Arcana: Enabling Private Posts on Public Microblog Platforms

Anirudh Narasimman[1], Qiaozhi Wang[2], Fengjun Li[2], Dongwon Lee[3], and Bo Luo[2]

[1] IBM Corporation, San Francisco, CA, USA
anirudh.narasimman1@ibm.com
[2] University of Kansas, Lawrence, KS, USA
{qzwang, fli, bluo}@ku.edu
[3] Penn State University, University Park, PA, USA
dlee@ist.psu.edu

Abstract. Many popular online social networks, such as Twitter, Tumblr, and Sina Weibo, adopt too simple privacy models to satisfy users' diverse needs for privacy protection. In platforms with no (i.e., completely open) or binary (i.e., "public" and "friends-only") access control, users cannot control the dissemination boundary of the content they share. For instance, on Twitter, tweets in "public" accounts are accessible to everyone including search engines, while tweets in "protected" accounts are visible to *all* the followers. In this work, we present *Arcana* to enable fine-grained access control for social network content sharing. In particular, we target the Twitter platform and introduce the "private tweet" function, which allows users to disseminate particular tweets to designated group(s) of followers. Arcana employs Ciphertext-Policy Attribute-based Encryption (CP-ABE) to implement social circle detection and private tweet encryption so that access-controlled tweets are only readable by designated recipients. To be stealthy, Arcana further embeds the protected content as digital watermarks in image tweets. We have implemented the Arcana prototype as a Chrome browser plug-in, and demonstrated its flexibility and effectiveness. Different from existing approaches that require trusted third-parties or additional server/broker/mediator, Arcana is light-weight and completely transparent to Twitter – all the communications, including key distribution and private tweet dissemination, are exchanged as Twitter messages. Therefore, with small API modifications, Arcana could be easily ported to other online social networking platforms to support fine-grained access control.

1 Introduction

Over the past decade, our communication and information sharing behaviors have been gradually and inevitably changed by online social networks (OSNs). The convenience of OSNs drives users to share their social, cultural, professional, and even personal content online. Some OSNs such as Facebook and Google+

provide access control functions for posting, which allow users to specify a dissemination boundary (e.g., public, friends, customized lists of friends, etc.) for each post. Unfortunately, quite a few popular OSNs, such as Twitter, Sina Weibo and Tumblr, provide no or very limited access control. For example, in Twitter, all "public tweets" are accessible to all Twitter users and search engines. The only privacy protection mechanism is to "protect" the entire account so that all tweets are only accessible to followers. Without fine-grained access control for tweets, it is impossible to specify a customized audience for each tweet.

The popularity of Twitter especially among the younger generation and its openness have introduced serious privacy concerns. Public tweets with private content are not only a treasure for stalkers and adversaries [36], but also may cause unexpected social incidents. For example, a student who posts a tweet complaining about a professor may only want to share it with her close friends. It can be embarrassed if her classmates see the post. Also, when one holds a birthday party with close friends but not coworkers, she may feel her party pictures would upset the uninvited and thus hesitates to post. In fact, these situations are not uncommon. Survey shows that users have target followers in their minds before posting [32]. But due to the absence of fine-grained privacy protection functions, users may decide to post with different accounts or not to post at all. The lack of usable privacy protection causes complication and frustration, which may eventually affect the overall usage of OSNs. Meanwhile, for users who decide to ignore the privacy expectations and post publicly for convenience, the absence of privacy protection may lead to significantly negative repercussions, such as private information leakage and damage of public image. For example, it is reported that college-admissions officers looked for applicants' activities on OSNs to check signs of bad behavior that may lead to bad publicity or a legal investigation [2]. Therefore, there are strong needs for a fine-grained privacy protection mechanism for open OSNs such as Twitter to balance the sharing and exposure of online social activities. In the rest of the paper, we use *private tweets* to denote OSN content that is intended to be shared with only a subset of followers instead of the public.

In this paper, we present *Arcana*, a fine-grained privacy enforcement mechanism for open social networks. We present our design and implementation based on Twitter, while the Arcana system can be ported to other social network platforms with small modifications on the interface with the OSN. Arcana allows users to assign followers into groups, and designate one or more groups of followers as the audience for each individual tweet. Such access-controlled tweets are first encrypted by Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [4, 33], and then embedded into a (random) host image as an invisible watermark [21]. Only followers in the designated groups could decrypt the message and see the plaintext tweet. The mechanism could be implemented as a browser plug-in, or a wrapper app for Twitter. All storage and communication are enabled by the native capabilities of Twitter. First, the keys are distributed over one-time messages using Twitter's Direct Message function. Secondly, the private tweets are disseminated as a regular (image) tweets through the standard tweet API.

Unlike other existing approaches [3, 12, 28], we do not need a trusted third-party, a private server, nor any other platform with storage or access control capabilities. With the "private tweet" function of Arcana, a user can conveniently send a private tweet to a designated subset of followers and maintain full control of it (i.e., she can delete it later). Lastly, Arcana converts the private tweet as an invisible watermark so that it seems like a regular tweet to the unintended users.

Our contributions are three-fold: (1) We design the *Arcana* system to enable private posts on a public microblogging platform without the assistance of a trusted third-party. (2) Arcana provides flexible, fine-grained attribute-based privacy protection so that a customized dissemination boundary could be specified for each private post. A key technology and innovation in Arcana is platform-independent, therefore, it could be adopted in any online social network platform. (3) We have implemented a first prototype of Arcana as a Chrome plug-in for Twitter and demonstrated its effectiveness.

2 Problem and Preliminaries

2.1 Problem Statement

It is observed that users often post private messages on Twitter [32], which are (implicitly) intended for a small group of friends or followers [24]. Such arbitrary subset of followers is known as a social circle, in which the message sender has complete control over the circle boundary [19, 30, 17, 34]. We define the tweets designated to pre-selected social circles as *private tweets*, which can be considered as a person-to-group communication. However, designed to be an open network, Twitter lacks effective support to private posting. By default, all the tweets are open to public (with the exception of a very coarse-grained "protected tweets" function). The only private communication mechanism on Twitter is "private message", which is a costly person-to-person communication between two users. To post a private message to a subset of N followers, N copies need to be sent. So, it is very laborious and expensive (in terms of communication and storage) to post to a large social circle or to many social circles at the same time. Therefore, it is important to develop a fine-grained privacy enforcement mechanism to enable private posting in open microblogging platforms.

Design goals. To tackle the problem, we present the Arcana system, which allows the users to disseminate private tweets to any arbitrary social circle. It provides fine-grained access control functions directly on top of the Twitter platform, without involving any other third party. In particular, Arcana is expected to provide the following key features: (1) *confidentiality*: only the intended followers are able to view the plaintext content of the private tweet; (2) *fine-grained*: any arbitrary dissemination boundary could be specified; (3) *self-sufficiency*: Arcana should not rely on any trusted third-party, external storage, mediation, or authentication platform other than Twitter.

2.2 Preliminary: CP-ABE

The Ciphertext-Policy Attribute-based Encryption (CP-ABE) scheme [4] allows a user to define dynamic access policies without knowing specific receivers in the system beforehand. In particular, each user is associated with a set of *attributes*, which are reflected in her secret key. The *access policies* are embedded in the ciphertext, which define the attributes that an authorized user should own. Therefore, one could specify an access structure over the set of attributes, and encrypt a message accordingly so that only the authorized users with attributes satisfying the access structure can decrypt it. In practice, access structures are described by monotonic access trees, whose leaves are the pre-defined attributes and non-leaf nodes are threshold gates. CP-ABE is resilient to collusion attacks. A message can be decrypted by a group of collusive users only when at least one of them could decrypt it on her own. In general, the CP-ABE scheme consists of four fundamental algorithms: Setup, KeyGen, Encrypt, and Decrypt [4]:

Setup: The setup algorithm generates a public key (a.k.a. the public parameters) PK and a master key MK. In particular, it chooses two randoms $\alpha, \beta \in Z_p$, a bilinear group G_0 of prime order p with a generator g, and the bilinear map $e : G_0 \times G_0 \rightarrow G_1$ to generate $PK = (G_0, g, h = g^\beta, e(g, g)^\alpha)$ and $MK = (\beta, g^\alpha)$.
KeyGen(MK, S): Assume a user has a set of attributes $S = S_1 \cup ... \cup S_m$, the key generation algorithm takes S and the master key MK as the input to generate the secret key SK, which is used to decrypt the messages designated for attributes S or a subset of S.
Encrypt(PK, $M, \mathbb{A}$): Message M is encrypted with public parameters PK and an access structure $\mathbb{A}$ defined over all possible attributes.
Decrypt(CT, SK): Ciphertext CT contains the access structure $\mathbb{A}$. The decrypt algorithm decrypts CT with SK if S satisfies $\mathbb{A}$.

Due to space limit, we do not include more details of the CP-ABE scheme but refer interested readers to [4, 18, 13, 35].

2.3 Preliminary: Digital Watermarking

Digital watermarking embeds information into a carrier signal such as images, videos, and text documents. Such watermarks are visually imperceptible in order to hide certain information. In Arcana, we adopt the Least Significant Bit (LSB) watermarking approach [7, 10] for its efficiency and simplicity. Besides LSB-based watermarking schemes that embed information into the spatial domain, other watermarking mechanisms insert hidden bits in frequency coefficients or the noisy blocks of the image. Please refer to [8] for more details of digital watermarks and other watermarking mechanisms.

In our work, an image is considered as an array of pixels, in which each pixel is represented by a numerical value, such as 8 bits for 256-gray-scale images, or 3×8 bits for 16M-color images. The LSB method substitutes the least significant bit of the carrier signal (the right-most bit of the pixel) with the bit value from the embedded message. For example, if a greyscale pixel is represented

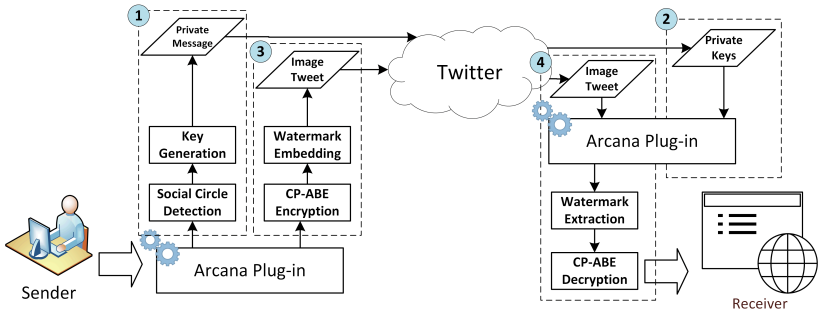


Fig. 1. Overview of the Arcana System.

as `0b00011100` and the information to be carried is `0b1`, the new pixel value becomes `0b00011101`. Ideally, we can insert 1 bit of watermark in each pixel of the image with no robustness consideration. Thus, with a common small-size image of 240×160 pixels, we can embed 4.8MB information, which is sufficient for common tweets (280 characters) and encrypted private tweets.

3 Arcana: Private Posts on an Open Microblog Platform

3.1 Arcana Overview

Arcana uses CP-ABE to encrypt tweets and invisible digital watermarking to embed ciphertext in host images. As shown in Figure 1, Arcana performs four main tasks: (1) The user (e.g., Alice) first invokes Arcana to categorize all her followers into social circles. In response, Arcana sets up the CP-ABE environment, generates cryptographic keys and distributes them to the followers through Twitter *direct messages*. (2) If a follower is assigned with at least one attribute by Alice, he will receive a private message which contains the cryptographic keys. (3) Alice sends the plaintext of the private tweet along with the designated social circle(s) to Arcana, which will encrypt the message, embed it into an image as an invisible watermark, and post to Twitter on behalf of Alice via an API call. (4) On the receiver side, Arcana detects if a received image in Bob's Twitter feed contains a private tweet watermark. If so, it extracts the binary watermark string (if exist), attempts to decrypt it with Bobs private key, and displays the recovered text together with the host image to Bob.

To be consistent with our prototype that implements Arcana as a Chrome extension, we illustrate Arcana as a browser plug-in in Figure 1. In fact, Arcana can be implemented as a Twitter client app for mobile devices, similar to Tweetbot or Twitterrific. Finally, it is worth noting that Arcana does not affect normal Twitter functions. If the sender uses Arcana but her follower does not, the follower cannot tell if Arcana is in use or view any private tweet.

3.2 Social Circle Detection and Key Distribution

To start using Arcana, the first step is to assign roles (i.e., attributes) to each follower for access control. However, manually assigning attributes to a large number of followers is tedious and labor-intensive. Therefore, automatic social circle detection is desired. As social circles have been studied recently to enable privacy protection [30], several automatic detection approaches have been proposed [23, 22]. In Arcana, we employ the SCSC multi-view clustering approach [22] to cluster the followers into non-overlapping groups and provide suggested social circles to the sender, who can manually refine the groups and assign an *attribute* to each social circle, such as *family*, *coworker*, *close friend*. Our scheme allows overlaps among social circles, e.g., a follower may be both *coworker* and *close friend*.

Arcana then invokes the Setup and Key Generation processes of CP-ABE to construct the cryptographic environment and generates the public, master and secret keys. In particular, a secret key SK_i is generated for follower f_i based on the attribute set S_i assigned by Alice. Arcana uses Twitter's *Direct Messages* function via an API call to distribute keys ($\{PK, SK_i\}$) encoded in Base91. It also embeds a note to indicate the key distribution message to Alice's followers. The follower stores the received keys locally, which are only used to decrypt private tweets. Since the keys are identified by the sender of the Direct Messages, they cannot be spoofed as long as Twitter is not compromised. Since Arcana does not store any personal information locally (in browser or on the client machine), Alice receives her cryptographic environment via Direct Message.

The key generation and distribution process is a one-time communication. Once the keys are distributed to the followers, they will be used to decrypt all future private tweets. If Alice wants to assign a new user to an existing circle, she needs to extract the master key MK from her own DM, and invoke **KeyGen(**MK**,** S**)** to generate and distribute a new secret key for this follower. It is worthy noting that this process does not affect any existing follower.

3.3 Posting a Private Tweet

To post a private tweet, the sender first submits the textual content of the tweet (plaintext M) to Arcana and specifies the access structure $\mathbb{A}$. Then, Arcana loads the cryptographic environment and invokes CP-ABE to encrypt this message as: $C = \textbf{Encrypt}(PK, M, \mathbb{A})$.

A naive approach would be directly converting the binary string C into a text string (e.g., using Base64 or Base91) and posting it to Twitter. However, this approach has two significant drawbacks: (1) the length of the CP-ABE ciphertext is significantly longer than the short plaintext C. For instance, the length of a Base91 encoded CP-ABE ciphertext of a regular tweet message ranges between 700 to 2000 characters, while each tweet is limited to 280 characters. Hence, directly posting this encoded ciphertext would take 3 to 8 tweets. (2) Although Base64 or Base91 encoded ciphertext are ASCII strings, they are not readable to users. Figure 2 (a) shows part of the Base91 encoded ciphertext for message

Algorithm 1: Posting a private tweet.

Data: Private message M; set of attributes **Attr**.
Result: Private tweet posted on Twitter.

```
1 begin
2     PK ⟵ loadKey()                          /* Load public parameters. */
3     𝔸 ⟵ CreatAccessStructure(Attr)
4     C ⟵ Encrypt(PK, M, 𝔸)                   /* CP-ABE encryption. */
5     C′ ⟵ strcat(C, MagicToken)
6     I ⟵ LoadImage()                         /* Load host image. */
7     I′ ⟵ Embed(I, C′)                       /* Embed watermark. */
8     TwitterAPI.post(I')
```

"Dinner at my place!" Posting a large number of tweets like this will create an awkward socialization situation. To tackle the usability problem, Arcana embeds the ciphertext into a host image as an invisible watermark, and posts the image to Twitter. Besides the ciphertext, Arcana also adds a special *Magic Token* (predefined binary string) to indicate the existence of Arcana watermark and mark the end point of the watermark. To a regular user, the encoded image looks no different from any other regular image.

The process for posting a private message is shown in Algorithm 1. First, the CP-ABE public parameters are loaded from Alice's own Direct Messages [L2]. Then, the CP-ABE access structure is created from user specified attributes [L3]. In [L4-5], the plaintext tweet is encrypted with the public parameters and the access structure, with a "Magic Token" appended to the end of the ciphertext. Next, the host image is loaded from the Arcana image library or uploaded by the user [L6]. Finally, the ciphertext is embedded in the host image and posted to Twitter [L7-L8].

3.4 Viewing Private Tweets

The decryption process contains three major steps: (1) loading private keys from Twitter Direct Messages, (2) filtering image tweets to identify and extract watermarks, and (3) decrypting and displaying the private tweet(s).

The decryption process is shown in Algorithm 2. In [L4], when a user Bob loads Twitter timeline, Arcana will scan through the Direct Messages to identify key distribution messages, and extract credentials and keys from such messages: $\{U_j, PK_j, SK_j\}$, where user ID U_j is is directly extracted from the sender field of the messages.Next, in [L6], each image tweet being loaded into the timeline is filtered by Arcana. We only scan candidate images posted by some users from whom we have received secret keys ($\mathbf{U} = \bigcup U_j$) [L7], and scan through the image until the binary Magic Token [L8]. Then, the corresponding keys are loaded and the watermark is extracted from the host image [L9-L10]. If the decryption process is a success, the plaintext of the private tweet is displayed together with the host image in the users browser [L11-L13].

Algorithm 2: Displaying a private tweet.

```
Data: Twitter timeline T.
Result: Plaintexts of private tweets displayed on Twitter.
1  begin
2      foreach dm in Twitter Direct Message do
3          if dm is Arcana key distribution then
4              {U_j, PK_j, SK_j} ⟵ ExtractKeys(dm)
5              j++
6      foreach Image tweet I in Twitter timeline do
7          if I.owner ∉ U then break
8          if VerifyMagicToken(I) then
9              {PK_j, SK_j} ⟵ LoadKey(U, I.owner)
10             C ⟵ ExtractWatermark(I, MagicToken)
11             M ⟵ Decrypt(PK_j, C, SK_j)
12             if M is text then
13                 Display(M)
```

3.5 Experiment and Performance Analysis

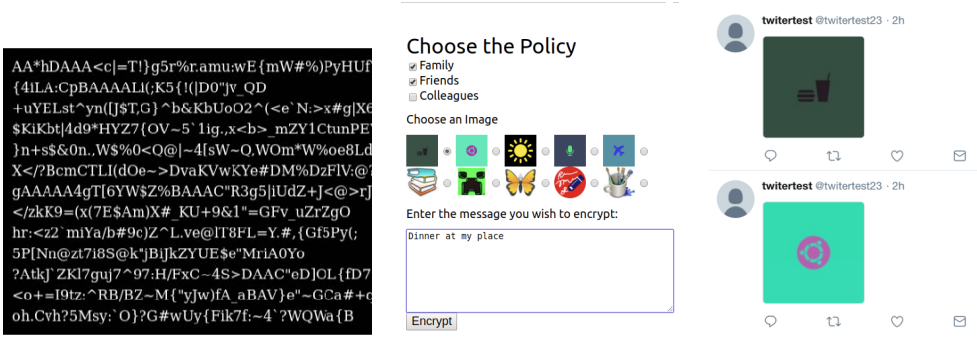
We have developed a proof-of-concept prototype of Arcana as a chrome extension. The interface for posting a private tweet is shown in Figure 2 (b).

In the example, the user has configured three attributes (social circles): *Family*, *Friends*, and *Colleagues*. To post a new private tweet, she first identifies the access policy by selecting one or more attributes, selects a host image from the library, and enters the text content of the private tweet. Figure 2 (c) shows the view of the private tweets from an unauthorized follower's perspective. As expected, the user who does not have any designated attribute only sees a regular image, but cannot tell it is a private tweet.

We then created a test user account "twittertest", who had three social circles, and posted three different messages from this account. Each of these private tweets are only intended for a subset of his followers, as follows:

Tweet message	Attributes (circles)
`Dinner at my place.`	Family, Friends
`New version of ubuntu is out.`	Colleagues, Friends
`We have a new ceo.`	Colleagues, Family

The test user "twittertest" has three followers: *Faye* is in the user's social circle "Family", *Frank* is in circle "Friends", and *Cole* is in "Colleagues". When these three followers log in to Twitter on browsers with Arcana, they see different private tweets based on their attributes. Figure 3 (a) shows part of Faye's twitter feed processed by Arcana. In this example, she can only see the messages distributed to "Family" i.e. "`Dinner at my place`" and "`We have a new ceo`". The second tweet cannot be decrypted by Faye's secret key, hence, no plaintext



(a) (b) (c)

Fig. 2. Private tweets: (a) Part of Base91 encoded CP-ABE ciphertext; (b) Screenshot in Arcana: posting a private tweet "Dinner at my place", which is only visible to Family and Friends; (c) View of the private tweets from an unauthorized follower.

is displayed with the image. Similarly Frank and Cole only see messages corresponding to their social circles, as shown in Figure 3 (b) and (c). This example has very few policies for the sake of convenience in demonstration, but it can be extended to add more host users, followers, attributes, and private tweets.

Efficiency and usability. In our experiments with a commodity Dell desktop, it takes approximately 2 seconds to encrypt and embed a 100-character tweet. As we have explained, private tweets could be very long, since a small image could carry relatively large amount of information. Encrypting and embedding a 1000-character message will take approximately 4 seconds. At the recipient's end, Arcana runs in the background, after the user logs in to Twitter from a browser with Arcana. The plaintext of private tweets are added to the Twitter feed on-the-fly. The decryption time is almost linear to the number of images. In our experiments, it took approximately 11 seconds to recover 20 private tweets. As Twitter loads tweets incrementally, Arcana only needs to examine and decrypt the top tweets, while more tweets are added on demand.

Arcana currently supports an "OR" (or "UNION") logic among the set of attributes. CP-ABE is capable of handling complicate logic of the attributes. For example, Alice could define that a private tweet is for followers who are (`Family` OR (`Friends` AND `Colleagues`)). It is relatively straightforward to extend Arcana's user interface to support complicate logic with operations such as AND OR and NOT.

4 Security Analysis and Discussions

Next, we present the threat model and examine Arcana's security guarantees.

Threat Model. We assume the Twitter platform is trusted and the data carried by Twitter private messages is secure. The Twitter users, however, can be honest-but-curious or malicious. So, we consider two types of intruders: (1) Eve is a

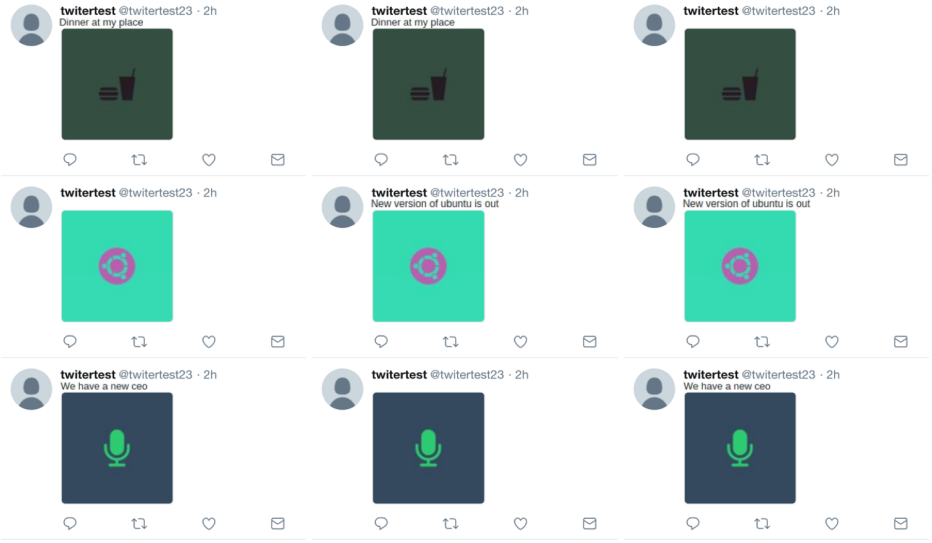


Fig. 3. Screenshots of Arcana: (a) Faye's view after decryption; (b) Frank's view after decryption; (c) Cole's view after decryption.

normal Twitter user (not necessarily Alice's friend) who attempts to learn the plaintext content of the private tweet messages that she is not authorized to read. (2) Chuck is a malicious user who attempts to impersonate Alice to send private tweets so that Alice's followers think it comes from Alice. We assume that neither Eve nor Chuck compromises other users' twitter accounts.

Key security. Cryptographic keys are generated in Arcana (in Chrome extension in our prototype), transmitted through Twitter's Direct Messages (DM), and stored in DM. In key generation, the memory copy of keys and key materials are destroyed once they are distributed. Key confidentiality during key transmission and storage relies on the confidentiality of Twitter DM, which is considered trustworthy. Keys in DM are considered secure, since the key distribution messages are rarely read by users, and shoulder surfers can hardly remember the keys, which appear like a long random strings. Keys are only accessible to Arcana when a user logs in to a Twitter account, hence, access to the keys relies on the access to the Twitter account. When keys are utilized to decrypt private tweets, they are erased after the user leaves Twitter. In summary, neither Eve nor Chuck is able to learn secret/master keys that are not distributed to them.
Data confidentiality. Data security in Arcana refers to the confidentiality of the private tweets, i.e., only authorized followers are allowed to view the plaintext of authorized tweets. In general, data security relies on five design features:

- Arcana binds credentials with Twitter accounts – anyone with access to Alice's Twitter account could access the keys generated by Alice or distributed

to Alice, and then access the corresponding private tweets. Security of credential binding (i.e., authentication of followers) relies on the authentication and account security in Twitter, which is considered trustworthy.

- Chuck may obtain the public parameters of Alice's key, and use it to encrypt messages. However, in Arcana, sender identity is bound to the Twitter account that posts the image. Hence, as long as Chuck does not compromise Alice's account, he cannot impersonate Alice to post the private tweet.
- The confidential dissemination of private tweets against eavesdroppers relies on the semantic security of CP-ABE cryptosystem. When Eve does not have the attributes and the corresponding secret key, CP-ABE guarantees that she could not learn any information from the encrypted private tweet.
- Arcana does not save any sensitive information on the client computer. Thus, getting access to the client computer does not lead Eve to the access of any Arcana keys/messages. However, a privileged user (root) is able to access or dump Arcana's memory content while Arcana is running and the keys are in memory. This is unavoidable for any web-based application.
- The security of Arcana is self-contained. Arcana does not use any third party service or trusted server for storage, authentication, key management/distribution, or encryption/decryption. Moreover, Arcana does not communicate to anyone except Twitter. That says, unlike other approaches in the literature, no third party is capable of accessing any sensitive information or private tweets.

Key Refreshment/Revocation. To remove member(s) from a group, we choose to generate and redistribute new private keys instead of revoking old private keys. The old keys have already been used to decrypt old private tweets by the members to-be-removed, hence, user's private information has already leaked to the users in old groups. Dynamic attributes and efficient key refreshment mechanisms are actively studied in cryptography community, new methods could be adopted in Arcana to enable efficient key revocation/refreshment [13, 35, 37].

Forwards and Replies. In Arcana, directly forwarding a private tweet would not compromise data confidentiality, since the encryption is not broken during forwarding. Unauthorized users cannot decrypt private content from the original image tweet or from a forwarded tweet. Meanwhile, if the image is altered, the watermark may be broken, hence, the sensitive information is destroyed, but never leaked. Currently, Arcana does not encrypt the replies to private tweets.

Side-channel. The *Magic Token* (Algorithm 1) may become a side-channel to expert followers who have intensive knowledge of Arcana. That is, if a follower manually examines the image and finds the Magic Token, he knows that he has been excluded from the audience of the private tweet. This may cause an issue in socialization. However, we would like to claim that: (1) it requires intensive knowledge of Arcana detect the side-channel, which is beyond the capability of regular Twitter users. (2) It is difficult, if not impossible, to completely avoid this side-channel. Arcana will need a mechanism to extract the watermark, which could always be invoked by the attacker to detect the existence of the watermark. And (3) this side-channel does not leak any sensitive information.

Arcana for other online social networking platforms. There exist other social networking platforms that do not have any access control functions, or only provide a coarse-grained privacy system, such as Tumblr or Sina Weibo. With limited modification of I/O to fit into the API of the specific OSN platform, Arcana could be deployed to work with almost any social networking platform that supports a private message function and an image posting function.

5 Related Works

Preserving user's privacy on OSNs has been intensively researched in recent years. The approach, frame and purpose of defending differ from case to case. In general, the literature of social network privacy research could be organized into two categories: *privacy models* and *privacy enforcement mechanisms*.

Privacy Models. Various privacy protection models have been proposed for social networks, such as [15, 6, 9]. There are also automatic tools to help users configure their privacy settings, such as *Privacy Wizards* [14], *A3P*[31]. Arcana is different from them that we introduce fine-grained privacy protection to open OSNs that employ very simple access models, such as Twitter or Tumblr.

Privacy Enforcement. Privacy enforcement can be classified into two categorizes: (1) *New designs of OSN architecture.* Privacy-aware OSNs protect users' privacy by storing private information on their own computers or distribute them across multiple administrative domains. Hence, they are decentralized, in stead of relying on one commercial OSN provider. There are also proposals of privacy-preserving SN architectures that add a layer of encryption outside the centralized server to prevent the service provider from peeking into users' information. This line of methods have been widely studied and there are several representative instances, such as Safebook[11], PeerSoN[5], Cachet[25], Hummingbird [12], and Twister[16]. However, their cost and availability disadvantages[27]are the major drawbacks. Detailed studies of decentralized privacy preserving OSNs are available at [1, 26]. (2) *Reforming commercial OSN platforms.* [3] allows users to share encrypted text, images and files. It relies on trusted third parties for key management and storage. Meanwhile, sharing an item to a group requires sharing group keys or publishing multiple ciphertexts encrypted with different keys. Twitsper[29] implements a wrapper for Twitter to allow users to share private whispers. It uses Twitter direct messages to transmit whispers, which means N copies of a whisper need to be sent in order to share it to N friends. Meanwhile, Twitsper employs an external supporting server to help manage whispers and replies. Twitterize[20] is an Android app that introduces anonymity and confidentiality for Twitter, through implementing an overlay network on top of Twitter, and using Android SQLite DB to store tweets and keys. Twitterize requires a secure channel for key distribution (QR code scanning or NFC), and significantly amount of local configuration and storage. It also posts Base64-encoded ciphertext on Twitter, which will be confusing to non-Twitterize users.

In summary, existing privacy-preserving social network approaches in the literature require (trusted) supporting servers (e.g., for authentication/authorization

[12], supporting group tweet functions [29], content storage [3]), a proprietary social network infrastructure (e.g., [6, 11]), or an additional secure channel (e.g., for key distribution in [3, 20]). The main novelty of our proposed Arcana lies in the fact that it does not need any of these.

6 Conclusion and Future Work

Arcana provides a private tweet function to enable fine-grained access control on Twitter. It allows the user to categorize followers into groups and post private tweets to selected groups. In its implementation, Arcana encrypts private tweets with CP-ABE, and embeds the ciphertexts as invisible watermarks in host images. Arcana is self-contained that all storage, authentication, key management, and encryption/decryption are handled locally or via Twitter. It does not need a trusted third party or an additional broker/mediator. We have implemented Arcana as a Chrome extension and discussed its privacy guarantees.

Arcana may be extended to enable fine-grained access control for any online social network platform. The extension requires modifications in the I/O interface to work with underlying SN platform, which is our future plan.

References

1. L. Bahri, B. Carminati, and E. Ferrari. Decentralized privacy preserving services for online social networks. *Online Social Networks and Media*, 6:18–25, 2018.
2. M. Bauerlein. Your dream college can see twitter, too. `https://www.bloomberg.com/view/articles/2013-11-08/your-dream-college-can-see-facebook-too`.
3. F. Beato, I. Ion, S. Čapkun, B. Preneel, and M. Langheinrich. For some eyes only: protecting online information sharing. In *ACM CODASPY*, 2013.
4. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*. IEEE, 2007.
5. S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta. Peerson: P2p social networking: early experiences and insights. In *ACM SNS Workshop*, 2009.
6. B. Carminati, E. Ferrari, and A. Perego. Enforcing access control in web-based social networks. *ACM TISSEC*, 13(1), 2009.
7. C.-K. Chan and L. Cheng. Hiding data in images by simple lsb substitution. *Pattern Recognition*, 37(3):469 – 474, 2004.
8. A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt. Digital image steganography: Survey and analysis of current methods. *Signal processing*, 90(3), 2010.
9. Y. Cheng, J. Park, and R. Sandhu. A user-to-user relationship-based access control model for online social networks. In *DBSec*, 2012.
10. I. J. Cox, M. L. Miller, J. A. Bloom, and C. Honsinger. *Digital watermarking*, volume 53. Springer, 2002.
11. L. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *IEEE Communications Magazine*, 2009.
12. E. De Cristofaro, C. Soriente, G. Tsudik, and A. Williams. Hummingbird: Privacy at the time of twitter. In *IEEE S&P*, 2012.
13. N. Doshi and D. Jinwala. Updating attribute in cp-abe: A new approach. *arXiv preprint arXiv:1208.5848*, 2012.

14. L. Fang and K. LeFevre. Privacy wizards for social networking sites. In *International World Wide Web conference(WWW)*, 2010.
15. P. W. Fong, M. Anwar, and Z. Zhao. A privacy preservation model for facebook-style social network systems. In *ESORICS*, 2009.
16. M. Freitas. Twister: the development of a peer-to-peer microblogging platform. *International Journal of Parallel, Emergent and Distributed Systems*, 31(1), 2016.
17. B. Gao and B. Berendt. Circles, posts and privacy in egocentric social networks: An exploratory visualization approach. In *IEEE/ACM ASONAM*, 2013.
18. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, 2006.
19. S. Kairam, M. Brzozowski, D. Huffaker, and E. Chi. Talking in circles: selective sharing in google+. In *ACM CHI*, 2012.
20. A. Karthick, D. Murali, A. Kumaraesan, and K. Vijayakumar. Twitterize: anonymous micro-blogging in computer systems and applications. *Advances in Natural and Applied Sciences*, 11(6 SI):96–103, 2017.
21. S. Katzenbeisser and F. Petitcolas. *Information hiding techniques for steganography and digital watermarking*. Artech house, 2000.
22. C. Lan, Y. Yang, X. Li, B. Luo, and J. Huan. Learning social circles in ego-networks based on multi-view network structure. *IEEE TKDE*, 29(8), 2017.
23. J. Leskovec and J. J. Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.
24. M. Madejski, M. Johnson, and S. M. Bellovin. The failure of online social network privacy settings. *Columbia University, Tech. Rep. CUCS-010-11*, 2011.
25. S. Nilizadeh, S. Jahid, P. Mittal, N. Borisov, and A. Kapadia. Cachet: a decentralized architecture for privacy preserving social networking with caching. In *ACM CoNEXT*, 2012.
26. T. Paul, A. Famulari, and T. Strufe. A survey on decentralized online social networks. *Computer Networks*, 75:437–452, 2014.
27. A. Shakimov, A. Varshavsky, L. P. Cox, and R. Cáceres. Privacy, cost, and availability tradeoffs in decentralized osns. In *ACM WOSN*, 2009.
28. I. Singh, M. Butkiewicz, H. V. Madhyastha, S. V. Krishnamurthy, and S. Addepalli. Enabling private conversations on twitter. In *ACSAC*, pages 409–418, 2012.
29. I. Singh, M. Butkiewicz, H. V. Madhyastha, S. V. Krishnamurthy, and S. Addepalli. Twitsper: Tweeting privately. *IEEE Security & Privacy*, 11(3):46–50, 2013.
30. A. Squicciarini, S. Karumanchi, D. Lin, and N. Desisto. Identifying hidden social circles for advanced privacy configuration. *Computers & Security*, 41:40–51, 2014.
31. A. C. Squicciarini, S. Sundareswaran, D. Lin, and J. Wede. A3p: adaptive policy prediction for shared images over popular content sharing sites. In *Proceedings of the 22nd ACM conference on Hypertext and hypermedia*, 2011.
32. J. Vitak, S. Blasiola, S. Patil, and E. Litt. Balancing audience and privacy tensions on social network sites. *Intl J. of Comm.*, 9, 2015.
33. B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography*. 2011.
34. J. Watson, A. Besmer, and H. R. Lipford. + your circles: sharing behavior on google+. In *SOUPS*. ACM, 2012.
35. Z. Xu and K. M. Martin. Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage. In *IEEE TrustCom*, 2012.
36. Y. Yang, J. Lutes, F. Li, B. Luo, and P. Liu. Stalking online: on user privacy in social networks. In *ACM CODASPY*, pages 37–48. ACM, 2012.
37. Y. Zhao, M. Ren, S. Jiang, G. Zhu, and H. Xiong. An efficient and revocable storage cp-abe scheme in the cloud computing. *Computing*, pages 1–25, 2018.