Sistema P2P de protección de la privacidad en motores de búsqueda basado en perfiles de usuario

Cristina Romero-Tris, Alexandre Viejo, Jordi Castellà-Roca, Youssef Benkaryouh

Universitat Rovira i Virgili, UNESCO Chair in Data Privacy
Departament d'Enginyeria Informàtica i Matemàtiques
Av. Països Catalans 26, E-43007 Tarragona, Spain
Email: {cristina.romero, alexandre.viejo, jordi.castella}@urv.cat
youssef.benkaryouh@estudiants.urv.cat

***Resumen*—Los motores de búsqueda en Internet (como por ejemplo Google, Bing, Yahoo, AOL, etc.) almacenan en sus servidores las consultas efectuadas por los usuarios. Esta información les permite crear perfiles, y así mejorar el servicio ofrecido (resultados personalizados, sugerencias, correcciones, etc.). Sin embargo, estos perfiles pueden también comprometer el derecho a la privacidad de los usuarios. La información que contienen puede servir para identificar a un usuario y así relacionar su identidad con consultas personales y confidenciales. Por esta razón, es necesario aplicar alguna medida de control que proteja la privacidad de los usuarios de motores de búsqueda. Este artículo presenta un entorno P2P diseñado para permitir que los usuarios se agrupen en diferentes categorías en función de su perfil y puedan ejecutar un protocolo para proteger su privacidad.**

Palabras clave*—Motor de búsqueda *(Web Search Engine)*, Perfil de usuario *(User Profile)*, Privacidad *(Privacy)*, Recuperación privada de la información *(Private information retrieval)*, Servicios personalizados *(Customized Services)*, Sistema P2P *(P2P System)

I. Introducción

Los motores de búsqueda en Internet (en inglés, *Web Search Engines* - WSEs) son una herramienta básica para encontrar contenidos y páginas en Internet. Los motores de búsqueda almacenan información de las páginas Web, la indexan y responden a las consultas de los usuarios con una lista de resultados que corresponden a los enlaces a las páginas que contienen la información buscada.

El éxito de un motor de búsqueda respecto a los otros reside en la adecuación de los resultados a los intereses del usuario. Por ejemplo, consideremos un usuario que consulta el término "Mercurio". Si el usuario está interesado en la astronomía, el motor de búsqueda debería mostrar los resultados relacionados con el planeta. Por el contrario, si sus intereses se centran en la química, los resultados deberían corresponder al elemento químico.

Por esta razón, para un motor de búsqueda es necesario conocer los intereses de sus usuarios. Para ello, basándose principalmente en el historial de consultas, los motores de búsqueda crean un "perfil" de usuario que permite personalizar los resultados de acuerdo con los intereses de cada usuario.

Aunque los perfiles sirven para mejorar la calidad del servicio, también pueden ser una amenaza para la privacidad. El historial de consultas puede contener información personal que permita identificar de forma única a un usuario. Por ejemplo, un usuario que busca su nombre completo, su número de seguridad social, su residencia, ocupación, etc. Además las consultas pueden contener información sensible como problemas de salud, la orientación sexual, política, religión, etc.

El almacenamiento en servidores remotos de esta información puede suponer un riesgo, y por lo tanto necesita ser protegida. Sin embargo, el escándalo de AOL [1] demostró que los usuarios no pueden confiar en la protección ofrecida por los motores de búsqueda. En este caso, se publicaron 20 millones de consultas realizadas por 658.000 usuarios. Algunos de ellos fueron identificados, y su identidad pudo ser relacionada con las consultas que habían realizado, quedando así su privacidad expuesta.

Los motivos anteriores ponen de manifiesto un compromiso entre la privacidad y la calidad del servicio recibido. Por un lado, los usuarios necesitan tomar alguna medida para proteger su privacidad. Por otro lado, los usuarios pueden ser reticentes a utilizar un sistema que les proporcione privacidad pero cuya respuesta sea lenta, o los resultados que les interesen no estén en las primera páginas. Si un usuario ofusca su perfil de manera significativa, obtendrá una buena protección de su privacidad, pero también recibirá un peor servicio, y viceversa.

En este artículo se propone un método que ofrece un compromiso aceptable entre privacidad, utilidad del perfil y tiempo de respuesta.

I-A. Estado del arte

La protección de la privacidad frente a los motores de búsqueda es un tema tratado con anterioridad en distintos trabajos. Una forma de clasificar estos trabajos es de acuerdo con el número de usuarios que participan en el protocolo: existen protocolos *single-party* y *multi-party*. Los protocolos *single-party* permiten que un usuario proteja su privacidad de forma individual. Los protocolos *multi-party* requieren un grupo de usuarios que colabore para proteger su privacidad.

Los protocolos *single-party* se basan en generar consultas falsas [2] o en modificar las consultas que son enviadas a los motores de búsqueda [3]. Sin embargo, algunas propuestas

ISBN: 978-84-9717-323-0

(p.e. [4], [5]) muestran que las consultas generadas por una máquina no tienen las mismas características sintácticas y semánticas que las consultas humanas. De acuerdo con los trabajos de [4] y [5], es posible detectar consultas automáticas con una probabilidad de error muy baja (alrededor de $0{,}02\,\%$).

Otra opción dentro de los protocolos *single-party* es usar un canal anónimo como por ejemplo *Tor* [6]. No obstante, en este caso el motor de búsqueda no es capaz de generar un perfil del usuario, ofreciendo una peor calidad del servicio (ver [7] para más detalles).

Por otro lado, los protocolos *multi-party* no están afectados por los errores de detección de consultas falsas, y generalmente son más rápidos que los esquemas basados en canales anónimos. En estos protocolos, los usuarios ofuscan su perfil con consultas falsas generadas por otros usuarios. La obtención de estas consultas falsas se realiza mediante la creación de grupos de usuarios, que pueden ser dinámicos [7], o estáticos [8], [9].

Los protocolos con grupos dinámicos utilizan un servidor para agrupar a los usuarios. Este nodo puede ser un cuello de botella, o puede suponer problemas de seguridad, o sufrir ataques de denegación de servicio.

En los protocolos con grupos estáticos, no es necesario un servidor que cree dinámicamente los grupos. Sin embargo, el problema en este caso es que los grupos contienen los mismos usuarios en cada ejecución del protocolo, existiendo la posibilidad de que usuarios deshonestos creen perfiles de los otros miembros del grupo.

I-B. Contribución y organización del artículo

Considerando las ventajas y desventajas de las soluciones con grupos dinámicos y estáticos, en este trabajo se propone una solución híbrida. El sistema presentado en este artículo clasifica a los usuarios de los motores de búsqueda en grupos, donde cada grupo representa una temática o categoría. Un usuario puede pertenecer a varios grupos (si su perfil, generado a partir de su historial de consultas, contiene diversas categorías) durante una "sesión", es decir, un período de tiempo variable en el que el usuario envía consultas al motor de búsqueda de forma regular. Cuando esta sesión finaliza, el usuario se desconecta del sistema, cambiando la topología de los grupos y haciéndola así dinámica.

La principal contribución respecto a trabajos anteriores es la creación y mantenimiento de perfiles dinámicos. El beneficio que obtiene el usuario del sistema es que enviará siempre consultas cuya temática esté ligada a sus intereses. Ésto protege su privacidad ya que ofusca su perfil (no contiene sus consultas reales), pero mantiene sus intereses (permitiendo la obtención de resultados personalizados).

En la Sección I-A se describen brevemente las principales propuestas para proteger la privacidad de los usuarios de los motores de búsqueda. En la Sección II se presenta la arquitectura propuesta y en la Sección III el protocolo de privacidad. La Sección IV presenta los resultados de la simulación del sistemas propuesto. Finalmente, las conclusiones se describen en la Sección V.

II. Arquitectura propuesta

El objetivo principal de nuestra propuesta es construir una arquitectura P2P que permita que los usuarios se agrupen en diferentes categorías en función de sus perfiles. En esta sección se explica en detalle la arquitectura que compone la red Peer-to-Peer de manera que los usuarios puedan agruparse según su perfil en redes no estructuradas. Para ello, a continuación se definen la estructura del perfil considerado y la topología de la red. En esta sección se explica en detalle la arquitectura que compone la red Peer-to-Peer.

II-A. El vector Perfil

Para definir el vector que caracteriza el perfil del usuario, el sistema utiliza las categorías definidas en el Open Directory project (ODP) [10]. ODP es una clasificación ampliamente aceptada de las diferentes categorías que existen de páginas Web. Esta clasificación se realiza a varios niveles. Por ejemplo, la consulta "Michael Jordan" estaría clasificada en ODP bajo las categorías (de más general a más específica) "deportes : baloncesto: profesional: NBA: jugadores". Por cuestiones de simplicidad, llamaremos L al nivel de la categoría, así la categoría "deportes" se encuentra a nivel $L = 1$, la categoría "profesional" a nivel $L = 2$, etc. Asimismo, llamaremos C_L al conjunto de categorías que se encuentran en un nivel, y s al número de categorías que contiene el nivel. Por ejemplo, en el nivel $L = 1$ hay $s = 16$ categorías, $C_1 = \{c_1, \ldots, c_{16}\}$=$\{$arte, negocios, ordenadores, juegos, salud, hogar, infancia y adolescencia, noticias, ocio, referencia, regional, ciencia, compras, sociedad, deportes, mundo$\}$.

Definimos también $Q_i = \{q_1, \ldots, q_k\}$ como el conjunto de consultas generadas por el usuario i, y $E_i^L = \{c_1, \ldots, c_k\}$ como el conjunto de categorías de nivel L a las que pertenecen las consultas de Q_i. Para obtener la categoría a la que pertenece cada consulta, se utiliza el método de análisis textual, como el propuesto en [11]. Este método aplica, para cada consulta, un análisis morfosintáctico y semántico, cuyo resultado final es la categoría ODP a la que pertenece la consulta.

Finalmente, definimos el perfil P_i^L de un usuario i en el nivel L como un vector, donde cada posición corresponde al peso w de una categoría del nivel L, $P_i^L = \{w_{c_1}, \ldots, w_{c_s}\}$. Cada peso w es el número de apariciones de esa categoría en E_i^L.

Por ejemplo, consideremos que nuestro sistema está fijado para $L = 1$. Consideremos también un usuario a que ha generado $k = 3$ consultas $Q_a = \{q_1, q_2, q_3\}$, con categorías E_a^1 =arte, deporte, arte. El perfil resultante sería $P_i^L = \{2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0\}$.

Este perfil es dinámico, es decir, a medida que el usuario vaya enviando más consultas, mayor información contendrá su perfil. Por ejemplo, si el usuario a envía otra query q_4 de arte, y otra q_5 de negocios, su perfil será $P_i^L = \{3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0\}$.

II-B. Estructura de la red

Los usuarios se conectan entre ellos usando una red P2P. Esta red está formada por varios *clusters*, donde cada *cluster*

representa cada una de las categorías de nivel L. Esto quiere decir que, por ejemplo, para $L = 1$ tendremos una red P2P formada por 16 *clusters*.

Dentro de cada *cluster*, existe un nodo llamado *superpeer* que mantiene la lista de direcciones de los nodos que pertenecen a ese *cluster*. Dependiendo de su perfil, un nodo puede estar conectado a uno o varios *clusters*.

Para acceder a la red, se utiliza un servidor *bootstrap*, encargado de mantener y proporcionar la lista de *superpeers* (direcciones IP) a los nuevos nodos.

A continuación se describe con mayor detalle las formas que existen de cambiar la estructura de la red.

II-B1. Nueva conexión de un nodo: Como muchas redes P2P, la red se construye gradualmente a medida que nuevos nodos se conectan al sistema. Los pasos para insertar un nuevo nodo en el sistema son:

1. Consideramos un nuevo usuario i que quiere enviar una consulta q_i. En primer lugar, el sistema extrae la categoría c_j a la que pertenece q_i y crea el perfil inicial del usuario $P_i^L = \{0_{c_1}, \ldots, 1_{c_j}, \ldots, 0_{c_s}\}$
2. El usuario i se conecta al *bootstrap*, que le envía la lista de *superpeers* existentes en el sistema. En este punto, dos situaciones pueden ocurrir:
   - *No hay superpeer para la categoría* c_j. En este caso, i se convierte en el nuevo *superpeer* de esa categoría, y envía un mensaje al servidor *bootstrap*. El *bootstrap* guardará la IP de i en la lista de *superpeers*, asociada con c_j.
   - *Existe un superpeer para la categoría* c_j. El usuario i manda una petición al *superpeer* de c_j para ser incluido en su *cluster*. El *superpeer* le envía la lista de direcciones IP de los nodos que ya pertenecen a c_j, y después incluye la IP de i en esta lista.
     El usuario i utiliza la lista de nodos que le envía el *superpeer* para ejecutar el protocolo de privacidad explicado en la Sección III.

II-B2. Modificación del perfil de un nodo: Hay dos maneras en las que el perfil de un usuario puede ser modificado:

1. *Realiza una nueva consulta de una categoría de su perfil con peso* $w_c > 0$. En este caso, se trata de una categoría ya existente en su perfil, y simplemente se incrementa en una unidad el peso w_c de la categoría.
2. *Realiza una nueva consulta de una categoría de su perfil cuyo peso* $w_c = 0$. En este caso, el perfil cambia el peso de la categoría a $w_c = 1$, y el usuario debe conectarse a un nuevo *cluster*. Utilizando la lista de *superpeers* proporcionada por el *bootstrap*, el usuario vuelve a ejecutar una de las opciones del paso 2.

II-B3. Salida de usuario: Cuando un usuario se desconecta del sistema, la estructura de la red cambia, y los siguientes cambios deben realizarse:

- Si el usuario que se desconecta es un *superpeer*, asumimos que antes de desconectarse envía dos mensajes. El primer mensaje contiene la lista de direcciones IP de los nodos del *cluster*. Este mensaje es enviado a uno de los nodos escogido al azar en el *cluster*, que se convertirá en el nuevo *superpeer*. El segundo mensaje se envía al servidor *bootstrap*, indicando la dirección IP del nuevo *superpeer*.
- Si el usuario que se desconecta es un nodo normal de un *cluster*, simplemente enviará una notificación a su *superpeer* para ser borrado de la lista de nodos del *cluster*. Los nodos que ya estaban en el *cluster* en el momento de la desconexión, borrarán al usuario de sus listas en el caso de que intenten conectarse con él y reciban un mensaje de error indicando que ya no está disponible.

III. Protocolo de Privacidad

La sección anterior explica la estructura de la red P2P. En esta sección, asumimos que cada usuario ya está conectado a la red, y posee la lista de direcciones IP de los nodos de uno o varios *clusters* a los que pertenece su perfil.

En este punto, utiliza un protocolo de privacidad para proteger las consultas que envía al motor de búsqueda. La idea general de este protocolo es que cuando el usuario genera una consulta, no la envía directamente al motor de búsqueda, sino que la envía a otro nodo de la red. Este nodo es escogido al azar dentro del *cluster* al cual corresponde la categoría de la consulta. A su vez, el nodo que recibe la consulta puede aceptarla o rechazarla. Si la rechaza, el usuario buscará a otro nodo del *cluster* que se la acepte. El nodo que acepta la consulta tiene dos posibilidades: enviarla al motor de búsqueda, o reenviarla a otro nodo del *cluster*. El nodo elegirá una opción u otra en base a su historial de consultas enviadas. Si decide reenviar la consulta a otro nodo, éste elegirá de nuevo entre las dos posibilidades, hasta que finalmente un nodo envíe la consulta al motor de búsqueda. Finalmente, los resultados para la consulta se devuelven al nodo que la generó por el camino inverso que siguió la consulta.

A continuación se describen las fases del protocolo anterior con más detalle:

III-A. Inicializaciones

Asumimos que un nodo i de la red tiene dos perfiles:

- *Un perfil real* P_i^L, construido a partir de las categorías de las consultas que genera, tal como se explica en la Sección II-A.
- *Un perfil ofuscado* Φ_i^L, con la misma estructura que el perfil real, pero construido a partir las categorías de las consultas que *envía* al motor de búsqueda.

El objetivo del protocolo de privacidad es hacer que el perfil ofuscado se parezca lo máximo posible al perfil real, ofuscando la información que el motor de búsqueda almacena, pero manteniendo su utilidad. Para controlar el nivell de distorsión del perfil se ha definido el parámetro de ofuscación z. Este parámetro representa la diferencia máxima que puede existir entre el peso de una categoría en el perfil real $w_{c_j^P}$ y el peso de esa misma categoría en el perfil ofuscado $w_{c_j^\Phi}$. Es decir, se tiene que cumplir la siguiente condición: $w_{c_j^P} < w_{c_j^\Phi} + z$.

Además, definimos otro parámetro ξ: la probabilidad de rechazo. Este parámetro indica la probabilidad (entre 0 y 1) que tiene un nodo de rechazar una consulta que recibe de otro usuario.

III-B. Envío de la consulta

Esta fase del protocolo se ejecuta cada vez que un usuario i genera una consulta q_i.

1. i calcula la categoría c_j a la que pertenece q_i.
2. i incrementa una unidad el peso $w_{c_j^P}$ de su perfil real.
3. i escoge al azar un nodo v del *cluster* que corresponde a c_j.
4. i y v ejecutan el siguiente protocolo de compromiso de bit para saber si v debería aceptar o rechazar la consulta. Sin este procedimiento, un usuario egoísta podría rechazar siempre las peticiones recibidas.
   a) i escoge al azar un valor X de longitud suficiente y aplica una función resumen criptográfica segura computacionalmente $H()$, obteniendo $x = H(X)$.
   b) i envía x al nodo v.
   c) v escoge al azar otro valor Y, y aplica la misma función de hash obteniendo $y = H(Y)$.
   d) v envía y al nodo i.
   e) i envía X al nodo v, para que verifique si $x == H(x)$.
   f) v envía Y al nodo i, para que verifique si $y == H(Y)$.
   g) Si alguna de las verificaciones falla, los nodos se desconectan y el usuario i recomienza esta fase del protocolo eligiendo otro v'.
   h) Si las verificaciones se llevan a cabo con éxito, i y v calculan un hash que concatena X e Y: $H(X||Y)$. El resultado de este hash se usa como entrada para un generador pseudo-aleatorio que genera un valor (λ) de manera aleatoria entre 0 y 1. Si $\lambda \geq \xi$, v acepta la consulta q_i. Si $\lambda < \xi$, v rechaza la consulta y el usuario i recomienza esta fase del protocolo eligiendo otro v'.
5. Suponiendo que el nodo que finalmente acepta la consulta es v, éste calcula $w_{c_j^{\Phi_v}} + z$, y lo compara con $w_{c_j^{P_v}}$.
   - Si $w_{c_j^P} \leq w_{c_j^\Phi} + z$, v envía la consulta al motor de búsqueda. Además, incrementa una unidad el peso $w_{c_j^\Phi}$ de su perfil ofuscado.
   - Si $w_{c_j^P} > w_{c_j^\Phi} + z$, v escoge al azar otro nodo r del *cluster* que corresponde a c_j. En este punto, el protocolo vuelve a ejecutarse entre v y r a partir del paso 4.

III-C. Recepción de los resultados

La fase del protocolo anterior se ejecuta hasta que uno de los nodos finalmente envía la consulta al motor de búsqueda. Este nodo es responsable de comenzar el reenvío de los resultados que recibe del motor de búsqueda. Ninguno de los nodos conoce el camino completo que siguió la consulta, simplemente reciben los resultados del nodo al que se la reenviaron, y los pasan al nodo del que la recibieron. Así, finalmente, la consulta llega hasta el usuario que la generó.

IV. Análisis del sistema

Con el objetivo de analizar el sistema propuesto, se ha implementado una aplicación que simula las consultas que enviaría cada usuario al WSE si utilizara este sistema. Además, esta aplicación analiza diversas estadísticas de las simulación, como el número de saltos que realiza una consulta antes de recibir la respuesta. Comparando el perfil original y el perfil obtenido en la simulación, se puede verificar si las categorías se mantienen en el perfil, permitiendo personalizar los resultados de futuras consultas. Otro de los parámetros analizados es el número de saltos de cada consulta, que permite estimar el tiempo de espera. Es decir, si el sistema requiere un gran número de saltos, esto supondría una gran espera y por lo tanto los usuarios serían reticentes a su utilización.

Los datos empleados en la simulación son los proporcionados por AOL. Primero, se han ordenado todas las consultas por la fecha en que fueron realizadas. El simulador recibe cada consulta en su tic correspondiente, es decir, para cada segundo. Los perfiles ofuscados de los usuarios de AOL serían los que hubieran obtenido si hubieran utilizado nuestro sistema.

Dado el gran número de datos de AOL únicamente se ha considerado un día entero para hacer la simulación. El trabajo futuro sería ampliar la simulación a más días.

El simulador y el protocolo incluyen diferentes parámetros que afectan a su comportamiento:

- *Ofuscación* (z): nivel máximo permitido de distorsión del perfil.
- *Probabilidad de rechazo* (ξ): un nodo rechazaría la consulta en función de una probabilidad fijada ξ. Esta probabilidad se obtiene mediante el protocolo de compromiso de bit.
- *Nivel de las categorías (L)*: se ha fijado el primer nivel $L = 1$ al hacer las simulaciones.

El nivel de las categorías es el mismo para todas las simulaciones realizadas, pero el nivel de ofuscación y la probabilidad de rechazo cambian para ofrecer un análisis más completo. Más concretamente, el sistema se ha simulado para tres valores de ofuscación ($z = 0, 5, 10$), y cuatro probabilidades de rechazo ($\xi = 0.25, 0.5, 0.75, 1$).

A continuación, se muestra el promedio de resultados para tres usuarios diferentes en cada una de las configuraciones. La Tabla I muestra el número de saltos promedio que ha necesitado una consulta para ser enviada al motor de búsqueda y recibir los resultados, es decir, ida y vuelta: el número de nodos que han reenviado la consulta y los resultados. Los resultados muestran que el número de saltos se encuentra sobre de los 4 saltos por consulta. También se observa que un nivel de ofuscación de $z = 0$ obliga a realizar un mayor número de saltos hasta encontrar un usuario que quiera enviar la consulta al motor de búsqueda. Esto ocurre porque los usuarios tienen menor flexibilidad para alejarse de su perfil real, restringiendo el número de consultas falsas que pueden enviar. Además, la tabla muestra que, para $z = 0$, la probabilidad de rechazo

ξ afecta ligeramente a los resultados. Cuanto mayor es la probabilidad de rechazo, mayor es el número de saltos. Por otro lado, los resultados no parecen estar afectados por esta probabilidad de rechazo cuando $z = 5$ o $z = 10$. Esto se debe a que los nodos tienen flexibilidad suficiente para alejarse de su perfil real, y pueden aceptar más consultas falsas.

Tabla I
PROMEDIO DE SALTOS DE CADA CONSULTA PARA DISTINTOS VALORES DE z Y ξ

Ofusc. (z) \ Prob. rech. (ξ)	**0,25**	**0,5**	**0,75**	**1**
0	4,32	4,36	4,45	4,62
5	3,48	3,46	3,51	3,42
10	3,43	3,41	3,45	3,41

El siguiente punto a analizar es la relación entre el número de consultas propias que el usuario ha enviado al motor de búsqueda, y el número total de consultas generadas. Por ejemplo, consideremos un usuario que genera 171 consultas, y ejecuta el protocolo propuesto para distribuirlas entre los usuarios de la red. Por distintas razones (e.g., la consulta le ha sido reenviada en un ciclo, o ningún usuario la ha aceptado), al final tiene que enviar él mismo 23 de esas consultas. Entonces, decimos que el motor de búsqueda conoce 23 de las 171 consultas reales generadas por el usuario, y por lo tanto, el nivel de conocimiento que tiene de su perfil es de $23/171 = 0{,}135$. La Tabla II muestra estos valores promedios para cada una de las configuraciones. En estos resultados podemos observar resultados muy similares para todas las configuraciones, independientemente de z y de ξ. Esto quiere decir que estos parámetros afectan al número de saltos que realizará la consulta, pero no a las razones por las que un usuario envía su propia consulta.

Tabla II
PROMEDIO DEL PORCENTAJE DEL PERFIL CONOCIDO POR EL MOTOR DE BÚSQUEDA PARA DISTINTOS VALORES DE z Y ξ

Ofusc. (z) \ Prob. rech. (ξ)	**0,25**	**0,5**	**0,75**	**1**
0	0,30	0,30	0,29	0,24
5	0,28	0,29	0,27	0,30
10	0,29	0,29	0,28	0,30

Por último, las simulaciones realizadas analizan la distancia entre el perfil real del usuario y su perfil ofuscado. Para ello, dado el perfil real $P_i^L = \{w_{c_1^P}, w_{c_2^P}, \ldots, w_{c_s^P}\}$, y el perfil ofuscado $\Phi_i^L = \{w_{c_1^\Phi}, w_{c_2^\Phi}, \ldots, w_{c_s^\Phi}\}$, la distancia entre ambos $d(P, \Phi)$ se calcula aplicando la siguiente fórmula:

$$d(P,\Phi) = \sqrt{(w_{c_1^P} - w_{c_1^\Phi})^2 + (w_{c_2^P} - w_{c_2^\Phi})^2 + \ldots + (w_{c_s^P} - w_{c_s^\Phi})^2}$$

La Tabla III muestra el promedio de distancias entre el perfil real y el ofuscado de cada usuario para cada configuración. De estos resultados podemos extraer que, cuanta más flexibilidad hay en el límite de ofuscación z, mayor será la distancia entre el perfil real del usuario y su perfil ofuscado. Por el contrario, la probabilidad de rechazo ξ no parece afectar de forma regular a los resultados. Esto se debe a que el hecho de aceptar más o menos consultas, no está relacionado con la ofuscación, y por lo tanto con la distanci entre perfiles.

Tabla III
PROMEDIO DE SALTOS DE CADA CONSULTA PARA DISTINTOS VALORES DE z Y ξ

Ofusc. (z) \ Prob. rech. (ξ)	**0,25**	**0,5**	**0,75**	**1**
0	14,46	8,23	9,89	19,53
5	22,34	19,30	24,78	21,21
10	22,94	23,02	27,79	21,54

V. CONCLUSIONES Y TRABAJO FUTURO

Los motores de búsqueda en Internet crean *perfiles* de sus usuarios para personalizar los resultados y ofrecer un mejor servicio. La información almacenada en el perfil puede suponer una amenaza para la privacidad del usuario. Por esta razón, en este trabajo se ha propuesto una arquitectura P2P que permite que los usuarios se agrupen según sus perfiles, permitiendo al mismo tiempo conservar un perfil ofuscado (o sintético) muy similar a su perfil real. Una vez en grupos, los usuarios ejecutan un protocolo con el que envían sus consultas de manera que el motor de búsqueda obtiene un perfil que no se corresponde al del usuario en lo que se refiere a las consultas pero si a las categorías. Esto sirve para favorecer una personalización de los resultados por parte del motor de búsqueda, mientras se protege la privacidad del usuario.

Para analizar el protocolo propuesto, varias simulaciones se han llevado a cabo con distintas configuraciones de parámetros. De estas simulaciones, se han mostrado los resultados que corresponden al número de saltos, al porcentaje del perfil que posee el motor de búsqueda, y a la distancia entre el perfil obtenido tras ejecutar el protocolo (ofuscado) y el perfil real del usuario. Los resultados muestran que (1) un nivel máximo de ofuscación nulo aumenta el número de saltos de las consultas, (2) los porcentajes de perfil que obtiene el motor de búsqueda no están afectados por el nivel de ofuscación ni la probabilidad de rechazo, y (3) la distancia entre el perfil real y el ofuscado está estrechamente relacionada con el nivel máximo de ofuscación z.

Como trabajo futuro, se prevee realizar más simulaciones ajustando otros parámetros del sistema, como el número mínimo de conexiones que debe tener un usuario o el procentaje de usuarios egoístas presentes en la red. Además, el trabajo futuro también incluye una propuesta para controlar el origen de consultas "ilegales", i.e., que un usuario pueda demostrar que no generó una consulta, sino que estaba enviándola para beneficio de otro usuario de la red.

REFERENCIAS

[1] M. Barbaro and T. Zeller, "A face is exposed for aol searcher no. 4417749," New York Times, August 2005.
[2] "TrackMeNot," http://mrl.nyu.edu/dhowe/trackmenot, 2013.
[3] J. Domingo-Ferrer, A. Solanas, and J. Castella-Roca, "h(k)-private information retrieval from privacy-uncooperative queryable databases," *Journal of Online Information Review*, vol. 33, no. 4, pp. 1468–1527, 2009.

[4] R. Chow and P. Golle, "Faking contextual data for fun, profit, and privacy," in *Proceedings of the 8th ACM workshop on Privacy in the electronic society – WPES'09*, 2009, pp. 105–108.
[5] S. T. Peddinti and N. Saxena, "On the privacy of web search based on query obfuscation: a case study of trackmenot," in *Proceedings of the 10th international conference on Privacy enhancing technologies – PETS'10*, 2010, pp. 19–37.
[6] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *Proceedings of the 13th conference on USENIX Security Symposium*, 2004, pp. 21–31.
[7] J. Castella-Roca, A. Viejo, and J. Herrera-Joancomarti, "Preserving user's privacy in web search engines," *Computer Communications*, vol. 32, no. 13–14, pp. 1541–1551, 2009.
[8] A. Viejo and J. Castella-Roca, "Using social networks to distort users' profiles generated by web search engines," *Computer Networks*, vol. 54, no. 9, pp. 1343–1357, 2010.
[9] A. Erola, J. Castella-Roca, A. Viejo, and J. M. Mateo-Sanz, "Exploiting social networks to provide privacy in personalized web search," *Journal of Systems and Software*, vol. 84, no. 9, pp. 1734–1745, 2011.
[10] ODP, "Open Directory Project," http://www.dmoz.org/, 2013.
[11] D. Sánchez, J. Castellà-Roca, and A. Viejo, "Knowledge-based scheme to create privacy-preserving but semantically-related queries for web search engines," *Inf. Sci.*, vol. 218, pp. 17–30, Jan. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2012.06.025