MEG Progress Report

Gregory Rehm, Jennifer Fowler, Michael Thompson

University of California Davis, Argonne National Laboratory
`grehm@ucdavis.edu` {`jfowler,thompsonm`}`@anl.ogv`

March 2016

1 Problem Statement

Few people realize today that the vast majority of emails transmitted today are sent completely unencrypted. This allows anyone who has access to the routers these communications are being sent over to read and even modify the contents of these communications. Encryption is a technology that is used liberally on the internet today. HTTPS ensures that bank communications are kept secure. SSH supplanted telnet because other actors cannot view and modify what actions that administrators perform on remote machines. So why is email an outlier? The problem does not lie in the infeasibility of encrypting email but rather the practicality of it. Existing email encryption schemes are clunky and can only be performed by committed users. This leaves the vast majority of non-technical and less security conscious users completely naked to their emails being spied. The Mobile Encryption Gateway (MEG) aims to solve this. MEG aims to take the difficulty of encrypting email and perform it all in the background while allowing the user to have as much ease as possible in performing secure communications. This is combined with an aspect of security and data privacy that MEG provides where all communications remain encrypted on a users mobile device. As a result of this security and ease of use MEG offers ordinary people their best chance going forward to have a reasonable encryption solution for their emails.

2 Work Accomplished

MEG is composed of three parts; the MEG server, Android application, and mail client. We have made fast strides with the MEG server. In addition to completing many of the web API's we needed on the server we have a solid suite of unit tests written that is helpful for facilitating rapid development of this component. We have also created an automated deployment project for the for the server component using Ansible. This will be invaluable when we take on the task of ensuring MEG is federated and able to withstand multiple nodes in the network failing. On the Android side of things we leveraged the already existing MEG skeleton app and started adding logic so that it could perform its work correctly. Current functionality of this app is as follows: It can recognize when the user has installed the app. The Android app can generate RSA key pairs so that we can perform encryption and decryption. The Android app can also receive messages from Google Cloud Messaging (GCM) when there is a new encrypted email message to receive from the server. This will trigger Android to call the server and request the message so the phone can perform its duties to decrypt the message and then send that decrypted message back to the client.

3 Work Remaining

There are still sizable amounts of work remaining to perform on this project. The server component is taking on a role of greater importance than we realized it would in the proposal phase and so additional work will have to be done on it to send messages that have been decrypted/encrypted by the phone back to the mail client. We will also need to provide additional APIs on the server side of things to facilitate encryption/decryption of mail that we wish to send or receive from others. The mail client itself still has yet to be started. We plan to finish it in the next quarter.

The Android app has become the largest project component yet. We currently have about 4358 lines of code either written by ourselves or imported (under GPL3) from other Android projects. The reason for this is that PGP is a very powerful and complex mechanism for data encryption and as a result is time consuming to code for. There is still a lot of work to be done though and while much of the groundwork has been laid for our success there still exists work in building core functionality off our foundation. Here are the things we need to to to get our Android app completed

- Ensure we can generate public and secret key rings on Android to store our keys
- Ensure we can handle a request to decrypt a message and then send the decrypted message back to the mail client
- From a QR code be able to generate a symmetric key for encrypting data transmissions between the mail client and the application
- Be able to handle encryption of messages for transmission to an email recipient.
- Handle process of generating and sending to the server a revocation key
- Create UI flow to handle revocation of keys.
- Perform storage of decrypted messages on the phone.

Some of these items will require less work to complete like performing storage of decrypted messages. Other items will require more like utilizing a QR code to create a symmetric key. Some of the work may be able to be done as we are progressing on the mail client. This may help to speed to process of the engineering as we can test the integration components of our project better this way.

4 Updates to Bibliography

No updates.

5 Project Difficulties

We initially anticipated it would be easier to network the MEG mail client application and the MEG Android application than it actually is in reality. We had based these expectations based on the condition that the mobile devices we would be communicating with would be networked over IPv6 and have a public address on the internet. However most mobile devices are not on IPv6, but rather IPv4. As a result we need to utilize the MEG server as an intermediary between the client and Android application. This requires additional engineering of the Android application and of the server application as well. Currently we have spent a week of time on this redesign of our process. In the future there will be additional APIs to engineer server side which may take another half week of work.

In the same vein, overall the Android application has proven to be slower to create than initially hoped for in the project proposal. I had hoped for more rapid progress on this component however much of the time I spent I was simply relearning how to think in terms of Java rather than Python and re-discover Android best practices. As such it may take a week or two extra to complete the Android application beyond what we had allotted for in the proposal. This may potentially mean the Android app will not be completed until 2016/04/11 or 2016/04/18. I do believe that this is not an explicitly negative thing. During our proposal phase we may have not had the best understanding of the amount of work necessary to be performed on Android. I believe this is more of a realignment of timing rather than an artifact that we are not completing work as quickly as we would like.