar retrieval effectiveness that the user receives without using ProxyTermsPWS facility. ProxyTermsPWS achieves user privacy and does not affect the effectiveness of queries.

## II. RELATED WORK

In recent years, many techniques and protocols have been proposed for protecting the web search privacy of users. In the following subsections, we present a detailed review of modern PWS techniques.

### A. PRIVATE INFORMATION RETRIEVAL TECHNIQUES

Chor *et al.* [29] proposed a private information retrieval technique by distributing the database into multiple servers. The proposed approach holds a realistic assumption that it works only when the multiple servers do not communicate with each other. This assumption is realistic in the context of web search as retrieval models rank the relevant documents from all retrieved documents. Kushilevitz and Ostrovsky [14] proposed a private information retrieval protocol by placing the database on a single server. However, Balsa *et al.* [19] found that this protocol cannot be used in the context of web search.

### B. PROXY BASED PWS TECHNIQUES

The PWS techniques of this category hide user identities by anonymous web browsing approach [16]–[18]. This is achieved by retrieving information from a web search engine (WSE) through proxy servers or dynamic
IP addresses. Tor [16] and RAC [18] are examples of this category. Although proxy-based PWS techniques hide users from WSEs, anonymizing the client identities does not solve the web search privacy (WSP) issue as the proxy servers can start logging clients’ web searches. These techniques move the privacy threat from WSE to proxy servers.

C. COLLABORATIVE PEER-TO-PEER PWS
The PWS techniques of this category involve multiple users collaborating together using a peer-to-peer protocol for issuing queries and retrieving information from a WSE [30], [31]. With this protocol, if a user wants to retrieve information from a WSE, then the user does not directly submit his/her query to the WSE. He/she first submits his/her query to some other user. The other user submits his/her query to WSE, and he/she submits a query of some other user. Using this way, the WSE cannot identify the true search intents of users as the users submit a query of some other peer each time. The deployment of collaborative-based PWS protocols is not realistic in practice as these protocols require the availability of the same set of users during issuing consecutive queries related to a similar topic [22]. The collaborative PWS protocols also suffer slow response times as multiple users exchange queries and retrieve documents from different locations. The protocols also require complex implementation of cryptography protocols on client devices and server.

D. QUERY SCRAMBLING BASED PWS TECHNIQUE
PWS techniques of this category generate a set of scrambled queries and retrieve information through scrambled queries [32], [33]. The scrambled queries retrieve information that corresponds loosely to the interest in response to a private query. The scrambled queries are generated through scrambler. Given a private query, the scrambler generates queries that are semantically more general rather than specific. The objective of scrambled queries is to hide the true search intents of web users. The scrambled queries are then issued to WSE for retrieving a set of documents called scrambled information. Given a set of scrambled information, the technique employs descrambling at the user machine which reconstructs a ranking similar to the one that the user receives without using the PWS technique. The query scrambling-based technique retrieves partial relevant documents; thus this technique provides low retrieval effectiveness.

E. QUERY OBFUSCATION BASED PWS TECHNIQUES
Another group of techniques that are recently explored for the web search domain is query obfuscation-based private web search (OB-PWS) [19], [20], [22], [24]. The OB-PWS techniques generate non-noisy cover queries for obfuscating true queries [34], [35]. Non-noisy queries are queries that are frequently issued by real web users. The OB-PWS techniques implement non-noisy queries generation mechanism by generating cover queries from a real query log if the techniques have access to a subset of the query log. Another approach for generating non-noisy cover queries is first crawling the world web pages for obtaining a collection of documents and then generating pseudo-real queries from the collection [22]. The quality of pseudo-real queries can be measured through query quality predictors [36].

TrackMeNot is an OB-PWS based technique [34]. It submits k random queries. GooPIR [35] works similar to TrackMeNot, however, GooPIR generates cover queries by selecting keywords of cover queries from a term dictionary. The main limitation of TrackMeNot and GooPIR is that both techniques do not consider relatedness in cover queries when a user issues consecutive queries related to a similar topic.

F. QUERY OBFUSCATION BASED PWS TECHNIQUE BY CONSIDERING RELATEDNESS WITH PAST QUERIES
OB-PWS techniques of the above category can be easily compromised if a user continues the current search task with multiple queries using single or multiple search sessions. Because the techniques generate cover queries for only the current query but do not consider whether the current query is related to previous queries. In this way, if an OB-PWS technique does not consider relatedness between consecutive queries, a WSE can easily classify true queries from cover queries by extracting related consecutive queries from non-related consecutive queries.

The PWS techniques of this category achieve this objective by storing a local query log on the client machine. Ahmad et al. [22] proposed an OB-PWS based technique that achieves similarity between consecutive cover queries. It generates related cover queries from a set of hierarchically organized topics. This ensures that the cover queries add multiple fake search intents. The main limitation of this technique is that it depends on indexing a big collection of documents for building the organized hierarchy of topics. This imposes an extra cost on the client machines. Additionally, different machines of a user can generate different cover queries for the same topic if the machines contain a different set of topic hierarchy.

Our proposed technique is different from related OB-PWS techniques. To generate consecutive cover queries, the proposed technique does not index big collections on client machines. The proposed technique transforms the true queries with the proxy queries. The IR system itself generates and processes all possible cover queries. Thus, the users do not submit true queries to the IR system. Furthermore, the terms mapping between the proxy and cover terms is achieved by the IR system by considering the similarity between related queries. Thus the consecutive cover queries generated by the IR system ensure high similarity between them. The similarity is automatically achieved when a user issues consecutive queries from a single search session, multiple search sessions, or from different devices.

III. PROPOSED TECHNIQUE: PROXY-TERMS BASED PRIVATE WEB SEARCH
To understand the proposed technique, let us review how a traditional IR system indexes a collection and how it processes
search queries and retrieves relevant documents. Given a collection of documents, an IR system indexes documents for the fast processing of queries. An inverted index is frequently used for efficient indexing [37]. The inverted index contains two parts: the dictionary and the posting lists. Dictionary contains the set of all terms which appear at least one time in the collection. The posting lists contain the occurrences of terms in the collection. The dictionary stores the head pointers of posting lists. For retrieving all matching documents of a query term, the IR system retrieves the head pointer of the term, traverses, and retrieves all the documents that are available in the elements of the posting list.

A topic in the collection is a set of similar documents, and it contains related terms. The language of a topic is a set of terms that appeared in the documents of the topic. In a typical IR system, if two topics contain different languages then the identifiers of documents of the topics are indexed at separate posting lists. Thus, when a user issues a query from the language of a topic A, the IR system only searches terms of topic A from the dictionary. This retrieves all matching documents of only topic A while ignoring the documents of other topics.

The proxy-terms based private web search (ProxyTermsPWS) maps the terms of cover topics with the terms of proxy topic. The mapping does not affect the effectiveness of queries: i.e., the ProxyTermsPWS retrieves a similar set of documents of a query as these are retrieved without proxy query. After achieving proxy-terms mapping, the ProxyTermsPWS constructs a proxy dictionary. It maps the terms of the proxy topic with the terms of cover topics. After mapping, the IR system makes publicly available the proxy dictionary to all users. The client machines download the proxy dictionary from the IR system. When a user issues a query to retrieve the information from any cover topic, the client machine transforms the terms of the true query with the proxy terms available in the proxy dictionary.

To materialize the ProxyTermsPWS technique, we need to explain three parts, namely: 1) How to identify proxy and cover topics, 2) how to map the terms of cover topics with the terms of proxy topics, and 3) how to transform true queries with the proxy queries. To explain the methodology of the proposed technique, a graphical example is provided in Figure 3 and Figure 4.

**A. IDENTIFYING PROXY AND COVER TOPICS**

Given a set of topics, the ProxyTermsPWS divides the topics into sets. The first set contains the topics with sensitive information. The second set contains the topics with non-sensitive information (Figure 3). The topics of the first set contain sensitive information such as information related to health, employment, political or religious beliefs, hacking, etc. To create terms mapping between cover and proxy topics, the ProxyTermsPWS randomly selects a proxy topic from the set containing non-sensitive information and randomly selects M cover topics from both sets. The terms of selected topics are placed in groups called proxy-term groups.

| Notation | Meaning |
|----------|---------|
| R        | Topics in collection |
| T        | Unique term set of topics |
| P        | Proxy-terms groups in proxy-dictionary |
| M        | Size of proxy-term group |
| Qs       | Exhaustive list of queries to retrieve information from topics containing sensitive information |

There can be any percentage of terms containing sensitive and non-sensitive information, however, there should be at least one term from the set containing sensitive information. Ideally, the best candidates for the proxy topics are the topics that are frequently used in daily searches such as topics related to weather, sports, cooking, news, automobile, tourism, etc.

**B. GENERATING PROXY DICTIONARY**

The objective of proxy-term mapping is to conceptually transform the terms of cover topics with the terms of proxy topics so that users can retrieve the information through proxy queries. Given T unique terms of proxy and cover topics, the ProxyTermsPWS partitions the T into P proxy-term groups. Each proxy-term group contains M terms with at least one term from the topics containing sensitive information (Figure 3). We want to achieve optimal term mapping in such a way that when a user issues a query for the topic containing sensitive information, the ProxyTermsPWS generates relevant cover queries. Given this objective the fitness function to find an optimal term mapping can be defined with the following two properties. Table 2 provides a list of important notations used in the rest of the article.

The values of M, T, and P shape the size of the proxy dictionary. A proxy dictionary that maps a large number of cover topics with the proxy topics achieves higher web search privacy than the proxy dictionary that maps a small number of cover topics with the proxy topics. However, on the other side, it should be noted that higher values of M, T, and P require high computation cost for generating optimal proxy dictionary, which becomes exponential when the value of P is large. This is because, given T terms and P proxy-terms groups, there are O(T^P) possible ways to map T terms with the P proxy-terms groups. In experiments, we generated the proxy dictionary with the P value of 6 and the analysis demonstrates it still achieves reasonable high web search privacy. In future work, our objective is to generate a proxy dictionary with high values of P using a high-performance computation facility. Moreover, it should be noted here that spending computation time for generating an optimal proxy dictionary is a one-time cost. Once an optimal proxy dictionary is available it can be distributed to users for achieving web search privacy.

1) Let Qs contains an exhaustive list of queries to retrieve information from the set containing sensitive information. A mapping between proxy and cover terms is optimal if the ProxyTermsPWS generates relevant cover queries to obfuscate each query in Qs. A query
is relevant if it retrieves related documents from the collection. This property also applies to consecutive queries when the users issue consecutive queries from a single search session or multiple search sessions.

2) For each query in $Q_s$, there should be a good percentage of related cover queries to retrieve information from the topics containing non-sensitive information.

Given the properties explained above the fitness function of the optimal term mapping is defined as:

$$fitness(Q_s) = \frac{\sum_{q \in Q_s} rcq(q)}{|Q_s|}$$  \hspace{1cm} (1)
map $T$ terms in the $P$ proxy-terms groups. Since the search space to find optimal proxy-term groups is exponential, therefore, we use a local search algorithm using greedy hill-climbing to find near to optimal solutions. The greedy hill climbing is an iterative optimization algorithm that starts with an arbitrary solution [40], [41]. It improves the initial solution by making incremental changes by exploring neighbor solutions. If the neighbor solution improves the current solution, the algorithm accepts the neighbor solution. Otherwise, it keeps the current solution. The algorithm continues exploring neighbor solutions until no further improvements can be obtained. The ProxyTermsPWS constructs the optimal proxy-term groups (using greedy hill-climbing based algorithm) as follows. Given $R$ topics and $T$ unique terms in $R$. The ProxyTermsPWS selects each term $t_i$ of $T$ and assigns the $t_i$ to a topic $r_j$ of $R$ if it contains the highest average term proximity [28] for the $t_i$ with the terms of seed queries of $r_j$. We estimate the average term proximity using the average distance between the $t_i$ and the terms of seed queries of $r_j$. Each topic of $R$ contains an equal number of terms. If the $t_i$ appears in both sets of topics the priority is given to the topic containing sensitive information even if the other topic holds higher average term proximity. The ProxyTermsPWS generates $P$ empty proxy-term groups, and randomly selects $M$ topics from the collection with at least one topic from the set containing sensitive information and places the terms of topics in the proxy-term groups using the following approach.

The ProxyTermsPWS sorts the terms in selected $M$ topics according to a number of proximity closed neighbors that the terms have with other topics in $R$. The ProxyTermsPWS places the sorted terms in the proxy-term groups. Each group contains terms from different $M$ topics. Once all the terms of the first $M$ topics have been placed, the ProxyTermsPWS selects new $M$ topics from the collection that have proximity closed neighbors with the terms of old topics of the previous iteration. If the topics of $R$ do not contain neighbors with the old topics, then the ProxyTermsPWS selects any random $M$ topics. For these $M$ topics, the ProxyTermsPWS sorts the terms according to a number of proximity closed neighbors that they have with other topics of $R$, and places the terms in new proxy-term groups. The ProxyTermsPWS continues placing the terms in the proxy-term groups until not all the terms of topics in $R$ have been placed. The terms of the topics containing sensitive information are always placed at the end. The ProxyTermsPWS uses the first term of each group as a proxy term and others as cover terms.

The initial mapping of terms in the proxy-term groups can be poor as there can be many queries in $Q_s$ for which the ProxyTermsPWS does not generate relevant cover queries. The ProxyTermsPWS improves the fitness of proxy-term groups using the following search space exploration heuristics. The ProxyTermsPWS keeps applying the heuristics until no improvement is obtained after performing many iterations.

**Algorithm 1 Heuristic#1: Switch Topics**

**Data:** $Q$, $R$, $T$, $M$

$i ← 0$;

while $i ≠ TerminationCriteria$ do

randomly select $r_i ∈ R$;

randomly select $r_j ∈ R$;

oldFitness ← fitness($Q_i$) using (1);

switch proxy groups of $r_i$ and $r_j$;

if fitness($Q_i$) < oldFitness then

switch back proxy groups of $r_i$ and $r_j$;

end

$i ← i + 1$;

end

1) **SWITCHING TOPICS**

Because the ProxyTermsPWS randomly selects topics and places them in the proxy-term groups, the initial selection can be poor. This heuristic (see Algorithm 1) improves the mapping by switching the proxy-term groups of topics using the following steps.

1) The ProxyTermsPWS randomly selects two topics $r_i$ and $r_j$ of $R$ from the topics containing non-sensitive information and switches the proxy-term groups of $r_i$ and $r_j$.

2) If the switch improves the solution, i.e., the terms combinations of $r_i$ and $r_j$ produce better query obfuscation (according to the fitness properties listed above) for the queries in $Q_s$, then the ProxyTermsPWS accepts the current switch. Otherwise, it returns back to the old proxy-term groups of $r_i$ and $r_j$. Switching terms of topics using pair of topics is better as it tracks which pairs provide a high improvement on the fitness of proxy dictionary. If an algorithm selects more than two topics for switching terms, then the algorithm needs to maintain additional information such as which pairs improve the fitness and which pairs do not improve the fitness. This will incur additional computation cost on the algorithm.

2) **SWITCHING TERMS OF TOPICS CONTAINING SENSITIVE INFORMATION**

This heuristic (see Algorithm 2) improves the mapping by switching the proxy-terms groups of only two terms.

1) The ProxyTermsPWS randomly selects a topic $r_{i_1}$ containing sensitive information, and randomly selects two terms ($t_x$ and $t_y$) of $r_{i_1}$.

2) The ProxyTermsPWS switches the proxy-term groups of $t_x$ and $t_y$.

3) If the switch improves the solution, i.e., the terms combinations of $r_i$ produce better query obfuscation for the queries in $Q_s$, then the ProxyTermsPWS accepts the current switch, otherwise, it returns back to the old proxy-term groups of $t_x$ and $t_y$. 
Algorithm 2 Heuristic#2: Switch Terms of Topics Containing Sensitive Information

Data: $Q, R, T, M$

\[
\begin{align*}
    i &\leftarrow 0; \\
    \text{while } i \neq \text{TerminationCriteria} \text{ do} \\
    &\text{randomly select } r_i \in R, \text{ where } r_i \text{ is a topic containing sensitive information}; \\
    &\text{randomly select } t_x \in r_i; \\
    &\text{randomly select } t_y \in r_i; \\
    &oldFitness \leftarrow \text{fitness}(Q_x) \text{ using } (1); \\
    &\text{switch proxy groups of } t_x \text{ and } t_y; \\
    &\text{if } \text{fitness}(Q_x) < oldFitness \text{ then} \\
    &\quad \text{switch back proxy groups of } t_x \text{ and } t_y; \\
    &i \leftarrow i + 1; \\
\end{align*}
\]

Algorithm 3 Heuristic#3: Switch Terms of Proxy-Term Groups

Data: $Q, R, T, M$

\[
\begin{align*}
    i &\leftarrow 0; \\
    \text{while } i \neq \text{TerminationCriteria} \text{ do} \\
    &\text{randomly select } t_x \in T, \text{ where the topic of } t_x \text{ contains sensitive information}; \\
    &T_x \leftarrow \text{top } \delta \text{ proximity neighbours of } t_x; \\
    &P_x \leftarrow \text{proxy-term identifiers of } T_x; \\
    &Q_x \leftarrow \text{cover queries using the cover terms in } P_x; \\
    &\text{sort cover term in } Q_x \text{ according to how best the queries in } Q_x \text{ obfuscate queries of } Q_x; \\
    &T_z \leftarrow \text{worst fitness } k \text{ cover terms of } P_x; \\
    &\text{randomly select } t_y \notin T_z; \\
    &T_y \leftarrow k \text{ proximity close neighbours of } t_y; \\
    &oldFitness \leftarrow \text{fitness}(Q_y) \text{ using } (1); \\
    &\text{switch proxy groups of terms in } T_y \text{ and } T_z; \\
    &\text{if } \text{fitness}(Q_y) < oldFitness \text{ then} \\
    &\quad \text{switch back proxy groups of term in } T_y \text{ and } T_z; \\
    &i \leftarrow i + 1; \\
\end{align*}
\]

3) SWITCHING TERMS OF PROXY-TERM GROUPS

This heuristic (see Algorithm 3) first retrieves the poor cover terms of proxy-groups that do not generate relevant queries. Then, it switches the proxy-groups of poor cover terms with the proximity neighbors of terms of proxy-groups that are not poor.

1) The ProxyTermsPWS randomly selects a term $t_x$ in $T$ from the topics containing sensitive information, and retrieves its top $\delta$ proximity neighbour terms ($T_x$).
2) The ProxyTermsPWS retrieves the proxy-term groups of $T_x$, and places the identifiers of proxy-term groups in the set $P_x$.
3) From the combinations of covers terms of $P_x$, the ProxyTermsPWS generates queries of $Q_x$ containing sensitive terms and stores in $Q_x$. For the $Q_x$ queries, ProxyTermsPWS sorts the cover terms in $P_x$ according to how best their combinations generate relevant queries to obfuscate queries of $Q_x$. This provides the fitness of cover terms. The ProxyTermsPWS then selects $k$ worst fittest cover terms of $P_x$ and places in the set $T_z$.
4) From the cover terms in $P_x$, the ProxyTermsPWS randomly selects a term $t_y$ that is not in $T_z$ and retrieves its proximity $k$ closed neighbours and places in the set $T_y$.
5) The ProxyTermsPWS then switches the proxy-term groups of the terms in $T_z$ and $T_y$.
6) If the switch improves the solution, i.e., the term combinations of $T_z$ and $T_y$ produce better query obfuscation for the $Q_x$, then the ProxyTermsPWS accepts the current switch, otherwise, returns back to the old proxy-term groups of $T_z$ and $T_y$.

Once near to optimal proxy-term groups have been achieved, the ProxyTermsPWS indexes the proxy-term groups in the proxy dictionary.

Example: Suppose a collection contains two set of topics. Let $\{a, b, \ldots, g\}$ are the terms of topics containing sensitive information, and $\{i, j, \ldots, x\}$ are the terms of topics containing non-sensitive information. Table 3 shows the proximity closed neighbours of the terms of collection. Let ProxyTermsPWS partitions the terms into eight proxy-term groups with a size of three terms. The initial random proxy and cover terms mapping is shown in Table 4. The initial term mapping is poor because if the user issues the true query $(a, b)$ then the ProxyTermsPWS generates many irrelevant proxy and cover queries $(p, l), (q, x), (q, l), (p, x), (a, l), (a, x), (p, b),$ and $(q, b)$ but does not generate any relevant cover query. The ProxyTermsPWS improves the initial terms mapping by switching the $\delta = 3$ poor cover terms of term $a$ with the $\delta = 3$ proximity closed neighbours of term $p$ (see Algorithm 3). Although the term $a$ has all poor cover terms, however, let $j, l, o$ are the top $\delta = 3$ poor cover terms. The ProxyTermsPWS then randomly selects a term $t_y$ from the cover terms of $a$ that is not in the $(j, l, o)$ and switches the $(j, l, o)$ with the proximity neighbours of $t_y$. Let $p$ is the random term and according to Table 3 $(m, n, o)$ are the proximity neighbours of $p$. The ProxyTermsPWS switches the proxy-term groups of $(j, l, o)$ with the proxy-term groups of $(m, n, o)$. Table 5 shows the proxy-term groups after switching terms. The switch generates better terms mapping because if the user issues the true query $(a, b)$, then the ProxyTermsPWS now generates a proxy query $(p, n)$ which has related terms. The other cover queries $(p, x), (p, b), (q, n), (q, x), (q, b), (a, n)$, and $(a, x)$ still have unrelated terms, however, ProxyTermsPWS can improve the terms mapping of these cover queries by applying terms switching.

C. GENERATING PROXY QUERIES

The proxy dictionary maps the terms of proxy topics with the cover topics. The IR system makes publicly available the
TABLE 3. Proximity closed neighbours of the terms.

| Term | Proximity Terms | Closed Neighbour |
|------|----------------|-----------------|
| a    | b, c, d        |                 |
| b    | a, c, d        |                 |
| c    | a, b, d, e     |                 |
| d    | a, b, c        |                 |
| e    | c, f           |                 |
| f    | e, g, h        |                 |
| g    | f, h           |                 |
| h    | f, g           |                 |
| q    | r, s, t        |                 |
| r    | q, s, t        |                 |
| s    | q, r, t        |                 |
| t    | q, r, s, u     |                 |

TABLE 4. Initial proxy-term groups of the collection.

| Proxy Terms | Cover Terms |
|-------------|-------------|
| i           | r, g        |
| j           | u, c        |
| k           | s, e        |
| l           | x, b        |
| m           | v, f        |
| n           | w, h        |
| o           | t, d        |
| p           | q, a        |

TABLE 5. Proxy-term groups after switching proxy-term groups of poor cover terms of term a with the neighbours of term p. The poor cover terms of a are (j, l, o) and the neighbours of term p are (m, n, o).

| Proxy Terms | Cover Terms |
|-------------|-------------|
| i           | r, g        |
| m*          | u, c        |
| k           | s, e        |
| n*          | x, b        |
| f*          | v, f        |
| t*          | w, h        |
| o           | t, d        |
| p           | q, a        |

proxy dictionary to all users. When a user issues a search query to the IR system, the client machine locates query terms in the proxy dictionary and transforms the terms of the user’s query with the proxy terms. The client machine issues proxy query to the IR system. The IR system receives the proxy query and generates cover queries from the proxy dictionary. Since ProxyTermsPWS indexes a term only once in the proxy dictionary, the true query can appear within any combination of proxy and cover terms. The IR system generates cover queries using all combinations of proxy and cover terms and ignores the combinations which do not appear in the web collection. Once the valid combinations are available, the IR system individually processes each query combination and prepares a rank list, and returns the rank lists to the client machine.

IV. DEPLOYING ProxyTermsPWS

ProxyTermsPWS provides the private web service facility using the following three architectures.

A. IR SYSTEM BASED ProxyTermsPWS

In this architecture, the IR system provides ProxyTermsPWS facility. Figure 5 shows the architecture of this approach. The IR system identifies proxy and cover topic and generates proxy dictionary. Given proxy dictionary, the client machines transform user queries to proxy queries and issue them to the IR system. The IR system generates cover queries from the proxy queries, retrieves and returns the rank lists to client machines. This approach provides the highest level of privacy as all users search for information through proxy queries. The IR system cannot reveal whether a user is retrieving information for proxy query or cover queries.

B. MIDDLEWARE BASED ProxyTermsPWS

This architecture is suitable when an IR system does not support ProxyTermsPWS. A middleware is used for providing ProxyTermsPWS facility. Figure 6 shows the architecture of this approach. The middleware collects proxy and cover topics by crawling documents from the web and generates a proxy dictionary. The middleware can be deployed at the organization or country level. The role of middleware is similar to the proxy server. It receives proxy queries from users and retrieves information from the IR system by generating cover and true queries with the help of a proxy dictionary. It returns back the retrieved information to users once the results of queries are available from the IR system. The middleware does not create a web search privacy threat to users because the users retrieve information from the middleware through proxy queries. For example, consider an organization that does not want to disclose true queries of its users to an external IR system. The organization can provide ProxyTermsPWS facility using a middleware. All users of the organization submit proxy queries directly to the middleware. The middleware generates cover queries from the proxy queries and forwards them to the IR system. The IR system retrieves rank lists and returns the rank lists to middleware. This approach provides a medium level of privacy as the IR system receives proxy queries only from the users of an organization but not from all web users. However, if the middleware is provided at the country level, this approach provides the highest level of privacy similar to the first architecture.
C. CLIENT MACHINE BASED ProxyTermsPWS

This architecture is suitable when there is no ProxyTermsPWS facility available from the IR system and middleware. Figure 7 shows the architecture of this approach.

The functionality of this architecture is similar to OB-PWS based architecture described in [22]. Under this scenario, a client machine locally crawls documents from the World Wide Web including a set of topics containing sensitive and
FIGURE 7. The architecture of client machine based ProxyTermsPWS.

non-sensitive information. It prepares cover and proxy topics for generating proxy dictionary. To retrieve information from the IR system, the client machine generates a proxy and cover queries, and submits them to the IR system. The IR system returns the rank lists of cover and proxy queries to the client machine. This architecture provides the lowest level of privacy as the client machine directly issues a true query to the IR system along with cover queries. For improving privacy, client machines can share their proxy dictionaries with other client machines.

V. EXPERIMENTS AND EFFECTIVENESS ANALYSIS

This section explains the effectiveness of ProxyTermsPWS on two collections. We compared the effectiveness of ProxyTermsPWS with two related query obfuscation-based PWS techniques, GooPIR and Qu-OB-PWS. GooPIR is an OB-PWS based PWS technique [35]. It submits $k$ cover queries to obfuscate true queries. It generates cover queries by selecting keywords of cover queries from a term dictionary. The main limitation of GooPIR is that it does not consider relatedness in cover queries when a user issues consecutive queries related to a similar topic. Qu-OB-PWS [22] is another query obfuscation-based PWS technique. It achieves relatedness between consecutive cover queries.² Qu-OB-PWS achieves relatedness by generating related cover queries from a set of hierarchically organized topics. This ensures that the cover queries add multiple fake search intents.

A. DOCUMENT COLLECTIONS

To analyze the effectiveness of proposed technique, we require a collection with a set of topics to classify the topics into cover and proxy topics. We generated two collections. For generating first collection, we manually selected a set of topics containing sensitive and non-sensitive information, and used these topics as seed queries to crawl and downloaded the documents related to topics using focused crawling [42]. To achieve this, first we defined a manual set of 30 topics containing sensitive information and 150 topics containing non-sensitive information. We wanted to construct two set of topics. One set for topics containing sensitive information and other set for the topics containing non-sensitive information. The selected topics are related to health, weapons, hacking, casino, torrent softwares, cooking recipe, vehicle maintenance, electric vehicles, tourism, movies, and pets. Table 6 shows an example list of topics that we used in experiments. We call this crawled collection.

We generated a second collection from the AOL query log dataset.³ We call this AOL collection. AOL collection has queries but no topics. For identifying topics from queries we grouped queries into clusters. Each cluster represents an independent topic. We clustered the queries based on the similarity of terms in queries. We ignored stop words during constructing clusters. For the queries that do not have any similar term with other queries. We retrieved the top 10 documents from the google search engine and then placed them to existing clusters based on the similarity between top frequent terms of retrieved documents and terms of clusters. After identifying topics (clusters) from queries we partitioned the topics into sensitive and non-sensitive topics. We placed all topics into sensitive category if the terms of clusters are related to health, disease, hacking, religion, weapon, politics, and employment.

²https://github.com/wasiahmad/intent_aware_privacy_protection_in_pws
³https://www.kaggle.com/dineshydv/aol-user-session-collection-500k
TABLE 6. Seed topics and seed queries.

| Topic              | Sensitive or Non-sensitive Topic | Seed Queries                                                                 |
|--------------------|----------------------------------|-------------------------------------------------------------------------------|
| Weapon             | Sensitive                        | Pistol disassembling, Weapon disassembling techniques, Gunsmith tools, Gunsmith techniques |
| Hacking            | Sensitive                        | With hacking, Email hacking tools, Hacking websites, Hacking tutorial, Proxy softwares, Hacker cost, Credit card hacking |
| Casino             | Sensitive                        | Gambling websites, Sports betting                                           |
| Torrent            | Sensitive                        | Torrent websites, Torrent softwares download, Torrent movies download, Earning money from Torrent, Proxy servers for torrents |
| Depression Treatment | Sensitive                      | Depression treatment, Personality disorder symptoms, Depression symptoms, Depression treatment therapy, Depression treatment medication |
| Pregnancy          | Sensitive                        | Pregnancy symptoms, Birth control pills, Missed period, Early miscarriage symptoms, Pregnancy tests |
| HIV                | Sensitive                        | HIV symptoms, HIV treatment, HIV blood tests, HIV medications                |
| Cooking Recipe     | Non-Sensitive                    | Roasted chicken, Tandoori chicken, BBQ spices, BBQ Fish, Vegetable pasta recipe, Baked vegetable recipe |
| Vehicle Maintenance| Non-Sensitive                    | Engine oil changing, CVT transmission problems, CVT maintenance, Engine fluid color, Replacing windshield wipers, Power steering fluid, Brake fluid, Transmission fluid |
| Tourism            | Non-Sensitive                    | Best cities in Europe, Hotel booking websites, Cheap airline tickets, Car rental websites |
| Pet                | Non-Sensitive                    | Online cat foods, Pet grooming, Cat toys, Pet vaccinations                   |

Figure 8 shows the distributions of documents relative to document length and vocabulary size. We removed all stop words from the documents and smoothed the remaining terms using the term stemming approach and indexed the roots of the terms in the proxy dictionary. For each topic, we indexed only the top 5000 most frequent terms in the proxy dictionary obtained from the documents of seed topics.

B. GENERATING PROXY DICTIONARY

Once the terms of proxy and cover topics have been obtained, the ProxyTermsPWS partitioned the terms into $P$ proxy-term groups using the proxy-terms mapping technique presented in section 3.2. Figure 9 shows best fitness($Q_i$) of ProxyTermsPWS over 150k iterations for generating optimal term mapping. Each group contains 6 terms with at least one term from the topics containing sensitive information.
Within the first few iterations, the search space heuristics quickly improved the fitness due to poor initial mapping between proxy and cover terms. However, once the term mapping stabilized, fitness improved very slowly over the course of a large number of iterations. We performed the experiments on Intel Core i7 7th Gen CPU with a processor speed of 2.11 GHz and main memory of size 16 GB. A single iteration took on average four seconds to calculate the fitness of term mapping. The experiments consumed around 167 hours to train an optimal term mapping. For generating $Q_s$ queries from the topics containing sensitive information, we exhaustedly combined the most frequent terms of the topics containing sensitive information into two and three terms combinations and used the combinations as search queries. We further removed all those term combinations from the $Q_s$ that retrieve less than five documents. Figure 10 shows several cover queries with a good query quality score for obfuscating queries of $Q_s$. According to results around 98% of queries have at least three high-quality cover queries, while 82% of queries have at least five high-quality cover queries. Quality of queries are estimated using $\text{MaxVAR}$ (maximum term weight variability) [38], [39]. We classified a query as a high-quality query if it has $\text{MaxVAR}$ score greater than 0.011 [36]. Figure 11 shows the distribution of the total number of queries per document relative to the vocabulary size of documents.

Since the focus of experiments is to preserve the privacy of search queries that appeared in the seed topics, therefore, the ProxyTermsPWS first indexed all the terms in the proxy dictionary which appeared in the seed topics. For the rest of the English dictionary terms which did not appear in the seed topics, we randomly mixed them in groups of 6 terms and indexed them in the proxy dictionary. Table 7, Table 8 and Table 9 show parts of proxy dictionary related to topics (containing sensitive information): Depression Treatment, Email Hacking, and Weapon Dissembling. The terms in the proxy-term groups are sorted based on topic identifiers. The terms of the topics containing sensitive information can be seen at the end of the tables.

C. MEASURING EFFECTIVENESS

To analyze the effectiveness of the proposed technique we generated a manual set of 100 test queries with consecutive 5 queries for each test query. Each query contained two or three terms. We obtained consecutive similar queries from the pseudo relevance feedback (PRF) documents [45]. For each test query, we first retrieved top 20 documents, and then we obtained top 30 terms that were proximity closed to the terms of test query [28]. The consecutive queries were then obtained by keeping one term from the initial terms and adding two or three (proximity closed) terms from the PRF documents. The objective of the experiments was to analyze the effectiveness of the system whether the system generates related consecutive cover queries when the users issue consecutive related queries.

We analyzed the similarity between consecutive queries using cosine similarity [46]. Mathematically, it measures
the cosine of the angle between two vectors projected in a multi-dimensional space. Cosine similarity is one of the most popular similarity measures applied to information retrieval applications and document clustering. The documents of two clusters are represented using term vectors. The similarity between the two queries corresponds to the average correlation between the term vectors, which is calculated as follows.

$$\text{Sim}(A, B) = \frac{\sum_{i=1}^{m} A_i B_i}{\sqrt{\sum_{i=1}^{m} A_i^2} \times \sqrt{\sum_{i=1}^{m} B_i^2}}$$

(2)

where $A$ and $B$ are m-dimensional vectors over the term set $T = \{t_1, \ldots, t_m\}$. Each dimension represents a term with its weight in the document, which is non-negative. As a result, the cosine similarity is non-negative and bounded between $[0, 1]$.

We measured the $\text{Sim}$ using top 20 documents of query and with the top 100 terms of the documents. A high value of $\text{Sim}$ indicates two queries are highly related to each other. Ideally, we want to achieve a high score of $\text{Sim}$ from the consecutive cover queries when the true consecutive queries of users are related to each other, and a low $\text{Sim}$ score when
the true consecutive queries are unrelated to each other. This reflects that the proposed system generates related or unrelated consecutive cover queries.

### D. Effectiveness Analysis

To analyze the effectiveness, we divided the test queries into five partitions. Each partition contained four sets of queries. The sets are generated using the following approach. In the first set of the first partition, we randomly placed five test queries. The query sets of this partition did not contain any consecutive queries, thus queries were highly unrelated to each other. The second partition contained four random test queries with an additional one related query for the first test query. The position of the related query was selected randomly. The third partition contained three random test queries with additional two consecutive related queries for the first test query. Similar to the second partition, the positions of related queries were selected randomly in the query set. The fourth partitions contained two random test queries with consecutive three queries related to the first query. The final partition contained a single random test query with consecutive four related queries. Queries of all partitions were issued to the IR system. The client machine transformed the test queries with the proxy queries and issued them to the IR system. The IR system generated cover queries and returned the retrieved documents to the client machine. The similarity scores of the consecutive test and cover queries are measured using (2).

Table 10 shows the average similarity scores of test and cover queries of all partitions. The partition#1 has unrelated test queries. We calculated the $\text{Sim}$ for all queries of partition#1 by assuming them as consecutive queries. The partition#1 has a low $\text{Sim}$ score for both test and cover queries. This reflects when a user issues queries related to different topics within a single search session then the proposed system generates unrelated cover queries. The partition#2 has one consecutive related query for each test query. We calculated the $\text{Sim}$ between the test and consecutive queries only. Similarly, for the cover queries, we calculated the $\text{Sim}$ score between the first cover query and other cover queries only. We used the same similarity calculation for the other partitions.

Table 10 and Table 12 show the effectiveness of PWS techniques on both collections when consecutive queries were issued from a single session. Table 11 and Table 13 show the effectiveness when consecutive queries were issued from multiple sessions. We simulated the multiple sessions by assuming that search sessions did not know the search queries of other sessions. On single-session experiments (Table 10 and Table 12), the results showed that Qu-OB-PWS and the proposed approach achieved high effectiveness for generating consecutive cover queries. As GooPIR does not consider the relatedness of current queries with the past queries, it showed low effectiveness in generating consecutive cover queries when the search session contained related queries.

On multiple search sessions experiments (Table 11 and Table 13), the proposed approach showed high effectiveness than Qu-OB-PWS and GooPIR. The Qu-OB-PWS showed less effectiveness because Qu-OB-PWS considers relatedness between queries in the current search session only. However, if a user continues the same search task from multiple sessions, the Qu-OB-PWS cannot capture relatedness between queries that span across multiple sessions. This is even more difficult to achieve when users issue queries from different devices.

According to the results, the ProxyTermsPWS achieved high similarity scores for the cover queries when there were different numbers of consecutive related queries in the partitions. Table 14, Table 15 and Table 16 show a sample of true, proxy and cover queries (for the crawled collection) generated from the ProxyTermsPWS for the partition#5 related to topics (containing sensitive information): Depression Treatment, Email Hacking and Weapon Disassembling.
TABLE 14. Table shows cover and original queries of a search session generated from the proxy dictionary for the topic Depression Treatment. Terms of the queries are ordered according to the best order found in the collection. Second column shows best consecutive cover queries of the session.

| Query#1 | Top#5 Relevant Proxy and Cover Queries | Top#5 Irrelevant Cover Queries |
|---------|----------------------------------------|--------------------------------|
| Depression Symptom (True Query) | Depression Cook | Cook Sofa |
| Cook Chicken (Proxy Query) | Lubricant Security | Electric Chicken |
| Engine Lubricant (Cover Query) | Chicken Engine | Symptom Cook |
| Sofa Design (Cover Query) | | |
| Mobile Security (Cover Query) | | |
| Electric Car (Cover Query) | | |

| Query#2 | Depression Treatment (True Query) | Depression Roast |
|---------|----------------------------------|-----------------|
| Chicken Roast (Proxy Query) | Chicken Depression | |
| Engine Oil (Cover Query) | Sofa Treatment | |
| Sofa Shop (Cover Query) | Mobile Roast | |
| Google Mobile (Cover Query) | Mobile Chicken | Truck Roast |
| Electric Truck (Cover Query) | | |

| Query#3 | Depression Treatment Medication (True Query) | Depression Roast Juicy |
|---------|-----------------------------------------------|-----------------------|
| Juicy Chicken Roast (Proxy Query) | Chicken Puel Room | |
| Engine Oil Fuel (Cover Query) | Chicken Puel Charge | |
| Room Sofa Shop (Cover Query) | Roast Sofa Software | |
| Google Mobile Software (Cover Query) | Mobile Roast Medication | Truck Juicy Medication |
| Electric Truck Charge (Cover Query) | | |

| Query#4 | Depression Therapy (True Query) | Depression Performance |
|---------|---------------------------------|------------------------|
| Chicken Lemon (Proxy Query) | Lemon Malware | |
| Engine Performance (Cover Query) | Depression Corner | |
| Corner Sofa (Cover Query) | Corner Emission | |
| Mobile Malware (Cover Query) | Sofa Malware | |
| Electric Emission (Cover Query) | Therapy Emission | |

| Query#5 | Depression Treatment Therapy (True Query) | Depression Malware Truck |
|---------|--------------------------------------------|--------------------------|
| Lemon Chicken Roast (Proxy Query) | Chicken Malware Corner | |
| Electric Engine Performance (Cover Query) | Therapy Malware Emission | |
| Corner Sofa Shop (Cover Query) | Treatment Mobile Roast | |
| Google Malware Malware (Cover Query) | Roast Sofa Malware | Lemon Mobile Truck |
| Electric Truck Emission (Cover Query) | | |

TABLE 15. Table shows cover and original queries of a search session generated from the proxy dictionary for the topic Email Hacking. Terms of the queries are ordered according to the best order found in the collection. Second column shows best consecutive cover queries of the session.

| Query#1 | Top#5 Relevant Proxy and Cover Queries | Top#5 Irrelevant Cover Queries |
|---------|----------------------------------------|--------------------------------|
| Email Account Hack (True Query) | Email Pay Temperature | Account Fashion Weather |
| HP Printer Review (Proxy Query) | Review Debit Temperature | Email Frame Debit |
| Football Group League (Cover Query) | Printer League Weather | Group Debit Temperature |
| Fashion Frame Brand (Cover Query) | | |
| Debit Pay Card (Cover Query) | | |
| Weather Temperature Forecast (Cover Query) | | |

| Query#2 | Password Hack (True Query) | Password Weather |
|---------|----------------------------|-----------------|
| Wireless Printer (Proxy Query) | Password Rain | Wireless Rain |
| Tournament League (Cover Query) | Wireless Weather | Credit Rain |
| Fashion Sunglasses (Cover Query) | Credit Pay | Printer Weather |
| Credit Card (Cover Query) | | |
| Rain Weather (Cover Query) | | |

| Query#3 | Online Email Hack (True Query) | Hack France Weather |
|---------|-------------------------------|---------------------|
| Home Printer Review (Proxy Query) | Home Debit Weather | |
| France Football League (Cover Query) | Printer Football Sunday | |
| Fashion Frame Trend (Cover Query) | Trend Point Sunday | |
| Debit Card Point (Cover Query) | | |
| Sunday Weather Forecast (Cover Query) | League Frame Forecast | Sunday Printer Weather |

| Query#4 | Email Hack Website (True Query) | Email Frame Storm |
|---------|---------------------------------|------------------|
| Color Printer Review (Proxy Query) | Website Qualify Storm | Color Debit Forecast |
| Football League Qualify (Cover Query) | Qualify Employee Weather | Quality Employee Weather |
| Women Fashion Frame (Cover Query) | Review Debit Storm | Review Debit Storm |
| Employee Debit Card (Cover Query) | Email Color Qualify | |
TABLE 16. Table shows cover and original queries of a search session generated from the proxy dictionary for the topic Weapon Disassembling. Terms of the queries are ordered according to the best order found in the collection. Second column shows best consecutive cover queries of the session.

| Query | Top#5 Relevant Proxy and Cover Queries | Top#5 Irrelevant Cover Queries |
|-------|---------------------------------------|-------------------------------|
| Query#1 | Pistol Disassembling (True Query) Pet Clinic (Proxy Query) Tpлинk Router (Cover Query) Vehicle Accident (Cover Query) Paint Board (Cover Query) Flower Fragrance (Cover Query) | Pistol Router Pet Router Tpлинk Flower Tpлинk Paint Tpлинk Flower |
| Query#2 | Pistol Disassembling Techniques (True Query) Pet Care Clinic (Proxy Query) Tpлинk Ethernet Router (Cover Query) Buy Accident Vehicle (Cover Query) Acrylic Paint Board (Cover Query) Flower Fragrance Warmer (Cover Query) | Pistol Pet Router Disassemble Pet Warmer Clinic Vehicle Flower Tpлинk Paint Flower |
| Query#3 | Firearm Disassembly Tools (True Query) Dog Clinic Location (Proxy Query) Mesh Router Device (Cover Query) Accident Repair Inspection (Cover Query) Palette Board Material (Cover Query) Spring Flower Scent (Cover Query) | Firearm Dog Repair Firearm Mesh Spring Dog Router Material Mesh Board Spring Device Material Spring |
| Query#4 | Gunsmith Tools (True Query) Vaccination Location (Proxy Query) Device Signal (Cover Query) Repair Location (Cover Query) Art Material (Cover Query) Fragrant Scent (Cover Query) | Gunsmith Signal Vaccination Signal Location Material Signal Price Signal Scent |
| Query#5 | Gunsmith Manual (True Query) Vaccination Service (Proxy Query) Signal Booster (Cover Query) Service Price (Cover Query) Art Stand (Cover Query) Flower Bloom (Cover Query) | Gunsmith Bloom Signal Offer Gunsmith Flower Vaccination Bloom Stand Bloom |

Engine Lubricant, Engine Oil and Engine Oil Fuel (see Table 14) which are highly related to each other.

VI. CONCLUSION

The paper presents a proxy-terms based query obfuscation technique ProxyTermsPWS for private web search. ProxyTermsPWS generates proxy dictionary which maps terms of topics containing sensitive information and topics containing non-sensitive information in such a way that users retrieve the information for the topics containing sensitive information through proxy queries. The major challenge is how to achieve optimal proxy-terms mapping so that the ProxyTermsPWS generates relevant related queries when the users issue related queries. As the search space to find optimal proxy-terms mapping is exponential, therefore, we propose greedy hill-climbing based heuristics to search near to optimal mapping.

For future work, we are currently investigating how to automatically classify queries containing sensitive and non-sensitive terms using text classification techniques. The focus of current experiments is to provide the query obfuscation only for the queries related to topics containing sensitive information. As future work, we are also investigating how to provide query obfuscation for the whole web search.

REFERENCES

[1] R. Khan, A. Ahmad, A. O. Alsayed, M. Binsawad, M. A. Islam, and M. Ullah, “Qupid attack: Machine learning-based privacy quantification mechanism for PIR protocols in health-related web search,” Sci. Program., vol. 2020, Jul. 2020, Art. no. 8868686.
[2] I. Weber, V. R. K. Garimella, and E. Borra, “Political search trends,” in Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR), 2012, p. 1012.
[3] I. Weber, A. Popescu, and M. Pennacchiotti, “PLEAD 2012: Politics, elections and data,” in Proc. 21st ACM Int. Conf. Inf. Knowl. Manage., 2012, pp. 2768–2769.
[4] M. Eirinaki and M. Vazirgiannis, “Web mining for web personalization,” ACM Trans. Internet Technol., vol. 3, pp. 1–27, Feb. 2003.
[5] F. M. Fucica and P. L. Lanzi, “Mining interesting knowledge from weblogs: A survey,” Data Knowl. Eng., vol. 53, no. 3, pp. 225–241, Jun. 2005.
[6] R. Jones, R. Kumar, B. Pang, and A. Tomkins, ““I know what you did last summer”: Query logs and user privacy,” in Proc. 16th ACM Conf. Conf. Inf. Knowl. Manage. (CIKM), 2007, pp. 909–914.
[7] B. Tancer, Click: What Millions People Are Doing Online Why it Matters. Hyperion, 2008.
[8] Z. Dou, R. Song, and J.-R. Wen, “A large-scale evaluation and analysis of personalized search strategies,” in Proc. 16th Int. Conf. World Wide Web (WWW), 2007, pp. 581–590.
[9] Y. Chen, D. Pavlov, and J. F. Canny, “Large-scale behavioral targeting,” in Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2009, pp. 209–218.
