AnonyFacebook - Liking Facebook Posts Anonymously*

Pedro Alves[1] and Paulo Ferreira[2]

[1] Opensoft / INESC-ID / IST, Rua Joshua Benoliel, 1 - 3D, Lisboa, Portugal
pedro.h.alves@gmail.com
[2] INESC-ID / IST, Rua Alves Redol 9, Lisboa, Portugal
paulo.ferreira@inesc-id.pt

Abstract. In several countries the simple act of liking (on Facebook) an anti-government article or video can be (and has already been) used to pursue and detain activists. Given such a scenario, it is of great relevance to allow anyone to anonymously "like" any post.

In this paper we present anonyFacebook, a system that allows Facebook users to "like" a post (e.g., news, photo, video) without revealing their identity (even to the social network administrators). Obviously, such anonymous "likes" count to the total number of "likes". Anonymous "likes" are ensured by means of cryptographic techniques such as homomorphic encryption and shared threshold key pairs.

Keywords: social networks, privacy, anonymity, homomorphic encryption.

1 Introduction

Social Networks (SNs) allow users to connect each other at an unprecedented scale. They are primarily being used to share media and keep in touch with friends, family and colleagues; however, they are also used to raise awareness and coordinate large communities around important topics, such as the political status of some countries [4,1,6]. For example, Egyptian activist Wael Ghonim credited Facebook with the success of the Egyptian people's uprising, in particular for its key role in organizing the most important protest on January 25th.[1]

Wael Ghonim was arrested for 12 days, shortly after the protest.[2] Like Wael, many activists suffered from their activities on SNs as Egypt (and several other countries) have been reported to track down activists on SNs to the point where bloggers died while in custody for their anti-government articles.[3]

Thus, we developed AnonyFacebook, a system for Facebook in which users can promote/raise awareness to news, links, photos or videos (which we will refer

* This work was partially supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under projects Pest-OE/EEI/LA0021/2013 and PTDC/EIA-EIA/113993/2009.

[1] http://huff.to/14v9xBd

[2] http://huff.to/hWjthA

[3] http://readwrite.com/2011/04/12/bahraini_blogger_dies_in_custody

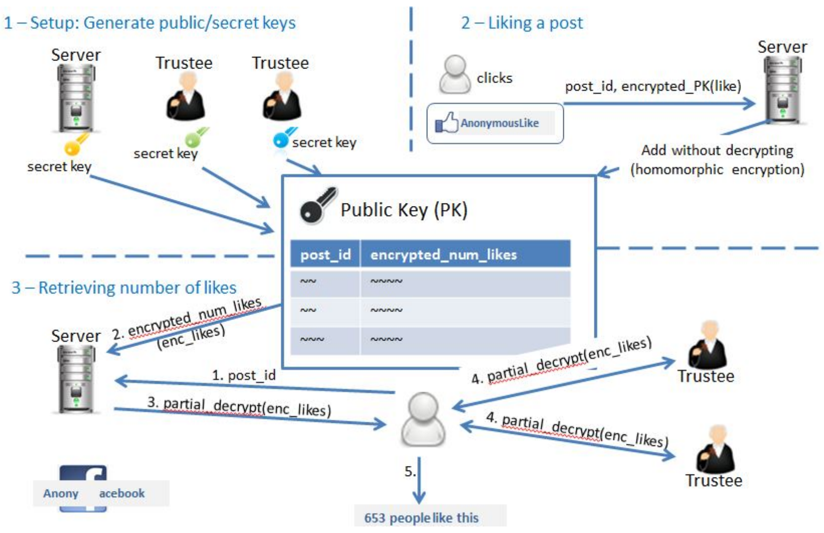
1 – Setup: Generate public/secret keys
Server
Trustee
Trustee
secret key
secret key
secret key
2 – Liking a post
clicks
AnonymousLike
post_id, encrypted_PK(like)
Server
Add without decrypting (homomorphic encryption)
Public Key (PK)
post_id
encrypted_num_likes
3 – Retrieving number of likes
Server
2. encrypted_num_likes (enc_likes)
1. post_id
3. partial_decrypt(enc_likes)
4. partial_decrypt(enc_likes)
Trustee
4. partial_decrypt(enc_likes)
Trustee
5.
AnonyFacebook
653 people like this

Fig. 1. AnonyFacebook global view

generically as posts) without revealing their identity. Such anonymity is ensured even to Facebook administrators. While implemented as a plug-in for Facebook, AnonyFacebook can be used in other similar SNs.

The interface supported by AnonyFacebook is similar to the one already existent in the original Facebook, and to which users are accustomed - a "like button", which can be embedded into a blog or news site, associated with a given post along with the number of "likes" already submitted by other users.

Despite such interface similarity, AnonyFacebook uses a completely different protocol than the original Facebook, whereby messages exchanged with the SN server are encrypted in such a way that:

- it is not possible to ascertain which post each user "liked";
- it is possible to know how many people "liked" a certain post.

These two aspects are ensured using a combination of homomorphic encryption [5] and shared threshold key pairs [3] as described in the next section.

2 Protocol

The global view of the AnonyFacebook protocol is illustrated in Figure 1: "likes" are encrypted with a key that is shared among the Facebook infrastructure (Facebook server, for short) and a set of independent entities (e.g., NGOs) so that none of these entities alone can decrypt it. By using a special property of the encryption algorithm, the Facebook server is able to sum the number of "likes" of a given post without having to decrypt them (i.e., the sum is also encrypted). When a user wants to know how many "likes" are associated with a given post, he coordinates all the entities to jointly decrypt the sum. We now present the details of the solution.

AnonyFacebook assumes a server S (hosted at facebook.com) and a set of Trustees T_i which can be public national institutes, NGOs, etc. Facebook clients exchange encrypted messages with S, and communicate with T_i to decrypt such messages (in particular, the number of "likes" of a given post).

The protocol has an initial setup process where a shared threshold key pair is generated across S and all T_i. This key will be published and used to encrypt all the "like" messages. Each entity (S and T_i) stores its part of the private key (also called shadow). To decrypt such "like" messages (or, being more precise, the result of operations on those messages), a certain number of T_i (depending on the threshold) must collaborate (i.e., Facebook server S alone is not be able to decrypt them).

After the setup stage, the system is ready to accept anonymous "like" messages. A "like" message in AnonyFacebook is a tuple $(post\_id, like)$, where:

- *post_id* is a unique identifier of the post, and
- *like* is an integer indicating if the user "liked" that post (value=1) or not (value=0).

On each one of these "like" messages, the *like* value is encrypted with the public key (generated in the initial setup phase) and the tuple is then sent to S. It's worthy to note that *post_id* has to be sent in cleartext to prevent duplicate "likes", as explained in the next paragraph. Thus, since the *post_id* is sent in cleartext, the AnonyFacebook client cannot send only the posts the user effectively "likes"; it has to also send posts the user didn't "like". Consequently, every time a user "likes" a given post, the client sends not only that *post_id* but also n random other *post_id*s, so that S is not able to know which exact post the user "liked". Therefore, S knows that the user "liked" one of those posts but is not sure which one.

Sending multiple *post_id*s each time the user "likes" a post is fundamental to prevent duplicate likes. Given that "likes" are encrypted, S can not know if the user already "liked" a given post. However, S knows that the user *potentially* "liked" a given post from those sent: every *post_id* sent to the Facebok server ("liked" or not "liked") is a potential "like" with probability $1/n$ (n is the number of *post_id*s sent for each "like").

To prevent duplicate "likes", S applies a probabilistic detection of multiple "likes" for the same post: it refuses *post_id*s that were already received, because there is a high probability that it is a "like" for a post the user previously "liked" (i.e. it is a duplicated "like"). Clearly, there may be false positives (i.e., S may refuse legitimate "likes") but the probability is low enough to guarantee the usability and usefulness of the system. Our estimates, based on Facebook numbers, show that (even being conservative) this probability is less than one collision per year for Facebook users; i.e., of all the "likes" the user does in Facebook during an year, one of them won't be successful.

The AnonyFacebook protocol allows users to see the number of "likes" of a given post, without knowing who did each individual "like". This functionality relies on the additive homomorphic properties of an ElGamal variant known as exponential ElGamal [2]. Using exponential ElGamal, it is possible to add two encrypted ("like") messages, without having to decrypt them first; in addition, the decryption of the encrypted sum equals the sum of the decrypted values. Thus, S applies this property by adding each encrypted "like" (note that the "like" value is either 1 or 0) with the already existent encrypted sum of that *post_id* (or zero if it is the first). When an AnonyFacebook user retrieves the number of "likes" of a given post, it is actually retrieving the encrypted sum associated with that post. Then, the client coordinates with the necessary number of Trustees (based on a pre-defined threshold) in order to decrypt the previously mentioned sum, for which each Thrustee uses its part of the private key.

3 Conclusion

In this paper, we propose AnonyFacebook, a SNA similar to Facebook that allows users to anonymously "like" any post; in other words, users can provide quantitative feedback without revealing their identity, even to the social network infrastructure itself. AnonyFacebook employs strong cryptographic techniques (homomorphic encryption, shared threshold key pairs) to guarantee users privacy. In particular, the AnonyFacebook server does not store individual "likes"; it does store the count of "likes" on any given post. Moreover, count of "likes" is encrypted in such a way that several entities (trustees) must collaborate to decrypt it (i.e., it is not possible for a single entity to decrypt it).

AnonyFacebook prevents duplicated "likes" using a *probabilistic* approach without breaking user's privacy. This is done by mixing the real "like" with several fake "likes": the client sends several post_ids for which the AnonyFacebook server does not know if they have been "liked" or not. This solution can wrongly prevent a legitimate "like"; however, as we have shown, this happens only once per year on average. So, we believe that our solution effectively prevents duplicated "likes" while achieving an acceptable tradeoff between privacy and usability.

We have implemented AnonyFacebook (within a Facebook replica), using an interaction model very similar to Facebook: developers can easily embed an anonymous "Like" button next to any content (blog post, video, etc.). The same mechanism also displays the current number of "likes" of a post. This implementation is publicly available (https://bitbucket.org/anonymousJoe/anonylikes); it can be used by any developer wishing to support privacy-preserving quantitative feedback in SNAs.

AnonyFacebook performance has been evaluated. We found that even though some encryption operations are computing intensive, the system can be implemented today without breaking user expectations (regarding the response time of client software) and usability for this kind of applications. In addition, since the performance of such encryption operations are mostly tied to CPU speed, it will tend to improve in the upcoming years with advances in processor technology.

References

1. Al-Ani, B., Mark, G., Chung, J., Jones, J.: The Egyptian blogosphere: a counter-narrative of the revolution. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (2012)

2. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. European Transactions on Telecommunications 8(5), 481–490 (1997)
3. Desmedt, Y.G., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
4. Lotan, G., Graeff, E., Ananny, M., Gaffney, D., Pearce, I., Boyd, D.: The Revolutions Were Tweeted: Information Flows During the 2011 Tunisian and Egyptian Revolutions. International Journal of Communication 5, 1375–1406 (2011)
5. Rivest, R.: On data banks and privacy homomorphisms. Foundations of Secure Computation 4(11) (1978)
6. Wulf, V., Misaki, K., Atam, M.: On the ground'in Sidi Bouzid: investigating social media use during the tunisian revolution. In: Proceedings of the 2013 ACM Conference on Computer Supported Cooperative Work, pp. 1409–1418 (2013)