IDIS (2009) 2:205–220
DOI 10.1007/s12394-009-0021-7

Sharing private data through personalized search

Kei Karasawa

Received: 24 March 2009 / Accepted: 26 July 2009 / Published online: 1 October 2009

Abstract A method for sharing private data through personalized searches is described. This method enables users to retrieve access-controlled private data as well as publicly available data by submitting a single query to a conventional search engine. Seamless integration of the method into current search services through a prototype on the Mozilla Firefox web browser, without any changes to existing search functions, such as crawling, indexing, and matching, is also described. Evaluations showed that the additional storage requirement is only 10% and that the system implementing the method responds in 10 s with 1,000 dummy results for anonymization.

Keywords Privacy · Digital identity · Identity-based hashing · Private information retrieval · Multi-factor authentication · Personalized search

Abbreviations

AJAX	Asynchronous JavaScript and XML
HTTP	HyperText Transfer Protocol
IBE	Identity-Based Encryption
PIM	Personal Information Management
PIR	Privacy Information Retrieval
PGP	Pretty Good Privacy
PSE	Personalized Search Engine
SAML	Security Assertion Markup Language
SNS	Social Network Services
SSL	Secure Socket Layer
TLS	Transport Layer Security

K. Karasawa (✉)
NTT Information Sharing Platform Laboratories, NTT Corporation, 3-9-11 Midori-cho, Musashino-shi, Tokyo 180-8585, Japan
e-mail: karasawa.kei@lab.ntt.co.jp

Introduction

The development of relational database management systems has been the focus of the data management community for decades, with spectacular results. A recent personal information management system can provide a keyword search over all of its data sources, similar to that provided by existing desktop search systems (Franklin et al. 2005.) Thus, finding worthwhile information in a large amount of data can be done using keyword search.

Recent research studies (Bao et al. 2007) and services have also used information about human activities, in addition to ordinary search-engine technologies, to separate good quality information from poor quality information. Accordingly, information on the Internet links to user contexts and identities when users store information, text, or images or search for them. For example, social bookmarking, in which people register articles worth reading in their shared bookmarks, shows current topics on the Internet, and it is quite effective in helping users search for news or weblogs, the value of which can change in a relatively short time. In addition, search technologies tend to personalize their results according to user context. User identities should be secure, and balancing privacy and usability in personalized searches is a requirement.

From the view point of sharing multimedia content, collaborative tagging systems, such as social bookmarking in deli.cio.us and photo sharing in Flickr, are useful and ad-hoc solutions in which many users add metadata in the form of keywords to shared content (Golder and Huberman 2005; Hammond et al. 2005.) This tagging is also known as folksonomy, short for "folk taxonomy". It is more effective than hyperlink evaluations for sorting news and weblogs and can be used as a recommendation system by means of a keyword-matching function when users performing a search for multimedia content handle that content in the same way as conventional search engines. Collaborative tagging services that are supported by like-minded people who have high information literacy are emerging and being developed, for example, an automated tag-clustering system (Begelman et al. 2006.)

However, users may be uncomfortable with exposing their personal information to search engines. For example, when we share private photos, particularly photos of our children, we might like to share them freely with family members but prevent unknown third parties from eavesdropping on them. Thus, private information is isolated from publicly accessible information that can be searched by commercial search engines, such as Google, in order to restrict access to it. However, finding and retrieving information that is shared by family members is inconvenient in this case because the members need to login beforehand to the restricted sites in which the information is stored. Due to this, it is impossible to unexpectedly find photos of interest as may often occur when using ordinary search engines.

The current approach for sharing content is for users in SNSs to be categorized or to create a sub-community voluntarily to come together and to recommend worthwhile information to each other. The categories are based on user attributes or buddy lists and help users with common interests and similar viewpoints find data. However, finding ambiguous data that can be categorized in several ways is difficult because of the isolation of service categories, particularly when users need to login to a separate service category before they can search for information. For

example, when we search for popular keywords, such as “Halloween”, on ordinary search engines, such as Google Image Search, we might like to be able to access images uploaded by acquaintances as well as any other available images on the Internet. This paper presents a method that can support a user in such a situation.

Problem statement

Target model

Digital devices, such as digital cameras, personal video recorders, and network attached storages, have increased in use and have enabled the storage of huge amounts of digital content at home. Some of that content can be shared with those outside the home. This paper presents a method to share that content securely by using current search engines and new home digital devices. There are three entities connected through the Internet in the target model: 1) search engines, 2) digital content holders, and 3) digital content viewers. Current sharing services for photos or videos combine entities 1) and 2). That is, the services have a database to store photos or videos and a user interface for retrieval. Viewers may have accounts with different services, so content holders need to open accounts with those services to post their data. In the model, search engines are independent of digital content holders. Users will not need to duplicate a huge amount of data and will be able to keep the same user interface for retrieval, that is, the same search engine, when they want to find photos or videos in the distributed home storages connected through the Internet.

For example, on the video-sharing service YouTube, after a user uploads content and chooses the option “Share your video with the world.”, any users are able to retrieve that content by using the search engine Google.

Security requirement

When users make their photos or videos available online, they should be able to protect their privacy, that is, choose who can view the photos and who cannot. In addition, the security protocol must prevent search engines and third parties from eavesdropping on the communication between the content holders and viewers. The security requirements are as follows.

(a) Query Privacy and Identity Privacy

Not only digital content itself but also queries on search engines and tags on digital content can reveal a great deal about a user. From the viewpoint of sharing private data among content holders and viewers, eavesdroppers and even search engines should be prevented from knowing what content viewers want and what content holders have.

(b) Access Control (Stalker Protection)

Content holders should be able to block content viewers from accessing their content, and content viewers should be able to block content holders from

recommending their content. In particular, content holders or content viewers should be able to perform these actions even if the actions were not instigated from the beginning.

For example, after a user uploads content and chooses the option "Private." on YouTube, the content is kept private. No one will be able to find the content on Google, but this also means that sharing it with acquaintances, particularly those who do not have an account with YouTube, is difficult.

This problem is solved by the method, presented in the following section, with which only acquaintances of the uploader can find the content by using a search engine, such as Google, without revealing the content, query, and identity of the searcher to third parties. For more flexible control of the access policy, a method that denies access to users who were previously permitted to access the same content is also presented.

Approach

The proposed method uses two types of identities: one for a digital content holder to register his or her content and one for a content viewer to retrieve the content. The identities are combined with tag information that is linked to the actual multimedia data, such as a photo or video. When a viewer searches digital content by using keywords, the method retrieves content tagged with both the keywords and the identities that exist in the viewer's buddy list or are the viewer's own. To control access to the content, a registered user permits searches by specific users by combining tags with the content viewer identities through which he or she wants to share the content. In other words, tags and identities could impose conditions on access rights. If you want to control the access rights, you can combine the tags with certain roles and places and so on. Complex combinations will achieve this control, but the comparison function commonly used in search engines should be simple so faster responses can be achieved, which are commonly required by users. If the function only requires full or partial matching of the bit stream being compared, integrating the function into the platform of current search engines seamlessly is easier.

Hiding query and identity

To protect user privacy, tags and identities as well as the digital content itself should be hidden from search engines and third parties by a randomizing function. However, to keep the comparison function simple, the hiding function should be deterministic. Of the current search indices, keyed hash algorithms are most suitable for hiding identities simply. For example, the Bloom filter (Broder and Mitzenmacher 2004) is a well-known algorithm for hiding queries while searching that uses multiple hash functions and random salts as an initial value. When content holders store data at home and make only hashed tags available to the public, search engines and third parties can collect only randomized indices. However, search engines can compare the bit stream of the hashed tags with that of hashed search queries by using the same hash key for registration as for retrieval. This paper introduces the concept of an identity as the shared key of the hashing function because both content holders and content viewers usually know each other's identities.

In addition, when you want to use a public identity, such as an e-mail address, as the key but also want to restrict the people who use it, the identity itself could be hashed by the trusted third party that controls access rights to the identity.

Anonymization via clustering

The hash function prevents search engines or third parties from knowing tags and identities, although they can still collect hashed values. The third party can also collect search results by resending hashed queries to search engines. Therefore, user identities may be leaked and privacy may be breached by dictionary attacks or social engineering. To reduce the risk of social engineering, the search results should be anonymized.

This paper introduces a bucket mechanism for anonymization (Fig. 1). Trimming tags and identities that have been hashed for privacy into a short bit stream makes the resulting bit stream the same as other hashed tags and identities. This paper defines a bucket as a set of bit streams that are the same. When content viewers retrieve content by sending the trimmed hashed queries (tags) to search engines, they download the entire bucket, which includes a lot of noise with the same bit stream as the hashed queries. When content viewers retrieve the matched tag from the bucket, they locally compare the full length of the bit streams of the hashed tags and identities, and the original bit stream is hidden from the search engine or the third party.

Anonymity depends on the number of tags in a bucket, and the number depends on the size of the hashed values. If the number of buckets is relatively small, hashed values collide more, and a bucket will have enough noise to create anonymity. However, a longer time is required to download the bucket and find the actual search results from the bucket. If the number of buckets is relatively large, a bucket will have a small number of tags, but search engines or third parties may be able to find some information about trimmed hashed values.

To maintain a certain security level, the number of buckets should be decided in accordance with the total number of hashed values in a database, where the number of buckets is higher than a threshold. However, the total number of hashed values continues to change, and recalculating the trimmed values whenever the database is modified is not practical. In addition, the level of anonymity cannot be increased after sending the results. Therefore, the hashed values should start small and be gradually increased to avoid sending less anonymized results.

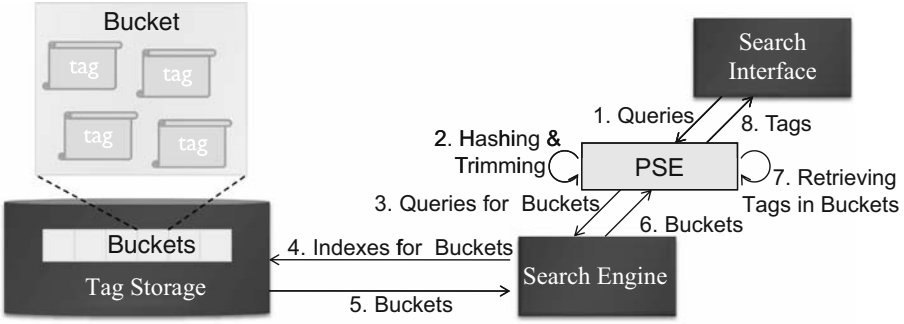

Fig. 1 Anonymization mechanism using buckets

Encryption

Third parties cannot know the trimmed and hashed tags and identities, although they can eavesdrop on digital contents that are transferred as plain data. If digital contents include identity-related information, such as someone's face or voice, the third parties can find the identity-related information by social engineering. Therefore, digital contents should be encrypted to prevent the leakage of private information. To maintain system consistency, the encryption function must be based on the same identities for registering or searching. In addition, the hashed values may include several users when the data is shared in a community. Identity-based encryption (Boneh and Franklin 2001) satisfies the above requirements. In this scheme, a private key generator generates private and public key pairs using a user's identity.

In addition, when data needs to be shared among several users or communities, broadcast encryption (Boneh et al. 2005) can be used, such as the PGP encryption system. However, broadcast encryption requires defining public keys for all users, such as identities in the IBE system, and the set of identity-related information can be disclosed to everybody. That is, there is a tradeoff between the amount of encrypted content and its security level.

Access control (stalker protection)

The above three methods enable content holders and content viewers to hide information from search engines and third parties. However, once content viewers know the available identities for accessing content, they can use them even if content holders want to reject previously permitted content viewers.

One solution is that digital content holders strictly manage registered hashed values to match the list of acceptable users. That is, a hashed key is mapped to a viewer identity and is not shared with anyone else. If content holders subsequently reject previously permitted content viewers, the hashed values are eliminated from the buckets. However, this method cannot adapt to a shared identity that is suitable for identifying a set of anonymous users, such as a community in a SNS.

Another solution is the stalker detection method, which has a threshold number of searches per user such that an identity manager rejects requests after the inner counter reaches a certain threshold. When malicious accesses are found, they can be stopped by filtering, and the malicious user can be found through Internet forensic techniques (Jones R 2005), if necessary. This algorithm can also protect against a dictionary attack from users who know the shared identities.

Implementation

General architecture

The previous section presented a method for personalized searches using identities. From the implementation point of view, the functions for keyed hashing are suitable and their codes can be distributed to clients by using JavaScript.

This section focuses on deployment of the method. Replacing an entire search engine or data storage device is quite difficult, so new functions should be added to current search systems. The method is implemented as a browser extension of Firefox and this extension is seamlessly integrated into current Internet services.

The structure of the functions of the proposed method is shown in Fig. 2. The grey blocks are new functions or data on current search systems.

A personalized search engine (PSE) has the following three functions: 1) hide a query and identity by hashing, 2) anonymize a query using buckets, and 3) encrypt or decrypt data using IBE with a public key.

PSE can be implemented on either the server side or client side. However, PSE is computing intensive, especially when data is encrypted using IBE. Therefore, PSE should be implemented on distributed servers or clients. If the function is written in JavaScript, it can be executed on both server and client sides and is easy to transport to the other side. PSE is implemented using JavaScript. In this implementation, the public JavaScript library is used for HMAC-SHA1 as the keyed-hash function and the anonymization function is executed for trimming the first two bytes of hashed values that include tags and identities. When a content viewer confirms the search result, PSE cannot decrypt the hashed tags and identities, although it can decrypt encrypted content. Therefore, PSE stores the original queries and restores them when the hashed values are matched with the search results.

In addition, we need to think about the implementation of identity management. An identity that is sent to some members could be spoofed, unless an appropriate authentication mechanism exists. For example, if identities were to be spoofed, content viewers could be exposed to the dangers of falsified information, such as SPAM mail. Therefore, identity management must be considered to build the system securely.

The identity manager authenticates users, sends identities to PSE, and stores policies for using identities. Users are authenticated before using identities, and identity information is sent to the authorized functions. For example, a content

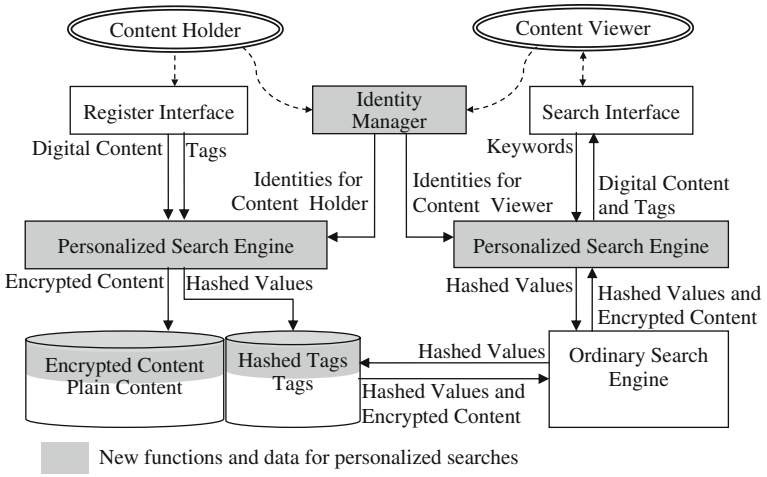

Fig. 2 Function architecture for personalized search

viewer finds recommended photos from a content holder when he or she has registered those photos with the viewer's identity through the identity manager, which permits the holder to use the viewer's identity as a shared key. The identity manager stores access policies using a stalker protection method as described above.

Prototype of PSE

The user interface of the prototype that is implemented in Firefox by using JavaScript and the AJAX method is shown in Fig. 3. The result of searching for a content holder's video on the public video-sharing service YouTube is shown. The PSE on the browser mash-ups two dialogs for the personalized search into the ordinary search results on YouTube. The search queries are "presentation" and "video".

In the top left corner of the window, Alice, an identity of a content viewer, is shown. The dialog is the interface to the identity manager, and a content viewer needs to login to the identity manager to authenticate him or herself.

In the middle of the window, a new box entitled "Personal Video" shows personalized search results as well as general search results for "Search // Presentation Video". This result shows two readable tags, "presentation" and "video", which PSE restored from search queries by fully matching hashed queries and returned hashed values, and 14 unreadable tags "-", which PSE is hiding from other viewers by the hash function on the browser. However, the search engine and third parties, such as eavesdroppers, will see 16 unreadable hashed tags. In the top

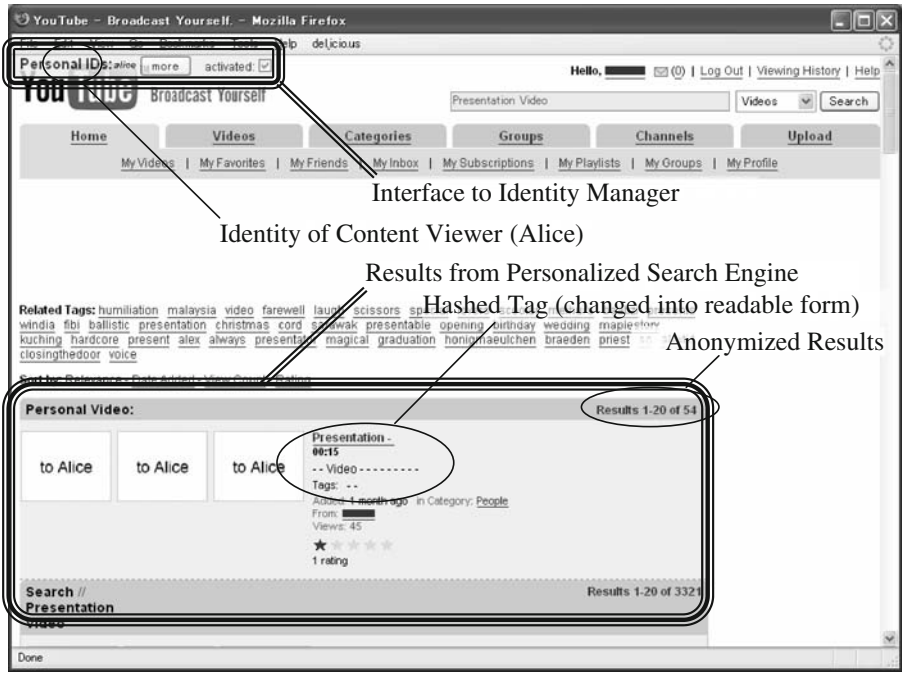

Fig. 3 Screenshot of personalized search result on prototype system

right corner of the box, the number of search results is shown. In this case, only one result matched the search, and the other 53 results were noise for anonymization. Therefore, the search engine and third parties can only know that the result is 54 videos, each of which has hashed tags starting with the same 2-byte bit stream, which are calculated through the anonymization function described above.

When a user sets his or her identity as Alice, the user will obtain the images tagged for Alice (Fig. 4). In this example, the user Alice searches for content with the queries "Test Video". Alice retrieves the private image "to Alice".

When a user sets his or her identity as Bob, the user will obtain the images tagged for Bob (Fig. 5). In this example, the user Bob searches for content with the same queries "Test Video". Bob retrieves the private image "to Bob".

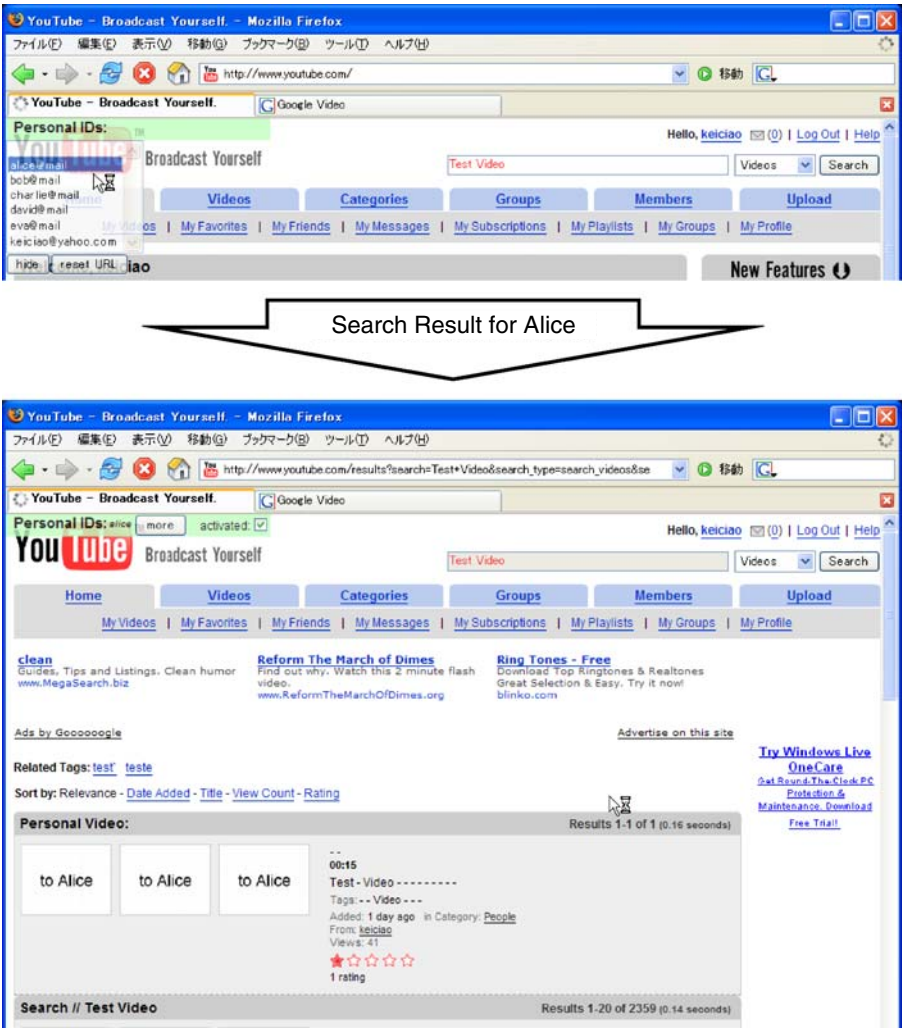

Fig. 4 Example of search personalized for Alice on YouTube

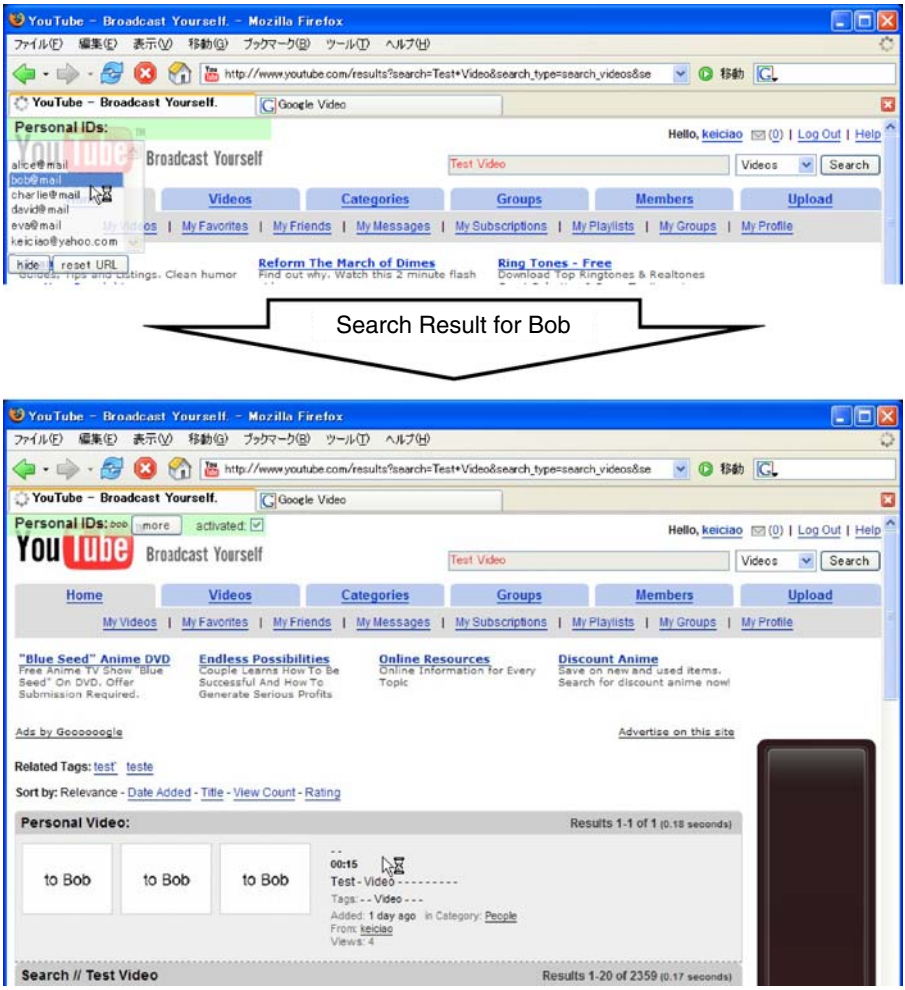

Fig. 5 Example of search personalized for Bob on YouTube

These examples show that only selected users can retrieve the contents and that the search result is personalised in accordance with the selected identity.

Connecting to identity manager

The permission to use identities should not be limited to the owner of the identities so that users in an SNS can share their private information with buddies that have permission. The authentication mechanism should be implemented independently of PSE, which should be distributed for effective information processing and privacy protection. The authentication function and PSE can be connected through the Internet. Therefore, handing over authentication information from an authentication function to a PSE is required. To implement the reuse of the authentication, we have some candidates in the identity-centric architecture, such as OpenID, and SAML. Both OpenID and SAML are leading candidates because they have been widely

deployed and provide many useful and secure functions, such as single sign-on, which enables a user to authenticate only once and gain access to the resources of multiple software systems. PSE uses SAML to get identity information from an identity manager without multiple authentications, which works as a policy decision point and policy enforcement point.

A login process is not suitable for search actions. To eliminate the login process, a user should use a secure token issued by an identity provider that is the identity manager. For example, cookies with a SAML assertion from the identity manager can store the token and transfer it through the search interface.

Stronger and easier authentication

Ideally, an identity manager itself should omit an explicit login process. A cookie for login is easy but vulnerable to fishing attacks. Once a cookie is intercepted, the eavesdropper can spoof the user anywhere.

SSL/TLS, which is the most widely adapted security protocol for the HTTP protocol, provides a strong and mutual authentication method by using digital certificates. IP address filtering is also effective as multi-factor authentication (Park and Redford 2007.) Cookies can be reinforced by combining them with the SSL/TLS mutual authentication protocol. However, managing certificates is usually quite expensive. In particular, certificates authorities must keep their private keys secret while issued certificates remain valid. The foundation of a certificate authority system usually includes building security systems and education systems for operators that are too expensive for a single server.

This paper introduces an easier method for managing certificates that is similar to the PGP system. PGP shares user public keys, and users rely on the public key repository. HTTP servers also have their own repository, usually for storing user profiles, such as identities and buddy lists. The proposed method is for an HTTP server to store user public keys that are included in user certificates in its repository, to accept the user connection by using the SSL/TLS client certificate authentication protocol, and to authenticate the user by comparing both stored certificates and accepted certificates. In this method, the certificates themselves do not need to indicate authenticity and are used only to comply with the SSL/TLS protocol. Therefore, certificates can be issued by anyone, such as the user him or herself. That is, the private key for issuing certificates can be visible to the public. In the method, when a user registers with an identity manager, he or she creates an account on the server and stores his or her profiles, including the certificate that was self-issued using the private key for issuing certificates that was broadcasted by the server without any authentication steps.

Evaluation

Response time for privacy protection

The privacy level depends on the bucket size, that is, the number of hashed values in a bucket. The response time of PSE mainly depends on the number of hashed values

because a search engine returns an entire bucket with a lot of noise to the client. The response time was measured on two commercial video-sharing services, “A” and “B”, to find a practical number of hashed values by changing query lengths. As Fig. 6 shows, the response time of service A increases linearly with the number of hashed values. The response can be hidden in about 1,000 dummy responses if users can wait for 10 s to obtain an exact result. However, in service B, the response time became exponentially slower with an increasing number of hashed values. It is speculated that the reason was the response being tuned to the common user behavior of only looking at the top ten results on the first page and not looking at the next page. Therefore, the proposed method is most suitable for personalized searches of service A because ordinary search engines return more results more quickly for this service.

Memory capacity for privacy protection

To control content viewers in the proposed system, tags for content are hashed by identities and duplicated by the number of identities. Therefore, the memory capacity may overflow for large amounts of content, but it will be suitable for small communities, such as communities in SNSs, family, or close friends. This paper evaluates the memory capacity according to the results of social bookmarking experiments.

If the number of hashed values were very large, the memory capacity to perform the calculations would not be practical. However, in a social bookmarking experiment, Michlmayr (2005) reported that the average number of tags per bookmark is 24.59, and for an enterprise service, Millen reported 2.3 tags per bookmark (Millen et al. 2006.) Therefore, the average number of tags per content in social bookmarking services can be assumed to be similar to 24.59 and is estimated to be about 1 KB per tag. Michlmayr also reported that only 0.51% of the bookmarks are stored by more than ten participants, 8.90% are stored by two participants, and 84% are stored by only one participant. Therefore, the average number of identities per content is also estimated to be about 100 even if the system is used in public services. Therefore, the number of hashed values can be 100 KB per content even though the data itself may be more than 1 MB, which is the average

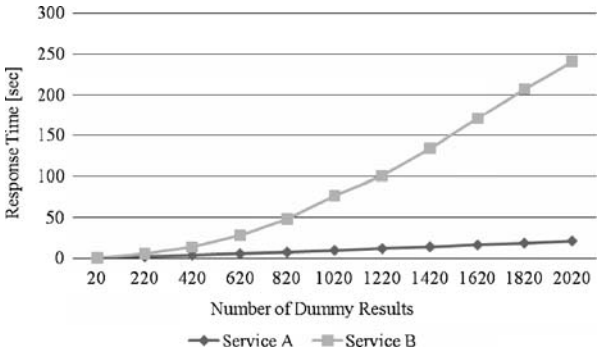

Fig. 6 Response time depending on number of hashed tags

size of multimedia content. That is, search engines only require 10% larger storage for a personalized search.

Related works

Private web searching (Saint-Jean et al. 2007; Xu et al. 2007) is the process of retrieving information from the Internet without revealing private information about the retriever. Private web searching anonymizes or hides a retriever's profile and/or identity, such as user name, gender, or IP address of the PC that the retriever uses for web searching. The anonymization solutions of private web searching can also be used in the proposed method instead of the bucket mechanism, although the proposed method differs from private web searching in several ways. First, the retrieved information from a search engine is allowed to include noise for anonymization because the tags on multimedia content are quite simple and the noise can be easily filtered. Second, in the proposed method, content folders need to have their content modified before publishing. Deploying the existing content could be difficult, although multimedia content will increase as network capacity improves.

To ensure privacy, two well-known privacy technologies have been proposed: privacy-preserving data publishing (Verykios et al. 2004; Zhong et al. 2005) and privacy information retrieval (Gasarch 2004.)

Privacy-preserving data publishing is the process of making a database of personal information available to the public for data mining without the problem of revealing identifying information about individuals. The solutions for avoiding this problem suggest that for the proposed method, sensitive raw data, such as identifiers, names, and addresses, should be modified. In the proposed method, however, the data mining algorithm must be simple because existing search engines are used.

PIR allows users to query a database without revealing the query. One simple PIR solution is for the database owner to send a copy of the database to the user. However, in the proposed method, the databases are huge and impossible to duplicate in their entirety. Therefore, a bucket is introduced in the method as a subset of a database. This is a practical solution using PIR and privacy-preserving data publishing.

A Bloom filter is a simple, space-efficient, randomized data structure for concisely representing a set of membership queries (Broder and Mitzenmacher 2004.) Search schemes based on Bloom filters and encryption methods by using Pohlig-Hellman encryption (Bellovin and Cheswick 2004) and IBE (Goh 2003) have been proposed. These methods encrypt Bloom filter indices by using a public key to protect against eavesdropping by third parties. Therefore, queries need to be decrypted. However, using this process is expensive because it requires many changes to search engines. The proposed method can be seamlessly integrated with current search engines because hashed tags can be handled by existing search functions, such as crawling, indexing, querying, and matching, and be independently calculated by users as necessary. That is, the method can be inexpensively deployed because it can use current search systems and users only need to add simple functions to create hashed tags if necessary.

A keyword search of encrypted data by using IBE has been proposed (Boneh et al. 2004), and a keyword search of streaming data has also been proposed (Ostrovsky and Skeith 2007; Baek et al. 2008). These methods enable third parties to determine whether only some keywords are in a document. However, the search time is linear in accordance with the size of the documents and keywords.

There are many research studies about personalized searches for web information. To personalize the results, Jeh and Widom (2003) have presented a method to put weight on a hyperlink that is used by a page rank algorithm, and a method has been proposed to augment the user query with information extracted from the search context by anonymous users (Kraft et al. 2006). However, these approaches have not directly handled user identities nor protected user privacy.

Future works

The presented method focuses on preventing third parties from eavesdropping on private searches, although some questions remain from the viewpoint of deployment.

Security analysis for identity management

In the proposed method, the security policy is described in the identity management mechanism for authentication of user identities. There are two types of use cases:

case 1, your own identity, which is yourself or the group that you belong to, and
case 2, the identity of another person or group to which you do not belong.

In case 1, the authentication mechanism can directly control the identity and its access right. That is, the mechanism can be simply implemented by using the usual authentication systems, such as RADIUS servers. However, there is no direct relationship between the identity and its access right in case 2. A user needs to control the access rights of the identities that he or she has or manage those rights on the basis of a policy decision point. For example, Alice permits Bob to use her identity when Bob personally recommends photos to Alice. As another example, Bob permits Charlie to use his identity only when Charlie searches for videos registered by Alice. Alice and Bob need to describe the roles and contexts. The policy is defined by the following five elements, although elements 4 and 5 are not essential: 1) agents—users' identifiers, 2) actions—registering, searching, and comparing (full or partial matching), 3) content—content with hashed tags and identities, 4) date—valid date and time, and 5) place—valid place. A user may define a wildcard to simplify the policy. As future work, the security policy of the proposed method will be described using previously proposed methods for privacy policies (Barth et al. 2006), and the security level will be clarified.

Social engineering

Adding random values to anonymize search keywords will not work in practice (Bellovin and Cheswick 2004.) However, if queries and their results are fully

randomized, an eavesdropper cannot obtain any private information. The proposed method can hide the tags and identities of searching or registering users, but hiding their IP addresses or URLs is difficult. IPv6 hosts may have a function for randomizing their addresses (Michlmayr 2005.) However, if the function is used, the network prefix cannot be randomized, and changing that network prefix so often is quite difficult because it causes Internet routing instability. A URL can be generated randomly in the directory, but a domain name is difficult to randomize because it depends on the DNS architecture. We use a proxy to anonymize source addresses from third parties when the proxy is trusted, but to make the system simpler, the anonymization function should be included in the client that has the source content that is being searched.

Conclusion

Internet searches usually handle private information about users, such as identities and contexts, to retrieve worthwhile information. This paper presented a method to protect user privacy on a useful personalized search platform. New storage devices to store hidden tags and identities as well as perform hash processing are required by this method. However, it was demonstrated that only 10% more storage is required and that the system implementing the method responds in 10 s with 1,000 dummy results for anonymization. In addition, the method can be implemented as a browser extension to be used with existing search services without changing those services. That is, the system can be used in the current computing environment.

For example, when a user uploads content as public onto YouTube, any users are able to find it. The proposed method enables only the uploader's acquaintances who have shared keys to find that content by using hashed tags on contents that are handled by search engines. The existing search engine Google and web browser Firefox can be used by installing a browser extension that implements the proposed method.

In addition, the search engine Google is prevented from collecting even hashed queries because the search queries are anonymized in a cluster named a "bucket" by shortening their hash lengths to the same as those of other hashed tags. The browser extension for Firefox can find the target contents in a "bucket" in a local PC.

The proposed method can be introduced to common search engines by choosing randomizing and anonymizing functions, such as hashing and trimming, suitable for both search crawlers and search engines. The implementation of the proposed method as an extension to an ordinary browser showed that the method is feasible because the hashing function is quite simple and effective.

Identity-centric architecture enables us to create an individual community easily, and a method was presented for users to securely and easily login to the identity management system without using an expensive trusted third party system.

Electronic devices in user homes store a lot of digital content and can be connected to the Internet through broadband channels. Sharing that content is easy, but finding worthwhile content on the Internet is difficult. This research supports users in finding and viewing valuable public content through recommendations by other people connected via the identity-centric system.

Acknowledgements I thank Prof. Dan Boneh and Dr. Neil Daswani for their comments on extensions to the "Approach" section.

References

Baek J, Safiavi-Naini R, Susilo W. Public key encryption with keyword search revisited. Lect Notes Comput Sci. 2008;5072:1249–1259.

Bao S, Wu X, Fei B, Xue G, Su Z, Yu Y. Optimizing web search using social annotations. Proceedings of the 16th International Conference on World Wide Web, May 2007, pp. 501–510.

Barth A, Datta A, Mitchell J, Nissenbaum H. Privacy and contextual integrity: Framework and applications. Proceedings of the 27th IEEE Symposium on Security and Privacy, May 2006, pp. 188–198.

Begelman G, Keller P, Smadja F. Automated tag clustering: Improving search and exploration in the tag space, collaborative web tagging workshop. Proceedings of the 15th International World Wide Web Conference, May 2006.

Bellovin S, Cheswick W. Privacy-enhanced searches using encrypted bloom filters, at: http://www.research.att.com/~smb/papers/bloom-encrypt.ps; 2004.

Boneh D, Franklin M. Identity-based encryption from the weil pairing. Lect Notes Comput Sci. 2001;2139:213–229.

Boneh D, Crescenzo G, Ostrovsky R, Persiano G. Public key encryption with keyword search. Lect Notes Comput Sci. 2004;3027:506–522.

Boneh D, Gentry C, Waters B. Collusion resistant broadcast encryption with short ciphertexts and private keys. Lect Notes Comput Sci. 2005;3621:258–275.

Broder A, Mitzenmacher M. Network applications of bloom filters: a survey. Internet Mathematics. 2004;1(4):485–509.

Franklin M, Halevy A, Maier D. From databases to dataspaces: a new abstraction for information management. ACM SIGMOD Record. 2005;34(4):27–33.

Gasarch W. A survey on private information retrieval. The Bulletin of the EATCS. 2004;82:72–107.

Goh E. Secure indexes, on the Cryptology ePrint archive, October 2003.

Golder S, Huberman B. The structure of collaborative tagging systems, at: http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cs/0508082, 2005.

Hammond T, Hannay T, Lund B, Scott J. Social bookmarking tools (I), at: http://www.dlib.org/dlib/april05/hammond/04hammond.html, April 2005.

Jeh G, Widom J. Scaling personalized web search, The 12th International Conference on World Wide Web, May 2003, pp. 271–279.

Jones R. Internet forensics, O'Reilly Media, Sebastopol, California/USA, 2005, Chapter 6 Web Servers.

Kraft R, Chang C, Maghoul F, Kumar R. Searching with context, The 12th International Conference on World Wide Web, May 2006, pp. 477–486.

Michlmayr E. A case study on emergent semantics in communities, The 4th International Semantic Web Conference, November 2005.

Millen D, Feinberg J, Kerr B. Dogear: Social bookmarking in the enterprise, The Conference on Human Factors in Computing Systems, April 2006, pp. 111–120.

Ostrovsky R, Skeith W III. Private searching on streaming data. J Cryptol. 2007;20(4):397–430.

Park H, Redford S. Client certificate and IP address based multi-factor authentication for J2EE web applications, Proceedings of the 2007 Conference of the Center for Advanced Studies on Collaborative Research, October 2007, pp. 167–174.

Saint-Jean F, Johnson A, Boneh D, Feigenbaum J. Private web search, workshop on privacy in the electronic society, October 2007, pp. 84–90.

Verykios V, Bertino E, Fovino I, Provenza L, Saygin Y, Theodoridis Y. State-of-the-art in privacy preserving data mining. ACM SIGMOD Record. 2004;33:50–57.

Xu Y, Wang K, Zhang B, Chen Z. Privacy-enhancing personalized web search, The 16th International World Wide Web Conference, May 2007, pp. 591–600.

Zhong S, Yang Z, Wright R. Privacy-enhancing k-Anonymization of customer data, The 24th ACM SIGMOD International Conference on Management of Data, June 2005, pp. 13–15.