Journal of Ambient Intelligence and Humanized Computing (2021) 12:9891–9907
https://doi.org/10.1007/s12652-020-02736-y

ORIGINAL RESEARCH

PAPIR: privacy-aware personalized information retrieval

Anas El-Ansari[1] · Abderrahim Beni-Hssane[1] · Mostafa Saadi[2] · Mohamed El Fissaoui[3]

Received: 4 March 2020 / Accepted: 20 November 2020 / Published online: 28 January 2021

Abstract
The problem of information overload on the Internet increased the need for personalized information retrieval (PIR) systems capable of providing information that corresponds to the user interests. Although, for most people, the word personalization comes with trust issues and privacy concerns. Since giving the user a personalized browsing experience usually comes at the cost of his privacy. Thus, most people are afraid of using such applications. To address this issue, we propose a new model for privacy protection in PIR systems. Our model aims at achieving a trade-off between the personalization quality and the privacy risk, to keep the latter under control. We have studied the assets and drawbacks of the existing profile-based PIR structures, from a privacy protection perspective, along with the possible privacy threats in this field in a threat modeling approach. The model we propose is based on the vector space model and targets profile-based PIR systems. It uses query expansion and re-ranking algorithms on the client-side to ensure personalization quality. While privacy protection is ensured during the personalization process, by taking into consideration the user's privacy requirements, and through encryption. We use the Advanced Encryption Standard (AES) algorithm to protect user data at-rest and a fully homomorphic encryption (FHE) scheme for data in-transit and in-use protection. To prove the feasibility and efficiency of our model, this paper includes a proof-of-concept implementation with proper experimental results.

Keywords Personalization · Privacy · Information retrieval · Homomorphic encryption

1 Introduction

It has become difficult for users to find information on the web that satisfies their needs since information resources continue to grow and have far exceeded human processing capabilities. The sheer abundance of information often prevents users from finding desired information, services and products, or aggravates making informed choices.

For those reasons, users need intelligent personalized web applications that simplify information access and content discovery, based on user preferences, and delivers services in a most valuable and convenient way. An example of personalized web applications that have recently become tremendously popular is the personalized web search system. These systems offer users personalized answers about services, products, and information.

Two main challenges face PIR systems in general. The first one is building a user profile that accurately represents the user's real interests, which was addressed in our previous work El-Ansari et al. (2020a). The second challenge is the privacy protection problem, since giving the user a personalized browsing experience comes at the cost of his privacy. Thus, most people are afraid of using such applications. According to the General Data Protection Regulation (GDPR) Voigt and Von dem Bussche (2017), data controllers must design information systems with privacy in mind. Also, the data subjects should have full access and control over the collected data. Yet most former research works in

✉ Anas El-Ansari
anas.elansari@gmail.com

Abderrahim Beni-Hssane
abenihssane@yahoo.fr

Mostafa Saadi
saadi_mo@yahoo.fr

Mohamed El Fissaoui
elfissaoui.m@ucd.ac.ma

[1] LAROSERI Laboratory, Computer Science Department, Sciences Faculty, Chouaib Doukkali University, El Jadida, Morocco

[2] LaSTI Laboratory, Univ Sultan Moulay Slimane, ENSA Khouribga, B.P 77, 25000 Khouribga, Morocco

[3] MASI laboratory, FP, Mohammed First University, Nador, Morocco

the field focused on improving the personalization quality disregarding user privacy.

In this paper, we propose a new model that aims at improving privacy protection on various levels. The purposes of our research, in general, are first to enhance the personalization quality by using accurate profiles capable of reflecting the user's changing interests. Second, to ensure user's privacy by protecting his sensitive data on the client-side, through the Internet channel, and more importantly on the server-side where most privacy risks come (data misuse, leakage, etc.).

PAPIR is a model that aims at improving both personalization quality and privacy protection. The first is ensured by building accurate user profiles used in query expansion and re-ranking algorithms. The second is modeled by design, taking into consideration each user's privacy requirements, and enforced with a FHE scheme.

This model can be implemented in numerous information retrieval (IR) applications, including search engines, natural language question answering systems, chatbots, library management systems, etc.

The paper is organized as follows; The next section discusses related works. Section 3 presents a comparative study of the existing personalization systems' structures focusing on user privacy, along with the privacy threat modeling. Our model is detailed in Sect. 4. The proof-of-concept implementation measures are discussed in Sect. 5, with experimental evaluation results in Sect. 6. The paper ends with the conclusion and perspectives.

2 Related work

This research relates to the field of Personalized Information Retrieval (PIR) systems, the process of delivering information or items to the user considering his specific needs and interests in the most adequate way and at the right time. These systems collect different types of user data continuously which raises many privacy protection concerns. We discuss former works in this section as follows: PIR applications, personalization methods, and privacy protection solutions.

- *PIR applications*

Some personalized systems were developed to help users find desired products (Amazon Smith and Linden (2017), eBay Greenstein-Messica and Rokach (2018)), improve search results (Yu et al. (2018)), browse news articles (Wu et al. (2019), Bountouridis et al. (2019)), find scientific and research papers (Mohseni et al. (2019)), or perform a combination of the above tasks.

- *Personalization methods*

Most personalized systems build user profiles by collecting and analyzing user's browsing history (visited Web pages as in Dennis et al. (2016) and ElShaweesh et al. (2017)). Other data sources have also been used such as images in Lully et al. (2018), or a combination of the visited Web pages, bookmarks, queries, and search results as in El-Ansari et al. (2020a). This combination boosts user profiling performance, especially when combined with Big Data technologies.

To construct the user profiles, a variety of learning techniques are used as including the probabilistic model Chaney et al. (2015), genetic algorithms Lv et al. (2016), clustering Liao and Lee (2016), deep learning techniques Singhal et al. (2017), and the well-known vector space model (Wu et al. 2019; El-Ansari et al. 2020b). Since the user profile might comprise irrelevant topics, Some of the above systems use concept filtering or rating algorithms to improve the profile's accuracy. A survey on user profiling techniques by Eke et al. (2019) provides more details on this subject.

- *Privacy protection solutions*

Most studies focus on improving the personalization quality disregarding the user's privacy issues. Zhu et al. (2010) tried to address the privacy protection problems in personalized systems by protecting user identification using techniques such as the pseudo-identity, the group identity, no identity, and no personal information. Most efforts focus on the second level. For example, Zhu et al. (2010) provided online anonymity for users by generating a group profile of k-users. De-identification techniques are mostly used in personalized systems collecting user's personally identifiable information (PII) and working on aggregate data. However, such techniques may be vulnerable to attacks like unsorted matching attacks, temporal attacks, and complementary release attacks. Tomashchuk et al. (2019) provided a detailed survey of de-identification techniques for privacy protection.

Since in our work we do not use PII, we focus on techniques protecting the user's sensitive data (user profile). In this type of privacy-preserving systems, three main technique lines are used. Differential privacy, Randomized perturbation, and Cryptographic methods.

Differential privacy (DP) has become widely accepted as a model for privacy protection during the past years, this solution preserves privacy by making it difficult for the adversary to infer the presence or absence of any individual in the data set. This technique is designed to work on aggregate data and is most suitable for big data.

For instance, Zhu et al. (2013) proposed another DP scheme for neighborhood-based CF that can select neighbor privately; however, fails to maintain a good trade-off between privacy and accuracy. Authors in Shen and Jin (2016) designed a privacy built-in client that perturb data on the user device. However, the utility of perturbed data may decrease due to the inhered volatility of the whole process. Another work in Zhang et al. (2020) proposes a probabilistic mechanism for mobility datasets releasing based on differential privacy, to give users control over privacy level in location-based services.

The main limitations of differential privacy are first, the amount of noise added to the data; more noise for higher privacy risks which reduces the data utility. It is also vulnerable to insider threats that come from within the personalization server. Desfontaines and Pejó (2020) presented a survey of DP techniques for more details on the subject.

Randomized perturbation (RP) is another noise-based technique that was proposed in Zhu et al. (2015). Authors claim that they can obtain accurate recommendations while adding randomness from a specific distribution to the original user data to counter information exposure. Though, the range of randomness is chosen by experience and does not have a provable privacy protection guarantee. Another work in Polatidis et al. (2017) proposed a multi-level privacy-preserving method for collaborative filtering systems by perturbing each rating before it is submitted to the server. Yet the results showed a decrease in utility. Authors in Liu et al. (2017b) presented a hybrid approach for privacy-preserving recommender systems by combining DP with RP to offer more privacy protection. However, the recommendation accuracy loss with this approach is significant. Most noise-based techniques share the problem of utility loss Siraj et al. (2019).

Both DP and RP techniques are designed for aggregate data privacy ignoring each user's privacy requirements and they both decrease the personalization accuracy. Lately, privacy concerns among users have increased due to unethical data aggregation practices of many recommendation systems. For this reason, in our work, we focus on individual data privacy.

As an *individual privacy (IP) solution*, Shou et al. (2014) claims that better results can be achieved with privacy guarantee if the personalization is only performed based on less sensitive user data. The idea is to expose only the insensitive part of the profile to the search engine. However, with this approach, a significant part of the user profile can be collected by an attacker or the server. Wu et al. (2020) proposed a framework for the privacy protection in book search based on the idea of constructing a group of plausible fake queries for each user query to cover up the sensitive subjects behind users' queries. This approach focused on limited queries with no support for general ones and no practical implementation.

Cryptographic techniques such as Homomorphic Encryption (HE), Searchable Symmetric Encryption (SSE), or Garbled circuits are used as a mechanism to protect private data confidentiality. Authors in Erkin et al. (2012) present a solution to generate private recommendations in a privacy-preserving manner using HE and data packing. While Anjali and Reeshma (2015) proposed a search framework based on Shou et al. (2014)'s model, enhanced with HE to protect user queries but no proper explanation on how HE is implemented. Also, authors in Liu et al. (2017a) used partially HE to design two protocols for privacy-preserving trust-oriented POI recommendation based on the off-line encryption and parallel computing. A recent work Wang et al. (2020) proposed a fully HE scheme for private IR in the cloud; the paper contains theoretical analysis, but no proof-of-concept implementation is discussed. A survey by Zhou et al. (2020) on keyword search with public key encryption can help understand the use of encryption in private IR.

Table 1 presents a comparative analysis of our model with the closely related works mentioned in this section.

Table 1 Comparative analysis with closely related works

Approaches Support (✓) No support (✗) Unknown (?) Homomorphic Encryption (HE) Individual Privacy (IP)	Shou et al. (2014)	Anjali and Reeshma (2015)	Wang et al. (2020)	Wu et al. (2020)	**PAPIR**
User Privacy Control	✓	?	✗	✗	✓
No trust assumptions	✗	✓	✓	✗	✓
Privacy protection solution	IP	IP	✗	IP	IP
Employed encryption technique	✗	HE	HE	✗	HE
Privacy protection guarantee	?	?	✓	?	✓
Personalization utility guarantee	✓	✓	?	✓	✓
Proof-of-concept implementation	✓	✗	✗	✓	✓
Scalability support	✓	✗	?	✗	✓

Our model focuses on countering privacy risks while preserving the personalization utility in PIR systems. The main contribution of our paper can be summarized as follows :

- Presents an overall study on privacy protection in PIR, detailing the structures of existing systems and the possible privacy threats in a threat modeling approach.
- Proposes a new model (based on the vector space model) that enhances the personalization quality by using accurate profiles capable of reflecting the user's changing interests. And ensures user privacy by protecting his sensitive data on the client-side, through the Internet channel, and on the server-side.
- Creates and stores user profiles on the client-side, which are used in query expansion and re-ranking algorithms without data perturbation or accuracy loss.
- Guarantees privacy protection by design, taking into consideration each user's privacy requirements, and uses a fast FHE scheme to protect user data in different states: at-rest, in-transit, and in-use with no assumption of trust in the server.
- Implements a FHE scheme with extended operations over ciphertexts such as comparison, Sum, etc.
- Includes a proof-of-concept implementation with scalability support.
- Provides theoretical and experimental analysis.

PAPIR can be easily adapted and implemented in numerous personalized IR applications; such as search engines, question answering systems, etc.

3 Privacy in personalized IR systems

User privacy is the user's ability to insulate himself, or some of his information and thereby express himself selectively. Privacy is becoming a big concern in many fields such as Social Networks, Cloud Computing, Personalized systems, etc. To protect user privacy in profile-based personalized IR systems, we must consider two contradicting effects. First, such systems improve search quality by collecting more user data. Second, must hide the sensitive data in the user profile to place the privacy risk under control. The storage of the user profile depends on the system structure.

3.1 Personalized IR systems' structures

In this section, we classify personalized search systems into three distinct structures. Based on the user profile's storage (on the client-side or the server-side) and use for personalization.

3.1.1 Server-side personalization

The first type of structure is Server-side personalization (Fig. 1), where the user profile is stored on the server. This structure requires the user to create an account to identify himself. The server creates and updates the user profile either from the user's explicit input (e.g., asking the user to specify his interests) or by implicitly collecting the user's search history (e.g., query and click-through history). The latter approach requires no additional effort from the user and contains a richer description of his interests.

Some search engines, like Google Personalized, adopted this architecture. Most systems with such a structure ask users to provide consent before collecting and using their data. If the user gives his permission, the search system will hold all the personally identifiable data possibly available on the server. Thus, from the user's perspective, this architecture does not have a minimum level of privacy protection.

The lack of protection for such data raises privacy issues. For instance, AOL query logs scandal when the AOL Research released a file on its website containing twenty million search keywords for over 650,000 users, intended for research purposes Wikipedia (2019). In the report, AOL did not identify the users. However, many users' queries contained personally identifiable information attributed by AOL to particular identifiable user accounts. The New York Times located an individual from the released search records by cross-referencing them with phone-book listings. AOL admitted it was a mistake and removed the file, yet others redistributed the file on mirror sites. This example not only raises panic among users but also dampens the data publishers' enthusiasm in offering improved personalized services.

+ Advantages:

- The search engine can use all of its resources (search patterns, document index) in the personalization algorithm.
- Also, the client software generally requires no changes.
- The server's high performance allows us to gain considerable time.

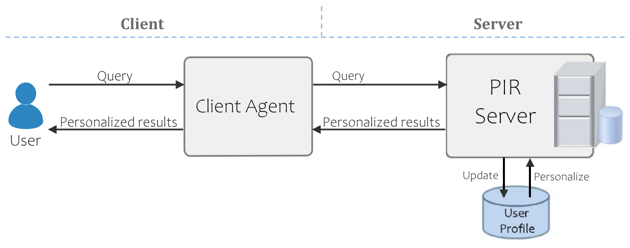

Fig. 1 Server-side personalization structure

– Drawbacks:

- From the user perspective, it does not have a minimum level of privacy protection.
- Most users are afraid of using such systems which can compromise their private data.

To address the drawbacks of this structure, especially the user privacy problems, the client-side structure can be a solution.

3.1.2 Client-side personalization

The second type of structure is Client-side personalization (Fig. 2), stores the user profile on the client-side. The client agent sends queries to the search engine and receives results, the same way as in the ordinary web search scenario. The client agent also performs a query expansion to generate a new personalized query before sending it to the search engine. Furthermore, as in Hawalah and Fasli (2015), the client agent ranks the search results to match user preferences.

+ Advantages:

- Offer a richer user profile: combining the user's search history with his contextual activities (visited web pages) and personal data (emails, bookmarks) and producing a richer user profile.
- Reduce privacy concerns since the user profile is on the client-side.
- Distribute the overhead in computation and storage for personalization among the clients.

– Drawbacks:

- The client usually receives many results from the search engine, which increases the re-ranking process time and reduce its efficiency.
- Besides, the personalization algorithm cannot use the server knowledge (Page-rank score of a result document).

Recent studies, to address the above drawbacks and improve the personalization quality without compromising user privacy, use a client–server collaborative structure.

3.1.3 Client-server collaborative personalization

The client–server collaborative structure (Fig. 3) is a balance between the past structures. The user profile is still on the client-side, but the server also participates in search personalization.

At query time, the client agent extracts a sub-profile from the user profile to send it to the search engine along with the query. The search engine then uses the received context to personalize the results.

Personalization research in this category is minimum, probably due to the relatively complex architecture. In Shou et al. (2014), the contextual information sent to the server is a generalized profile that specifies the user search preferences without exposing the sensitive data in the user profile. The client agent extracts a sub-profile relevant only to a particular query. This sub-profile is a condensed version of the original user profile (generally a few terms or a weight vector from a user search history). Thus, such a structure can reduce the personal data obtained by the search engine.

+ Advantages:

- Offers better privacy protection than a server-side structure because the amount of user data collectible on the server-side is lower than in the case of a server-side personalization.
- It allows the use of a server's internal resources (common search patterns, document index) in the personalization algorithm.
- It presents a more personalized set of results.

– Drawbacks:

- The condensed contextual information is not as efficient as the original user profile.

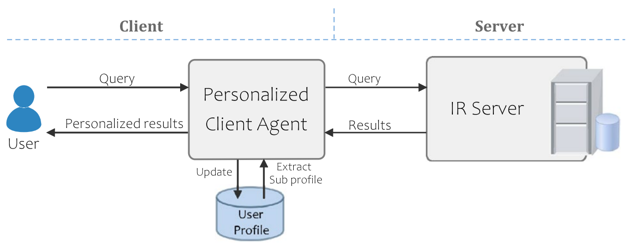

Fig. 2 Client-side personalization structure

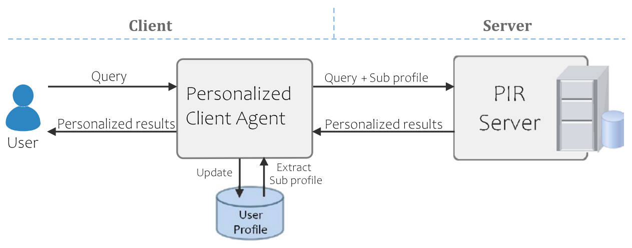

Fig. 3 Client–server collaborative structure

- Presents a privacy risk; though the original user profile is not exposed, the generalized ones can still be collected on the server-side or with an eavesdropping attack (Fig. 4).

3.2 Privacy threats in personalized IR systems

Personalized information retrieval (PIR) systems pose several risks to user privacy. In this section, we discuss potential privacy threats based on the location of the breach on the user's sensitive data.

3.2.1 Threats on the client-side

Exploiting a user's device to store sensitive data (user profile) by personalized IR applications, introduces risks. Some of these threats may lead to serious problems for both users and service providers.

Cross-Site Scripting (XSS) is one of the prevalent vulnerabilities in recent web applications. An attacker can execute scripts within the context of the website under attack. Different types of XSS exist with the same result; allowing for the execution of malicious codes in the browser of the user. This may allow the attacker to access sensitive user data.

Client-side SQL injection (csSQLi) is a new form of well-known SQL injection attacks that have emerged recently due to the introduction of database-support on clients (Google Gears and HTML 5 Web SQL Databases). A popular mechanism that's often used in conjunction with SQL injection is a mechanism called stacked queries. A stacked query allows an attacker to execute his query, fully irrespective of the original query the application executes. Stacked queries are added to an original query through the use of a semicolon. SQL injection with stacked queries can be very powerful, as it allows the attacker to execute arbitrary SQL commands on the database especially on old browsers' versions.

Client-side data corruption or leakage is when the user, or an attacker controlling his device, changes or corrupts the stored data, or simply retrieves sensitive information. Data leakage/corruption can be caused by numerous attacks including malware, XSS, csSQLi, and threats exploiting vulnerabilities in the user's browser or device.

To ensure data security and lower the risk of exploiting client-side data storage vulnerabilities, both users and service providers are advised to implement preventive measures (e.g., encryption, digital signatures, and access control mechanisms). Furthermore, output encoding mechanisms and parameterized queries are used to prevent XSS and csSQLi respectively. Users also need to install anti-malware solutions and keep the device's system and browser updated to avoid new vulnerabilities.

3.2.2 Threats on the server-side

Reports of privacy breaches on the server-side affecting personalized systems dominate the news with increasing frequency. Since most personalized systems including personalized search engines and social media store the collected massive user data on their servers making it the first target for attackers. We can classify server-side privacy threats into two categories: insider threats and outsider threats.

An insider privacy threat comes from within the service-providing organization (malicious or negligent insiders, infiltrators, or even the organization's intentions). The threat may involve data leakage (AOL scandal Wikipedia (2019)), data misuse (Facebook–Cambridge Analytica scandal Tuttle (2018)) when the user data are used for other purposes, data brokerage (sourcing and aggregating data, and reselling the most valuable categories of users to third parties), the theft of confidential or commercially valuable information, etc.

Outsider threats are ever-present, constant, and do pose a danger. Any system connected to the internet is at risk. Historically, the data breaches that make the news are typically carried out by outsiders. In 2016, search engine and email giant organization Yahoo had their system compromised by a data breach, resulting in stealing the information of about 500 million users Trautman and Ormerod (2016). eBay too reported that an attack exposed its entire account list of 145 million users in May 2014 Minkus and Ross (2014). Many more companies on the web have suffered from data breaches (Adobe, Canva, LinkedIn, Zynga, etc.). While these breaches can cost millions of dollars, outsider threats are generally the ones addressed with traditional security measures (Firewall, Passwords, Encryption, etc.) used to prevent potential attacks (Malware Attacks, Phishing, XSS, SQL injections, Password attacks, etc.).

However, securing a server is a difficult and challenging task that cannot be fully accomplished. Failure to implement proper security controls such as patches and updates or secure configurations, changing default accounts, or disabling unnecessary back-end services can compromise data confidentiality and integrity. Moreover, introducing an additional solution to enhance a server's security can increase vulnerability and exposure to further threats. One answer to the problem is to understand server vulnerabilities and start implementing a risk-mitigation approach taking into consideration insider and outsider threats.

3.2.3 Communication channel threats

The internet serves as the electronic chain linking a client to a server. Messages on the internet travel a random path from a source node to a destination node. The message passes through several intermediate nodes on the network before reaching the final destination. It is impossible to guarantee that every computer on the internet, through which messages pass, is secure and non-hostile.

Web applications often use the HTTP protocol for client–server communication, which communicates all information in plain text. Even when they provide transport-layer security through the use of the HTTPS protocol, if they ignore certificate validation errors or revert to plain-text communication after a failure, they can jeopardize security by revealing data or facilitating data tampering.

Compression side-channel attacks such as CRIME and BREACH presented concrete and real-world examples of HTTPS vulnerabilities in 2012 and 2013. Since then, security experts confront new attacks on TLS/SSL every year especially with servers using version 1.2 of the TLS communication protocol, which supports encryption and compression. The combination of encryption and compression algorithms presented security flaws that allowed the attackers to open the content of the encrypted HTTP header and use the authentication token within the cookie to impersonate a user Yang and Gong (2019).

These attacks, among others, can lead to a privacy threat called *eavesdropping* (also known as a sniffing or man-in-the-middle attack) that is a theft of information as it is transmitted over a network by a computer, smartphone, or another connected device. The attacker takes advantage of unsecured network communications to access data as it is being sent or received by its user.

Moreover, the advancement towards future 5G mobile networks is rapid and expected to offer high data speed and low latency, which will eventually result in huge data flows in the communication channels between users and servers. This fact increases the user's concerns about privacy protection in these networks.

3.3 Privacy threat model

Since implementing effective privacy protection models requires understanding the range of potential privacy threats in this field. It is important to study and define a privacy threat model. The threat model described in this section is based on the aforementioned privacy threats in PIR systems and structures. The idea is to model how an attacker could threaten the user's privacy and conduct an attack. As described in Fig 4, we investigate three different threat scenarios:

1. The attacker controls the user's device and have access to the whole user profile.
2. The communication channel is insecure and the attacker can eavesdrop and collect the user's queries with portions of his profile, then using auxiliary information (Online ontology) and social engineering skills he can guess the user's profile.
3. The server is compromised, and the attacker can collect data sent by the user, then guess the original user profile as in the second scenario.

Section 3.2 summarized potential methods that an attacker can use to gain access to the client's device, the server, or the communication channel. Since the user profile is stored on the client-side, encryption is mandatory to secure the user's data and reduce the risk in the first scenario.

During a browsing session in the personalized IR system, a user sends many queries to the server, with short portions of his profile. As we mentioned in the second and third scenarios, an attacker can obtain a significant part of the original profile by collecting the sub-profiles and using the online ontology to figure the rest.

Considering the user profile P, each time a user enters a query q, the system sends a part of P. If the attacker captures each generalized profile G_i, it is possible after a number of queries n to guess a significant portion of the user profile P using the online taxonomy. And, even if the generalized profile G_i contains no private data, the attacker can still obtain the user profile by comparing the G_n to the ontology.

$$\sum_{i=1}^{n} G_i = G_n \rightarrow P \tag{1}$$

Where n is a number of queries (depends on the user activity and time).

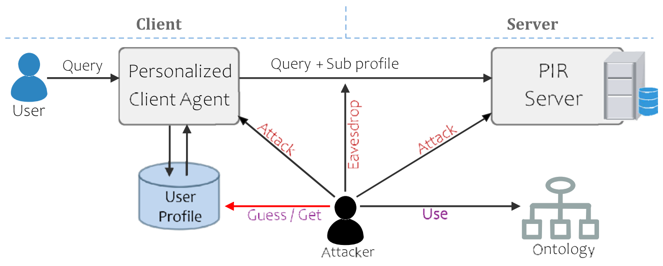

Fig. 4 A privacy threat model in PIR

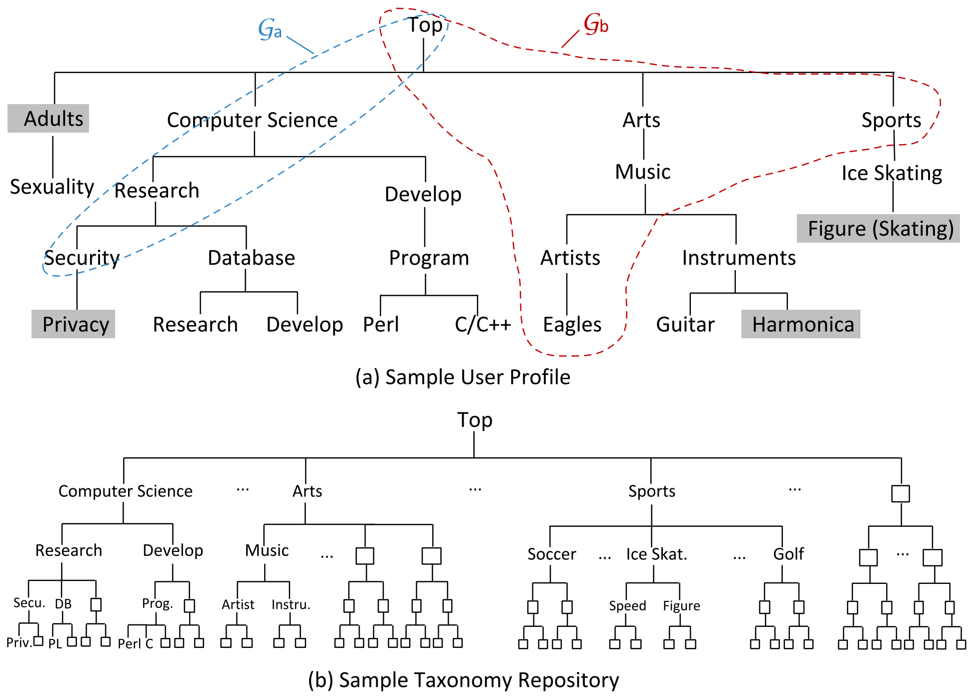

Fig. 5 Ontology-based user profile

To illustrate how an attacker can breach user privacy, Fig. 5 shows an example of a user profile (a) with two generalized profiles (G_a and G_b). The gray concepts in this figure reflect the user's private data. And the generalized profiles contain no sensitive data because the system stops at the parent nodes. However, in G_a for example, the attacker can retrieve the sub-tree of *Security* relying on the taxonomy (b) in the same Fig. 5, where *Security* is the parent of two nodes including a private one (*Privacy*). Therefore, if the probability of touching both branches is equal, the attacker has 50 percent confidence on *Privacy*, leading to high privacy risk.

To reduce the aforementioned risks in our PAPIR model, we combine two techniques discussed in Sect. 2; individual privacy solution with encryption techniques to protect the user profile and limit the possibility of the three scenarios in the threat model.

4 The proposed PAPIR model

This section presents the structure of our privacy-aware model for personalized IR systems. We use the vector space model (VSM) in the user profile construction process and also in the information retrieval. In the VSM, each document, query, or user interest is an n-dimensional vector where n is the number of distinct terms over all the documents and queries. Vectorized data processing helps with developing faster IR systems by making efficient utilization of CPU cache. Figure 6 describes the PAPIR model structure that combines four main components:

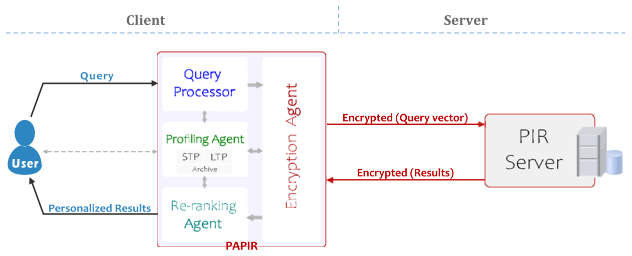

Fig. 6 PAPIR model structure

C1. Profiling agent
C2. Query processor
C3. Encryption agent
C4. Re-ranking agent

PAPIR enhances the Client-Server collaborative structure (Fig. 3) to offer more security and better privacy protection while keeping a good personalization quality. It also addresses the drawbacks of the previous models:

- It uses a rich profile of tree layers for enhanced personalization.
- It eliminates the privacy risk of the generalized user profiles that can still be collected on the server-side or with an eavesdropping attack.
- The client receives a set of personalized and re-ranked results.

- The personalization algorithm on the server-side can use all the knowledge that is only available on the server-side (Page-rank score of a result document).
- Our model also preserves the user's privacy on different levels using encryption.

4.1 Profiling agent

Building user profiles consist of learning from user browsing behaviors. Hence this process is activated after each browsing session as described in our previous work El-Ansari et al. (2020a). In our model, the profiling agent creates and stores the user profile on the client-side (User's device) in the form of a tree of concepts or topics. Each topic of interest is represented with a word vector. For reasons we discuss later in this section, instead of a single user profile, we segregate the user's interests into three parts:

- Short-term profile (STP).
- Long-term profile (LTP).
- The archive.

After each browsing session, the profiling agent prepares and classifies a list of visited concepts to the short-term and long-term layers. A user might show interest in a topic and abandon it after a while. For example, when the football world cup starts, sports fans focus on this event. Once the contest ends, they would likely turn their interest to other sports.

The short-term profile includes the user's recent interests. One way to discover these interests is to define a threshold, and the system classifies all new concepts with weights above it to the short-term profile.

The long-term interests are more stable than the short-term ones. For example, programming languages are stable interests for a user who works as a programmer. Therefore, the short-term profile contains the changing user interests, while the long-term contains the stable ones.

The archive contains interests that are no longer important to the user. Concepts that are gradually losing their weight values (importance) eventually move to the archive. We used separate profiles in our model for the following assets:

- The three profiles will help the system adapt to the user's interest change.
- With the short and long-term profiles, we separate stable interests from the occasional ones.
- Helps preserve user privacy by minimizing the risk of guessing the user's whole profile from the query vector sent to the server.

In most previous works, the same level of privacy protection is afforded for all individuals. However, it is common that the data subjects have quite different expectations regarding the acceptable level of privacy for their data. In our model, once the profiling agent creates the initial profile, the user is allowed to specify his privacy requirements by specifying a sensitivity value for each topic in his profile through a graphical user interface.

4.2 Query processor

This component is a key element in our model, as it is responsible for two main tasks: query preparation and query expansion.

4.2.1 Query preparation

Once a user enters a query q, the first step is to prepare a query vector Q. As we mentioned before, each document, user query, or user interest is an n-dimensional vector where n is the number of distinct terms over all the documents. Normally, users express queries in natural language with a set of words. Each query is tokenized and processed to remove stop-words and stemmed with a Porter stemming algorithm to clear common suffixes Swain and Nayak (2018).

Let $Q_0 = (0_1, 0_2, \dots, 0_n)$ be a query vector with 0 as a weight value for all n terms. For each query term t_i we change the corresponding weight value from 0 to 1 to mark it's presence in the user query. The same operation is performed on the term t_i's synonyms to improve the retrieved results' accuracy. For example, a query q contains 2 key-terms t_5 and t_7, and t_5 is a synonym of the term t_9. The resulting initial query vector would be as follows:

$$Q_i = (0_1, 0_2, 0_3, 0_4, 1_5, 0_6, 1_7, 0_8, 1_9, 0_{10}, \dots, 0_n)$$

4.2.2 Query expansion

Information Retrieval is concerned with the identification of documents in the collection that are relevant to the user's information needs. Queries formed by users are generally short and vague, making it difficult to estimate the exact user need. Information retrieval may improve their effectiveness by using the process of query expansion, which automatically adds new terms to the original query posed by the user.

In our query expansion algorithm, the terms we add to the user query are extracted from the user profile reflecting his interests and needs. Selecting terms (to add) is done by computing the similarity between the query vector and each topic in the user profile (STP) taxonomy to extract related topics of interest. The following algorithm describes the query expansion process:

Algorithm 1: Query Expansion

```
Input: $Q_i$, STP, $\mu$, $\rho$
Output: $Q_e$
$Q_e \leftarrow Q_i$;
$x \leftarrow 1$;
foreach $C$ in $STP$ do
    if $Sensitivity(C) < \rho$ then
        if $CosSim(C, Q) > \mu$ then
            $Q_e \leftarrow Q_e + C$;
            $x \leftarrow x + 1$;
        end
    end
end
$Q_e \leftarrow x^{-1} \times Q_e$;
return $Q_e$;
```

We compute the cosine similarity metric between the query vector Q and each concept's vector C in the STP as follows:

$$CosSim(C, Q) = \frac{C \cdot Q}{\|C\| \times \|Q\|} = \frac{\sum_{i=1}^{n} C_i \times Q_i}{\sqrt{\sum_{i=1}^{n} C_i^2} \times \sqrt{\sum_{i=1}^{n} Q_i^2}} \tag{2}$$

A high cosine value indicates that the query is closely related to the topic of interest in the user profile. To achieve a balance between personalization quality and individual user privacy, we need to take into consideration two important metrics thresholds: Utility μ and Privacy risk ρ.

Utility metric (μ) is used to enhance the personalization quality and increase the system's performance by avoiding unnecessary computations. For some queries, called distinct queries, this whole process of personalization contributes little or even reduces the search quality.

A distinct query is a clear one that needs no personalization and is used by most users to look for the same result. For example, by the question (who is the director of the Titanic movie), users are looking for the same answer (*James Cameron*). The query expansion and randomized perturbation tasks, in this case, are unnecessary. We use a predefined utility threshold (μ) in the query expansion algorithm to avoid such unnecessary computations. To select concepts from the STP that are relevant to the query q; $CosSim(C, Q) > \mu$.

Privacy risk (ρ) metric is based on the user's privacy requirements. It helps decide which topics in the user profile can be added to the query vector. The user specifies a sensitivity value S for each concept in his profile. The privacy risk metric ρ is the average sensitivity value in the user profile. We calculate the similarity between the query vector and a concept C_i's vector, only if C_i's sensitivity is under the threshold: $S_i < \rho$.

To generate the expanded query vector Q_e, we calculate the new weight (nw_i) for each term i as follows:

$$nw_i = \frac{w_i + \sum_{j=1}^{x} W_{ij}}{x + 1} \tag{3}$$

Where :

w_i : is the weight of term i in the initial query vector.
x : is the number of selected concepts.
W_{ij} : is the weight of term i in the concept j's vector.

4.3 Encryption Agent

The encryption agent is responsible for the protection of the user data in its different states; at-rest, in-transit, and in-use.

In our model, the user data that require protection at rest are mainly the user profiles(STP, LTP, and the archive). For data-at-rest encryption, we use the AES algorithm with a 256-bit key. The computational requirements of this approach are low, and it offers strong protection making it the algorithm of choice for governments, financial institutions, and security-conscious enterprises around the world.

The query vector and search results on the other hand require in-transit and in-use protection. Our approach to protecting user data in the last two states is based on the Homomorphic encryption, which enables cryptographically secure computations in untrusted environments.

We use a Fully Homomorphic Encryption (FHE) scheme called CKKS (proposed by Cheon, Kim, Kim, and Song in Cheon et al. (2017)) also know as HEAAN (Homomorphic Encryption for Arithmetic of Approximate Numbers). CKKS allows additions and multiplications on encrypted real or complex numbers, and it is implemented in a widely used library called SEAL (2020) actively developed by Microsoft. The CKKS's plaintext space is $\mathbb{C}^{n/2}$ for some power-of-two integer n. Allowing us to compute on encrypted rational weights in the query and documents' vectors.

Cheon et al. (2017) proposed plaintext encoding/decoding methods to deal with the complex plaintext vector efficiently, which exploits a ring isomorphism:

$$\phi : \mathbb{R}[X]/(X^n + 1) \rightarrow \mathbb{C}^{n/2}$$

Encoding Let $V = (v_1, v_2, \ldots, v_{n/2}) \in \mathbb{C}^{n/2}$ be a plaintext vector and $\Delta > 1$ a scaling factor, the plaintext vector is encoded as a polynomial $m(X) \in R := \mathbb{Z}[X]/(X^n + 1)$ by computing $m(X) = \lfloor \Delta \cdot \phi^{-1}(V) \rceil \in R$ where $\lfloor \cdot \rceil$ denotes the coefficient-wise rounding function.

Decoding Given the message polynomial $m(X) \in R$ and a scaling factor $\Delta > 1$, the message polynomial is decoded to a complex vector $V = (v_1, v_2, \dots, v_{n/2}) \in \mathbb{C}^{n/2}$ by computing $V = \Delta^{-1} \cdot \phi(m(X)) \in \mathbb{C}^{n/2}$.

Where the scaling factor Δ enables controlling the error of encoding/decoding, caused by the rounding process.

SEAL (2020) implements the CKKS scheme algorithms; *KeyGen*, *Encrypt*, *Decrypt*, *Add*, and *Multiply* among others. For a positive integer q, let $R_q := R/qR$ be the quotient ring of R modulo q. Let χ_s, χ_r and χ_e be distributions over R which output polynomials with small coefficients. These distributions, the initial modulus Q, and the ring dimension n are predetermined before the key generation phase as encryption parameters.

KeyGen:

Input:

- Sample a secret polynomial $s \leftarrow \chi_s$

- Sample a (resp. a') uniform randomly from R_Q (resp. R_{PQ}), and $e, e' \leftarrow \chi_e$

Output: secret key, public key, and evaluation key

$$sk \leftarrow (1, s) \in R_Q^2$$

$$pk \leftarrow (b = -a \cdot s + e, a) \in R_Q^2$$

$$evk \leftarrow (b' = -a' \cdot s + e' + P \cdot s^2, a') \in R_{PQ}^2$$

Encrypt:

Input:

- an ephemeral secret polynomial $r \leftarrow \chi_r$

- a message polynomial $m \in R$,

Output: a ciphertext (ct)

$ct \leftarrow (c_0 = r \cdot b + e_0 + m, c_1 = r \cdot a + e_1) \in R_Q^2$

Decrypt:

Input: a ciphertext $ct \in R_q^2$,

Output: a message (m')

$m' \leftarrow \langle ct, sk \rangle (\text{mod } q)$.

Add:

Input: 2 ciphertexts ct_a and $ct_b \in R_q^2$,

Output: a ciphertext (ct_{add})

$ct_{add} \leftarrow ct_a + ct_b \in R_q^2$.

where $Dec(sk, ct_{add}) \approx Dec(sk, ct_a) + Dec(sk, ct_b)$.

Multiply:

Input: 2 ciphertexts $ct_a = (ca_0, ca_1), ct_b = (cb_0, cb_1) \in R_q^2$,

Output: a ciphertext (ct_{mult})

$(d_0, d_1, d_2) = (ca_0cb_0, ca_0cb_1 + ca_1cb_0, ca_1cb_1)(\text{mod } q)$

$ct_{mult} \leftarrow (d_0, d_1) + \lfloor P^{-1} \cdot d_2 \cdot evk \rceil \in R_q^2$.

where $Dec(sk, ct_{mult}) \approx Dec(sk, ct_a) \cdot Dec(sk, ct_b)$.

Other function are also available in SEAL to reduce the ciphertext noise after operations (Add, Multiply) such as Rescaling or Bootstrapping, MultiplyMany and AddMany, which either multiply together or add together several ciphertexts in an optimal way. SEAL also provides functions such as $AddPlain(ct, m_{add})$ and $MultiplyPlain(ct, m_{mult})$ that, given a ciphertext ct encrypting a plaintext polynomial m, and unencrypted plaintext polynomials m_{add}, m_{mult} output encryptions of $m + m_{add}$ and $m \cdot m_{mult}$, respectively. For more details we refer the reader to the documentation available in the SEAL (2020) library link.

We also use a comparison function $Comp(ct_1, ct_2)$ to compare two encrypted numbers. This function is a new approximate representation of the comparison function with a rational function proposed by Cheon et al. (2019b) and improved in Cheon et al. (2019a) with optimal asymptotic complexity and an acceleration method. This improved comparison algorithm only require $O(log(1/\epsilon)) + O(log(\alpha))$ computational complexity to obtain an approximate comparison result of $a, b \in [0, 1]$ satisfying $|a - b| \geq \epsilon$ within $2^{-\alpha}$ error.

We implement this comparison function in a sorting algorithm $Sort(EIF, N)$ executed in the server-side to sort documents by their interest score I'_{score} and return the Top-N relevant ones. The sorting algorithm takes as *input* the encrypted index file *EIF* containing in each row the following encrypted document data: $D_{Title}, D_{url}, D_{vec}, I_{score}$ (Sect. 5.1). The I_{score} reflects the dot product of the document d_i with a given query vector Q. The following algorithm describes the encrypted search process using SEAL and the above-mentioned algorithms and functions:

Algorithm 2: Homomorphic search

```
Client-side
1. Set encryption parameters $Enc_{param}$;
2. $(sk, pk, evk) \leftarrow KeyGen(Enc_{param})$;

Input: Expanded query vector $(Q_e)$
Output: Encrypted query vector $(Q_{enc})$, $pk, evk$
$Q_{enc} \leftarrow Encrypt(pk, Encoding(Q_e))$;
return $Q_{enc}, pk, evk$;
Server-side
Input: $Q_{enc}, Enc_{param}, pk, evk$
Output: $R_{enc}$
$IF \leftarrow Load(Encrypted.Index.File)$;
for $i \leftarrow 1$ to $IF.count()$ do
    for $j \leftarrow 1$ to $n$ do
        $Mult[j] \leftarrow Multiply(Q_{enc}[j], IF[i][D_{vec}][j])$;
    end
    $IF[i][I_{score}] \leftarrow AddMany(Mult)$;
end
$R_{enc} \leftarrow Sort(IF, N)$;
return $R_{enc}$;
Client-side
Input: $R_{enc}$
Output: $R_{pt}$
$R_{pt} \leftarrow Decoding(Decrypt(sk, R_{enc}))$;
return $R_{pt}$;
```

Setting the encryption parameters Enc_{param} along with the *KeyGen* are operations executed in the preparation phase on the client-side. The encryption agent then communicates the Enc_{param}, pk, evk to establish secure communication. On the other side, the server prepares an index file *IF* in the data-set indexing process. Next, the *IF* is encoded and encrypted using the public key *pk*. These operations are executed in the preparation phase. In the encrypted search process, once the user query is prepared and expanded as a vector, the encryption agent encodes and encrypts this query vector using the public key *pk*. The encrypted query vector is then sent to the server. The search program on the server-side loads a copy of the index file *IF* encrypted also with the same public key. In this file, each document in the data set is represented with an encrypted document vector D_{vec}. The search program then computes the dot product of the query vector and each document vector. This encrypted value is then stored in the I_{Score} column for each document and used next by the $Sort(IF, N)$ function to sort documents based on the interest score in the index file and return top N rows. The returned encrypted results are then sent back to the encryption agent for decryption with secret key *sk* and decoding. These results are then processed by the re-ranking agent as described next (Sect. 4.4).

Security proof The CKKS scheme's indistinguishability under chosen-plaintext attack (IND-CPA) is based on the ring learning with errors (RLWE) problem hardness assumption, the ring variant of the very promising lattice-based hard problem Learning with errors (LWE). For more details, the bit security of the CKKS scheme based on known attacks was estimated by Albrecht (2017)'s LWE estimator.

4.4 Re-ranking agent

The PIR server returns a set of N results that are relevant to the query vector as shown in Fig. 6. After the decryption step, the returned set of results R_s enters a re-ranking process. We use both STP and LTP to compute an interest score for each document.

Before computing the new I_{Score} for each result, we use a function $Merge(STP, LTP)$ to merge concepts in STP and LTP. In the end, we sort the results based on the new I_{Score} with the $Sort()$ function.

Algorithm 3: Re-ranking process

```
Input: $R_s$, STP, LTP
Output: $R_r$
$P \leftarrow Merge(STP, LTP)$;
for $i \leftarrow 1$ to $R_s.count()$ do
    foreach $C$ in $P$ do
        $Sim \leftarrow CosSim(C, R_s[i][D_{vec}])$;
        $R_s[i][I_{Score}] \leftarrow R_s[i][I_{Score}] + Sim$;
    end
end
$R_r \leftarrow Sort(R_s)$;
return $R_r$;
```

After presenting the personalized results to the user, the profiling agent collects the chosen documents along with the user query and updates the user profile.

5 Proof of concept implementation

We implement our model as a web application based on the client–server collaborative structure operating with a browser extension installed on the client-side. In the implementation phase, different measures are taken both on the server-side and the client-side.

5.1 Server-side measures

As a local server, we used a computer with an i5 processor (4CPU) having a clock speed of 2.5 GHz and 12GB RAM running a Linux server environment.

Dataset preparation We choose DBpedia (a structured and semantic version of Wikipedia) as a data source for our system; due to its great richness of information. Given that the version we used (DBpedia (2015)) contains descriptions of over 1.9 M concepts within the English version of Wikipedia; including titles, abstracts, and links to the corresponding documents in Wikipedia.

Indexing process The goal of the indexing process is to reduce the time needed to perform a search operation. It consists of five basic steps: tokenization, dropping stop words, lemmatization, term weighting, and index creation. Hence each document in the dataset is processed to generate a document vector $D_{vec} = (w_1, w_2, \dots, w_n)$ where n is the number of distinct terms in the dataset, and w_i is the weight of a term i calculated using the well-known $TF \times IDF$ formula. The result of this process is an index file *IF* containing in each row the following document data:

- D_{Title}: contains the document title.
- D_{url}: is the link to the Wikipedia page.
- D_{vec}: is the vector representation of the document.
- I_{score}: Interest score of the document for a given query.

The encryption program implemented on the server-side is responsible for encrypting each element of the index file using the public key *pk*.

Homomorphically encrypted search is also assured by the encryption program implemented on the server-side based on the SEAL (2020) library. As described in algorithm 2, this program returns the top N documents based on the dot product of the query vector and the documents vectors. We have tested different values of N to test its effect on the results' relevance and computation time.

Scalability A recent project called SparkFHE (2020) integrates Apache Spark with fully homomorphic encryption to enable performing efficient computation on large encrypted datasets. This project is being developed by the SpiRITlab, in collaboration with the Microsoft research team developing SEAL (2020) and working on a similar project called SparkSEAL. Although both projects (SparkFHE (2020) and SparkSEAL) are not officially released, the available version of SparkFHE (2020) can be tested and offer some examples. To test the distributed computation effect on the performance of our PAPIR model, we implemented a 2nd version of the previously mentioned encryption program using the SparkFHE project on our local server. The results are discussed in Sect. 6.

5.2 Client-side measures

The main components of our model are implemented on the client-side (as shown in Fig. 6). We used some features that we have developed in previous works;

Query processor In El-Ansari et al. (2017) we presented a natural language question-answering system over multiple knowledge bases (including DBpedia (2015)). We use a modified version of the question processing feature from this system in the current Query processor to process and transform natural language questions into a query vector. This component includes a query expansion feature (Algorithm 1) to personalize and expand the user query based on his profile.

Profiling agent also uses a profiling method presented in El-Ansari et al. (2020a) to create dynamic user profiles reflecting each user's interests and preferences. This component is developed as a browser extension to track user browsing activities. Along with a graphical user interface (GUI), giving the user access and full control on his profile to provide feedback, change, or update his privacy requirements regarding sensitive topics.

Encryption agent is responsible for protecting user data confidentiality in different states: At-rest, In-transit, and In-use. We implement the user profile protection At-rest using the AES algorithm with a 256-bit key, offering strong protection with lower computational requirements. To protect the user personalized query vector both In-transit and In-use along with the search results, we used homomorphic encryption that showed promising results in one of our previous works Makkaoui et al. (2017). We use the CKKS scheme implemented in the open-source SEAL (2020) library. A recent widely used library supporting efficient FHE and actively developed by the Microsoft research team. The key for a successful implementation with this library is choosing the right encryption parameters (Enc_{param} in Algorithm 2), which is an iterative task. These parameters are:

- *poly_modulus* : a polynomial $x^n + 1$; n a power of 2;
- *coeff_modulus* : an integer modulus q which is constructed as a product of multiple distinct primes;
- *plain_modulus* : an integer modulus t;
- *noise_standard_deviation* : a standard deviation σ;
- *random_generator* : a source of randomness.

In most cases developers only need to set the *poly_modulus* n (1024, 2048, 4096, ..., 32768), *coeff_modulus* (bit-length: 128-bit, 192-bit, and 256-bit), and *plain_modulus* (a positive integer $t \geq 2$ and at most 60 bit-length) parameters.

Both *random_generator* and *noise_standard_deviation* have good default values and are, in most cases, not necessary to specify explicitly. The choice of encryption parameters affects the performance, capabilities, and security of the encryption scheme.

Re-ranking agent As described in algorithm 3, this agent performs two operations; computing the new I_{score} based on the user profile (STP + LTP), and sorting the results based on the new I_{score}. For the first operation, we use the cosine similarity measure in formula 2. The second operation is based on the Quicksort algorithm Hossain et al. (2020) also known as partition-exchange sort with time complexity of $O(n \log n)$.

6 Experiments and results discussion

This section describes the experiments and the corresponding results of evaluating our PAPIR system. In particular, we conduct a Cost/Efficiency analysis to compare the costs of implementing our model to the outputs it can achieve.

6.1 Cost analysis

We analyze the communication and computation cost for a single search operation both on the client-side and server-side. Algorithms 1 and 3 both run separately on the client-side. Algorithm 1 computes the similarity between the query vector with user profile concepts' vectors of the same dimension. The vectors' size is the number of distinct terms in the dataset which can be considered constant, while the number of concepts in the STP is not. Therefore, the time complexity for the first algorithm is $O(n)$ meaning that it is a linear time algorithm.

Algorithm 3, as we mentioned before, is based on the Quicksort algorithm with time complexity of $O(n \log n)$ making it a linearithmic time algorithm.

Estimating the run time for Algorithm 2 is different since it is executed on the client-side and server-side. The computation cost for this algorithm depends on several variables, including the encryption parameters. From our experiments, the computation time is mostly affected by the *poly_modulus* degree.

We investigated the computation time for the main CKKS scheme algorithms that we use in Algorithm 2 (Encrypt,

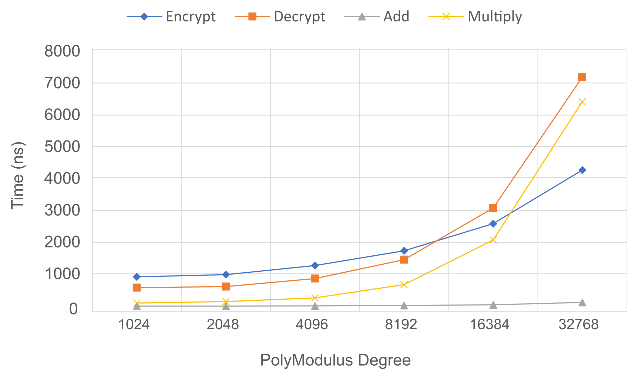

Fig. 7 Effect of the PolyModulus degree on the computation time of the four main algorithms in the CKKS scheme

Decrypt, Add, and Multiply) under different *poly_modulus* degree values. The results are shown in Fig. 7. While using a larger *poly_modulus* degree n decreases performance, it also increases the security level. Therefore, finding a balance value is important for an efficient and secure implementation of the CKKS scheme.

We noticed a significant improvement in the computation time of these algorithms when using SparkFHE (2020), as shown in Fig. 8. This result allowed us to increase the *poly_modulus* degree value in our implementation from 4096 to 8192, reinforcing the security level of our algorithm.

Considering the implementation settings and the hardware limitations, the overall computation cost is reasonable. The average response time for a single user query is 1.36 s, which is acceptable, giving the fact that the implementation was on a local machine with limited hardware capabilities.

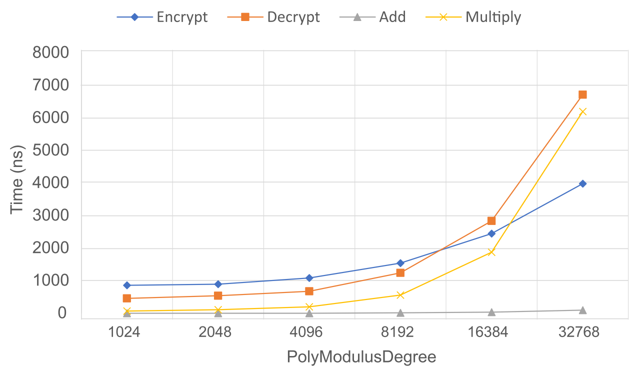

Fig. 8 Effect of the PolyModulus degree on the computation time of the four main algorithms in the CKKS scheme using SparkFHE

6.2 Efficiency analysis

Our model aims to enhance the search accuracy and results' relevance for each user based on his profile while preserving the user's privacy. To evaluate the system's efficiency, we conducted a series of A/B testing experiments with heuristic evaluation. With the help of 35 participants (mainly colleagues and students) who were asked to browse freely for several sessions to construct user-profiles and specify the privacy requirements. Then use different system versions to perform search operations and provide feedback on the search result and the user profile. The collected feedback data are used to analyze the system's accuracy and privacy protection.

6.2.1 Utility evaluation

Another important parameter in algorithm 2 is N representing the Top-N documents retrieved from the server. This number N affects both performance and accuracy since a higher value increases the computation cost and a lower one decreases the accuracy. To evaluate the search accuracy and relevance, we use three well-known metrics in the IR field: Precision, Recall, and F-measure.

Precision reflects the fraction of retrieved documents that are relevant to the user query:

$$Precision = \frac{Relevant\_docs \cap Retrieved\_docs}{Retrieved\_docs}$$

Recall is the fraction of the relevant documents that are successfully retrieved:

$$Recall = \frac{Relevant\_docs \cap Retrieved\_docs}{Relevant\_docs}$$

F-measure is the harmonic mean of precision and recall:

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

In a first experiment, we evaluated the effect of N on the F-measure value as shown in Fig. 9

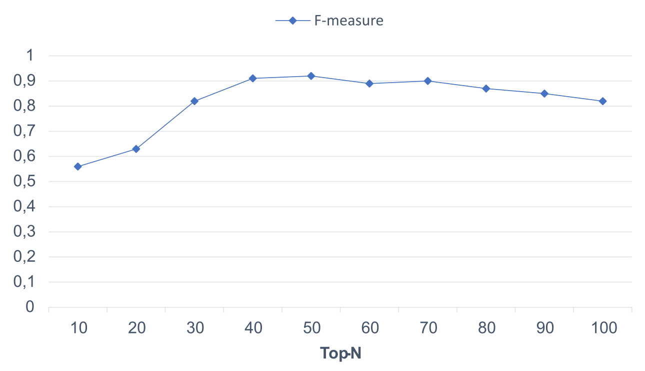

Fig. 9 Effect of N (in Top-N) on the F-measure

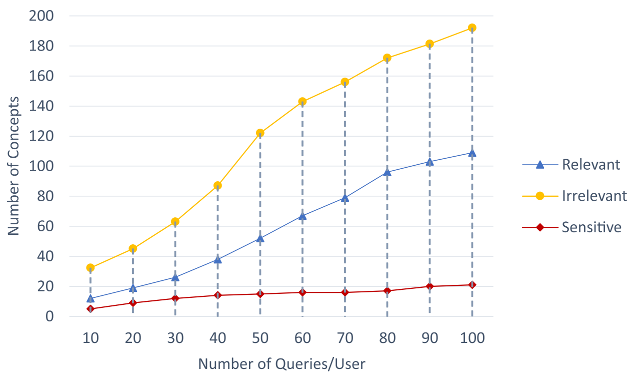

Fig. 10 Privacy loss evaluation results

Table 2 Accuracy A/B testing results

	Precision	Recall	F-measure
A	0.88	0.97	0.92
B	0.73	0.82	0.77

Based on the first experiment results, we conducted a second A/B testing experiment where we proposed two system versions to the subjects.

A: The PAPIR based system with (Top-N=50)

B: a personalization free system where we ignore the algorithm 1.

Based on the subjects' feedback, we computed the average precision, recall, and F-measure value for both versions, as shown in Table 2.

Initial experiments prove that our model significantly improves the accuracy and relevance of the search results, as shown in the above table.

6.2.2 Privacy loss evaluation

To test the privacy protection of our model, we conduct an experiment based on an implausible assumption. Assuming that the homomorphic encryption security is, somehow, broken and an attacker can retrieve the secret key and decrypt the user queries. How much sensitive data will the attacker (or the server) be able to collect?

To answer this question, we disabled the encryption agent and asked the subjects to perform search operations. By mapping the accumulated search queries for every user with the dataset and user profile. The aim is to identify the average number of sensitive, relevant, and irrelevant concepts an attacker can collect after numerous browsing sessions (Fig. 10).

Even without the homomorphic encryption layer, our model reduces the amount of sensitive data that an outsider can collect. Since the personalization process, in our model, is performed in two separate steps Query expansion and Re-ranking. The first one only uses the STP and transforms the user query into an expanded vector. This form also complicates the task for an attacker trying to eavesdrop and collect user data.

Overall, the initial experiments proved the feasibility and efficiency of our model for privacy protection in personalized information retrieval systems.

7 Conclusion and future work

The work presented in this paper focuses on protecting the user's privacy by protecting the sensitive data in his profile. The model we propose counters various privacy threats on different levels. The use of homomorphic encryption as an extra protection layer reinforces the user's privacy protection, which is still one of the major issues in the field of personalized IR systems. Experimental results prove the feasibility and efficiency of our model.

A personalized IR system must ensure privacy protection to earn the user's trust. Otherwise, only a minority of users will use it, to whom the personalized experience is more important than their privacy.

In the future, we plan to improve the implementation of our model by taking into consideration large scale datasets. We are also interested in building a fully HE library adapted for BigData and Cloud technologies.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Albrecht MR (2017) On dual lattice attacks against small-secret lwe and parameter choices in helib and seal. In: Annual international conference on the theory and applications of cryptographic techniques, Springer, pp 103–129

Anjali S, Reeshma K (2015) An efficient privacy preserved personalized web search model using fully homomorphic encryption. Int J Innov Res Comput Commun Eng. https://doi.org/10.15680/ijircce.2015.0307030

Bountouridis D, Harambam J, Makhortykh M, Marrero M, Tintarev N, Hauff C (2019) Siren: A simulation framework for understanding the effects of recommender systems in online news environments. In: Proceedings of the conference on fairness, accountability, and transparency, ACM, New York, NY, USA, FAT* '19, pp 150–159. https://doi.org/10.1145/3287560.3287583,

Chaney AJ, Blei DM, Eliassi-Rad T (2015) A probabilistic model for using social networks in personalized item recommendation. In: Proceedings of the 9th ACM conference on recommender systems, ACM, New York, NY, USA, RecSys '15, pp 43–50. https://doi.org/10.1145/2792838.2800193,

Cheon JH, Kim A, Kim M, Song Y (2017) Homomorphic encryption for arithmetic of approximate numbers. In: International conference on the theory and application of cryptology and information security. Springer, pp 409–437

Cheon JH, Kim D, Kim D (2019a) Efficient homomorphic comparison methods with optimal complexity. IACR Cryptol ePrint Arch p 1234

Cheon JH, Kim D, Kim D, Lee HH, Lee K (2019b) Numerical method for comparison on homomorphically encrypted numbers. In: International conference on the theory and application of cryptology and information security. Springer, pp 415–445

DBpedia (2015) Dbpedia (release 2.0). https://wiki.dbpedia.org/data-set-20

Dennis WL, Erwin A, Galinium M (2016) Data mining approach for user profile generation on advertisement serving. In: 2016 8th international conference on information technology and electrical engineering (ICITEE), pp 1–6. https://doi.org/10.1109/ICITEED.2016.7863269

Desfontaines D, Pejó B (2020) Sok: differential privacies. Proc Priv Enhanc Technol 2:288–313

Eke CI, Norman AA, Shuib L, Nweke HF (2019) A survey of user profiling: state-of-the-art, challenges, and solutions. IEEE Access 7:144907–144924

El-Ansari A, Beni-Hssane A, Saadi M (2017) A multiple ontologies based system for answering natural language questions. In: Rocha Á, Serrhini M, Felgueiras C (eds) Europe and MENA cooperation advances in information and communication technologies. Springer, Cham, pp 177–186

El-Ansari A, Beni-Hssane A, Saadi M (2020a) An improved modeling method for profile-based personalized search. In: Proceedings of the 3rd international conference on networking, information systems & security, pp 1–6. https://doi.org/10.1145/3386723.3387874

El-Ansari A, Beni-Hssane A, Saadi M (2020b) An ontology-based profiling method for accurate web personalization systems. J Theor Appl Inf Technol 98(14):2817–2827

ElShaweesh O, Hussain FK, Lu H, Al-Hassan M, Kharazmi S (2017) Personalized web search based on ontological user profile in transportation domain. In: International conference on neural information processing. Springer, pp 239–248

Erkin Z, Veugen T, Toft T, Lagendijk RL (2012) Generating private recommendations efficiently using homomorphic encryption and data packing. Trans Info For Sec 7(3):1053–1066. https://doi.org/10.1109/TIFS.2012.2190726

Greenstein-Messica A, Rokach L (2018) Personal price aware multi-seller recommender system: evidence from ebay. Knowl Based Syst 150:14–26. https://doi.org/10.1016/j.knosys.2018.02.026

Hawalah A, Fasli M (2015) Dynamic user profiles for web personalisation. Expert Syst Appl 42(5):2547–2569. https://doi.org/10.1016/j.eswa.2014.10.032

Hossain MS, Mondal S, Ali RS, Hasan M (2020) Optimizing complexity of quick sort. International conference on computing science, communication and security. Springer, Berlin, pp 329–339

Liao CL, Lee SJ (2016) A clustering based approach to improving the efficiency of collaborative filtering recommendation. Electron Commerce Res Appl 18:1–9. https://doi.org/10.1016/j.elerap.2016.05.001

Liu A, Wang W, Li Z, Liu G, Li Q, Zhou X, Zhang X (2017a) A privacy-preserving framework for trust-oriented point-of-interest recommendation. IEEE Access. https://doi.org/10.1109/ACCESS.2017.2765317

Liu X, Liu A, Zhang X, Li Z, Liu G, Zhao L, Zhou X (2017b) When differential privacy meets randomized perturbation: a hybrid approach for privacy-preserving recommender system. In: International conference on database systems for advanced applications. Springer, pp 576–591

Lully V, Laublet P, Stankovic M, Radulovic F (2018) Image user profiling with knowledge graph and computer vision. In: European semantic web conference. Springer, pp 100–104

Lv G, Hu C, Chen S (2016) Research on recommender system based on ontology and genetic algorithm. Neurocomputing 187:92–97. https://doi.org/10.1016/j.neucom.2015.09.113

Makkaoui KE, Beni-Hssane A, Ezzati A, El-Ansari A (2017) Fast cloud-rsa scheme for promoting data confidentiality in the cloud computing. Procedia Comput Sci 113:33 –40, https://doi.org/10.1016/j.procs.2017.08.282. (the 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017) / The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2017) / Affiliated Workshops)

Minkus T, Ross KW (2014) I know what you're buying: Privacy breaches on ebay. In: International symposium on privacy enhancing technologies symposium. Springer, pp 164–183

Mohseni M, Maher ML, Grace K, Najjar N, Abbas F, Eltayeby O (2019) Pique: Recommending a personalized sequence of research papers to engage student curiosity. In: Isotani S, Millán E, Ogan A, Hastings P, McLaren B, Luckin R (eds) Artificial intelligence in education. Springer, Cham, pp 201–205

Polatidis N, Georgiadis CK, Pimenidis E, Mouratidis H (2017) Privacy-preserving collaborative recommendations based on random perturbations. Expert Syst Appl 71:18–25

SEAL (2020) Microsoft SEAL (release 3.5). https://github.com/Microsoft/SEAL, microsoft Research, Redmond, WA

Shen Y, Jin H (2016) Epicrec: Towards practical differentially private framework for personalized recommendation. In: Proceedings of the 2016 ACM sigsac conference on computer and communications security, ACM, New York, NY, USA, CCS '16, pp 180–191. https://doi.org/10.1145/2976749.2978316

Shou L, Bai H, Chen K, Chen G (2014) Supporting privacy protection in personalized web search. IEEE Trans Knowl Data Eng 26(2):453–467. https://doi.org/10.1109/TKDE.2012.201

Singhal A, Sinha P, Pant R (2017) Use of deep learning in modern recommendation system: a summary of recent works. CoRR abs/1712.07525, arXiv:1712.07525

Siraj MM, Rahmat NA, Din MM (2019) A survey on privacy preserving data mining approaches and techniques. In: Proceedings of the 2019 8th international conference on software and computer applications, pp 65–69

Smith B, Linden G (2017) Two decades of recommender systems at amazon.com. IEEE Internet Comput 21(3):12–18. https://doi.org/10.1109/MIC.2017.72

SparkFHE (2020) Sparkfhe (release 2.0). https://github.com/SpiRITlab/SparkFHE-Examples

Swain K, Nayak AK (2018) A review on rule-based and hybrid stemming techniques. In: 2018 2nd international conference on data science and business analytics (ICDSBA), IEEE, pp 25–29

Tomashchuk O, Van Landuyt D, Pletea D, Wuyts K, Joosen W (2019) A data utility-driven benchmark for de-identification methods. In: International conference on trust and privacy in digital business. Springer, pp 63–77

Trautman LJ, Ormerod PC (2016) Corporate directors' and officers' cybersecurity standard of care: The yahoo data breach. Am UL Rev 66:1231

Tuttle H (2018) Facebook scandal raises data privacy concerns. Risk Manag 65(5):6–9

Voigt P, Von dem Bussche A (2017) The eu general data protection regulation (gdpr). A practical guide, 1st edn. Springer, Cham

Wang X, Luo T, Li J (2020) An efficient fully homomorphic encryption scheme for private information retrieval in the cloud. Int J Pattern Recogn Artif Intell 34(04):2055008

Wikipedia (2019) Aol search data leak. https://en.wikipedia.org/wiki/AOL_search_data_leak

Wu C, Wu F, An M, Huang J, Huang Y, Xie X (2019) NPA: neural news recommendation with personalized attention. CoRR abs/1907.05559, arXiv:1907.05559,

Wu Z, Li R, Zhou Z, Guo J, Jiang J, Su X (2020) A user sensitive subject protection approach for book search service. J Assoc Inf Sci Technol 71(2):183–195

Yang M, Gong G (2019) Lempel-ziv compression with randomized input-output for anti-compression side-channel attacks under https/tls. In: International symposium on foundations and practice of security. Springer, pp 117–136

Yu P, Ahmad WU, Wang H (2018) Hide-n-seek: An intent-aware privacy protection plugin for personalized web search. In: The 41st international ACM SIGIR conference on research & development in information retrieval, ACM, New York, NY, USA, SIGIR '18, pp 1333–1336. https://doi.org/10.1145/3209978.3210180,

Zhang J, Yang Q, Shen Y, Wang Y, Yang X, Wei B (2020) A differential privacy based probabilistic mechanism for mobility datasets releasing. J Ambient Intell Hum Comput. https://doi.org/10.1007/s12652-020-01746-0

Zhou Y, Li N, Tian Y, An D, Wang L (2020) Public key encryption with keyword search in cloud: a survey. Entropy 22(4):421

Zhu Y, Xiong L, Verdery C (2010) Anonymizing user profiles for personalized web search. In: Proceedings of the 19th international conference on World Wide Web, ACM, New York, NY, USA, WWW '10, pp 1225–1226. https://doi.org/10.1145/1772690.1772886

Zhu T, Li G, Ren Y, Zhou W, Xiong P (2013) Differential privacy for neighborhood-based collaborative filtering. In: 2013 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM 2013), pp 752–759. https://doi.org/10.1109/ASONAM.2013.6785787

Zhu J, He P, Zheng Z, Lyu MR (2015) A privacy-preserving qos prediction framework for web service recommendation. In: Proceedings of the 2015 IEEE international conference on web services, IEEE computer society, Washington, DC, USA, ICWS '15, pp 241–248. https://doi.org/10.1109/ICWS.2015.41,