# Secret Picture: An Efficient Tool for Mitigating Deletion Delay on OSN

Shangqi Lai[1](✉), Joseph K. Liu[2], Kim-Kwang Raymond Choo[3,4], and Kaitai Liang[5]

[1] Department of Computer Science, The University of Hong Kong, Hong Kong, Hong Kong
aquas@connect.hku.hk
[2] Faculty of Information Technology, Monash University, Clayton, Australia
joseph.liu@monash.edu
[3] School of Information Technology and Mathematical Sciences, University of South Australia, Adelaide, Australia
raymond.choo@unisa.edu.au
[4] INTERPOL Global Complex for Innovation, Singapore, Singapore
raymond.choo@fulbrightmail.org
[5] Department of Computer Science, Aalto University, Espoo, Finland
kaitai.liang@aalto.fi

**Abstract.** With the increasing popularity of online social networks (OSNs) and the ability to access and exchange sensitive user information, user privacy concerns become an important issue which have attracted the attention of researchers and policymakers. For example, deleted pictures or pictures in deleted posts may not be deleted from the OSN server immediately, and hence accessible to another unauthorized user. In this paper, we highlight the deletion delay issue in seven popular OSNs, namely: Facebook, Instagram, MySpace, Tumblr, Flickr, Google+ and Weibo, which can be exploited by another unauthorized user to gain access to these pictures. To ensure OSN users are able to achieve a higher level of privacy, we propose a conceptual privacy-preserving tool for photo sharing, without compromising on transparency and real-time sharing features. We demonstrate the utility of the tool by prototyping a browser extension, which does not require modification of existing OSN systems.

**Keywords:** Online social networks · Deletion delay · Privacy attacks · Privacy-preserving for social networks · Photo sharing

## 1 Introduction

Online social network (OSN) providers, such as Facebook, Twitter, and Google+, provide an effective platform for its users to conduct real-time communication. For example, users can update their status, check-in, post a comment and upload other user-generated content (e.g. text, picture, and video) in the social networks.

DOI: 10.1007/978-3-319-29814-6_40

It is no surprising that the popularity of OSN has extended to users of different ages, countries and cultures.

However, when users disclose personal or sensitive information about themselves on the OSN, they are often unaware of the privacy implications. For example, who is able to access these information, and how these information can be mined or abused by, say, a cyberstalker or a criminal (e.g. publicly accessible home address information, holiday pictures, and wall posts such as "I am on holidays at Puerto Rico" will be targeted by opportunistic burglars). Consequently, OSN users may suffer financial loss, physical harm, etc. Although major OSN providers have put in place privacy and other related policies to ensure the security and privacy of user data, most users may not be familiar but the privacy settings. For example, OSN providers, such as Google+, can have in place extensive policies and measures to prevent accidental leakage of user information (e.g. public, and list of friends), for example, by providing users with the visibility of their data; users may not always choose the appropriate settings to preserve the privacy of their data. In addition, malicious actors, such as spammers and phishers, are on the constant lookout for ways to learn user data from OSNs.

In this paper, we discuss about the privacy issues associated with user-uploaded pictures as pictures may contain rich information, such as metadata, that can be extracted by a third-party (including a cybercriminal). For example, in a group picture taken at a kid's birthday party and uploaded to an OSN may provide a stalker with information such as the exact date of birth for the kid, who the parents and friends are, what school the kid go to, where the kid stay, what are the kid's interests, etc. More specifically, we want to highlight a less understood risk – **Deletion Delay**. In deletion delay, deleted pictures or pictures in deleted posts are preserved by the OSN providers and these pictures may be accessible to anyone with knowledge of the picture's URL.

In this paper, we study the deletion delay issue in popular OSNs. We then propose a privacy-preserving tool to address the deletion delay. In our solution, we design and implement the tool in a browser extension, which allows one to encrypt pictures at the user-end prior to uploading to the OSN. A third-party server is then used to store the secret key and sharing list of the user. By encrypting the online pictures, even if one of your friends decides to share the pictures with others, a new sharing list will be required to access these pictures.

## 2 Background

### 2.1 Problem Statement

Before introducing the problem, we briefly review the basic features available in OSNs [10]:

– Post/share picture: Almost all popular OSNs allow registered users to upload and share their pictures (and/or videos). For instance, Google+ users can click on the "photo" button and select a picture on the local device or a Google album to be posted. Users can generally control the extent of sharing. If they only choose to share the picture with a specified list of friends, then only friends in the list can access the picture. Hence, users can make the picture private by including only themselves in the sharing list. According to [10], OSNs such Facebook and Instagram use a similar mechanism to manage user access permission.

– Obtain URL of picture: Authorized users can access the pictures uploaded to OSNs via the URLs, which can be obtained by right-clicking on the particular picture and copying the URL (i.e. "Copy URL"). This method is available in popular OSNs, such as Facebook, Twitter, Tumblr and MySpace [10]. Although some OSNs providers do not provide such a feature, browsers such as Google Chrome has the "Inspect element" feature that allows the user to access the html code of the website, and locate the URL in the source code. Needless to say that the access control mechanism employ by these OSNs providers will not stop unauthorized user access to the pictures via URLs.
– Delete picture/post: The user can simply click "Delete" to remove the picture/post from the OSNs.

**Deletion delay** refers to the problem that when an user attempts to delete their posts. While the post will disappear from the user's profile (e.g. in Facebook, the user's wall) immediately, the picture is still stored on the OSN servers, perhaps with the exception of Twitter as noted in [10]. Therefore, anyone can access the deleted picture by accessing the original URL.

Table 1 details our study of eight popular OSNs. We found that with the exception of Twitter, most of the OSNs have a deletion delay of over three days. In the case of Weibo (the most widely used OSN in mainland China), many developers use this as a third-party Image Storage Service [15] because it has a long deletion delay. We found that pictures in the deleted post three years ago could still be accessible online.

**Table 1.** Deletion delay of different online social networks, adapted from [10]

| Platforms | Day(s) |
|---|---|
| Facebook | 7 |
| Twitter | Immediately |
| Instagram | 3 |
| MySpace | $> 30$ |
| Tumblr | $> 30$ |
| Flickr | 14 |
| Google+ | $< 1$ |
| Weibo | $> 1124$ |

If the deleted pictures have been shared with other OSN users prior to deletion, the thumbnail images or a copy of the deleted picture may still be recover-able and accessible, as shown in recent studies (e.g. from Windows devices [11] and from Android [9] devices).

Observations from our study are as follow:

1. Some OSN providers may copy the link of the original picture directly. Hence, if the source platform has a deletion delay, then the target platform will have the same delay issue;
2. Some OSN providers use shared links associated with the original copy of the shared picture. These shared links will expire when the original copy is deleted; and
3. Although shared links can be removed, some OSN providers provide a thumbnail of the shared picture in the destination platform timeline. The thumbnail will be kept in the timeline when the original copy is deleted;

## 2.2 Countermeasures

While deletion delay can potentially compromise the privacy of a user's privacy, there are several potential solutions.

Eliminating the deletion delay is an obvious solution, and in theory, this is an action that can be easily undertaken by OSN providers. However, in practice, it had been noted by various independent sources (see [5,14]) that Facebook maintain copies of the user pictures on their servers after the users have decided to delete the pictures and posts. It was noted that from the calendar years 2010 to 2012, the deletion delay decreased from three years to 30 days, and in 2015 [10], the picture became unavailable seven days after the deletion. But from the perspective of the OSN providers, the practicality of this approach relies on the data structure. Searching and removing a single picture in more than a million resources is a very challenging task. Deletion requests may result in significant overhead and delay, as these requests needed to be forwarded to different data centers all over the world, and each data center deals with thousands of requests at any one time.

Users can also choose to encrypt the picture prior to uploading, such as using the traditional public key encryption [7,12], identity-based encryption [4,13], and attribute-based encryption [3,8]. It is more reliable as these measures ensure that any pictures uploaded and shared online are encrypted. In practice, ONS providers never offer such service, therefore, with the increasing number of friends and shared photos, the key management will become increasingly complex and unwieldy, if user should do it on his own. The user would need to deliver the decryption key to all friends in the sharing list after the user has uploaded an encrypted picture. Once the user decides to revoke this sharing, the user would need to use a new encryption key to encrypt the pictures and request the server to delete the old encrypted pictures. This is an inefficient and impractical solution.

In this paper, we address the following research challenge: "Can we design a new mechanism to mitigate the overhead in key generation and distribution, while managing shared list in a simple and efficient way?".

## 3 Our Conceptual Privacy Preserving Tool

### 3.1 Our Approach

To address the research challenge identified in the preceding section, we consider the principles. Firstly, there is no doubt that local encryption (i.e. encryption at the user-end) is preferred, as it provides the user more control over the picture (including shared links and thumbnails). Secondly, the solution should not burden users with key management, and this can be achieved using a sharing list and the encryption/decryption key approach as we will describe in this section (see our browser extension).

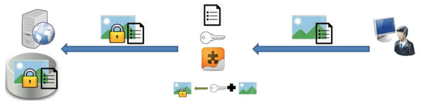

**Fig. 1.** Posting of picture

*Posting of picture (see Fig. 1).* Once a user posts / uploads a picture to an OSN platform using a browser, the browser extension generates a symmetric key randomly, and encrypts the picture using a pre-determined symmetric encryption algorithm. In our prototype described in Sect. 3.2, we use Advanced Encryption Standard (AES)-256 bit encryption [1]. The picture is then posted to the OSN platform. Upon receiving the request and encrypted picture from the browser, the OSN platform will store the encrypted picture and generate a URL for accessing the encrypted picture. The browser extension will call the corresponding API to upload the picture and obtain the URL, as well as storing the key-value pair of the symmetric key and the picture identity (i.e. the URL of original copy) on the local device by calling localStorage API in HTML5 [2]. Therefore, other OSN users will only be able to see an encrypted picture. Even if the users have the URL of the encrypted picture, they will not be able to view the picture without having access to the decryption key.

*Accessing encrypted picture (see Fig. 2(a)).* When the picture owner and users who have been granted the access right wish to access the encrypted picture, the OSN server will send the encrypted picture to the respective user's browser extension. This (plug-in) extension verifies the access rights specified by the sharing list. If the user is in the sharing list or the user is the owner, the extension will use the corresponding symmetric key to decrypt the encrypted picture, and present the (decrypted) picture to the user.

*Deleting encrypted picture (see Fig. 2(b)).* When the picture owner decides to delete the picture, then a "delete" request will be sent to the browser extension and the OSN platform. The extension then deletes the symmetric key associated with the picture. When an user who have been previously granted access to the

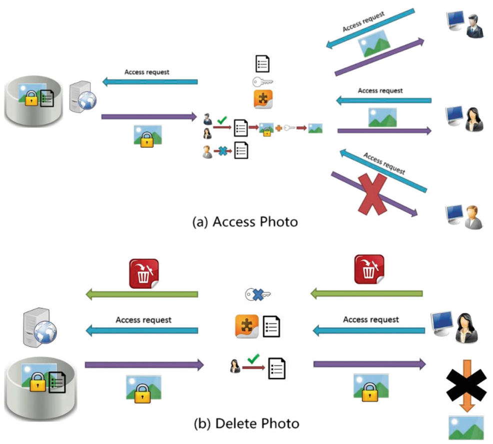

**Fig. 2.** Accessing and deleting the encrypted picture

deleted picture wishes to view the picture, the extension will check and determine that the picture has been deleted (i.e. corresponding symmetric key cannot be located), and therefore, not able to decrypt the encrypted picture.

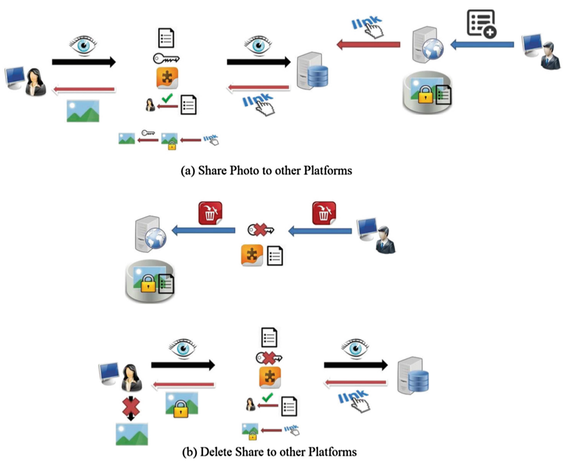
(a) Share Photo to other Platforms
(b) Delete Share to other Platforms

**Fig. 3.** Share and delete photo on other platforms

*Sharing encrypted picture to other OSN platform (see Fig.* 3*(a)).* When the (encrypted) picture is shared on other OSN platforms, the original OSN server generates a link for the third-party platform to access the encrypted picture. The browser extension obtains a new key-value pair of the symmetric key and picture identity. If a user on the third-party OSN platform requests for the encrypted picture, the extension will check against the sharing list and decrypt the picture if the requesting user is included in the sharing list. Otherwise, the decryption will not take place.

*Deleting encrypted picture from other OSN platform (see Fig. 3(b)).* When the user decides to delete the picture, the user will send a "delete" request to the browser extension and the OSN platform. Upon receipt, the OSN platform will delete the shared link. At the same time, the browser extension will also delete associated symmetric key. Subsequent request to access the deleted picture will be unsuccessful, as the decryption key associated with the encrypted picture has been deleted by the browser extension.

### 3.2 Our Prototype

We design a plug-in extension for Google Chrome – see Fig. 4.

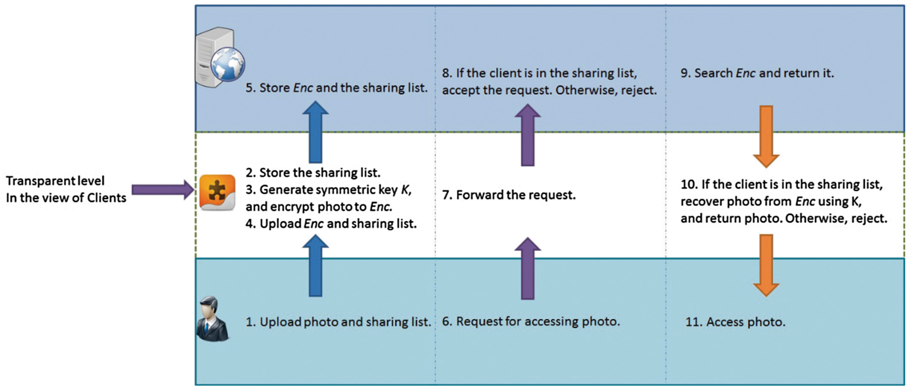
5. Store *Enc* and the sharing list.
8. If the client is in the sharing list, accept the request. Otherwise, reject.
9. Search *Enc* and return it.
Transparent level
In the view of Clients
2. Store the sharing list.
3. Generate symmetric key *K*, and encrypt photo to *Enc*.
4. Upload *Enc* and sharing list.
7. Forward the request.
10. If the client is in the sharing list, recover photo from *Enc* using K, and return photo. Otherwise, reject.
1. Upload photo and sharing list.
6. Request for accessing photo.
11. Access photo.

**Fig. 4.** An overview of our approach

The extension (see Fig. 5) uses the API from the respective OSNs, and consists of three components, namely: a popup interface, background and content script.

The popup interface allows an user to authorize the extension to access the respective API. Users also manage the sharing list using the popup interface. The users post / upload pictures using the page, and the pictures will be encrypted by Forge [6] prior to calling the corresponding OSN API. Immediately following this, the extension stores the sharing list and key-value pair on the local device.

Background is an invisible page, but both popup and the content script are able to access this page resource. Background plays the role of the daemon process to store sensitive information, such as the sharing list and key-value pair records. It also runs the decryption asynchronously.

Content script monitors specific webpage by registering listener. When the content script detects a picture identity, it will check against the sharing list, and

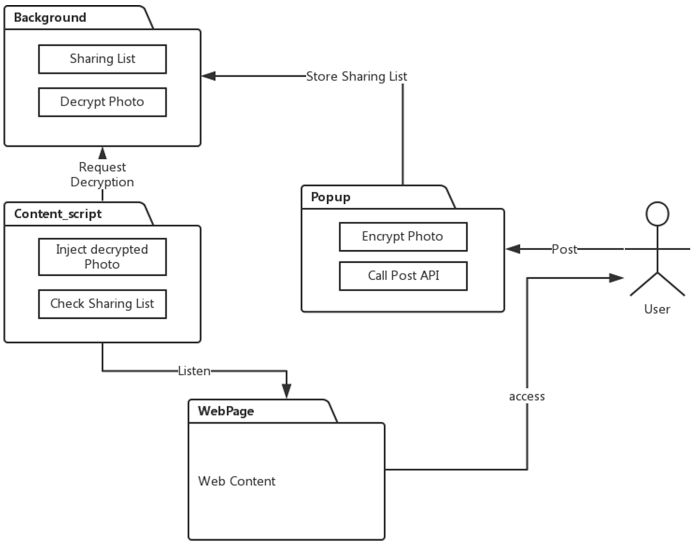

**Fig. 5.** An overview of our prototype

forward the ciphertext with the request message to the background to obtain the original picture. Once this happens, the content script will display the decrypted picture to the user.

## 4 Evaluation

### 4.1 Experiment Setup

To conduct the evaluation, we establish a developer account in Sina Weibo and for Google Chrome. We create a new App in Sina Weibo to facilitate the extension obtaining authorization and calling of the API. Meanwhile, we create an extension in Google Chrome store to obtain the application ID and provide callback URL for authorization.

In the evaluation, we upload pictures saved in JPG format to avoid compression by the OSN, and we change the settings to remove Weibo's watermark.

We measure the following performance metrics: time taken to encrypt pictures of different file sizes (as this affects performance), and the loading time of the encrypted picture.

### 4.2 Evaluation Result

Figure 6 shows the evaluation results of a picture saved in different resolutions and hence, file sizes. At a resolution of $3647 * 2736$, we need 20.424 s to encrypt the picture. However, we remark that users seldom upload such high resolution pictures to Weibo. If the picture is compressed to $2048 * 1536$, it only takes a third of the time to encrypt the picture. The encryption time required descreases exponentially with the picture resolution. For example, a picture with $1024*768$ pixels only requires 1.736s, which may not be noticed by the average user.

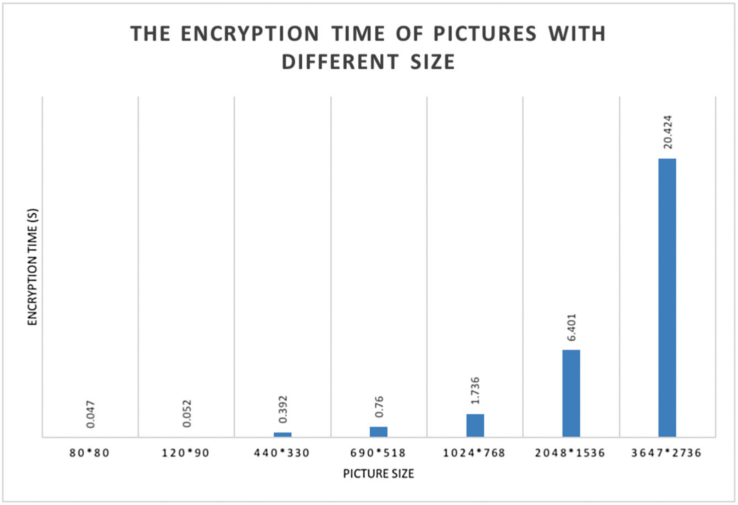

**Fig. 6.** The average encryption time of pictures with different size

When a webpage loads on the browser, it will firstly load the document object model (DOM, which assists front-end code to build the HTML) by parsing the respective element. It will then apply the CSS and execute the JavaScript code with the page load event, prior to rendering the webpage. Users can view the page contents, although the page may be incomplete during the page load stage. Figure 7 shows the average load time observed in our evaluations. As our extension will check the sharing list during the DOM loading stage, this process may slow down the DOM loading for up to 5 s, which will not adversely affect the average user's experience since users can continue to browse the webpage during this loading stage.

## 5 Conclusion

In this paper, we highlighted an understudied risk due to deletion delay of pictures from popular OSNs. We then presented the design of a novel privacy-preserving too to address the deletion delay issue. Based on the evaluation of our prototype (i.e. a Google Chrome plug-in extension), we demonstrated that our approach is practical and suitable for real-world deployment.

Future work will include improving the Google Chrome plug-in extension, building plug-ins for other popular browsers, and extending the support to more OSNs. We will also study the viability of other general approaches not restricted by API (i.e. only can be called several times per hour) to address deletion delay.

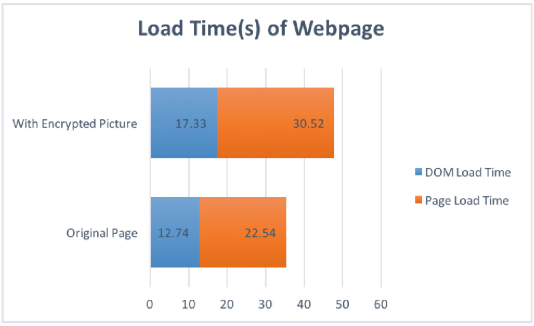

**Fig. 7.** The average load time comparison of original webpage and webpage with encrypted picture

**Acknowledgments.** Joseph K. Liu is supported by National Natural Science Foundation of China (61472083). Kaitai Liang is supported by privacy-aware retrieval and modelling of genomic data (PRIGENDA, No. 13283250), the Academy of Finland, Finland.

## References

1. Announcing the ADVANCED ENCRYPTION STANDARD (AES). No. 197 in Federal Information Processing Standards Publication, United States National Institute of Standards and Technology (NIST) (2001)
2. HTML5 Specification. World Wide Web Consortium (W3C) (2015)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-Policy Attribute-Based Encryption. In: 2007 IEEE Symposium on Security and Privacy (S&P 2007), pp. 321–334, Oakland, California, USA (2007)
4. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
5. Cheng, J.: Ars Technica: Three Years Later, Deleting Your Photos on Facebook Now Actually Works. http://arstechnica.com/business/2012/08/facebook-finally-changes-photo-deletion-policy-after-3-years-of-reporting
6. Digital Bazaar, I.: Forge: Javascript security and cryptography. http://digitalbazaar.com/forge/
7. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theor. **31**(4), 469–472 (1985)
8. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, pp. 89–98. ACM, Alexandria, VA, USA, October 30 - November 3, 2006
9. Leom, M.D., D'Orazio, C.J., Deegan, G., Choo, K.K.R.: Forensic collection and analysis of thumbnails in android. In: 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp. 1059–1066. IEEE Computer Society Press (2015)
10. Liang, K., Liu, J.K., Lu, R., Wong, D.S.: Privacy concerns for photo sharing in online social networks. IEEE Internet Comput. **19**(2), 58–63 (2015)

11. Quick, D., Tassone, C., Choo, K.R.: Forensic Analysis of Windows Thumbcache files. In: 20th Americas Conference on Information Systems, AMCIS 2014, Savannah, Georgia, USA, 7–9 August, 2014
12. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems (reprint). Commun. ACM **26**(1), 96–99 (1983)
13. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
14. Whittaker, Z.: ZDNet: Facebook Does Not Erase User-Deleted Content. http://www.zdnet.com/blog/igeneration/facebook-does-not-erase-user-deleted-content/4808
15. Zhu, Y.: Sina app: Weibo tuchuang (in Chinese). https://hk.v2ex.com/t/44453#reply149