Privacy preserving cooperative computation for personalized web search applications

Nesrine Kaaniche, Souha Masmoudi, Souha Znina, Maryline Laurent, Levent Demir

Proceedings Paper:
Kaaniche, N. orcid.org/0000-0002-1045-6445, Masmoudi, S., Znina, S. et al. (2 more authors) (2020) Privacy preserving cooperative computation for personalized web search applications. In: Proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing (SAC2020). The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), 30 Mar - 03 Apr 2020, Brno, Czech Republic. ACM Digital Library , pp. 250-258. ISBN 9781450368667

https://doi.org/10.1145/3341105.3373947

Privacy Preserving Cooperative Computation for Personalized Web Search Applications

Nesrine Kaaniche
Department of Computer Science,
University of Sheffield
Sheffield, United Kingdom

Souha Masmoudi
Telecom SudParis, Institut
Polytechnique de Paris
Evry, France

Souha Znina
Telecom SudParis, Institut
Polytechnique de Paris
Evry, France

Maryline Laurent
Member of the Chair Values and
Policies of Personal Information
SAMOVAR, Telecom SudParis,
Institut Polytechnique de Paris
Evry, France

Levent Demir
Qwant Research
Paris, France

ABSTRACT

With the emergence of connected objects and the development of Artificial Intelligence (AI) mechanisms and algorithms, personalized applications are gaining an expanding interest, providing services tailored to each single user needs and expectations. They mainly rely on the massive collection of personal data generated by a large number of applications hosted from different connected devices. In this paper, we present CoWSA, a privacy preserving Cooperative computation framework for personalized Web Search peripheral Applications. The proposed framework is multi-fold. First, it provides the empowerment to end-users to control the disclosed personal data to third parties, while leveraging the trade-off between privacy and utility. Second, as a decentralized solution, CoWSA mitigates single points of failures, while ensuring the security of queries, the anonymity of submitting users, and the incentive of contributing nodes. Third, CoWSA is scalable as it provides acceptable computation and communication costs compared to most closely related schemes.

CCS CONCEPTS

• **Security and privacy** → **Security services**; *Privacy-preserving protocols*; • **Computer systems organization** → Peer-to-peer architectures;

KEYWORDS

privacy, web search engines, personalized services, collaborative computation, decentralized architectures, Interest-based networks

XXX, XXX, XXX

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/xx.xxx/xxx_x

ACM Reference Format:
Nesrine Kaaniche, Souha Masmoudi, Souha Znina, Maryline Laurent, and Levent Demir. 20XX. Privacy Preserving Cooperative Computation for Personalized, Web Search Applications. In *XXX*. ACM, New York, NY, USA, Article X, 9 pages. https://doi.org/xx.xxx/xxx_x

1 INTRODUCTION

Web Search Engines (WSEs) enable internet users to retrieve useful information based on submitted queries and a set of related data contents. In fact, most of existing WSEs record several information, based on submitted queries, known as *query logs*. Generally, these logs include the query's keywords, IP address, browser type and language, query's context information, i.e.; date, time and location of the request, as well as other identifying information such as a reference to cookies included in the user's browser. Relying on these collected information, WSEs are then able to deduce correlations and interest patterns, while relying on specific dedicated algorithms [18]. Indeed, users' profiles are created and regularly updated, in order to classify users based on their interests.

User profiles present a valuable information for WSEs. First, they permit to improve the results for users' queries. For instance, in [14], Agosti et al. stated that users generally provide short queries, meaning that the data query does not contain a sufficient number of useful terms/words that permit to discard irrelevant contents. Consequently, WSEs mainly rely on *users' profiles* to enrich short queries and discard irrelevant results to the requesting user. For instance, let us consider the word *apple*. This term can refer to the fruit or to the high-tech company. Thus, based on the user profile, WSE is able to infer the correct sense of the word used by the querying entity. In [5], Cooper detailed several applications of users' profiles permitting to improve the user experience while receiving personalized results. For example, based on the submitted query coupled with the user's profile, several personalized and relevant advertisements and/or news are shown up. It is worth noting that these systems have become extremely efficient engines for online activities. In [14], Mackenzie et al. claimed that 35 % of what consumers buy on Amazon and 75 % of what they watch on Netflix is attributable to personalized recommendations.

These recommendations and personalized relevant results are usually the result of a massive collection of users' personal data. In fact, in several cases, queries may contain identifying information that permits to precisely identify the user, e.g., name, geolocalization, name and address of the employer, etc. Other queries may contain sensitive information revealing the religion, health status, ethnic origin, sexual orientation, $\cdots$ that are generally considered as highly sensitive to owners.

Contributions — In this paper, we propose to set up a multi-hop computation method between a client and the service provider, in order to ensure users' privacy against both curious service providers and malicious entities. In fact, we introduce CoWSA, a cooperative computation framework that combines two main privacy-enhancing techniques, namely data perturbation and collaborative secure computation. The proposed framework has several advantages. First, it enables end-users to manage and minimize the amount of disclosed personal data to third parties. In fact, users' profiles are stored locally at the user side and are never shared with service providers. Indeed, only some relevant users' attributes are disclosed to service providers, w.r.t. each submitted single query. Second, CoWSA relies on a decentralized architecture. It allows end-users to securely submit their queries, while relying on random intermediate nodes. These entities are encouraged to join a particular query w.r.t. their relevant interests, and to add their interests to the query's associated profile, as they receive in return a panorama of personalized ads and news, w.r.t. the aggregated profile of interest to them. This mitigates single points of failures, while ensuring the anonymity of the quering users. Third, CoWSA is scalable as it provides acceptable computation and communication costs compared to most closely related schemes.

Paper organization — Section 2 details the functional and security requirements and Section 3 presents the related work. Section 4 gives an overview of CoWSA and details the main procedures and algorithms. Section 5 provides a detailed security analysis and Section 6 discusses implementation results, before concluding in Section 7.

2 DESIGN GOALS

The design of CoWSA is motivated by ensuring a good trade-off between the privacy (defined in section 2.2 as covering itself three requirements) and data utility, i.e. the capacity of the service providers (WSEs, third parties) to give satisfaction to users with personalized search answers, ads and news - adapted to the user's interests. This leads to identifying the following functional and security properties.

2.1 Functional Properties

To ensure a good trade-off between the privacy and data utility, two main functional properties are considered:

- **accurate group profile:** the proposed protocol should line-up users w.r.t. their interests, i.e., users are grouped according to similar profiles. The accurate grouping permits WSE to improve clients' and users' experience and provides personalized results. Note that several works propose grouping users, relying on social networks. However, these assumptions may be strong in the sense that users do not necessarily have a social network.
- **flexibility:** as a Peer to Peer (P2P) [9] based solution, it is important to ensure an efficient users' management (i.e., joining and leaving a group). Thus, flexibility also requires a continuous presence of a certain number of on-line devices, such that a set of devices have to be connected to the Internet to ensure relaying other users' queries.

2.2 Security and Privacy Requirements

In order to appropriately define security and privacy requirements, we consider three main adversaries, as follows:

- **curious web search engine:** tries to identify the requesting user, in order to build a *precise* profile.
- **malicious user:** attempts to learn the queries of the initially requesting client, or, as an insider attacker, attempts to inject false inputs to the aggregate profile.
- **malicious client:** attempts to learn the profile of the intermediate users.
- **selfish user:** tries to get benefit from the protocol without participating. Indeed, a selfish user uses the network to only submit his own queries, i.e., without collaborating to forward other users' queries. Note that such nodes' behavior may cause a denial of service.

Both curious WSE and malicious users are mainly considered against privacy and data leakage, while the selfish user adversary is considered against performance concerns. The proposed CoWSA protocol has to ensure the following security and privacy requirements:

- **anonymity:** ensures that neither the requesting user, nor intermediate relaying nodes should be identified by the WSE and/or malicious users.
- **queries' confidentiality:** queries' contents have to remain secret to intermediate relaying nodes.
- **unlinkability:** ensures that neither a curious WSE nor a malicious outsider is able to link a query to its related/corresponding user.

2.3 Performance Requirements

Two main requirements are defined to ensure a good quality of service for CoWSA users:

- **scalability:** the proposed protocol should provide acceptable communication overheads, hence, maintaining an acceptable response time, even with a high number of participating entities.
- **no single point of failure when initializing users' groups:** as a P2P solution, CoWSA manages users' groups with no need for a central node.
- **single point of failure mitigation when accessing to WSE:** in case of selfish users, the client needs to select another group of users for getting the WSE service.

3 RELATED WORK

Several works have been proposed, in the literature, to preserve privacy and minimize the amount of data collected by service providers while providing services tailored to user profiles [3, 4, 11, 13, 19, 20], [16] and [17].

In [3], Castella *et al.* introduced a multi-party protocol, called Useless User Protocol (UUP), based on dynamic groups. The main idea behind this scheme is that a central node creates a group of users after receiving n queries. Afterwards, users can securely exchange their queries, while submitting queries on behalf of each other. As a result, the web search engine cannot create a profile of a particular user. To enhance confidentiality, several measures are performed on queries before their distribution, namely encryptions, re-maskings and permutations. The response is then distributed to all group members. Each user selects only his response and eliminates other search results. Even though the [3] proposal supports several security properties, it is not resistant against selfish users and malicious clients. Later, in [20], Shou *et al.* introduced a client-side privacy-protection system for personalized service. The proposed construction captures users' profiles in a hierarchical taxonomy, w.r.t. several privacy-level assurance. However, the proposed system cannot resist to malicious users, with broader knowledge, such as richer relationship among queries' domains, e.g., exclusiveness, sequentiality, etc.

In [19], Romero *et al.* presented a Peer-to-Peer (P2P) system to ensure privacy-preserving web search services. The proposed scheme classifies users into groups according to their interests, in order to create a group profile. The classification of users into groups is performed by a set of nodes, called super-peers, that mainly refer to distributed brokers. Thus, the proposed system does not support the privacy property of end-users against curious super-peers, as they are able to create users' profiles and collude with both WSEs and third parties, i.e., advertisements and news servers.
In [4], the authors proposed a privacy-preserving web search system, called blind web-search, based on a Fully Homomorphic Encryption (FHE) [10] scheme and Private Information Retrieval (PIR) [12, 22]. The [4] proposed a single keyword queries with exact matches, and a generic construction for multiple-keyword searches [21]. However, based on advanced cryptographic techniques, the proposed framework generates high computation and communication overheads.

Recently, Lai et al. proposed a decentralized search system, called QueenBee [13]. The proposed framework defines a new search engine business model by offering incentives to both content providers and peers that participate in QueenBee's page indexing and ranking operations. Even the proposed framework is promising, it requires building a new infrastructure, thus generating high processing overheads and induces inter-operability concerns w.r.t. existing infrastructures.

In [17], Pires et al. present a decentralized Private Web search solution, called CYCLOSA. The proposed solution involves an SGX-based [1] browser extension in the web browsers. The main idea behind CYCLOSA is to send fake queries to the WSE taking into account the sensitivity of the user query. Indeed, the CYCLOSA browser extension firstly calculates a score k w.r.t. the sensibility of the user query. Then, it selects $k+1$ random peers to which it sends k fake queries and the real one. Afterwards, the peer receiving the query, stores it in a local table of past queries and, as a proxy, forwards it directly to the WSE. The latter sends back the search result to the last node which forwards the answer to the initial sender. Finally, CYCLOSA retrieves the response corresponding to the real query and displays it to the user. Although the CYCLOSA solution deals with several security challenges, namely unlinkability and indistinguishability, it does not ensure the sender's anonymity w.r.t. relay nodes as they know the real source of the query. Furthermore, the CYCLOSA solution degrades the level of personalization of the search results as the WSE response matches the relay node profile and not the query sender's one. Moreover, in CYCLOSA the $k+1$ established connections for each user query generate more communication overhead and bandwidth consumption.

[1] https://software.intel.com/en-us/sgx

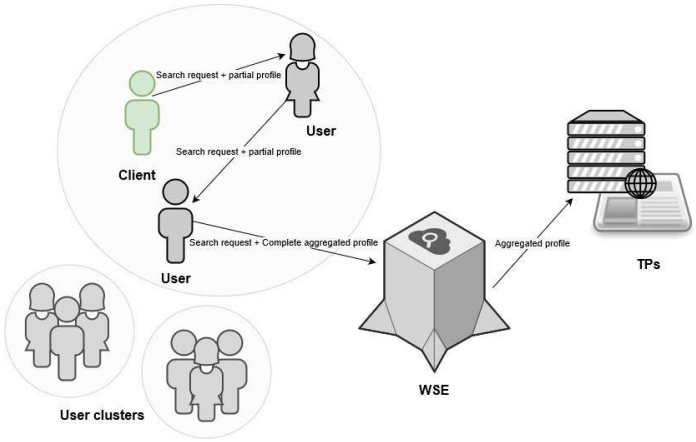

Figure 1: Network Model

4 COWSA DESCRIPTION

In this section, we first present the network model in subsection 4.1. Then, we give an overview of different phases and algorithms in subsection 4.2.

4.1 Network Model

For our network model, we assume that each client obtains a characterizing profile, encompassing several categories. On the other hand, we assume the service provider is a search engine, interacting with an advertising and news agency. Note that ads and news are also categorized and annotated with keywords.

As depicted in Figure 1, CoWSA involves five different entities, defined as follows:

- **Client (C):** submits queries to the web search engine, and protects his own privacy from malicious relay nodes and curious service providers.
- **Users (U):** are intermediate nodes that relay clients' queries, while enclosing their sub-profiles, that are relevant to the client's submitted query.
- **Web Search Engine server (WSE):** is responsible for providing most relevant responses to clients' queries, w.r.t. the

aggregated profile. It also serves as a proxy between users and third parties.

- **Third Parties (TPs):** represent the advertising and news providers. TPs provide ads and news, w.r.t. the aggregated profile transferred by the WSE. Their main objective is to better adapt the recommendations to received profiles.
- **Trusted Authority (TA):** is a trusted entity that is responsible for the system initialization: i.e., key generation algorithms. Note that TA is not involved in queries' processing procedures.

We assume that users are organized into clusters. The organization of users into clusters is out of the scope of this paper. Meanwhile, to fit the *totally* decentralized architecture, we propose to build groups of users, within Publish and Subscribe networks (Pub/Sub) [8, 15]. The Pub/Sub paradigm is a well-known approach for disseminating information between multiple interested entities in a decoupled and asynchronous manner. Message producers submit messages to a broker network which routes those to interested subscribers. In our setting, we suppose that subscribers are users that express their interest in specific queries' domains/categories by issuing subscriptions. Thus, CoWSA mainly relies on the use of multi-broker Pub-Sub networks [6, 7].

4.2 Overview

CoWSA is set upon a decentralized architecture. It extends proxy-based WSE solutions to provide privacy preserving decentralized WSE framework. CoWSA relies on three main procedures: *Sys_Init*, *Query_Submit* and *Query_Resp*.

The *Sys_Init* phase occurs once to setup the system. It involves the creation of the different groups of users, with respect to their shared interests.

The *Query_Submit* corresponds to the process of submitting a query to the WSE server. Indeed, when a client wants to issue a data query to the WSE, he attaches a sub-profile, i.e., the category of his profile that corresponds to the domain of the request. Instead of directly sending the data query to the WSE server, the client chooses a random path, i.e., a set of users expected to relay the data query. At each hop, the intermediate user has to increase the profile of the user by his own sub-profile, including the category of his profile that corresponds to the domain of the request.

The *Query_Resp* phase occurs when the WSE receives the request and the aggregated profile. This latter transfers the profile to the TPs, i.e., the advertising and news agencies. The results are then transmitted via the same path. Thus, each intermediate user receives the news corresponding to his sub-profile, while deciphering only the matching information associated with his sub-profile. The end user, meanwhile, receives the result of his data query by displaying some recommended advertisements.

For ease of presentation, the different notations used in this paper are listed in Table 1.

Table 1: Notations used in this paper

Notation	Description
C	Client
U	User
WSE	Web Search Engine server
TPs	Third Parties
$\mathcal{P}_U$	Users' profiles
C	A category belonging to a user profile $\mathcal{P}_U$
$\mathcal{I}$	An interest belonging to category C
λ	A security parameter
sk	A private key
pk	A public key
q	The query content
pth	The query path
Q	The query vector
$\mathcal{A}$	A random subset of interested users U
enc	An encryption algorithm - $\text{enc}_{s_{C-WSE}}(m)$ for symmetric encryption of m with s_{C-WSE} and $\text{enc}(pk, m)$ for asymmetric encryption of m with public key pk
dec	A decryption algorithm with similar notations as *enc*
s_{C-WSE}	A session key between a client C and a Web Search Engine server WSE
$\mathcal{M}$	A routing pair composed of the predecessor and next relay users
q_e	The encrypted query
$\mathcal{P}_{Q,agg}$	The aggregated profile associated to the query q_e
$\mathcal{S}$	A search result
$\mathcal{R}_{\mathbb{A},\mathcal{P}_{Q,agg}}$	The ads' result associated to the aggregated profile $\mathcal{P}_{Q,agg}$
$\mathcal{R}_{\mathbb{N},\mathcal{P}_{Q,agg}}$	The news' result associated to the aggregated profile $\mathcal{P}_{Q,agg}$
$\mathcal{R}_{\mathbb{Q},\mathcal{P}_{Q,agg}}$	The encrypted search result associated to the query q
$\mathcal{R}$	The query responses' vector

4.3 CoWSA Phases

In this section, we detail the main three CoWSA phases.

4.3.1 Sys_Init Procedure. The *Sys_Init* procedure initializes the whole system, relying on two main algorithms, referred to as: `set_prf` and `key_gen`.

The `set_prf` consists on building users' profiles $\mathcal{P}_U$. Note that several profiles can be assigned to each user. Each profile $\mathcal{P}_U$ is defined as a set of categories C_i (i.e., $i \in \{1, \cdots, l\}$ and l is the maximum number of supported categories). Each category consists on a set of interests $\mathcal{I}_j$ (i.e., $j \in \{1, \cdots, m\}$ and m is the maximum number of supported interests), defined as follows:

$$\mathcal{P}_U = \{C_i\}_{i\in\{1,\cdots,l\}} = \{\mathcal{I}_{i,j}\}_{i\in\{1,\cdots,l\},j\in\{1,\cdots,m\}}$$

As stated above, clients' and end-users' profiles are out-of-scope of this paper. Indeed, CoWSA is built on *Masq* [1], a client-centric profiling tool, introduced in 2019, by Qwant [2], a privacy-preserving WSE. Note that clients and users' profiles can be updated, with respect to each entity's interests and interaction with different applications.

The `key_gen` algorithm is executed by a trusted authority. It takes as input the security parameter λ and outputs a pair of a public and private keys for each system's entity, namely clients, users and WSE server, defined as: (sk_i, pk_i), where $i \in \{U, C, WSE\}$.

4.3.2 Query_Submit Procedure. The *Query_Submit* procedure includes three main algorithms, namely: `set_path`, `set_query` and `relay_query`.

When a client (C) wants to submit a query to WSE, he first defines the query vector Q, defined as follows:

$$Q = [\text{pth}, \{\mathcal{I}_{i,j}\}, q]$$

where `pth` represents the query path, $\{\mathcal{I}_{i,j}\}$ and q is the query content.

The client first runs the `set_path` algorithm. For this purpose, (C) points out a set of categories $\{C_i\}$ that are relevant to the content of the query to be submitted to WSE. Based on his selected interests w.r.t. defined query's categories, the client (C) publishes these categories, relying on the Pub/Sub infrastructure. Thus, brokers perform the matching between query's categories and potential active users' interests [15]. (C) is then notified, by brokers, with

interested active users that may relay his query. Afterwards, (C) selects a set of potential relay users, and sets up the query path as a set of encapsulated identities and public keys, as shown in Algorithm 1.

Algorithm 1 `set_path` algorithm

Input: the set of interested active users $\{(U_i, pk_{U_i})\}$
Output: the encapsulated query path `pth`
`pth` $\leftarrow (U_C, pk_C)$;
pick random a subset of potential interested users to set up the query path $\mathcal{A} \subseteq \{(U_i, pk_{U_i})\}$;
for all $i \in \{1, \cdots, |\mathcal{A}|\}$ **do**
  `pth` $\leftarrow (U_i, \mathsf{enc}(pk_{U_i}, \mathtt{pth}))$, where `enc` is an asymmetric encryption algorithm;
end for
return `pth`

The defined `pth` represents the encapsulation of the public keys of different active users, that have been selected by the client to relay his query. Note that `set_path` algorithm is optional and (C) might choose no-relay users for his query, thus leading to `pth` = (U_C, pk_C).

Once the `pth` is set, the client performs the `set_query` algorithm. To do so, (C) first generates a session key, denoted by s_{C-WSE}. Then, he encrypts the query content q and the session key using the public key of WSE such that $q_e = enc(pk_{WSE}, \{q, s_{C-WSE}\})$ and the query vector is hence updated and defined as:

$$Q = [\mathtt{pth}, \{\mathcal{I}_{i,j}\}, q_e]$$

Note that the use of a session key, shared by only the submitting client and the WSE, enables to keep secret the query' results from relaying nodes. That is, only (C) is able to decrypt the result of the query, received from the WSE.

Once the query vector Q is set, the client executes the `relay_query` algorithm prior to sending Q to the first relay user, which repeats the same actions - `relay_query` and forwarding - upto the last user on the path. As shown in Algorithm 2, each intermediate node relays the query while adding his interests and updating the aggregated profile, w.r.t. his expected rewarding ads and news.

In order to fulfill the personalization requirements and to leverage the trade-off between privacy and utility, CoWSA relies on the *aggregated* profile, built w.r.t. cooperative hop-by-hop process between different interested users. As mentioned above, contrary to existing collaborative solutions, CoWSA defines a rewarding function, such that the relay users receive personalized ads and news, w.r.t. aggregated profile. Thus, these users are encouraged to join a particular query w.r.t. their relevant interests, and to add their interests to the query's associated profile, as they receive in return a panorama of personalized ads and news, w.r.t. the aggregated profile, that interest them.

For ease of presentation, we define the aggregated profile as a clear content. However, to better ensure the security of the submitted query and the privacy of involved parties, CoWSA enables clients and users to share a session group key gk, such that the query's relevant interests $\{\mathcal{I}_{i,j}\}$ are encrypted and updated relying on gk and a symmetric encryption algorithm. We assume that both client and users securely store the group key gk.

Algorithm 2 `relay_query` algorithm, executed by User U_k

Input: the query Q, the pair of public and private keys of U_k (sk_{U_k}, pk_{U_k})
Output: routing pair $\mathcal{M}_k = (prd, nxt)$, updated query Q
set $prd = pk_{U_{k-1}}$; $k-1$ represents the index of the predecessor relay user;
update $\mathcal{M}_k$;
for $C_i = \{\mathcal{I}_1, \cdots, \mathcal{I}_j\}$ **do**
  if $C_i \in \mathcal{P}_{U_k}$ **then**
    $C_i \leftarrow C_i \cup \mathcal{I}_{j+1}$; $\mathcal{I}_{j+1} \in \mathcal{P}_{U_k}$
    update Q;
  end if
end for
`pth` $\leftarrow \mathsf{dec}(sk_{U_k}, \mathtt{pth})$;
pick nxt from `pth` and update $(\mathcal{M}_k, Q)$;
return $(\mathcal{M}_k, Q)$

Arriving at the last relay user, the query vector is complete. At this point, the initial query is considered accompanied by an aggregated profile. The query is then sent to WSE, to perform the *Query_Resp* procedure, detailed hereafter.

4.3.3 Query_Resp Procedure. The *Query_Resp* procedure involves three different algorithms, namely: `proc_query`, `relay_rsp` and `get_rsp`.

The `proc_query` algorithm is executed by WSE server, upon receiving the query Q from the last relaying user U_n, defined as:

$$Q = [\mathtt{pth}, \mathcal{P}_{Q,agg}, q_e]$$

where `pth` = (U_n, pk_{U_n}), $\mathcal{P}_{Q,agg}$ is the aggregated profile associated to the query, and q_e is the encrypted query content along with the session key (generated by the submitting client). The `proc_query` algorithm consists in three sub-tasks, defined as follows:

- First, the WSE server extracts the query content q from q_e. For this purpose, he first decrypts q_e using his private key sk_{WSE} to retrieve the shared session key s_{C-WSE} and query content q. Then, the WSE selects and ranks the search results, denoted by $\mathcal{S}$, that are associated to $\mathcal{P}_{Q,agg}$. Afterwards, he encrypts $\mathcal{S}$, as follows:
$$\mathcal{R}_{Q,\mathcal{P}_{Q,agg}} = enc_{s_{C-WSE}}(\mathcal{S})$$
- Second, the WSE server shares the aggregated profile with TPs, namely advertisement and news servers. Each TP then provides a set of ranked results, associated to the $\mathcal{P}_{Q,agg}$, referred to as $\mathcal{R}_{\mathbb{A},\mathcal{P}_{Q,agg}}$ and $\mathcal{R}_{\mathbb{N},\mathcal{P}_{Q,agg}}$, for ads and news respectively.
- Third, upon receiving TPs' responses, the WSE server creates the query responses' vector $\mathcal{R}$, defined as follows:
$$\mathcal{R} = [\mathcal{R}_{Q,\mathcal{P}_{Q,agg}}, \mathcal{R}_{\mathbb{A},\mathcal{P}_{Q,agg}}, \mathcal{R}_{\mathbb{N},\mathcal{P}_{Q,agg}}]$$

As introduced above, the WSE creates $\mathcal{R}$ and returns it in the opposite direction, i.e., following the opposite sense of the query path,

i.e., $\mathcal{A} = \{U_n, \cdots, U_1, C\}$. Thus, WSE starts the `relay_rsp` algorithm and sends $\mathcal{R}$ to the user U_n, that submitted the Q. Each relay user U_i, then, performs the `relay_rsp` algorithm, while retrieving results from TPs, namely $\mathcal{R}_{\mathbb{A},\mathcal{P}_{Q,agg}}$ and $\mathcal{R}_{\mathbb{N},\mathcal{P}_{Q,agg}}$, w.r.t. the aggregated profile. Then, based on the routing pair $\mathcal{M}_i = (prd, next)$, U_i picks $prd = pk_{U_{i+1}}$ and sends Q to the next hop, until reaching the client (C).

When the client receives the query response, he performs the `get_rsp` algorithm. Indeed, he retrieves personalized results $\mathcal{R}_{\mathbb{A},\mathcal{P}_{Q,agg}}$ and $\mathcal{R}_{\mathbb{N},\mathcal{P}_{Q,agg}}$, associated to $\mathcal{P}_{Q,agg}$ and executes `get_rsp` as detailed in Algorithm 3.

Algorithm 3 `get_rsp` algorithm

1: **Input:** the query response $\mathcal{R}$, the session key s_{C-WSE}
2: **Output:** search results $\mathcal{S}$
3: extract $\mathcal{R}_{Q,\mathcal{P}_{Q,agg}}$ from $\mathcal{R}$;
4: $\mathcal{S} \leftarrow dec_{s_{C-WSE}}(\mathcal{R}_{Q,\mathcal{P}_{Q,agg}})$;
5: **return** $\mathcal{S}$

5 SECURITY ANALYSIS

In this section, we discuss the resistance of the proposed CoWSA framework against curious service providers and malicious entities, introduced in section 2.

5.1 Against Curious WSE

Curious WSE servers aim at precisely identifying the submitting client, in order to build a fine-grained profile, and provide personalized results. This attack is considered against the privacy property, which involves the anonymity and unlinkability requirements.
As CoWSA framework relies on a fully decentralized architecture, users and clients are connected to each other, such that they share the same interests, based on Pub/Sub networks. Thus, two main consequences are identified:

- As each client submits his queries on behalf of other users, by executing `relay_query`, the WSE server cannot distinguish which query belongs to each user/client. Thus, the unlinkability property is preserved.

- Let us assume the worst case, where a client (C) chooses not to run the `relay_query` algorithm, as it is considered as optional. (C) submits his queries directly to the WSE server, during two different sessions α and β, such that:

$$Q^{(\alpha)} = [(U_C, pk_C), \mathcal{P}^{(\alpha)}{}_{Q,agg}, q_e{}^{(\alpha)}]$$

$$Q^{(\beta)} = [(U_C, pk_C), \mathcal{P}^{(\beta)}{}_{Q,agg}, q_e{}^{(\beta)}]$$

  Based on different queries, received from the same client (i.e., pk_C), the WSE server cannot precisely confirm if the query comes from the submitting client or a relaying user. So that, the anonymity requirement is ensured.

5.2 Against Malicious Users

Malicious users aim at learning the queries that are submitted by the client or identifying the originating client.

On one hand, the main information that a malicious user U_i can possess, is the routing pair $\mathcal{M}_i = (prd, nxt)$. The routing pair corresponds to the previous relaying user (prd) and the next one (nxt). Knowing this information, a malicious user cannot discern if the query was generated by the relaying predecessor or if that user was relying it on behalf of another client. In addition, the malicious user cannot deduce the topology of a cluster, i.e., users that are interested to relay a client's query, nor the number of relaying users from the client to the WSE server. Furthermore, the system is dynamic and relies on a Pub/Sub network, where users are able to join a query based on shared interests. This prevents a malicious user from finding the real source of a query.

On the other hand, queries' contents are concatenated with a session key, randomly generated by the client. The result is then encrypted using the public key of WSE server. This prevents malicious users to learn the content of the query as well as the associated results, since search results are enciphered using the generated session key.

5.3 Against Malicious Clients

The main goal of malicious clients consists in learning the profile of relaying users. However, once the query vector Q is set, the submitting client cannot learn users' interests, nor the updated aggregated profile sent to the WSE server.
On the other hand, the client may learn some generic interests about a relaying user, when setting the query path $\mathcal{A}$, based on brokers' notifications. However, this also does not constitute a relevant amount of information that permits to build a precise profile of other users, since users express interest in relaying queries, w.r.t. generic categories.

5.4 Against Selfish Users

Selfish users try to get benefit from the protocol without collaborating with other users, for instance by submitting only their own queries. In order to mitigate against selfish users, CoWSA proposes to reward each relaying user, as he receives *personalized* ads and news from TPs, w.r.t. a closely related profile. Even though selfish users are still able to use the anonymity mask benefit to submit their own queries, the presence of such users will not cause any denial of service, since CoWSA is built upon a decentralized architecture and users join a query w.r.t. shared interests.

6 PERFORMANCES ANALYSIS

Table 2 presents the processing costs, supported by each involved entity, w.r.t. different CoWSA algorithms. It is worth stating that the computation cost depends on the selected encryption algorithms, which is strongly related to the security level. This latter is determined via the brute-force attack which consists in checking all possible keys until the correct one is found (i.e; with a key of length k bits, there are 2^k possible keys). Thus, k denotes the security level in symmetric cryptography. In public-key cryptography, the security level of an algorithm is defined with respect to the hardness of solving a mathematical problem such as the Integer Factorization for the RSA scheme, implemented by the CoWSA prototype and detailed hereafter.

Table 2: Processing Costs of CoWSA algorithms

CoWSA phases	*Sys_Init*		*Query_Submit*			*Query_Response*		
CoWSA Algorithms	set_prf	key_gen	set_path	set_query	relay_query	proc_query	relay_rsp	get_rsp
TA	–	$\mathbb{U}\gamma_K$	–	–	–	–	–	–
C	×	–	×	$\gamma_k + (n+1)\gamma_{E_a}$	γ_P	–	–	γ_{D_s}
WSE	–	–	–	–	–	$\gamma_P + \gamma_{D_a} + \gamma_S + \gamma_{E_s}$	γ_P	–
U	×	–	×	–	$\gamma_{D_a} + \gamma_P$	–	γ_P	–
TP	–	–	–	–	–	γ_S	–	–

Note: – indicates that this entity is not involved in processing the associated algorithm; × indicates that the processing cost is not evaluated ; $\mathbb{U}$ represents the number of all the system's entities; n is the number of relaying users; γ_{E_s} and γ_{D_s} represent the computation cost of the encryption and decryption operations of a symmetric encryption scheme, respectively; γ_{E_a} and γ_{D_a} represent the computation cost of the encryption and decryption operations of an asymmetric encryption scheme, respectively; γ_K and γ_k represent the cost of key generation w.r.t. a symmetric encryption and asymmetric encryption schemes, respectively; γ_S is the cost of search results with respect to a user's profile; γ_P is the cost of processing the query or response vector, by appending interests or extracting users' identities.

From Table 2, we deduce that the enforcement of a collaborative search approach does not impact the WSE's processing overhead. Instead, it requires extra-computation costs from the submitting client, and involved relaying users. This trade-off between performances and privacy is settled by ensuring a better personalization of services. In fact, contrary to most-closely related solutions, CoWSA rewards involved users, as they receive personalized ads/news, while relaying a particular query. It also provides better results' accuracy associated to the aggregated profile, by enforcing the engagement of different interested users.

In this section, we discuss the implementation results of the proposed CoWSA framework. For this purpose, we implemented different algorithms and procedures and built the CoWSA test-bed, to show the feasibility of our proposition, in real-world settings. The CoWSA test-bed is built upon Ubuntu 16.04 machine - with 2 CPUs, 4*GB* memory and 10*GB* hard disk - hosting 6 Docker [2] containers, referring to a client (C), four users (U) and a server (WSE), communicating through network sockets. Based on Python version 2.7, and the associated cryptographic library *Pycrypto* [3], we implemented three python scripts, namely *client.py*, *user.py* and *server.py* - one for each main actor of our model. Note that, for ease of presentation, we consider that each search request involves a single client and three intermediate users.

In the following, we detail the developed algorithms and procedures, w.r.t. *Sys_Init*, *Query_Submit* and *Query_Resp* in subsection 6.1, subsection 6.2 and subsection 6.3 respectively. Then, we provide a detailed discussion for the implementation results.

6.1 *Sys_Init* Implementation

The *Sys_Init* phase consists on the system initialization. As detailed in Section 4.3, this phase includes the users' profiles creation and key generation procedures. As CoWSA is built on *Masq* [1], we set and implemented the key_gen algorithm, such that each client (C) and relaying user (U) possesses the following elements:

- a pair of a public and private keys (sk_i, pk_i), where $i \in \{U, C\}$, generated using the key_gen algorithm of the asymmetric RSA scheme. Note that the public keys, used in CoWSA test-bed, are 1024-bits length.
- a client/user profile extracted from *Masq*, that defines a set of categories and sub-categories, i.e., interests. Table 3 represents an example of profiles of client and users sharing similar interests.
- access to a shared directory that associates the system's entities IP addresses to their public keys. This allows (C) to map each interested user to his public key and identify the WSE server's public key, in order to set the path pth and submit the query Q to the corresponding server.

Table 3: Example of a Client and Users' Profiles considered as Close

Client	cinema (comedy, drama), sport (football, swimming), music (classical, pop)
User 1	cinema (horror, comedy, drama), literature (poetry, novel, theater), music (classical, jazz, pop)
User 2	literature (novel, theater), sport (basketball, hokey), music (rap, reggae, rock)
User 3	cinema (drama, science fiction,), sport (football, basketball, swimming), music (electro, pop)

6.2 *Query_Submit* Implementation

The *Query_Submit* phase relies on three main algorithms, namely set_path, set_query and relay_query, and several sub-tasks, detailed here-after. Recall that, for our CoWSA prototype, we suppose that the query path pth is already set, and it encompasses 3 different users. Thus, we implemented several functions, w.r.t. the set_query and relay_query algorithms, defined as follows:

6.2.1 Initializing the Query Session. In a classical web-search scenario, the query session is initialized by triggering the search engine from a web browser. For the CoWSA prototype, a session is initiated by executing the *client.py* script. Once the query session is initialized, the query's path pth is defined as a list of three IP addresses of the selected relaying users, w.r.t. the *client.py* script.

6.2.2 Defining the Query Vector. The query's vector definition involves several functions and sub-tasks. First, the client executes AES.key_gen algorithm, to derive a $128 - bit$ session key, denoted

[2] https://www.docker.com
[3] https://pypi.org/project/pycrypto/

by s_{C-WSE}. In order to support the encryption of the query's content and the derived session key, we rely on three functions: (i) PadMessage and WrapMessage permit to add the padding and enable the concatenation of several sub-keywords/results, respectively; and (ii) PackHostPort converts the IP address and port formats to $32 - bit$ binary format, thus enabling a multi-layer encryption of the query path. Note that for the CoWSA prototype, we rely on the onion routing concept, while executing the AddAllLayers function.

Second, the client creates the aggregated profile sub-vector, as a set of associated categories and interests that will be then updated by the relaying intermediate users. In addition, in order to mitigate against malicious and curious external adversaries, the CoWSA prototype involves the encryption of the aggregated profile sub-vector, using a group key denoted by gk. The group key is shared between the relaying users. For this purpose, the AES.key_gen algorithm is executed and a $128 - bit$ group key is generated. Then, the aggregated profile sub-vector is randomly filled, w.r.t. the client's categories. To serialize the query vector for further encryption, we rely on the *dumps* function, imported from *pickle* library.

Finally, in order to differentiate the client's query from the search engine's response for the same search session, we added a flag. It takes f for *forward* or b for *backward* to identify the type of the transmitted vector.

6.2.3 Relaying the Query Vector. To submit the generated query vector, the client opens a socket with the next relaying user using the IP address and the port number belonging to the first item, as defined in `pth`. Once the connection is established, the query vector is sent to the next hop, U. An active user U, i.e., waiting for incoming connections, receives the query vector, the IP address and the port of the submitting user/client. Based on these pieces of information, the *user.py* script is executed to both retrieve the corresponding public key from the shared directory and extract the different parts of the encrypted query vector. Once the different elements are extracted, the *user.py* script runs three main functions: (i) UnWrapMessage and UnPadMessage are executed to return the session group key and the decrypted content, and remove the padding characters respectively; and (ii) UnPackHostPort outputs the IP address and port of the next relaying user. Afterwards, the *user.py* script executes the AES.decrypt algorithm, to decipher the categories' vector using the group key, and the *AES* module imported from the *Crypt.cipher* library. Note that the de-serialization of the vector is performed by the `loads` function, imported from *pickle* library, and the update of the aggregated profile sub-vector is provided by a random filling of a sub-set of categories.

6.3 *Query_Resp* Implementation

The third procedure of CoWSA prototype consists on processing the query and relaying the response vector to the submitting client. It involves three algorithms, i.e., `proc_query`, `relay_rsp` and `get_rsp`, detailed hereafter.

6.3.1 Processing the Query. The first algorithm of *Query_Resp*, i.e., `proc_query` is executed by *server.py* upon the reception of Q from the query's ultimate relaying user. For this purpose, the *server.py* script extracts the IP address and related port number from the received query. Then, based on the size of the query vector, two main sub-vectors are extracted: the aggregated profile vector $\mathcal{P}_{Q,agg}$ and the encrypted query q_e. Afterwards, both UnWrapMessage and UnPadMessage are executed to return the session key and the query content, and to remove the padding characters respectively. After retrieving search results, and receiving results from TPs, the *server.py* script starts defining the result vector $\mathcal{R}$. Note that, for our CoWSA prototype, WSE and TPs' results are simulated and presented as follows: *Here is your search result* and *Here are your ads*, respectively. Let us note that *server.py* script encrypts the generated message, corresponding to the search results, using the extracted session key s_{C-WSE} retrieved when receiving the request, and the AES.encrypt algorithm, by enforcing *AES* module imported from the *Crypt.cipher* library. Recall that a flag b is also added to the created response vector, referring to a *backward* message. The obtained result constitutes the WSE response and is sent to the IP address, extracted by the *server.py* script when receiving the query vector.

6.3.2 Relaying the Response Vector. The response vector is transmitted to the client, by passing through the different path's nodes, by performing the `relay_rsp` algorithm. That is, an involved user has to keep the active status, i.e., waiting for incoming connections, to receive the response vector from the predecessor user/WSE server. Thus, once receiving $\mathcal{R}$, the *user.py* script applies two main testing operations. The first test is performed on the first character in order to check the flag value. In the sequel, a new session key is generated. The second test is applied on the IP address of the previous relaying user. Indeed, if $\mathcal{R}$ comes from the WSE server, the PadMessage function is first executed to add the padding characters, then the WrapMessage function is used to concatenate the received message (except the first character), the public key of the next relaying user and the new generated session key. Otherwise, if the message comes from another relaying user, the *user.py* script first performs the UnWrapMessage function, taking as input the received message (except the first character) and his private key, in order to extract the *Here are your ads* message sent by the *server.py* script. Finally, the *user.py* script concatenates the resulting $\mathcal{R}$ with the b flag, before transmitting to the next user.

6.3.3 Retrieving the Search Results. The *client.py* script retrieves the search results associated to his query, by performing the `get_rsp` algorithm. Indeed, to receive the response vector, the client has to keep the active status, i.e., waiting for incoming connections. The received response message should include the b flag. Thus, the *client.py* script executes the UnWrapMessage function to retrieve the encrypted search results. Then, using the shared session key s_{C-WSE}, the message is decrypted, by executing AES.decrypt algorithm. Finally, to retrieve the clear-text, *client.py* script removes the padding characters by using the UnPadMessage function. Thus, the message *Here is your search result*, originally sent by the WSE server, is displayed on the client's screen.

6.4 Implementation Discussion

Several tests are conducted, in order to show the feasibility of CoWSA prototype in real-world settings. The performed tests are

based on several varieties of the same search query, simulating the categories that would be introduced by the aggregated profile. Indeed, the addition of keywords that correspond to categories already used by the WSE server for indexing, returns, in most cases, more relevant results. This remains valid for a limited number of unambiguous categories. However, note that introducing these categories as keywords is not as efficient as processing them as a precise user's profile, typically processed with conventional WSEs.

Therefore, it is worth deducing that the integration of CoWSA prototype provided better *personalized* results, i.e., more relevant to the real preferences of the user. The use of this cooperative approach enhances the customization of the results obtained but does not compromise users' privacy especially that unlike traditional solutions, the CoWSA prototype does not rely on a central entity collecting personal data. In addition, the client's profile is not shared with the WSE and is locally stored at the client side. Thus, queries cannot be associated with the identity of the originator of the query.

The privacy preservation is becoming an arising concern for end-users. However, this protection should not jeopardize the quality of the service as this will ultimately reduce its usability. For this reason, the CoWSA prototype seeks to minimize the impact of using cryptographic blocks on response times. Even though, the introduction of new resource-consuming mechanisms and crypto-blocks inevitably increases the communication overheads, results on the simulated environments showed that response's times are still acceptable.

7 CONCLUSION

Under the personalization and improvement of users' experience concerns, the massive collection of personal data by different web-applications, driven by the establishment of precise users' profiles has raised several privacy challenges.

A novel privacy-preserving cooperative computing framework for web-search applications, CoWSA, is introduced. It presents several security features that make it suitable for several web-search services such as advertisements and news. CoWSA allows clients to securely submit their queries, while relying on random relaying users. These entities are encouraged to join a particular query w.r.t. their relevant interests, and to add their interests to the query's associated profile, as they receive on the way back a panorama of personalized ads and news. This mitigates single points of failures, while ensuring the anonymity of submitting clients.

The security of CoWSA has been discussed and the proposed framework has been proven to be resistant to several data leakages and forgery attacks performed either by a curious WSE, a malicious user and client or a selfish user. Finally, a proof of concept is presented, under the CoWSA test-bed, in order to show the feasibility of the proposed framework.

ACKNOWLEDGMENTS

We are thankful to Qwant for the valuable discussions we had on empowering users to control their personal data, and to achieve higher confidence in digital technologies.

REFERENCES

[1] 2019. Data store for Qwant Masq. https://github.com/QwantResearch/masq-app/.
[2] 2019. Qwant web browser. https://www.qwant.com/?l=en.
[3] Jordi Castellà-Roca, Alexandre Viejo, and Jordi Herrera-Joancomartí. 2009. Preserving user's privacy in web search engines. *Computer Communications* 32, 13-14 (2009), 1541–1551.
[4] Gizem S Çetin, Wei Dai, Yarkin Doröz, William J Martin, and Berk Sunar. 2016. Blind Web Search: How far are we from a privacy preserving search engine? *IACR Cryptology ePrint Archive* 2016 (2016), 801.
[5] Alissa Cooper. 2008. A survey of query log privacy-enhancing techniques from a policy perspective. *ACM Transactions on the Web (TWEB)* 2, 4 (2008), 19.
[6] Shujie Cui, Sana Belguith, Pramodya De Alwis, Muhammad Rizwan Asghar, and Giovanni Russello. 2018. Malicious entities are in vain: Preserving privacy in publish and subscribe systems. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 1624–1627.
[7] Shujie Cui, Sana Belguith, Pramodya De Alwis, Muhammad Rizwan Asghar, and Giovanni Russello. 2019. Collusion defender: preserving subscribers' privacy in publish and subscribe systems. *IEEE Transactions on Dependable and Secure Computing* (2019).
[8] Christian Esposito and Mario Ciampi. 2014. On security in publish/subscribe services: A survey. *IEEE Communications Surveys & Tutorials* 17, 2 (2014), 966-997.
[9] Geoffrey Fox. 2001. Peer-to-peer networks. *Computing in Science & Engineering* 3 (2001), 75–77.
[10] Craig Gentry et al. 2009. Fully homomorphic encryption using ideal lattices.. In *Stoc*, Vol. 9. 169–178.
[11] Nesrine Kaaniche and Maryline Laurent. 2016. Attribute-based signatures for supporting anonymous certification. In *European Symposium on Research in Computer Security*. Springer, 279–300.
[12] Nesrine Kaaniche and Maryline Laurent. 2017. Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms. *Computer Communications* 111 (2017), 120–141.
[13] Ziliang Lai, Chris Liu, Eric Lo, Ben Kao, and Siu-Ming Yiu. 2018. Decentralized Search on Decentralized Web. *arXiv preprint arXiv:1809.00939* (2018).
[14] Ian MacKenzie, Chris Meyer, and Steve Noble. 2013. How retailers can keep up with consumers. *McKinsey & Company* (2013).
[15] Emanuel Onica, Pascal Felber, Hugues Mercier, and Etienne Rivière. 2016. Confidentiality-preserving publish/subscribe: A survey. *ACM computing surveys (CSUR)* 49, 2 (2016), 27.
[16] David Pàmies-Estrems, Nesrine Kaaniche, Maryline Laurent, Jordi Castellà-Roca, and Joaquin Garcia-Alfaro. 2018. Lifelogging protection scheme for internet-based personal assistants. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 431–440.
[17] Rafael Pires, David Goltzsche, Sonia Ben Mokhtar, Sara Bouchenak, Antoine Boutet, Pascal Felber, Rüdiger Kapitza, Marcelo Pasin, and Valerio Schiavoni. 2018. CYCLOSA: Decentralizing Private Web Search Through SGX-Based Browser Extensions. *IEEE 38th International Conference on Distributed Computing Systems* (2018), 467–477.
[18] Sören Preibusch. 2015. The Value of Web Search Privacy. *IEEE Security & Privacy* 13, 5 (2015), 24–32.
[19] Cristina Romero-Tris, Damià Castellà, Alexandre Viejo, Jordi Castellà-Roca, Francesc Solsona, and Josep María Mateo-Sanz. 2015. Design of a P2P network that protects users' privacy in front of Web Search Engines. *Computer Communications* 57 (2015), 37–49.
[20] Lidan Shou, He Bai, Ke Chen, and Gang Chen. 2012. Supporting privacy protection in personalized web search. *IEEE transactions on knowledge and data engineering* 26, 2 (2012), 453–467.
[21] Nazatul Haque Sultan, Nesrine Kaaniche, Maryline Laurent, and Ferdous Ahmed Barbhuiya. 2019. Authorized keyword search over outsourced encrypted data in cloud environment. *IEEE Transactions on Cloud Computing* (2019).
[22] Hua Sun and Syed Ali Jafar. 2018. The capacity of symmetric private information retrieval. *IEEE Transactions on Information Theory* 65, 1 (2018), 322-329.