Why Can't Online Social Networks Encrypt?

Ero Balsa*, Filipe Beato*, and Seda Gürses†
* KU Leuven ESAT/COSIC, iMinds, Leuven, Belgium
†NYU, USA

ABSTRACT

Online Social Networks (OSNs) are popular communication channels that reach a diverse strate of society. At the same time, their architecture is based on centralization of large amounts of user information, which poses several privacy risks, due to breaches, leakages and increasing normalization of surveillance programs. Despite recent breaches into their systems, OSNs providers are reluctant to offer E2EE services to their users hiding behind claims such as complexity of use and implementation challenges. We argue that these claims are not justified. We consider three different possible roles OSN providers can play in providing E2EE for large numbers of users using their platforms. We sketch different advantages and challenges posed by each of the models and consider associated threats. We conclude with a discussion on whether we would want OSNs to play this role and some open questions.

1. INTRODUCTION

Online Social Networks (OSNs), such as Facebook, Google+, and Twitter have become prominent communication channels for millions of users. Most OSNs provide their user base efficient and reliable channels to easily distribute and share information. Given their prominent role and their design, OSN providers end up centralizing the collection and processing of large amounts of data, insufficiently protected with coarse-grained privacy settings. This prompts several relevant privacy issues enumerated by many studies [1] including unwanted disclosure or use of "private messaging" features of OSNs [14, 21]. Reports of mass breaches of large data sets of personal information are increasingly common, as evident in recent accounts of the NSA surveillance programs like Prism [23] and the recent iCloud Breach [17].

All these worrisome issues motivate the need to (at least) implement existing cryptographic techniques to protect user's privacy in OSNs, such as end-to-end encryption (E2EE). By allowing users to encrypt private messages and select posts, OSNs can step up to their responsibility to protect user data. E2EE on OSNs would ensure that, especially for those posts that users want to keep confidential from a greater public, OSNs practices do not lead to unintended disclosures and privacy breaches. It would also allow users to have a more granular choice in using OSNs, as they would be able to select which posts they do not want to disclose to the OSN itself.

However, most OSNs have been reluctant to even protect communication on the wire, e.g., by turning on TLS. Moreover, they have actively argued against integrating E2EE claiming it is too complex to implement and cumbersome to use [20].

These claims against E2EE seem at best to be strategic. For example, in the same breath, Facebook representatives argued that they want to remain accessible to governments through the "front door" [20]. The resources committed to and the speed at which security features get implemented also suggest that these features are not a priority. Further, we are not convinced that the complexity of implementing E2EE is a viable excuse for OSNs not to implement E2EE, neither do we think it is that complex. Rather, given the series of news about the inability of various service providers to secure their databases, it should be part of their responsibility to also support users in taking additional measures to protect their information. In addition, while not all users may have the necessary skills, interest or time to use E2EE, this is not an argument against providing E2EE to those users who do. In fact, encryption savvy users may have a positive role in popularizing the feature, a practice typically used to go viral with new features on OSNs. In fact, OSNs are in a prime position to seize the momentum for E2EE and join efforts to make E2EE more accessible to the masses.

In this light, we explore three models through which OSN providers could proactively enable the integration of E2EE into their services. For each model, we sketch the responsibilities and challenges for an OSN like Facebook to implement E2EE. In addition, we sketch a preliminary threat analysis, asking ourselves how the advantage of an OSN offering E2EE may transform the threat model of E2EE as envisaged for asynchronous federated systems like email. Finally, we conclude with a discussion on whether, given these threat models, it would be desirable to have a centralized OSN like Facebook implementing E2EE.

We believe a discussion on implementing E2EE is important and yet limited. While end-to-end encryption à la PGP is desirable, there are many known problems with it, e.g., establishing a web of trust. Further, PGP based models only offer authentication and encryption of communications,

This work was supported by the Research Council KU Leuven GOA TENSE (GOA/11/007), in addition to the Flemish Government, IWT SBO SPION and IWT SBO MobCom, FWO G.0360.11N Location Privacy and FWO G.068611N Data mining. At NYU it was supported by Intel Science and Technology Center for Social Computing.

while many other properties such as perfect forward secrecy, deniability and concealment of communication partners are valuable too. We are also well aware that the source of the problem is the centralization of these services that function as walled gardens, monopolize said services and attract malicious parties. Encrypting data is one of a slew of mechanisms like decentralization, free access to source code, interoperability with other web standards, democratic governance, a commitment to user privacy and autonomy, community protection that have been proposed to address the multiplicity of problems associated with such centralized services. Yet, as we develop OSNs based on those mechanisms, we believe it is necessary to demand from centralized OSNs to be more responsible hosts of the data they hold in their large databases. Part of this can be accomplished through providing users the opportunity to negotiate their trust in these OSNs or distribute their trust through end-to-end encryption.

2. END-TO-END ENCRYPTION ON OSNS

End-to-end encryption (E2EE) is a (tele)communication paradigm that requires uninterrupted encryption between two parties, such that the encrypted data sent by the originator is only accessible by intended recipients.

Currently, OSNs implement HTTPS to protect the communication from client (user) to the server (OSN). However, with the OSN in the middle, any exchange between two communication partners Alice and Bob is available in clear text to the OSN. Hence, while HTTPS is useful against third parties eavesdropping on the communication between a user and the OSN, it does not protect against the OSN and attacks on the OSN.

Implementing E2EE on OSNs should ensure that only intended recipients are able to decrypt the messages. A successful implementation of E2EE provides confidentiality, authenticity and protection against man-in-the-middle attacks.

Since OSNs hold a strong identity base and demand users to log in to use their services, we do not consider *anonymity*, *(plausible) deniability* and *forgeability* as feasible or desirable properties in centralized OSN. Note that *pseudonimity* goes against Facebook's real name policy [24] but is technically possible. Yet, persistent use of the pseudonymous identities, together with the application of graph analysis techniques, is likely to lead to re-identification [**?**], especially in the absence of strategies like obfuscation.

Currently Facebook stores private messages indefinitely —not even users are allowed to delete them— so the whole history of previous messages is available. Ideally, the users should have the choice to decide on the availability of their private messages on an OSN.

For security reasons, an E2EE protocol would provide *perfect forward secrecy*. The question is what would be a reasonable design to allow users to use perfect forward secrecy, and yet be able to negotiate (expectations of) longer term availability of some messages. Would users be able to adapt to default non-availability with the option of selecting for availability, e.g., as in applications like SnapChat? For the sake of simplicity, we do not consider perfect forward secrecy here, but leave it as an open discussion point.

Lastly, as any E2EE implementation ultimately depends on the OSN to ensure the availability of the encrypted messages, the OSN can, at any given moment launch a denial-of-service attack. We hope that most OSNs have business incentives that strongly weigh against performing such attacks. Nevertheless, by virtue of being a centralized service, OSNs may be vulnerable to such attacks devised by other adversaries, e.g., by governments. Depending on the type of attack, different mitigation strategies may be considered, e.g., censorship circumvention, mirroring, which are important beyond the scope of this paper.

3. MODELS FOR E2EE ON OSNS

We propose and analyze three different models of how E2EE can be implemented on an OSN. Various models of secure messaging exist [22] and were presented at the first EFF CUP [9]. Our intention at the W3C workshop focuses on necessary means to integrate E2EE into or around a centralized OSN.

The models are focused on matters of user trust towards the provider that enables the encryption services, be it a centralized OSN or a third party encryption service provider. We examine the role of these parties, the technical challenges we foresee in implementing E2EE, and provide a preliminary sketch of the resulting threat model.

3.1 OSN as PKI

Description. The OSN is a trusted central authority, providing users with their public-private key pair or the means to generate them,[1] e.g., a client-side tool. The public key is linked to the user profile, certified and stored by the OSN, whereas the private key is managed and stored by the user. The OSN provides E2EE either integrated on the site or using client side scripting. This minimizes the complexity of key management and its impact on user's experience.

Challenges and Responsibilities Users may lose their private keys or access the OSN from different devices. In order to cater to the needs of users in such instances, the OSN may want to provide key recovery, backup and synchronization mechanisms, e.g., providing a public-private key pair per device, similarly to iMessage [16]. The OSN may also implement an Identity-Based Encryption (IBE) system using the users IDs as public keys, thus becoming a private key generator [7].

Possible Threats. Despite the convenience of this model with respect to key management, the OSN remains the main trusted party. As IBE private key generator, the OSN is able to decrypt all private messages. If the OSN secures the keys well, this could protect user data against third parties and attackers, yet the keys become more attractive to attackers. If the OSN only provides a key generation tool on the client side, it may introduce backdoors. In both versions of this model, the OSN may be compelled by law to retrieve the private key. In addition, as certifier of public keys, the OSN can certify its own keys and use them to impersonate a user, launching an effective and hard-to-detect man-in-the-middle attack.

3.2 OSN as Federated ID-Based PKG

Description. The OSN is part of a federated system formed by multiple, independent, entities. Using multiple trusted authorities IBE [10], each entity provides, upon user request

[1]Note that two different key pairs are needed: one for encryption and one for signing. For the sake of simplicity, we mainly refer in this paper to the encryption key pair.

and based on a public key chosen by the user (e.g., her OSN ID), a *share* of an ID-based secret key. The private key is constructed with the combination of t out of n shares, which the user can do client-side. The ultimate goal is that none (or a partial coalition) of the providers alone is able to retrieve the key [6].

This model relies on the convenient key management enabled by IBE while preventing the OSN from, on the one hand, being able to retrieve the private key and, on the other hand, impersonating users as easily (as public keys can be certified by different authorities and are "human-readable").

Challenges and Responsibilities. User-friendly solutions should not require or hold users responsible for the *manual selection* of the entities to be entrusted with the shares of the secret key. Methods need to be developed to support the user in this process. Also, the legitimate user should be required to *authenticate* with each of these entities so that only she is able to collect the shares of her secret key. A possible solution to both challenges is to provide, within the end-to-end encryption tool, a series of *connect* buttons to several popular service providers that the user may have an account with. Then, users connect with each of these providers to obtain a share of their secret key. A key requirement is that these providers should operate under different jurisdictions to avoid coercion from a government to reveal their shares, e.g., Twitter (US), Spotify (Sweden/UK), Shazam (UK) or SoundCloud (Germany), Privalia (Spain). In contrast to classic public key infrastructure, if a public key in IBE is revoked, the user would no longer be able to use that identifier for encryption, e.g., the Facebook ID. To support revocation an expiration date can be concatenated to the identifier [7].

Possible Threats. Although trust on the OSN is minimized by relying on other authorities, this model works under the assumption that t authorities will not collude. A security analysis of this model should consider key factors such as the number n of different entities which hold shares of the private key, the incentives of these entities to collude and the existence of an entity that may coerce these entities to disclose their shares, e.g., a government or supranational entity. An analysis of the security provided by a trans-jurisdictional distribution of key shares would ideally inform the trust model of the share distribution.

3.3 OSN as Supporter

Description. The OSN supports third-party initiatives. Although this represents Facebook's current and official position [20], it is unclear what specifically Facebook does to support E2EE provided by third-parties.

Third party tools that provide E2EE encryption for OSNs are typically browser plug-ins conceived or developed by researchers and other independent developers [5, 13, 15, 18]. Such plug-ins must be developed and maintained for different browsers, ideally both for desktop and mobile. Developers of these tools typically rely on limited time and resources to ensure a constant and smooth maintenance of these tools, and keeping up with constant browser and OSN updates is hard. Therefore, the OSN must push changes on the site with care, transparently, trying to minimize as much as possible the impact on the operation of these plug-ins.

A key element in the deployment of third-party plug-ins is the parsing of the OSN site. These plugins need to capture each HTML page before it is displayed to the user so that the encrypted content can be decrypted and replaced by its corresponding cleartext. The OSN can greatly facilitate this task by providing parsing functionalities through an API. An API provides a stable, black-box way of parsing the site, that developers can rely on independently from changes on the HTML.

The OSN can also help advertise and boost the visibility of these plug-ins. Similarly to previous Facebook trials of new services [26], the OSN can recruit a subset of users for early testing.

Challenges and Responsibilities. The weaker involvement of the OSN presents additional challenges, amongst which PGP-like key management is specially sensitive. Third party tools need to provide a mechanism to allow users to verify the authenticity of public keys. Expecting non-tech-savvy users to import public keys from a key server and perform offline fingerprint verification is illogical and unrealistic [25]. PGP's web of trust is also an inadequate mechanism when it comes to the general Internet user [27], as well as plainly insecure [25]. Thus, the tool should deal with key verification, loss and revocation effectively. Verification can be performed in a similar fashion as Google's recent proposal for E2EE for their email, i.e., using PGP along with the concept of Certificate Transparency for key verification [**?**]. With the same aim, others proposed the use of DNSChain [**?**].

User adoption represents another challenge. Whereas the OSN can seamlessly integrate the tool in the site and even turn it on by default, third-party plug-ins must be found and installed by the users. Users may not be aware of the existence of these plugins or realise that end-to-end encryption tools provide better protection than what is currently offered by Facebook [4]. Also, a user depends on her friends to install the tool, too, which may not be trivial. Third party tools thus ultimately depend on users being able to find them, understand what these tools provide and, socialize their use.

If different third parties offer different E2EE tools, *interoperability* is also an issue. This could be tackled by agreeing on an open standard that all developers can base their tools on. Otherwise, the landscape of tools available may be so fragmented that users require several tools to communicate with their OSN friends [3].

Possible Threats. The involvement of the OSN ranges from doing nothing whatsoever to supporting the third party tools by implementing certain changes on their site, e.g., an API for page parsing. Even if its involvement is minimal, the OSN still has the potential to hamper the adoption of E2EE in several ways. It can compromise both the integrity and the availability of the encrypted messages in the network, simply by modifying or removing cyphertext from their servers. Also, if the OSN serves as trusted public key directory, they can attempt to impersonate users with a man-in-the-middle attack. If fingerprints are not checked out of band, this attack cannot be detected. A separate threat model similar to that of PGP with email would apply to the third party providers.

4. DISCUSSION & CONCLUSION

The role that an OSN may play on the implementation of E2EE ranges from trusted third party (when it provides E2EE and acts as PKI) to directory and storage service (when it merely supports third party tools). From a security

and privacy perspective, the first model is unacceptable, as the OSN provider can both impersonate and decrypt messages. This model could spread a false sense of security among users and would be deceptive towards the public if it would be called E2EE.

If the OSN or a third party provides a tool that generates the keys at the client-side, the code must be open source. Otherwise, the user has to now trust the OSN application generating the private keys and is as susceptible as the earlier model. If the preferred model is one in which the OSN also provides the E2EE application and it is open source, then we would like to have a discussion on how the oversight of this code will be organized and sustained throughout time and what happens if this mode of oversight fails?

From a security engineering perspective, the less trust users need to place on the OSN, the better. Yet the users actual perception may be different. A recent study [4] showed that many users may be unaware of the security advantages of open source code and independent developers, and that they would more willingly trust a powerful and popular entity such as Facebook. This may play a significant role in the adoption of third party plugins, as users may not trust a *random bunch* of developers. The participants of the study further argued that even if these parties had good intentions, they believed a powerful OSN provider such as Facebook is more likely to be able to guarantee strong and robust security mechanisms. Besides, the users stated, no third party tool is likely to feature the same degree of integration and reliability on the site.

We infer from the user study that OSNs can play an important role in "normalizing" E2EE by integrating it into their main sites, making it easy to use, and assisting users' with their mistakes [27]. If we want to make E2EE popular, one way would be to make the protocols more convenient for the "general" Internet user. For users not acquianted with PGP like encryption, whose mental models are limited to Facebook's convenience oriented UX, this project may require loosening the classical E2EE model. For example, in order to match users' existing mental models, users should be able to authenticate their communication partners in an intuitive, fast and reliable way. This process should be similar to verifying whether the people that send them friend requests on Facebook are who they claim to be. But even if these users may carefully verify the identity of the people they communicate in the OSN, it will be a challenge to design a simple mechanism with which users can recognize man-in-the-middle attacks.

To ensure E2EE, the OSN provider should not have access to users private keys and users should remain the sole entity to access and control their private keys. However, most users are not accustomed to handling private keys. Users may lose their keys, e.g., by losing the device where the keys are stored. Further, installing a plug-in on different devices may require them to import the keys generated upon first use. Saving and importing the private key can be done by automatically encrypting and uploading the key to the OSN (or a storage service) encrypted with a password. However, users typically choose insecure passwords [2] and tend to forget them [11], often suffering no consequences (as OSN users have been taught to take password recovery for granted). Another alternative would be to consider protocols that support multiple keys. But even with multiple keys, users would not be able to access content with newly generated keys if there is no server in the middle [16].

In order to address problems with passwords, secure password managers may have to be integrated to manage (multiple) keys across devices [12]. In any case, a precondition of any of these models would be to have developers and manufacturers dedicating resources to developing mechanisms for storing and importing private keys securely on mobile devices.

Further, data recovery may be implemented independently of key recovery. Most of the data uploaded to social networks is *shared* with others. Therefore, even if a user forgets or loses her password/private key, friends' copies of these data can be considered as way to recover these data. Tracking issues with such data recovery models would also have to be considered, if OSNs would play a role in implementing such a feature. Finally, if E2EE is not the default, users should be made perfectly aware of when E2EE is *on* or *off*.

If we would take an optimistic position, we could argue that not only can OSNs technically support the integration of E2EE to their platforms, but they could also promote the use of E2EE and contribute to its normalization among a greater set of users. In order to reach this goal, the "classical" model of the E2EE UX can be loosened to better match the existing UX of users on OSNs, e.g., by stating that keys can be stored on multiple devices some of which have bad security track records, by storing private keys in an encrypted repository, or introducing mechanisms to reover encrypted data from other users. None of these solutions should however come at the price of giving access to private keys in the clear to an OSN or other third party.

If we would take a skeptical position, we could say that there is no good way to solve the E2EE problem that balances convenience with security in a way that does not require greater trust in the centralized OSN. Even if the OSN is only in a supporter role, should, for example, OSNs promote certain third-party E2EE applications over others? If so, according to which criteria? And, wouldn't OSNs heightened role in providing E2EE on OSNs contradict our original goal to provide users with the ability to negotiate trust with OSNs? We are also not sure if, given the convenience vs. security argument, and the communication culture based on the UX of OSNs, E2EE can be implemented without compromising the aspired security properties. In fact, given the way in which threat models change with a centralized OSN in the middle, decentralization of OSNs may have to be a precondition for implementing E2EE with the objective of distributing trust and better protecting user data in OSNs.

To further substantiate both perspectives, and to discover other viewpoints, is what we hope to do in the future. While we have not covered all possible models of integrating E2EE into centralized OSNs and our threat models are incomplete sketches, we hope that this paper will allow us to discuss more systematically the matter at hand.

5. REFERENCES

[1] A. Acquisti, B. Van Alsenoy, E. Balsa, B. Berendt, D. Clarke, C. Diaz, B. Gao, S. Gurses, A. Kuczerawy, J. Pierson, F. Piessens, R. Sayaf, T. Schellens, F. Stutzman, E. Vanderhoven, and R. De Wolf. The SPION Project. `http://bit.ly/1aiefXs`.

[2] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[3] Chris Ballinger. Fixing the xmpp push problem. April 08, 2014. `https://chatsecure.org/blog/`, Accessed Oct. 31, 2014.

[4] Ero Balsa, Laura Brandimarte, Alessandro Acquisti, Claudia Diaz, and Seda Gürses. Spiny cactos: Osn users attitudes and perceptions towards cryptographic access control tools. In *USEC*, page 10, 2014.

[5] Filipe Beato, Markulf Kohlweiss, and Karel Wouters. Scramble! your social network data. In *PETs*, pages 211–225. Springer, 2011.

[6] Filipe Beato, Stijn Meul, and Bart Preneel. Pratical Identity Based Encryption for Online Social Networks. Cosic internal report, 2014.

[7] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *CRYPTO 2001*, pages 213–229. Springer, 2001.

[8] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3), 2003.

[9] Joe Bonneau. A recap of the first eff cup workshop. August 13, 2014 =http://bit.ly/1xIVLIf, Accessed Oct. 31, 2014.

[10] Liqun Chen, Keith Harrison, David Soldera, and Nigel P Smart. Applications of multiple trust authorities in pairing based cryptosystems. In *Infrastructure Security*, pages 260–275. Springer, 2002.

[11] Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, pages 657–666. ACM, 2007.

[12] Paolo Gasti and Kasper Bonne Rasmussen. On the security of password manager database formats. In *ESORICS 2012*, 2012.

[13] Saikat Guha, Kevin Tang, and Paul Francis. Noyb: Privacy in online social networks. In *Proceedings of the first workshop on Online social networks*, pages 49–54. ACM, 2008.

[14] Xeni Jardin. US orders twitter to hand over account data on wikileaks and multiple wikileaks supporters. January 7, 2011 `http://bit.ly/1357EyV` Accessed Oct. 27, 2014.

[15] Nadim Kobeissi and Arlo Breault. Cryptocat: Adopting accessibility and ease of use as security properties. *arXiv preprint arXiv:1306.5156*, 2013.

[16] Greg Kumparak. Apple explains exactly how secure imessage really is. February 27, 2014 `http://tcrn.ch/1mJq6VZ` Accessed Oct. 27, 2014.

[17] Dave Lewis. icloud data breach: Hacking and celebrity photos. Sept. 2, 2014 `http://onforb.es/1Cmngvl`, Accessed Oct. 6, 2014.

[18] Wanying Luo, Qi Xie, and Urs Hengartner. Facecloak: An architecture for user privacy on social networking sites. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 3, pages 26–33. IEEE, 2009.

[19] Zach Miners. End-to-end encryption needs to be easier for users before facebook embraces it. Mar. 19, 2014 `http://bit.ly/1iCBi2i`, Accessed Oct. 27, 2014.

[20] Zach Miners. End-to-end encryption needs to be easier for users before facebook embraces it, March 2014. [Online at pcworld.com].

[21] Will Oremus. Facebook sued for "reading" your private messages. Jan. 3, 2014, `http://slate.me/1evQXN1`, Accessed Oct. 27, 2014.

[22] Trevor Perrin. Recent developments in secure messaging and other person to person communication. July 9, 2014 `https://www.eff.org/files/SOUPS.pdf`, Accessed Oct. 31, 2014.

[23] Washington Post. NSA slides explain the PRISM data-collection program. June 6, 2013 `http://wapo.st/J2gkLY`. Accessed Sept. 6, 2013.

[24] Facebook Account Settings. What names are allowed on Facebook? At `https://www.facebook.com/help/112146705538576`, Accessed Oct. 29, 2014.

[25] Secure Share. 15 reasons not to start using PGP. `http://secushare.org/PGP`, Accessed Oct. 28, 2014.

[26] Nick Summers. Facebook trials tweaked single-column Timeline design and new 'Like PageâĂŹ button in New Zealand. Feb. 28, 2013 `http://tnw.co/1yGTiPF`, Accessed Oct. 28, 2014.

[27] Alma Whitten and J Doug Tygar. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *Usenix Security*, volume 1999, 1999.