Autenticación implícita eficiente y con privacidad

Alberto Blanco-Justicia
Departament d'Enginyeria Informàtica i Matemàtiques
Universitat Rovira i Virgili
Email: alberto.blanco@urv.cat

Josep Domingo-Ferrer
Departament d'Enginyeria Informàtica i Matemàtiques
Universitat Rovira i Virgili
Email: josep.domingo@urv.cat

***Resumen*—La autenticación implícita es la capacidad de un sistema de autenticar a sus usuarios no sólo a partir de su identidad y credenciales, sino a partir de cómo estos interactúan con sus dispositivos. Esta interacción se puede inferir recogiendo y combinando datos de varios sensores del dispositivo, como los patrones de escritura o las antenas de telecomunicaciones visibles. En trabajos recientes se introdujo el concepto de autenticación implícita con privacidad, cuyo objetivo es proteger los perfiles de comportamiento de los usuarios frente a los proveedores de servicios que deben autenticarlos. Los mecanismos propuestos hasta la fecha, sin embargo, implican un alto coste computacional o están muy limitados en los tipos de características del usuario que pueden tratar. En este trabajo proponemos un mecanismo basado en filtros de Bloom, que hacen posible la autenticación implícita con privacidad y sin pérdidas sustanciales de precisión manteniendo un bajo coste computacional.**

***Palabras clave*—Autenticación implícita, Filtros de Bloom, Privacidad.**

I. Introducción

La autenticación implícita es la capacidad de un sistema informático de autenticar individuos basándose en la manera en que éstos interactúan con sus dispositivos, es decir, en su comportamiento. En nuestro contexto, el comportamiento de un usuario se puede determinar analizando diferentes características, como los patrones de escritura, el historial y la configuración del navegador, las direcciones IP, la localización, las antenas visibles, etc. Podemos entender la autenticación implícita como un complemento de los mecanismos de autenticación habituales, basados en identificadores y credenciales.

La autenticación implícita está ganando importancia junto al mercado de los teléfonos inteligentes. Típicamente, los teclados de los smartphones son pequeños e incómodos, lo que dificulta la elección y escritura de contraseñas suficientemente fuertes. Esta situación, junto al hecho de que los usuarios suelen elegir contraseñas débiles, tal y como indican numerosos estudios y artículos de prensa, hacen que el uso de mecanismos complementarios de autenticación sea prácticamente obligatorio. Entre estos mecanismos, los más utilizados son la autenticación biométrica y la autenticación en dos pasos con contraseñas de un solo uso. La autenticación biométrica requiere sensores especializados en el dispositivo del usuario y que la entidad autenticadora (el servicio) tenga acceso a los patrones biométricos del usuario. Por otra parte, la autenticación en dos pasos tiene un problema intrínseco: el canal secundario (email, SMS, aplicación móvil, etc.) utilizado para enviar los códigos de acceso temporales suele ser accesible desde el mismo dispositivo que accede al servicio (típicamente un smartphone conectado a Internet), por lo que si se compromete la seguridad de uno de los canales, no es sencillo asegurar que el canal secundario seguirá siendo seguro.

La autenticación implícita también tiene sus problemas. El más destacable es la posible exposición de los datos privados de los usuarios, que deben ser analizados para obtener los patrones de comportamiento de referencia, respecto a los cuales serán autenticados los usuarios. Los mecanismos de autenticación implícita presentados en [12], [4] tratan de proteger los datos privados de los usuarios respecto al proveedor de servicios. En estas propuestas, los perfiles de referencia de los usuarios se almacenan cifrados en el proveedor, y las muestras obtenidas por el dispositivo se comparan con estos perfiles de referencia. Estas propuestas resuelven el problema de privacidad de la autenticación implícita, pero su coste computacional, tanto para el proveedor como para el dispositivo del usuario, dificulta su implementación práctica.

I-A. Contribuciones

En este trabajo, proponemos un mecanismo de autenticación implícita, computacionalmente eficiente, que preserva la privacidad de los usuarios, ya que sólo se revelan huellas digitales de sus perfiles, en lugar de los perfiles originales. Mediante filtros de Bloom, construimos estas huellas digitales de los perfiles y, aprovechando las propiedades de estos filtros, calculamos las distancias entre los perfiles de referencia almacenados por los proveedores y las muestras obtenidas por el dispositivo del usuario. La privacidad de los usuarios se preserva asumiendo que las funciones de hash criptográficas son seguras.

Nuestro mecanismo produce huellas digitales de los conjuntos de características de los usuarios. Estas huellas son compactas y pueden integrarse fácilmente en protocolos de autenticación existentes, por ejemplo como cabeceras en los paquetes HTTP.

En la sección II presentamos una visión general de temas relacionados, incluyendo la autenticación implícita, los sistemas de autenticación implícita que protegen la privacidad de los usuarios, y los mecanismos de minería de datos con privacidad. La sección III describe los filtros de Bloom en detalle. La sección IV enumera los tipos de características que podemos encontrarnos en los perfiles de usuario, y cómo podemos comparar esos perfiles dependiendo del tipo de

datos. En la sección V presentamos nuestro mecanismo de autenticación implícita. La sección VI presenta los resultados experimentales, específicamente la pérdida de precisión que implica el uso de nuestro sistema. Finalmente, presentamos las conclusiones y las posibles líneas de investigación futuras en la sección VII.

II. Trabajos relacionados

II-A. Autenticación implícita

En un sistema de autenticación implícita, el proveedor de servicios autentica a sus usuarios comprobando si su comportamiento es suficientemente similar a su comportamiento típico registrado. En [8] se presentan evidencias empíricas de que las características recogidas del historial del dispositivo del usuario (su historial de navegación, sus localizaciones típicas, sus patrones de escritura, etc.) son suficientes para autenticarlo.

Aquí, como en muchos otros casos, nos encontramos con el dilema de la seguridad en contra de la privacidad. Almacenar el perfil del usuario en su dispositivo conlleva el riesgo de que un intruso pueda suplantar al usuario legítimo. Por otro lado, almacenar los perfiles en los servidores externos es más seguro para el usuario, pero va en contra de su privacidad.

II-B. Autenticación implícita con preservación de privacidad

En el sistema de autenticación implícita propuesto en [12] el dispositivo del usuario cifra tanto su perfil inicial, enviado durante el registro al servicio, como las muestras enviadas en cada intento de autenticación. El problema de seguridad que mencionábamos en la sección anterior queda resuelto porque el perfil nunca se guarda en el dispositivo del usuario. De igual modo, el problema de privacidad también queda resuelto, ya que el perfil que se envía al proveedor está cifrado.

El proceso de autenticación implica calcular un índice de similitud para cada característica individual, que es luego agregado y comparado con un umbral predefinido. Este índice de similitud se calcula en los servidores del proveedor, garantizando que: i) el dispositivo sólo aprende la desviación típica media de las características; ii) el servidor sólo aprende cómo están ordenadas las características respecto al perfil de referencia.

Este protocolo requiere el uso de dos esquemas de cifrado distintos y está restringido a características numéricas, lo que supone una desventaja, ya que algunas características, como el historial de navegación, no son características numéricas.

El sistema propuesto en [4] intenta resolver las limitaciones del protocolo anterior: utiliza un único esquema de cifrado, no revela el orden de los valores de las muestras ni su desviación típica y puede lidiar con características no numéricas. Para ello, se basa en el protocolo de cálculo de distancias entre funciones de preferencia de [1].

En este protocolo, el dispositivo del usuario cifra el perfil del usuario utilizando el criptosistema de Paillier y lo envía al proveedor, junto a diversos valores auxiliares. Para autenticarse, el usuario envía una muestra cifrada de su actividad al proveedor, y ambos ejecutan un protocolo similar al de [1] para calcular la distancia entre ambos perfiles. Nótese que el resultado de la operación nunca se descifra, sino que se comprueba utilizando los valores auxiliares.

Aunque [4] resuelve algunos de los problemas de [12], lo hace a cambio de un mayor coste computacional. De hecho, calcular los valores auxiliares comporta invertir matrices, lo que en la práctica acaba limitando el tamaño de los perfiles por cuestiones de complejidad.

III. Filtros de Bloom

Los filtros de Bloom [2] son una estructura de datos probabilística y eficiente en cuanto a espacio utilizada para codificar conjuntos de datos, y que permite consultas de membresía. Estas consultas están sujetas a cierta probabilidad de falsos positivos, proporcional al número de elementos codificados. Un filtro de Bloom consiste en una lista de bits $B = b_0, \ldots, b_{m-1}$ de longitud m, con todos los bits inicializados a 0, acompañada por $k \ll m$ funciones de hash con rangos en $[0, \ldots, m-1]$. Para insertar un elemento e, se obtienen k índices calculando el valor de hash de e para cada una de las funciones de hash. A continuación, se asigna el valor de 1 a los bits correspondientes $b_{h_0(e)}, \ldots, b_{h_{k-1}(e)}$ (ver Figura 1). Del mismo modo, una consulta de membresía para el elemento e se resuelve comprobando si $b_{h_0(e)} = \cdots = b_{h_{k-1}(e)} = 1$.

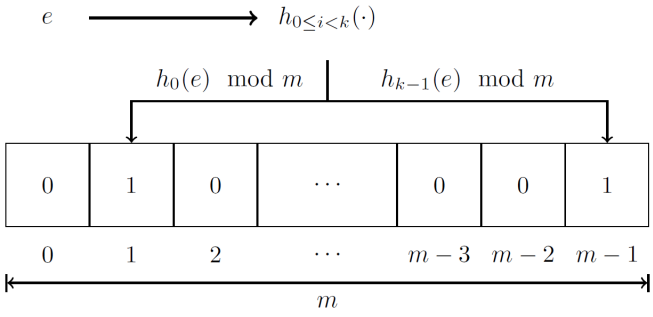

Figura 1. El elemento e se inserta en un filtro de Bloom de tamaño m

La probabilidad de falsos positivos (es decir, de que una consulta de membresía para un elemento que no está en el conjunto devuelva que sí lo está) viene dada por la ecuación

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k, \tag{1}$$

que es la probabilidad de que los índices obtenidos calculando el hash con las k funciones de hash correspondan a bits del filtro de Bloom que sean ya iguales a 1, bien a causa de otro único elemento (muy poco probable) o bien a causa una combinación de otros elementos insertados.

En [14] se da la función que aproxima el número de elementos insertados en un filtro de Bloom (es decir, el cardinal del conjunto codificado):

$$|S| \approx -\frac{m}{k} \ln\left(1 - \frac{H(B_S)}{m}\right). \tag{2}$$

donde S es el conjunto codificado, B_S es la lista a de bits resultante de codificar S, $H(\cdot)$ es el peso de Hamming de una lista de bits (el número de bits iguales a 1), m es la longitud del filtro de Bloom y k es el número de funciones de hash.

III-A. Filtros de Bloom en minería de datos con privacidad

Los esquemas de emparejamiento privado (*private matching schemes*) son una primitiva recurrente en los protocolos de computación multiparte en minería de datos privados. Estos esquemas son útiles para calcular *equi-joins* entre tablas de diferentes entidades, que no quieren intercambiar todos sus datos, sino sólo aquellos de los que ya comparten cierta información. También han sido utilizados en búsquedas de documentos por palabras claves, emparejamiento en redes sociales y, como en nuestro caso, para comparar los perfiles de comportamiento de diferentes usuarios.

De entre las diferentes implementaciones de esquemas de emparejamiento privado, el trabajo de [6] es posiblemente el más relevante. En este trabajo, los autores proponen distintos protocolos para calcular la intersección de conjuntos privados, así como el cardinal de la intersección de conjuntos, bajo distintos modelos de adversario. En estos protocolos, los conjuntos se codifican como polinomios y se cifran bajo Paillier, cuyas propiedades homomórficas permiten la evaluación de polinomios cifrados. En [1], los autores se basan en estos protocolos para permitir el cálculo de distancias entre funciones privadas, incluyendo funciones numéricas. Por desgracia, estos protocolos requieren el cálculo de operaciones criptográficas poco eficientes y no están, por lo tanto, indicadas para grandes conjuntos de datos.

En [5], los autores proponen un esquema de emparejamiento privado basado en una variante de los filtros de Bloom y en protocolos de transferencia inconsciente, que resuelve en gran medida los problemas de escalabilidad de propuestas anteriores. En este protocolo, los conjuntos de datos son codificados utilizando una variante de los filtros de Bloom. Las dos partes entonces ejecutan un protocolo de transferencia inconsciente para calcular un nuevo filtro de Bloom que codifica la intersección de los conjuntos originales. En [11], los autores mejoran aún más la escabilidad del protocolo. El mecanismo de [9] utiliza las propiedades homomórficas del criptosistema de Goldwasser-Micali para ejecutar consultas de membresía en filtros de Bloom cifrados bit a bit.

III-B. Unión e intersección de conjuntos

La unión e intersección de conjuntos codificados se puede calcular fácilmente aplicando las operaciones $\vee$ (O) y $\wedge$ (Y) bit a bit, respectivamente. Tomando los conjuntos A y B, y sus respectivas codificaciones B_A y B_B con los mismos parámetros (mismo tamaño m y mismas k funciones de hash), el filtro de Bloom

$$B_{A\cap B} = b_{A,0} \wedge b_{B,0}, \ldots, b_{A,m-1} \wedge b_{B,m-1}$$

representa la codificación de la intersección $A\cap B$. Del mismo modo, la la unión se puede obtener calculando la operación $\vee$ bit a bit.

Nótese que los filtros de Bloom obtenidos de la codificación de $A\cap B$ o $A\cup B$ no son necesariamente iguales a los filtros de Bloom obtenidos al calcular las operaciones anteriores. Ésto se debe a la naturaleza probabilística de los filtros de Bloom, que podría causar la pérdida de algunos elementos, o incluso aumentar la probabilidad de falsos positivos.

Finalmente, aplicando la Expresión (2) al filtro de Bloom resultante, podremos obtener estimaciones de los cardinales de la unión y la intersección de los conjuntos. Nuestro mecanismo hará uso de la cardinalidad para calcular la similitud entre dos conjuntos.

III-C. Seguridad y privacidad

En [7] se introducen las definiciones de seguridad para los filtros de Bloom, que derivan de las nociones de seguridad para las funciones de hash criptográficas. Primero, definimos el *soporte* de un vector B de tamaño m, representado por $\mathsf{supp}(B)$, como el conjunto de los índices del vector en los que el valor es diferente a 0:

$$\mathsf{supp}(B) = \{i \in [0, \ldots, m-1], b_i \neq 0\}.$$

A continuación, definimos los ataques de *preimagen* y *segunda preimagen* sobre filtros de Bloom:

Definición 1 (Preimagen de un filtro de Bloom): Dado un filtro de Bloom B con un único elemento insertado, una preimagen del filtro es la cadena $y \in \{0,1\}^*$ con índices $I_y = \{h_0(y), \ldots, h_{k-1}(y)\} \subseteq \mathsf{supp}(B)$.

Definición 2 (Segunda preimagen de un filtro de Bloom): Dado un filtro de Bloom B con un solo elemento $x \in \{0,1\}^*$, con índices $I_x = \{h_0(x), \ldots, h_{k-1}(x)\}$, insertado, una segunda preimagen del filtro es una segunda cadena $y \neq x$ con índices $I_y = \{h_0(y), \ldots, h_{k-1}(y)\}$ tal que $I_y \subseteq \mathsf{supp}(B)$. La diferencia con respecto a la preimagen en la Definición 1 es que el adversario conoce x.

Los ataques de preimagen básicamente pretenden recuperar los elementos codificados de un filtro de Bloom. Los ataques de segunda preimagen intentan encontrar elementos cuya codificación es igual a la de otro elemento insertado.

En el mecanismo de autenticación que proponemos, los proveedores de servicio almacenan los perfiles de los usuarios en forma de filtros de Bloom. Para que la privacidad de los usuarios se mantenga, es importante que el proveedor, o cualquier otro atacante con acceso al proveedor, no sea capaz de obtener los perfiles de los usuarios a partir de sus codificaciones. Por lo tanto, necesitamos que los filtros de Bloom resistan ataques de preimagen para asegurar la privacidad de los usuarios. Por otra parte, resistencia a ataques de segunda preimagen garantiza que un atacante no sea capaz de forjar un perfil cuya codificación sea igual a la de un usuario legítimo.

Los autores de [7] recomiendan el uso de códigos de autenticación de mensajes, como HMAC, para prevenir ataques contra los filtros de Bloom. Utilizar HMAC incrementa el dominio de las funciones de hash y, además, ralentiza su ejecución, lo que es deseable desde el punto de vista de la seguridad.

IV. Similitud de conjuntos

En esta sección describimos métodos para calcular la similitud o diferencia entre conjuntos a partir del cardinal de su intersección. Estos métodos se basan en los resultados de [1].

Como en el citado artículo, consideramos tres tipos de características: características categóricas independientes, características categóricas correlacionadas, y características numéricas independientes. En este trabajo nos limitaremos al primer y tercer casos, ya que el segundo requiere la codificación de multi-conjuntos, algo que no podemos conseguir con filtros de Bloom.

IV-A. Características categóricas independientes

Sean X e Y conjuntos de valores categóricos independientes y binarios, es decir, tales que la relación entre dos valores es la igualdad o nada. Este tipo de conjuntos podrían representar el historial de navegación del usuario, antenas de comunicación visibles, aplicaciones instaladas, etc. La diferencia (normalizada) entre X e Y se puede calcular como la distancia de Jaccard, es decir $d_J(X,Y) = (|X \cup Y| - |X \cap Y|)/|X \cup Y|$.

Parece obvio que, a mayor número de coincidencias entre X en Y, menor será la distancia calculada, que estará en el rango $[0, 1]$.

IV-B. Características numéricas independientes

En este caso, el perfil del usuario es un conjunto de valores numéricos, por ejemplo datos de sensores, un histograma de visitas a páginas web (historial de navegación con frecuencias asociadas a cada página), preferencias del usuario en diversos temas, etc.

Dados los conjuntos $X = \{x_1, \cdots, x_t\}$ e $Y = \{y_1, \cdots, y_t\}$, una manera de calcular la distancia entre ellos es calcular $\sum_{i=1}^{t} |x_i - y_i|$. Si X e Y representan histogramas normalizados, esto es, $0 \leq x_i \leq 1$ para todo i y $\sum_{i=1}^{t} x_i = 1$, la distancia calculada puede ser normalizada, ya que conocemos la máxima distancia posible (que es 2).

V. Autenticación implícita usando filtros de Bloom

En esta sección describimos el sistema que proponemos para autenticar implícitamente a usuarios según su comportamiento almacenado. Utilizamos filtros de Bloom para codificar los perfiles de usuario para que el proveedor de servicios no sea capaz de obtener el perfil de usuario original a partir de su codificación; ésto garantiza la privacidad. La autenticación del usuario dependerá de la distancia entre el perfil de referencia almacenado y las muestras aportadas por el dispositivo de usuario, que será calculada por el proveedor. El usuario podrá acceder al servicio si la distancia es inferior a cierto umbral definido a priori.

Primero, describimos cómo implementamos los filtros de Bloom. Luego, describimos el protocolo de autenticación para los casos definidos en la sección anterior.

V-A. Construcción de los filtros de Bloom

Seguimos las recomendaciones de [10] para construir nuestros filtros de Bloom. Básicamente, las k funciones de hash $h_i(x)$ se construyen a partir de dos funciones de hash independientes, $g_1(x)$ y $g_2(x)$, de la siguiente manera: $h_i(x) = g_1(x) + ig_2(x) \mod m$. Esta estrategia reduce el tiempo de computación para codificar los conjuntos, mientras que se mantiene la probabilidad de falsos positivos como si se utilizaran k funciones de hash diferentes. Además, para asegurar que los perfiles de usuario no pueden ser recuperados a partir de su codificación, elegimos una función de hash con clave para g_1, específicamente HMAC con SHA-512, y SHA-512 para g_2. Por tanto, las k funciones de hash quedan definidas de la siguiente manera: $h_i(x, \mathsf{key}) = \mathsf{HMAC}(x, \mathsf{key}) + i\mathsf{SHA\text{-}512}(x) \mod m$.

Por último, los valores óptimos de m y k se pueden calcular en función del número máximo N de elementos que esperamos insertar, y un valor fijo para la probabilidad de falso positivo ρ para las consultas de membresía. Tal como se indica en [7]:

$$m = -\frac{N \ln \rho}{(\ln 2)^2}, \qquad k = \frac{m}{N} \ln 2. \tag{3}$$

V-B. Autenticación implícita para características categóricas independientes

V-B1. Protocolo de registro: Sea el perfil de usuario el conjunto $R = \{r_1, \ldots, r_n\}$ de tamaño $n \leq N$. El valor máximo permitido de elementos N, así como los valores k y m, los fija y publica el proveedor de servicios. El protocolo de registro se desarrolla de la siguiente manera:

1. El usuario inicializa el filtro de Bloom $B_R = \{b_{R,0}, \ldots, b_{R,m-1}\}$ de tamaño m fijando todos los bits a 0.
2. Luego, el usuario elige una contraseña y ejecuta un protocolo de derivación de claves para obtener la clave key.
3. Para cada característica $r_i \in R$, el usuario calcula el conjunto de índices
$$I_{r_i} = \{h_0(r_i, \mathsf{key}), \ldots, h_{k-1}(r_i, \mathsf{key})\}$$
y asigna un 1 a los bits correspondientes.
4. Finalmente, el usuario envía B_R al servidor y elimina B_R y key del dispositivo.

V-B2. Protocolo de autenticación implícita: Para autenticar al usuario, el proveedor calcula la distancia de Jaccard entre el conjunto de referencia almacenado R y la muestra S enviada por el usuario. Para ello, el proveedor calcula los cardinales de la unión y la intersección de los conjuntos R y S:

1. Sea el conjunto $S = \{s_1, \ldots, s_n\}$ la muestra obtenida por el dispositivo del usuario. El usuario obtiene key ejecutando un algoritmo de derivación de claves sobre su contraseña, y construye un filtro de Bloom B_S como en la fase de registro.
2. Después, el dispositivo envía B_S al proveedor y lo elimina, junto a la clave key.

3. El proveedor calcula $|R \cap S|$ y $|R \cup S|$ tal y como se describe en la sección III-B y aplicando la Expresión (2).
4. El proveedor autentica al usuario si $d_J(R,S) < t$, siendo t el umbral predefinido.

V-C. Autenticación implícita para características numéricas independientes

V-C1. Protocolo de registro: En este caso, el perfil del usuario es un conjunto de valores enteros $U = \{u_1, \ldots, u_n\}$. En el caso de que las características del perfil no fuesen valores enteros, se aplicaría una transformación sobre ellos, que podría implicar cierta pérdida de precisión. El usuario construye el conjunto $R = \{(i,j) : 1 \leq j \leq u_i\}$ for $1 \leq i \leq n$. Este conjunto tiene un tamaño igual a $\sum_{i=1}^{n} u_i$. El protocolo de registro continúa del mismo modo que el descrito para características categóricas independientes, descrito en la sección V-B, utilizando el nuevo conjunto R.

V-C2. Protocolo de autenticación implícita: La muestra tomada por el dispositivo del usuario es el conjunto de características numéricas $V = \{v_1, \ldots, v_n\}$. El proveedor calculará la diferencia entre U y V como $d = \sum_{i=1}^{n} |u_i - v_i|$. El usuario construye el conjunto $S = \{(i,j) : 1 \leq j \leq v_i\}$ for $1 \leq i \leq n$ y procede igual que en el caso anterior, pero utilizando el conjunto S (véase sección V-B).

El proveedor, por su parte, calcula $|R|$, $|S|$ y $|R \cap S|$ utilizando la Expresión (2), más el procedimiento de la sección III-B, para el caso de $|R \cap S|$. Nótese que:

$$\begin{aligned} |R \cap S| &= |\{(i,j) : u_i, v_i > 0 \text{ y } 1 \leq j \leq \min\{u_i, u_j\}\}| \\ &= \sum_{1 \leq i \leq n} \min\{u_i, v_i\}. \end{aligned}$$

Por lo tanto, el proveedor puede calcular

$$\begin{aligned} &|R| + |S| - 2|R \cap S| = \\ &\quad = \sum_{i=1}^{n} (\max\{u_i, v_i\} + \min\{u_i, v_i\}) - 2\sum_{i=1}^{n} \min\{u_i, v_i\} \\ &\quad = \sum_{i=1}^{n} (\max\{u_i, v_i\} - \min\{u_i, v_i\}) = \sum_{i=1}^{n} |u_i - v_i|. \end{aligned}$$

El proveedor autentica al usuario si la distancia calculada mediante la expresión anterior está por debajo de un umbral t predefinido.

VI. Análisis experimental

Con tal de probar la aplicabilidad de nuestro mecanismo, lo sometimos a varias pruebas para determinar las pérdidas de precisión en las decisiones de autenticación y el tiempo de ejecución, tanto para el protocolo de registro como para el de autenticación. El protocolo está implementado en Python, y sigue la descripción de la sección V-A.

En un primer test, comprobamos el tiempo de ejecución de los protocolos de registro y de autenticación implícita para diversos valores de n, el tamaño de los perfiles de usuario. Los parámetros m y k se fijaron tal y como describen las ecuaciones (3). Los resultados se muestran en la Figura 2.

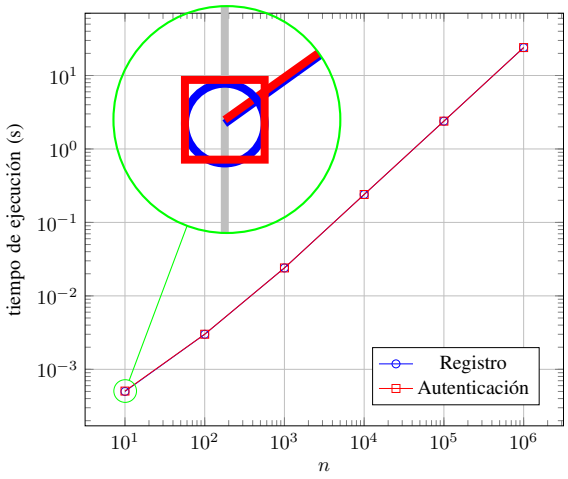

Figura 2. Tiempos de ejecución para diferentes valores de n.

Como podemos observar, ambos protocolos tienen unos tiempos de ejecución prácticamente iguales. Tal y como explicamos en la sección anterior, ambos protocolos conllevan la construcción de sendos filtros de Bloom, cuyo tiempo de ejecución domina por completo el tiempo total de ambos protocolos (23.4 s cuando $n = 1{,}000{,}000$). La comparación de los perfiles codificados tarda menos de 1 milisegundo.

En un segundo test, medimos la pérdida de precisión introducida por nuestro mecanismo en el caso de perfiles con características categóricas independientes. Para ello, generamos 5000 pares de perfiles de usuario con un tamaño de $n = 50$ con características categóricas independientes. El primer perfil de cada uno de los pares es el perfil enviado al proveedor durante el protocolo de registro, mientras que el segundo es la muestra enviada al proveedor durante el protocolo de autenticación. Cada una estas muestras fue alterada aleatoriamente, modificando hasta un 50 % de sus características. A continuación, y para un umbral $t = 0{,}3$, clasificamos los pares de perfiles en dos clases: *aceptados* si la distancia de Jaccard entre los perfiles de cada par es inferior al umbral, y *rechazados* en el caso contrario. Finalmente, ejecutamos nuestro protocolo para cada uno de los pares, con diferentes valores de m y k (en este caso no usamos los óptimos), y contamos cuántos pares son clasificados incorrectamente respecto a la clasificación hecha anteriormente. Los resultados de este experimento (acompañados de los tiempos de ejecución) se muestran en la Figura 3.

Las líneas verticales discontinuas en $m \approx 2^{9{,}5}$ se sitúan en el valor óptimo de m según las Expresiones (3), para $N = n = 50$ y $\rho = 0{,}001$. Observamos que para valores de m iguales o mayores al valor óptimo, la proporción de clasificaciones incorrectas cae por debajo del 5 %; de hecho, llega al 0 % para m cercana a 2^{20}. Nótese que para valores de $m = 2^4$ y $k = 4$ o $k = 8$, la proporción de error es 1. La razón de ello es que para valores de m por debajo del valor óptimo, el filtro de Bloom se satura (todos sus bits son iguales a 1), y por lo tanto, y acorde con la Expresión (2), la estimación

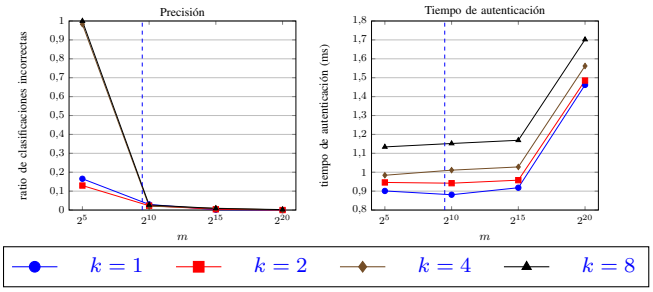

Figura 3. Precisión y tiempos de ejecución para valores de m y k

de elementos codificados tiende a ∞ ($-\ln 0$). Por lo tanto, y siempre partiendo del valor óptimo de m, cuanto mayor sea el valor, mejores resultados de clasificación obtendremos. Habría que considerar, pues, aspectos de eficiencia, tanto en tiempo como en tamaño de los filtros. Cabe decir que los tiempos de ejecución son moderados hasta para valores grandes como 2^{20} y que para lidiar con el tamaño de los filtros, tanto en el transporte como en el almacenamiento, éstos pueden comprimirse.

VII. Conclusiones

En este trabajo, hemos propuesto un mecanismo de autenticación implícita que preserva la privacidad de los usuarios y que es computacionalmente eficiente. Nuestro protocolo parte del trabajo realizado en [4], pero evita la gran carga computacional de dicha propuesta, que limitaba el tamaño de los perfiles de usuario. Con tal de agilizar los cálculos, hemos utilizado las propiedades de los filtros de Bloom para calcular los cardinales de la intersección y unión de conjuntos codificados. Nuestro protocolo es sencillo y rápido, y por lo tanto aplicable a sistemas reales. Además, los perfiles de los usuarios no pueden recuperarse a partir de sus respectivas codificaciones.

Sin embargo, perdemos la seguridad semántica respecto a esta propuesta, lo que podría, en casos extremos, tener un impacto negativo en la privacidad de nuestra propuesta. Planteamos para un futuro el uso de protocolos de transferencia inconsciente y esquemas de cifrado homomórfico para solucionar este problema.

Otra posible línea de trabajo es la de encontrar el modo de calcular las distancias entre conjuntos de características categóricas correlacionadas (por ejemplo, si una de las características consideradas son las antenas de telecomunicaciones, las antenas cercanas se pueden considerar más similares entre sí que las lejanas) mediante filtros de Bloom.

Agradecimientos y descargo

Agradecemos las subvenciones de los organismos siguientes: Gobierno de España (proyectos TIN2014-57364-C2-1-R “SmartGlacis” y TIN2015-70054-REDC), Comisión Europea (proyectos H2020-644024 “CLARUS” y H2020-700540 “CANVAS”). Templeton World Charity Foundation (proyecto TWCF0095/AB60 “CO-UTILITY”), Generalitat de Catalunya (ayuda 2014 SGR 537, beca FI-DGR al primer autor y Premio ICREA Acadèmia al segundo autor). Los autores pertenecen a la Cátedra UNESCO de Privacidad de datos, pero las opiniones en este artículo son suyas y no son necesariamente compartidas por UNESCO o los organismos financiadores.

Referencias

[1] A. Blanco-Justicia, J. Domingo-Ferrer, O. Farràs and D. Sánchez. “Distance computation between two private preference functions.” In *Proc. of 29th IFIP TC 11 Intl. Conference (SEC 2014).* Springer, 2014. 460–470.

[2] B. H. Bloom. “Space/time trade-offs in hash coding with allowable errors.” *Communications of the ACM* 13.7 (1970): 422–426.

[3] A. Broder and M. Mitzenmacher. “Network applications of Bloom filters: a survey.” *Internet Mathematics* 1.4 (2004): 485–509.

[4] J. Domingo-Ferrer, Q. Wu and A. Blanco-Justicia. “Flexible and robust privacy-preserving implicit authentication.” In *Proc. of the 30th IFIP TC 11 Intl. Conference (SEC 2015)* Springer, 2015. 18–34.

[5] C. Dong, L. Chen and Z. Wen. “When private set intersection meets big data: an efficient and scalable protocol.” In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security.* ACM, 2013. 789–800.

[6] M. J. Freedman, K. Nissim and B. Pinkas. “Efficient private matching and set intersection.” In *Advances in Cryptology-EUROCRYPT 2004.* Springer, 2004. 1–19.

[7] T. Gerbet, A. Kumar and C. Lauradoux. “The power of evil choices in Bloom filters.” In *45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2015).* IEEE, 2015. 101–112.

[8] M. Jakobsson, E. Shi, P. Golle and R. Chow. “Implicit authentication for mobile devices.” In *Proc. of the 4th USENIX Conf. on Hot Topics in Security.* USENIX Association, 2009.

[9] F. Kerschbaum. “Outsourced private set intersection using homomorphic encryption.” In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security.* ACM, 2012. 85–86.

[10] A. Kirsch and M. Mitzenmacher. “Less hashing, same performance: building a better Bloom filter.” In *Algorithms–ESA 2006*. Springer, 2006. 456–467.

[11] B. Pinkas, T. Schneider and M. Zohner. “Faster private set intersection based on OT extension.” In *23rd USENIX Security Symposium (USENIX Security 14).* 2014. 797–812.

[12] N. A. Safa, R. Safavi-Naini and S. F. Shahandashti. “Privacy-preserving implicit authentication.” In *ICT Systems Security and Privacy Protection.* Springer, 2014. 471–484.

[13] R. Schnell, T. Bachteler and J. Reiher. “Privacy-preserving record linkage using Bloom filters.” *BMC Medical Informatics and Decision Making* 9.1 (2009): 41.

[14] S. J. Swamidass and P. Baldi. “Mathematical correction for fingerprint similarity measures to improve chemical retrieval”. *Journal of Chemical Information and Modeling* 47.3 (2007):952-964.