Proposal for MEG - The Mobile Enryption Gateway

Gregory Rehm, Jennifer Fowler, Michael Thompson

University of California, Davis grehm@ucdavis.edu

Argonne National Laboratory {jfowler,thompsonm}@anl.gov

February 2016

1 Title Page / Introduction

Gregory Rehm, Jennifer Fowler, Michael Thompson
University of California, Davis grehm@ucdavis.edu
Argonne National Laboratory {jfowler,thompsonm}@anl.gov

Keywords: Email Encryption, Mobile Computing, Data privacy, Open Source Software

In the 21st century email encryption still remains a hard problem to scaleably solve. The problem doesn't lie in the developing technologies to perform the encryption, but rather in the usage of the encryption method. Modern methods of email encryption have a high barrier of entry for the average user. Because of this people still transmit their emails completely unencrypted able to be read or modified by anyone with access to a routing node. MEG, the Mobile Encryption Gateway aims to solve these problems and to set up a usable encryption method so that mobile users will have access to strong email encryption, regardless of their email client.

Total Budget: $24,214.56

List of deliverables:

- Literature Review
- MEG server
- End of Quarter Progress Report
- MEG Android app
- MEG Gmail client plugin
- Final Project Report
- Research Conference / Journal Paper

2 Project Summary

Keywords: Email Encryption, Mobile Computing, Data privacy, Open Source Software

MEG, the Mobile Encryption Gateway aims to ensure that email communications are encrypted for everyone at the cost of installing a free app. The philosophy behind MEG is: to ensure truly secure communications private keys must live with the users and not be accessible to hostile actors. To accomplish this the MEG server will house public keys in an openly accessible server. A users private key will live on their mobile phone. The MEG android app aims to perform all encryption services from a users mobile device. The MEG client will act as a proxy for transmitting and receiving encrypted emails. MEG can conceivably pair with any other email app like Gmail or outlook to transmit emails. In this spirit, we can keep communications private while adding as little cost as possible to enable end to end encryption of user emails. This is possible because MEG will handle all security details of the certification, decryption, and encryption allowing the user to focus on their primary purpose, composing their email.

3 Motivation

This project touches on a real vulnerability in data privacy. Government officials, corporations, and criminals have plenty of incentive to sift the internet for peoples emails. Ordinary users on the other hand have little recourse to actually encrypt their emails unless their job enables them to encrypt mail or they understand how to set it up for themselves. These usability problems are what prevents non-technical people from using email encryption. It just has too high a barrier of entry. There are paid services providing encryption on the web but they do not make their code open source and thus are not open to audit. If the amount of privacy these sources offer is unknown we ask why should people trust them. MEG on the other hand, offers itself as an open source, free, and completely end to end encrypted alternative to these systems. To ensure that such great technology is used by even non-technical people we hope to make MEG as easy to use as possible with hopes of making it accessible to anyone.

Getting a system to the point where it will be accepted as a publicly viable way of encrypting emails will be difficult. So the first steps we will take will be getting a proof of concept of the MEG server and client off the ground so that we can show that we have a strong idea. The most challenging part will be the amount of work that needs to be done. However I believe that with the right amount of motivation and luck that a proof of concept will be possible.

4 Previous Work

The first email encryption scheme released was PEM. Users could publish their public keys and then these keys could be signed by the private key of a certifying authority. Certificate Authorities (CAs) worked by creating a hierarchical system of trust eventually ending in a single trusted root authority [7]. However PEM was never truly implemented when the security concerns and potential liabilities of a single root authority became evident [3, 9]

After the failure of PEM, S/MIME was the next encryption standard to emerge and is now supported by many mail clients. The main difference in between PEM and S/MIME is S/MIME drops the necessity for a single root CA and acknowledges the existence of multiple independent authorities [7]. Because CAs can reliably validate the identity of who someone is contacting S/MIME is currently recognized as one of the best ways to encrypt mail. The only problem with S/MIME is that CAs charge expense for their services and the vast majority of users find the process for obtaining a certificate onerous [8]. As a result S/MIME is utilized in a limited sense. Many mail clients like Outlook support it but a majority of users have never obtained an S/MIME certificate for their personal use [1].

The alternative to S/MIME is PGP, and in a perfect world users would just be able to perform the service of encrypting and validating all mail for themselves using PGP. Given that PGP is decentralized and available to anyone researchers wanted to test how people would be able to use the technology for themselves. One of the first studies used to determine whether users could encrypt their email communications with PGP came in 1999 titled "Why Johnny can't Encrypt". This study decisively found that ordinary users are awful at encrypting messages themselves. This is true even when people are given ample amounts of time to try to perform the task of PGP encryption [11]. Even worse, the few who were able to encrypt their communications exposed their private key in plain text [11]. A follow-up study conducted in 2006 on a more modern interface still found users could not accomplish the task of encrypting and sending an email [10]. So while PGP clearly offers advantages that S/MIME doesn't; that it is free and decentralized users still cannot use it natively off the shelf.

As a result of usability difficulties with PGP and S/MIME alternative schemes for email encryption have surfaced. A lightweight key distribution system [4] is one such system devised. A proxy gateway that has knowledge of the user's private key is another system that can be adapted for email encryption [5]. The main problem with these systems is that they will be implemented by the mail provider. We now know post-Snowden, and in light of the shutdown of Lavabit [2], that mail providers either actively collaborate with governments or are bound by warrant to release private user information. Paid services like ciphermail could have offered reprieve but their source code is not open to audit and thus we cannot verify the level of privacy a consumer actually has when using their service. Hushmail is a perfect example of this and only received a 1 out of 7 score from the Electronic Frontier Foundation for their commitment to user privacy [6]. Understanding this, we need to have the source code of any encryption service be open source or at least available to security audit. And most importantly, to ensure data privacy we need to have the private keys for the user live somewhere that is inaccessible to prying eyes or robots.

5 Specific Aims

MEG aims to address many of the problems that stem from using PGP, S/MIME. MEG uses PGP but operates in the background and handles everything automatically for the user. MEG handles the problem of automatically encrypting/decrypting messages, alleviating the usability issues surrounding PGP. The problem with S/MIME where it is costly to generate keys is solved by automatically generating a key pair with PGP. Instead of a certificate authority vouching for the identity of a contact MEG will operate with PGP's web of trust. This way users can validate the authenticity of their contacts. The storage of the private key occurs on the mobile device. The public key is distributed to the MEG server. Keys can be signed by other MEG users to build the web of trust. This way MEG users can have trust in communicating with users they do not necessarily know.

Since MEG relies on PGP to handle all of the heavy lifting of performing encryption and decryption of messages the first task we need to perform is to set up a server that can manage all public keys under the MEG system. This server will likely be a web app with a database behind it to ensure we can have storage of public keys and revocation lists. Since we intend the web server to be federated so that losing a MEG server will not result in the loss of the entire service we will provide database replication across MEG servers so that they all have the same information. The web app will provide the API for interacting with MEG. We hope to provide comprehensive API documentation and unit testing to ensure that the web server is both performing its job and is secure.

The next major item we need to engineer is the MEG app on an Android device. The only thing this app will do is perform decryption of messages and serve as the email gateway. It will need to have knowledge of the users private key and understand how to access it to perform decryption. It will also need to have a symmetric key to communicate securely with the MEG client plugin.

MEG needs to be compatible with any email service available. This is where the client plugin comes in. We want this app to be compatible with Gmail so that it has out of the box compatibility with a major mail provider. The client plugin will serve as a proxy for sending encrypted messages back to the phone so they can be decrypted. The client plugin will also send encrypted messages off to a recipient. To make the web of trust easier to understand for non-technical users messages will be color coded depending on the level of confidence we have in the contact. Red means a message is from an untrusted or unencrypted source. Yellow means we do not trust the contact directly but they are validated through web of trust. Green means that we trust the contact completely.

6 Plan

Finishing the MEG server will likely be the easiest part of the entire project. Building a server with LAMP is a well understood process and there exist plenty of languages to write our web app with. We will probably choose the Django framework for this project because the author already has experience with the Django and is an experienced Python programmer. The Android app may be tricky to create but should not constitute too much effort because it acts as a relay for the client plugin. Since the client plugin is where the end user interaction happens in MEG this will likely be the most difficult item to engineer.

Our plan is to first create the MEG server. If we have time in our schedule we will ensure the service is federated so that if a node dies MEG will stay alive. However if we do not have time federation is not necessary for the proof of concept. Next we will create the Android app to ensure a phone can serve as a decryption gateway. There may be some production critical things we need to cut from the Android app due to time constraints like end-to-end encryption between the client plugin and the android app. Finally the client plugin will tie all the pieces of the project together to ensure MEG is truly operational. If we can get a minimal viable prototype of the client plugin that is able to intuitively visualize the web of trust then the author will be very happy.

7 Deliverables

For the project we aim to deliver two major products:

- Literature Review: A full literature review on email encryption.
- MEG server: A server that can act as the certificate authority and sign user public keys
- Progress Report: A report detailing the things we have accomplished at the end of winter quarter.
- MEG android app: An android app that acts as the MEG gateway and handles decrypting mail for the user.
- MEG Gmail client plugin: The client plugin will interface with Gmail as a proof of concept. It will have UI elements so that the user can visualize their Web of Trust.
- Final Project Report
- Research Conference / Journal Paper

8 Issues

The amount of work posed by this project is the greatest issue we can foresee at this moment. There are many tiny sub-elements critical for MEGs operation that collectively may take time to code and then debug. For example sending a message is a ten step process in MEG operating over three separate components; the MEG server, android app, and client plugin. If any one of these things is incomplete in some way then the service will not behave as intended. To complicate matters there will be only one full time researcher working on MEG. So to get a proof of concept may entail cutting production critical items like end-to-end encryption between the client plugin and the MEG android app. We can also cut MEG server federation if necessary. However if we do have time for these items we will happily place them into the proof of concept.

References

[1] Email encryption. https://en.wikipedia.org/wiki/Email$_{e}ncryption.Accessed:$ 2016/02/18.

[2] Lavabit. https://en.wikipedia.org/wiki/Lavabit. Accessed: 2016-02-12.

[3] Privacy-enhanced electronic mail. https://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail. Accessed: 2016-02-12.

[4] Ben Adida, Susan Hohenberger, and Ronald L Rivest. Lightweight encryption for email. In *SRUTI*, 2005.

[5] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology—EUROCRYPT'98*, pages 127–144. Springer, 1998.

[6] Electronic Frontier Foundation. Secure messaging scorecard. https://www.eff.org/secure-messaging-scorecard. Accessed: 2016-02-17.

[7] Simson L Garfinkel, David Margrave, Jeffrey I Schiller, Erik Nordlander, and Robert C Miller. How to make secure email easier to use. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 701–710. ACM, 2005.

[8] Simson L Garfinkel and Robert C Miller. Johnny 2: a user test of key continuity management with s/mime and outlook express. In *Proceedings of the 2005 symposium on Usable privacy and security*, pages 13–24. ACM, 2005.

[9] Bryan D Payne and W Keith Edwards. A brief introduction to usable security. *Internet Computing, IEEE*, 12(3):13–21, 2008.

[10] Steve Sheng, Levi Broderick, Colleen Alison Koranda, and Jeremy J Hyland. Why johnny still can't encrypt: evaluating the usability of email encryption software. In *Symposium On Usable Privacy and Security*, pages 3–4, 2006.

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

9 Biographic Sketch

Gregory Rehm is a Masters student at UC Davis currently interested in a wide variety of topics like computational medical research, data analysis, and systems security. His professional experience consists of 3 years working at Location Labs in Emeryville CA where he was a software engineer and DevOps engineer at various points in his career. Greg is currently working with a team lead by Dr. Jason Adams at UC Davis Medical Center to implement analytics for patients on ventilators. Greg is also working with Dr. Cindy Rubio Gonzalez at UC Davis to perform analysis on how continuous integration affects open source software projects.

10 Schedule

From the the time this proposal is submitted to the time our final report is finished we have about 12 weeks to work on this project. The proposed schedule for this project is:

- Literature Review: 2016/02/19 - 2016/02/26
- MEG server: 2016/02/19 - 2016/03/20
- Progress Report: 2016/03/04 - 2016/03/11
- MEG Android app: 2016/03/21 - 2016/04/04
- MEG Gmail client plugin: 2016/03/21 - 2016/05/15
- Final Project Report: 2016/05/07 - 2016/05/26
- Research Conference / Journal Paper: 2016/05/27 - 2016/06/30

11 Budget

Item	Cost
Salaries for two quarters	$9,626
Benefits	$126
Conference Travel	$1,250
Basic Android Phone	$200
Fees and Tuition	$12,130
Indirect Costs	$882.56
Total	**$24,214.56**

12 Appendix A.

We can submit this paper to SOUPS, or Symposium on Usable Privacy and Security. A similar paper submitted here was "Johnny 2: a user test of key continuity management with S/MIME and Outlook Express," DOI: 10.1145/1073001.1073003.

13 Broader Impact

Email encryption has long been something considered hard to implement *en masse*. The hope of this project is that we can bring secure email communications to anyone who wants it at cost of installing a mobile app. Later we can move towards integrating additional mail clients into our Android app and also can create an iPhone app as well. Commonplace email encryption can prevent criminal, government, or corporate actors from reading or modifying our personal emails in transit. More commonplace email encryption can also be a major victory for privacy advocates. The ability of activists to communicate securely will be critical to democracy in the 21st century. So widespread encryption will help further their causes as well. We believe MEG will place many of these things in reach by lowering usability barriers and providing a free and technologically secure service.