flyByNight: Mitigating the Privacy Risks of Social Networking

Matthew M. Lucas
matthew.lucas@alumni.illinois.edu

Nikita Borisov
nikita@illinois.edu

Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
1406 W. Green Street
Urbana, IL 61801-2918
+1 (217) 333-2300

ABSTRACT

Social networking websites are enormously popular, but they present a number of privacy risks to their users, one of the foremost of which being that social network service providers are able to observe and accumulate the information that users transmit through the network. We aim to mitigate this risk by presenting a new architecture for protecting information published through the social networking website, Facebook, through encryption. Our architecture makes a trade-off between security and usability in the interests of minimally affecting users' workflow and maintaining universal accessibility. While active attacks by Facebook could compromise users' privacy, our architecture dramatically raises the cost of such potential compromises and, importantly, places them within a framework for legal privacy protection because they would violate a user's reasonable expectation of privacy. We have built a prototype Facebook application implementing our architecture, addressing some of the limitations of the Facebook platform through proxy cryptography.

Categories and Subject Descriptors

K.4.1 [**Computer and Society**]: Public Policy Issues – *Privacy*.

E.3 [**Data**]: Encryption – *Public key cryptosystems*.

H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *Commercial Services, Web-based services*.

H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *Web-based interaction*.

General Terms

Security, Legal Aspects.

Keywords

Social networks, privacy.

1. INTRODUCTION

Social networking applications are quickly becoming a pervasive communication medium. Starting only a few years ago, several networks now boast tens of millions of users and are continuing to grow. Although the active populations of individual social networks can be dynamic, as some networks wane in popularity while other new entrants quickly accumulate large user bases, we can expect social networking to remain a popular paradigm due to the new opportunities for communication and information sharing they present.

Along with new opportunities come new risks. Social network providers maintain vast repositories of personal information. Since social networks mimic in-person interactions, people are often willing to reveal many more private details than they would otherwise [6]. There are many possible ways that the privacy of a social network user's information can be compromised: publication of specific information on the network to unintended recipients due to poorly understood defaults, accidental data release, intentional use of private data for marketing purposes by the social networking site, court order, and so on.

These privacy breaches are all made possible by the centralized architectures of social networking websites. These architectures require all information being transmitted through the social network to flow through central servers operated by the service provider. An alternative, decentralized social network architecture might not present any risk at all of accumulation of private data by third parties and would not be susceptible to these privacy breaches.

But for several reasons we did not attempt to build a new, decentralized social network and convince users to abandon centralized services – namely, Facebook – for it in the interest of privacy. The benefits of the centralized social network architecture are obvious and controlling. Centralized social networks are accessible from anywhere; they have large, established user bases; they have already invested in infrastructure; they hide networking and other technical information from their users and thereby maintain excellent usability. Realistically, today, only the very most privacy-conscious users abstain from using Facebook because of security concerns. The benefits clearly outweigh the privacy risks. We accepted that Facebook is going to be transmitting personal information for many, many people for at least the foreseeable future. With this in mind we considered how to mitigate the privacy risks of using Facebook by adding privacy protection features to it while retaining its existing benefits.

As our solution for mitigating the privacy risks of using Facebook, we implemented flyByNight, a Facebook application for encrypting and decrypting sensitive data using client-side JavaScript. Our architecture ensures that sensitive data never

WPES'08, October 27, 2008, Alexandria, Virginia, USA.

touches Facebook servers in an unencrypted form. Our implementation had to address some challenges that result from using JavaScript and the Facebook application platform; in part, our work was helped by using proxy cryptography.

We maintain universal accessibility and ease of use at the cost of sacrificing some security and privacy, leaving the application vulnerable to several types of attack. Nevertheless, we argue that our architecture significantly raises the costs to carry out privacy breaches and, importantly, puts the sensitive data under a different legal framework because a user may hold a reasonable expectation of privacy of the data. Therefore, we are able to significantly mitigate the privacy risks associated with using a site such as Facebook.

2. THE FACEBOOK ENVIRONMENT

Facebook is a popular social networking site, with a population of 90 million users [19]. Facebook allows users to share information in various ways. Each user maintains a profile with biographical information and interests. Users can also update their statuses as a way to describe their daily activities and whereabouts. They can upload photos, send messages to other users, or hold public conversations by writing on each other's "walls."

Each of these methods involves transmission of potentially sensitive information, and Facebook gives users the ability to control the dissemination of various parts of their profile, photos, and status updates. Common "privacy settings" allow limiting information flow to within friends of friends or within a given "network," meaning everyone at the same university, company, city, and so on. Previous work has shown that many users are unaware of the extent to which information is shared in the default configuration and overestimate the protection their information enjoys [1]. However, even when users diligently configure their privacy settings, privacy breaches are possible, especially since the privacy controls only limit information flow within the Facebook interface and have no effect on how Facebook handles the data on the backend. Privacy controls do not at all limit Facebook Inc. from aggregating private data from its users.

Facebook has several times introduced architectural features that revealed information in ways that made users uncomfortable. The most famous example was the introduction of a "news feed" in which the activities of one's friends are aggregated into a single page. The ability of others to monitor in detail things such as changes to one's profile or friend relationships upset many users. Facebook introduced an option to turn such data reporting off, but the news feed remains a popular feature today. A more egregious privacy violating was the Beacon system, where shopping history at partners' web sites was added to a person's news feed.

Other potentials for violations exist. A subpoena or court order could compel Facebook to reveal certain information from its servers to the police, or it could release information that is stored on its servers due to a programming error, break-in, or scrape. There are documented cases of such breaches occurring at Facebook and other social networking sites [21].

2.1 Potential Legal Privacy Protection

Having shown that the risk of privacy breaches is real, we pause briefly to consider whether there already exists a legal framework for protecting the data that users transmit through Facebook – that is to say, we consider whether there is a precedent indicating that Facebook has an obligation to protect the personal privacy of the information that its users transmit through it. The ultimate answer we find is "no."

The constitutional principle governing private information would appear to be the "reasonableness standard" from Justice Harlan's concurring opinion in U.S. v. Katz (1967), which holds that the two requirements that a plaintiff must satisfy in order to demand legal privacy protection are, first, to show that he "exhibited an actual (subjective) expectation of privacy" and, second, to show that the "expectation be one that society is prepared to recognize as 'reasonable.'" [11] The question of what is and what is not "reasonable" is open.

Many cases can be applied to Facebook and similar corporations if we accept the premise that the corporations are service providers and their users are their customers. There are statutes and precedents that govern the access e-mail providers may have to their users' messages, for example [12]. But nevertheless it is difficult to argue that these cases endow social network communications with legal privacy protection because the same cases evince the "legitimate business purposes" doctrine, which holds that entities such as Facebook can analyze and disclose communications made through their systems to third parties without user consent in pursuit of a reasonable economic interest of the entity [8]. Under this doctrine, which governs American privacy protection, the burden falls upon users to analyze the possible economic motives held by the digital social networking provider for disclosing their personal information and to adjust their expectations of privacy accordingly.

This weakness in privacy law introduced by the legitimate business purposes doctrine renders it difficult for users to argue that they have a reasonable expectation of privacy of the information that they reveal to Facebook, since Facebook's business model involves selling personal information and using it to target advertisements. Users are legally informed of this through the arcane license agreements to which they agree when registering.

Thus, Facebook (and social network providers like it) can lay legal claim over the privacy status of the contents of communication traces using a legitimate business purposes argument; they can gain "joint access or control" over the information; and they can distribute it, having negated the reasonableness of the expectation of privacy [8]. Of course, ordinarily we would not even expect Facebook to have to legally justify itself for breaching its users' privacy. Facebook's introduction of privacy controls for the news feed was a response to a public relations problem, not an attempt to forestall legal action. Most privacy violations are invisible and therefore would not merit any similar public relations response from Facebook. There would be little pressure for Facebook to withhold, for example, information from police officers lacking a warrant or private investigators. This we find troubling.

Thus it is our understanding that having an expectation of privacy in the information one transmits in the standard way through Facebook is unreasonable.

Our work is based, in part, on shifting the model of interaction with Facebook in order to make reasonable the expectation of privacy in the transmitted information. By creating a reasonable expectation of privacy, we better satisfy the first prong of the Katz test. Our architecture allows users to register an intent to keep certain information private and allows them to apply cryptographic tools that can be reasonably depended upon to keep information private. To do this, we use the Facebook Platform.

2.2 Facebook Applications

Facebook allows developers to create "Applications" to extend the types of information that can be stored, manipulated, and shared using the Facebook interface. These Applications are built using the "Facebook Platform," an API that can be used by developers to obtain information from the Facebook databases and deliver custom web pages to end users via Facebook servers. We used the Facebook Platform as the foundation on which to build the flyByNight privacy architecture.

The way the Facebook Platform API works is that is such that Facebook acts as a middleman for all interaction between application providers and end users. End users initiate contact with an application provider through a Facebook URL such as `http://apps.facebook.com/appname`. Facebook servers interpret input data sent along with these requests and pass their interpreted data along via the Internet to the application servers, whose addresses are registered with Facebook by the application developers. The application server then performs requested actions based on the Facebook-interpreted user input, perhaps including database operations. The application server then delivers to Facebook an output page consisting of HTML and Facebook-specific markup, including scripts. Facebook then interprets this output page, replacing the Facebook-specific markup with standard HTML and JavaScript, and delivers the interpreted output page to the end user.

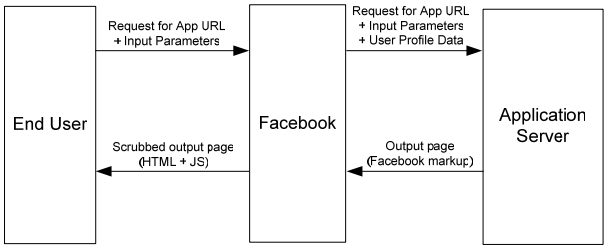


Figure 1. Facebook Application dataflow.

Because Facebook servers act as middlemen for all communication in the Facebook Platform, several significant security concerns are introduced. We address these in Section 3.1.

3. FLYBYNIGHT ARCHITECTURE

Having the opinion that the law does not offer adequate privacy protection for information that flows through Facebook, we decided to resort to technical methods to protect the privacy of Facebook communications. We designed the flyByNight application with the following goals:

- Protect personal information transmitted to Facebook by means of encryption
- Ensure that Facebook servers never store cleartext data or private key material and that cleartext data never appear on the Internet
- Support one-to-one and one-to-many communication
- Maintain universal accessibility and ease of use of Facebook
- Allow Facebook to manage social network friend relationships
- Use the Facebook interface for key management, storing as much key information on the server as possible

The last two goals represent a compromise between usability, security, and privacy. A fully private solution would not reveal even friendship relationships to Facebook; however, this would require abandoning the Facebook architecture, along with its large existing network of friends and useful search features. Similarly, using the Facebook interface for key management introduces some risks, but this design allows users to easily access and use the application from any web browser, requiring only the additional burden on the user of having to remember a single extra password.

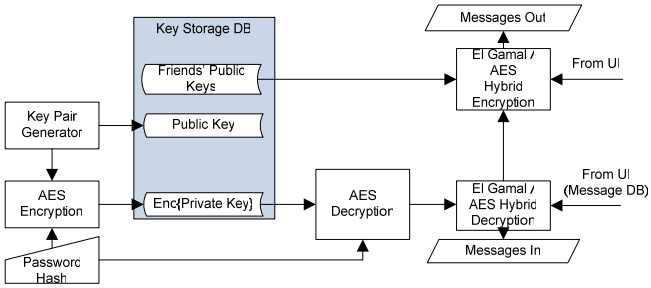


Figure 2. flyByNight Architecture.

Our architecture is shown in Figure 2. When a user first interacts with the application, he generates a public/private key pair and provides a password. The key generation and other cryptographic operations are performed in client-side JavaScript. The password is used to encrypt the private key. The encrypted private key is then transmitted to the flyByNight application server via Facebook servers and stored in a key database on the flyByNight server.

When a user wishes to send a private message or make an update to profile or status information, he first enters the message text into the application. The application always has access to a complete list of the user's friends who have also installed the application and their public keys, and from this list the user selects which friends to whom he would like the message to be delivered (or in cases such as status and profile updates, all of them). The client-side JavaScript encrypts the message with the necessary public keys (only one if proxy encryption is used; see Section 4.2) and tags the encrypted versions with the ID numbers of their intended recipients before sending them via Facebook to a message database on the flyByNight server, where they reside. The existence of these messages is public, but their contents are encrypted.

To read messages, the user queries the message database for a list of all encrypted messages sent to him and receives a list of message handles. When a user requests a specific message, the ciphertext of the message is delivered. The user decrypts his own private key by supplying his password, and he decrypts the message using his private key. This model can easily be adapted to handle profiles and status messages by allocating in the message database for each user a field to hold his profile text, a field to hold his status message, and so on, with the contents of that field being replaced with a new "message" containing the user's new profile or status every time he makes an update. When other users want to see the profile of the user, they can simply request the current contents of the user's profile or status field in the message database. To make this system work requires the "one-to-many" encryption operation we describe in Section 4.2.

3.1 Threat Analysis

The main feature of the architecture that protects users' privacy is that Facebook never sees unencrypted messages or keys and is therefore unable to recover the contents of communication. However, there is still significant trust placed in the Facebook servers because they necessarily act as middlemen. If it wished to compromise users' privacy, Facebook would have two approaches it could take. First, it could replace the JavaScript code of the flyByNight implementation such that it sends a copy of the user's password or unencrypted messages back to Facebook. Second, it could replace the public keys of friends with ones of its own, so that it can decrypt the private messages. Both of these attacks are relatively easy to carry out, and thus at a first glance, one might imagine that flyByNight does not provide any protection from privacy compromise by Facebook. But in the more expansive view, a user who goes so far as to utilize flyByNight to encrypt a message satisfies (in our opinion) both requirements of the reasonableness test, and therefore using flyByNight to send information through Facebook may endow that information with legal privacy protection that compensates for its technical vulnerability.

The first requirement of the reasonableness test, that a user intends to keep information private, is clearly satisfied; there can be no doubt that the user did not intend Facebook to have his private information when he went to the trouble of using an application that encrypts all stored data. The second requirement, that society be prepared to recognize the expectation of privacy as reasonable, is also satisfied. Facebook cannot claim a legitimate business purpose in surreptitiously replacing the client-side code of a Facebook application or secretly replacing in values it transmits to an end user on behalf of an application server. We imagine that members of the general public would find it reasonable to expect Facebook to refrain from covertly performing such operations. Thus the risks introduced by the technical vulnerabilities of flyByNight are mitigated because the exploitation of those vulnerabilities involves actions that invoke legal protection instead.

In addition, the use of flyByNight mitigates several other privacy risks. An accidental breach of information is less likely; an error in the Facebook implementation might reveal stored data to the wrong user, but the user will not have access to the keys necessary to decrypt it. Similarly, a data compromise of the Facebook servers will result only in the leakage of encrypted data. Another attack prevented by the architecture is the casual browsing of private information by Facebook employees [18]. With the use of flyByNight, employees would have to be very determined to carry out such an attack, as it would involve both implementing a man-in-the-middle attack and potentially exposing themselves to legal liability. Finally, it is not clear that even a court order could produce decrypted private data; we discuss a similar scenario in Section 5.

3.2 Additional Possible Safeguards

Savvy users can take extra measures to mitigate the risks of Facebook's middleman role. To eliminate the risk of Facebook replacing the JavaScript code with a trojan that compromises security, users can inspect the code delivered by Facebook. This task can be automated with a browser plugin; for example, JavaScript inserted by the Greasemonkey plugin [5] can verify the integrity of the code by methods similar to those listed in [16]. Attacks on key management by Facebook can likewise be mitigated by judicious verification of the users' public keys. Our prototype implementation enables such verification by displaying the public keys of all friends, allowing users to perform out-of-band verification of the keys (say, by means of a telephone call) as well as detection of any changes to public keys between uses. To simplify this process, our prototype could be extended to use key fingerprints or hash visualization techniques [15]. Another method of detecting changes would be to store friends' public keys once they are first retrieved, authenticated by a MAC derived from the users' password.

Of course, these checks require significant effort on the part of the user and will likely be carried out only by the more technically savvy (and more paranoid) users. However, even occasional monitoring for covert attacks by Facebook would be sufficient to discourage them generally, since news of such attacks would be a public relations nightmare and would additionally possibly carry legal liability.

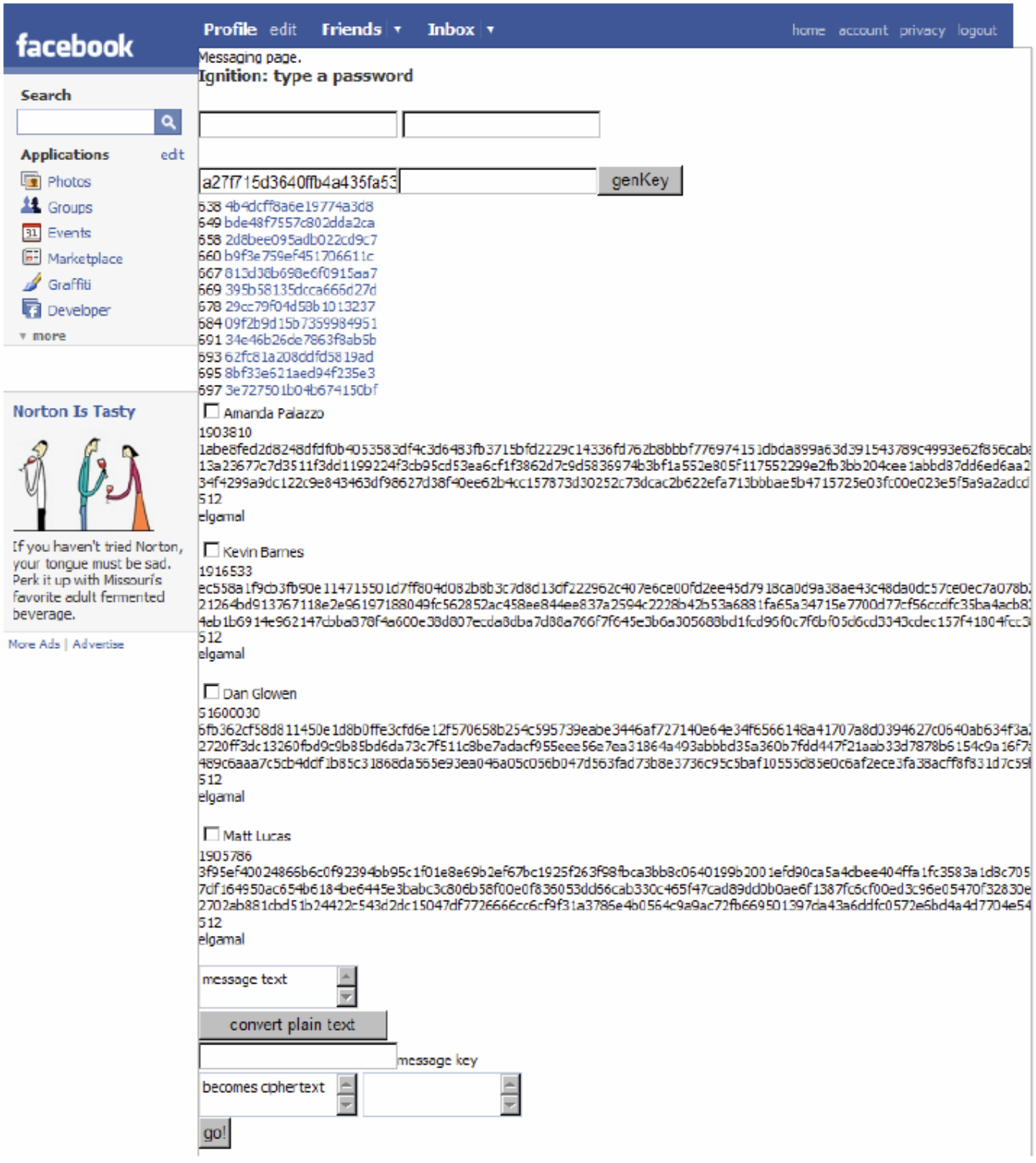
4. IMPLEMENTATION

We built a fully functional prototype implementation of the flyByNight architecture as a Facebook application. The application is accessible through Facebook under the URL `http://apps.facebook.com/flybynight`. A screenshot of the application in action is shown in Figure 3. Our current prototype has a user interface that exposes the cryptographic operations to better demonstrate its functionality; the interface would need to be made more user-friendly when released for the general public.

4.1 Cryptographic Tools

Our implementation is based upon open source JavaScript implementations of AES [20] and RSA [7] and on our own implementation of El Gamal, built on top of the math routines contained in the RSA implementation. The first challenge we faced was the limitations imposed on JavaScript by the Facebook platform. Facebook rewrites the JavaScript supplied by the application developer to reduce the possibilities of cross-site

Figure 3. flyByNight Facebook Application. To send a message, the user checks the boxes corresponding to the intended recipients of the message, types the message text, and clicks "go!" A list of the messages a user has received, indexed by the first blocks of their ciphertexts, appear at the top; to read one of these, the user types his password and clicks the ciphertext block. The plaintext then appears in a dialog box.



scripting attacks. Variables are renamed and certain functions are disabled. In addition, Facebook imposes a limit on the total size of the JavaScript code that an application is able to use.

After adjusting our implementation to deal with this issue, we ran into a second bottleneck: key generation performance. RSA key generation requires the creation of two random primes. This process can take several seconds in applications such as PGP or SSH. However, due to our use of the restricted JavaScript architecture, our implementation ran several orders of magnitude slower. Generating 1024-bit keys took at least 10 minutes on a 1.6 GHz Pentium IV; even 512-bit keys required more than a full minute. Compounding the problem, browsers interrupt script execution every 10 seconds, asking the user to abort a probably broken script; expecting users to dismiss this dialog every 10 seconds for 10 minutes is clearly unacceptable. To remedy this problem, we switched our encryption algorithm to El Gamal. Key generation in El Gamal involves only a single modular exponentiation, which in our implementation takes less than 10 seconds. A disadvantage of El Gamal, as compared with RSA, is that public key encryption and decryption is significantly slower than RSA. RSA encryption is very fast due to small public exponents, and even decryption runs faster because the use of the Chinese Remainder Theorem allows modular operations to be performed with 512-bit numbers, rather than 1024-bit. El Gamal is still fast enough to perform a single encryption or decryption without raising a browser alert, but for one-to-many communication, its poor performance still presented a problem.

4.2 One-to-Many Communication

In Facebook, a user typically makes a single update to a profile, status message, or wall that must immediately become visible to all his friends at once. To provide a private way to perform such actions, flyByNight supports a "one-to-many" operation that encrypts a single message for a group of friends. This operation cannot be handled by simple iteration because users commonly have a hundred friends or more, and so a simple implementation of "one-to-many" encryption would therefore be a hundred or more times slower than a single encryption would, since the message would be individually encrypted for each friend. In addition to the performance issue, the storage requirements would quickly become prohibitive. A 1024-bit El Gamal-encrypted message is 256 bytes in length; thus, one hundred versions of an encrypted status message, originally only 50-100 bytes in length, would require over 25 kilobytes of storage. While this is a small number for a single status message, when multiplied by tens of millions of users, this creates a significant storage overhead.

To address both these problems, we made use of proxy cryptography [2, 10]. Proxy cryptography allows a party to take a message encrypted with one key and re-encrypt it with another key, without knowing either key or learning the contents of the message. We describe how this works with El Gamal. El Gamal uses a private key of x and a public key g^x, where g is a public generator of a group in which the discrete logarithms are hard to compute. Encryption of a message m generates a pair (a,b) as follows:

$$a = g^r, b = m \cdot (g^x)^r$$

where r is randomly chosen for each message. To decrypt the message, the recipient computes:

$$\frac{b}{a^x} = \frac{m \cdot g^{rx}}{g^{rx}} = m$$

Given two El Gamal keys, g^x and g^y, a proxy key p that allows conversion of messages encrypted under g^x to ones encrypted under g^y would be the value $p = x - y$. Given a message (a,b) encrypted under g^x, the proxy key can be used to compute a new message (a',b') as follows:

$$a' = a, b' = \frac{b}{a^p} = \frac{m \cdot g^{rx}}{g^{r(x-y)}} = m \cdot g^{ry}$$

We can use such proxy techniques to reduce the client-side computation and storage requirements of one-to-many encryption to be $O(1)$ in the number of recipients. A user, Alice, creates a group key pair, (x_g, g^{x_g}). Then, to add Bob to the group, Alice creates another key pair for Bob, $(x_{a\to b}, g^{x_{a\to b}})$, and a proxy key $x'_{a\to b} = x_g - x_{a\to b}$. She then gives the proxy key to the flyByNight application, while the private key x_b is sent to Bob by encrypting it under Bob's public key, g^{x_b}. She repeats this process for every other friend, creating keys $x_{a\to c}$, $x_{a\to d}$, … and proxy keys $x'_{a\to c}$, $x'_{a\to d}$, … .

To send a message to a group of friends, Alice encrypts it under the public key g^{x_g} and sends this to flyByNight, which stores it in this form. When Bob wishes to decrypt the message, he first asks flyByNight to use its proxy key $x'_{a\to b}$ to transform it to one encrypted under the key $g^{x_{a\to b}}$. He then retrieves the encrypted key $x_{a\to b}$ and decrypts it using his own private key, and then finally decrypts the message.

Note that Alice performs only a single encryption for each message she sends, and only a single encrypted message needs to be stored on the Facebook server. The server will have to perform $O(n)$ proxy encryptions for each of Alice's n friends who read the message, but this can be implemented in an efficient language, such as C, and each iteration need only be done on demand when each friend requests the message. No time is wasted on performing proxy re-encryptions for friends who do not check the message, as might be the case when Alice updates her status several times before Bob checks Facebook. The server does need to store $O(n)$ proxy keys and $O(n)$ encrypted keys, but this is a one-time storage cost. Similarly, Alice must generate these keys for each of her friends, but she can do this process incrementally as new friends are added. A more challenging operation is revocation, when a friend relationship is severed, since a new group key and new proxy keys for each remaining friend would need to be generated. In practice, such operations are rare enough that we imagine that they would not present a significant usability barrier.

4.3 User Interface Concerns

In our design and implementation of flyByNight we took several usability concerns to be of paramount importance. The most important requirement was that the implementation had to be universally accessible as a Web application, just as Facebook is, without any technical knowledge required of the user or binary

code. It was imperative that flyByNight maintain the simplicity and platform-independence afforded by Facebook's web interface.

It was also imperative that our application would not significantly burden the performance of the Facebook site as perceived by the end user – that is, we were determined not to slow down the end user's workflow. Sending an encrypted text message, for example, would have to be perceived as instantaneous, since the standard Facebook messaging application gives that impression. The performance constraint was critical because regular users who are not particularly privacy-conscious will not use encrypting versions of social networking functions if they are any more difficult or time-consuming than standard versions.

Finally, we were determined to require of our users as little technical knowledge as possible and to place on them the smallest possible additional memory burden. Facebook's success is in no small part due to the fact that the only information that a user must commit to memory in order to use Facebook is his user name and password, and the only information he needs in order to navigate the network is the real-world names of his friends. We did not reach the optimal case of flyByNight placing absolutely no additional burden on its users than does Facebook itself. Indeed, we believe that it is impossible for a user to secure information from Facebook without the use of secret information that he never makes available to Facebook, and by definition, secret information is an additional burden placed upon the user. But we attempted to keep this burden as small as possible. We require the user to remember only a single additional password, which we use to generate a key to encrypt and store invisibly from users all the other secret information used to make flyByNight work. We do not believe that a better compromise can be made. Requiring anything less than a password would, after all, reduce the entropy of the key space used to secure users' private keys and thus render the entire system less secure.

We worried that the critical usability requirements of requiring the application to be a Web application without binary code and requiring it to perform well enough so that users' workflows are not at all slowed down would be irreconcilable, in which case we were prepared to resort to using Java applets or something like them in the vein of Hushmail [9] (see Section 5). However, in the end we were able to build the entire application out of scripts while satisfying all our usability concerns.

4.4 Images

One goal we were unable to achieve with the flyByNight implementation is encryption of photographs. Photos on Facebook have great potential to be compromising to the individuals pictured in them, and providing them with the same protection as status updates or one-to-one communication would clearly be desirable. Unfortunately, the JavaScript architecture makes doing so quite difficult. In particular, JavaScript is unable to read local files from a disk. Files must be uploaded to a server first before they can appear in a JavaScript buffer. Such an upload over the Internet in the clear would obviously defeat the purpose of encryption. A helper application to upload photos could be used, but then that would restrict the accessibility of the social network, since users would not be able to upload photos from dumb clients such as mobile phones. This may be a trade-off that more privacy-conscious users are willing to make. To display the decrypted images in a standard browser without storing them on an intermediate server, perhaps the `data:` protocol could be used, but we did not examine this in any more detail.

5. RELATED WORK

Acquisti and Gross studied privacy in social networks [1, 6]. They highlighted a number of privacy vulnerabilities and found that users greatly overestimate the privacy of their data. Using an application such as flyByNight would give users more explicit control over their information and hopefully mitigate this problem.

Several architectures for protecting information on social networks have been proposed. Apres [14] is a system for anonymous presence information, to be used with instant messaging systems. It allows users to communicate their status updates to others through a centralized server without revealing either the status or the friend relationships to the server. The system thus provides stronger privacy protection than flyByNight, as we still allow Facebook to maintain friendship relationships; however, switching to such a system would mean that users would need to register with a new service, losing the benefits of the existing communities of friends on Facebook. Furthermore, it is difficult to make the case that an expectation of privacy of one's friendship relationshipsis reasonable, as those friendship relationships can often be inferred from other public sources. Frikken and Golle have designed a system for privacy-preserving social network analysis that also protects social network relationships while allowing the detection of patterns [4]. Their system once again requires a shift away from the Facebook platform.

Felt and Evans [3] have designed a system that insulates personal information stored on the central server from third-party application servers. This system prevents third-party application developers from obtaining personal information about the users of a social network, but it does not address the question of whether the central social network provider itself, e.g., Facebook, can be trusted to handle personal information. In contrast, in our system, data transmitted through the social network are never visible in the clear to Facebook, and they are also never visible to the flyByNight servers. In fact, in our system, the information is never available on any machine besides the client machines of senders and recipients. Thus our system does not depend on the trustworthiness of Facebook.

Hushmail [9] provides an email privacy service that is similar in philosophy to our design. It uses a Java applet to implement encryption and decryption of mail using public key techniques. Interestingly, the use of Java was onerous enough for many users that they recently implemented a different version in which the encryption operations are performed on their server. This service was recently the subject of a court order that compelled Hushmail to capture private information and turn it over to the police [17]. Hushmail stated that, had their original, client-side encryption system been used, they would not have been able to comply with the court order, which may mean that the same would be true for flyByNight. Our use of JavaScript to implement encryption operations means that we require less of users and their machines than we would if we used Java, and it is our intention to require so little of our users and machines that they do not demand a server-side alternative that would be susceptible to court order.

Finally, our use of proxy cryptography is inspired by the SELS mailing list software [13], in which proxy cryptography is used to reencrypt a mailing list message for each recipient. Their proxy encryption scheme is more complex than ours, as it handles authentication and admission control to the list.

6. CONCLUSION

We have designed an architecture for mitigating the privacy risks of using a social networking site such as Facebook, and we implemented the architecture using a prototype Facebook application. Our design strikes a balance between protecting the users' privacy and maintaining Facebook's usability. We argue that, although some privacy and security risks remain, the threat of privacy compromise is greatly reduced because our architecture raises the cost of technical privacy attacks and shifts the communication medium to one in which a user may hold a reasonable expectation of privacy and thereby enjoy an accompanying legal framework for privacy protection.

7. REFERENCES

[1] Acquisti, Alessandro and Ralph Gross. Imagined Communities: Awareness, Information Sharing, and Privacy on the Facebook. In George Danezis and Philippe Golle, editors, Workshop on Privacy Enhancing Technologies, volume 4258 of Lecture Notes in Computer Science, Cambridge, UK, June 2006. Springer.

[2] Blaze, M., G. Bleumer, and M. Strauss. Divertible Protocols and Atomic Proxy Cryptography. Lecture notes in computer science, pages 127-144.

[3] Felt, Adrienne, and Evans, David. Privacy Protection for Social Networking APIs. University of Virginia, 2008.

[4] Frikken, Keith and Philippe Golle. Private Social Network Analysis: How to Assemble Pieces of a Graph Privately. In Roger Dingledine and Ting Yu, editors, Workshop on Privacy in Electronic Society, pages 89-97, New York, NY, 2006. ACM.

[5] Greasemonkey. http://www.greasespot.net, 2008.

[6] Gross, Ralph, Alessandro Acquisti, and H. John Heinz III. Information Revelation and Privacy in Online Social Networks. Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, 2005, p. 71-80.

[7] Hanewinkel, Herbert. PGP / GnuPG / OpenPGP message Encryption in JavaScript. http://www.hanewin.net/encrypt/, 2005.

[8] Hodge, Matthew J. The Fourth Amendment and Privacy Issues on the "New" Internet: Facebook.com and MySpace.com. Southern Illinois University Law Journal.

[9] "Hushmail – Free Email with Privacy." Hush Communications Corp, 2008. http://www.hushmail.com.

[10] Ivan, A. and Y. Dodis. Proxy Cryptography Revisited. Proceedings of the Network and Distributed System Security Symposium (NDSS), February, 2003.

[11] Katz v. United States, 389 U.S. 347 (1967).

[12] Kerr, Orin S. "A User's Guide to the Stored Communications Act and a Legislator's Guide to Amending it." George Washington University Law Review, 2004: 1208-1227.

[13] Khurana, Himanshu, Adam Slagell, and Rafael Bonilla. SELS: A Secure E-mail List Service. In Security Track of the ACM Symposium on Applied Computing (SAC), March 2005.

[14] Laurie, Ben. Apres: A System for Anonymous Presence. http://www.apache-ssl.org/apres.pdf, 2004.

[15] Perrig, Adrian and Dawn Song. "Hash Visualization: a New Technique to Improve Real-World Security." In Proceedings of the International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC), Hong Kong, July 1999.

[16] Reis, Charles, Steven D. Gribble, Tadayoshi Kohno, and Nicholas C. Weaver. "Detecting In-Flight Page Changes with Web Tripwires." Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI '08), San Francisco, CA, April 2008.

[17] Signel, Ryan. "Encrypted E-mail Company Hushmail Spills to Feds." http://blog.wired.com/27bstroke6/2007/11 /encrypted-e-mai.html, 2007.

[18] "Scoop: Facebook Employees know what profiles you look at." Valleywag, October 10, 2007. http://valleywag.com/tech/scoop/facebook-employees-know-what-profiles-you-look-at-315901.php.

[19] "Statistics | Facebook." Facebook Inc. http://www.facebook.com/press/info.php?statistics, 2008.

[20] Walker, John. Javascrypt: Browser-based Cryptography Tools. http://www.fourmilab.ch/javascrypt/, 2005.

[21] "You've Been Poked by University Police." Daily Illini, July 25, 2006. http://media.www.dailyillini.com/media/storage/paper736/news/2006/07/25/Opinions/Editorial.Youve.Been.Poked.By.University.Police-2133945.shtml