Global Secure Messaging App (GSEM)

Louis Thomas Kavouras

E-mail: louis.kavouras1@marist.edu

Abstract

Today's modern day, just about everyone has a smart device of some kind. Along with these smart devices, users have applications from utilities to social media. However, with this great technology, many of us are sacrificing our data's security for usability. In additon, with all the great social media platforms, messaging services, other forms of communication, we are prone to attacks. Facebook had in the past was in the cross hairs for having user passwords left in plaintext form, which left their clients very vulnerable to fraud and other attacks. Global Secure Messaging App (GSEM) for short, is a solution to all of those worries by incorporating AES and RSA algorithms, communicaton from one client to another is all secured on the client and server side of the application.

Keywords: AES, Advanced Encryption Standard, RSA, Rivest Shamir Adleman, Android, Firebase, Hashmap, Socket, Java

1. Introduction

Global Secure Messaging App also refered as GSEM, can be thought of as a form of communication that keeps privacy in mind. Using a combination of an AES and RSA together to provide end to end security for the clients when they wish to communicate with one another.

Security in today's day is very ugly and not the cleanest and most intuative which leaves it prone to human error. GSEM is currently designed for the Android operating system, and is designed in a way that is intutative and modern to the end user. I have based the user inteface to look like facebook or some other similar social media platform. I did this by incorporating the following pages; Welcome page, Login, Register, Profile, Chat, and People.

On the people tab, the users can find other people on the app, send friend requests and communicate by sending text messages to oneanother. If an user forgets their password, they can simply press the recover text view and a dialog will popup to walk them through a very simple recovery process within two clicks in the android application.

2. Background

During the very early phase of this project, I had several ideas that would help make this implementation as smoothly as possible. Many of those ideas did not work out as planned, which led me to this social media based design. As I stated before, there exists an option to send people friend requests, but what I did not mention is the significance.

I have chosen to use friend requests as the delivery method for exchanging the public keys of the end users. After many trials and errors, I am confident to say that I will be continuing the hybrid approach of both AES and RSA encryption since it appears to be accomplishing the goal.

3. Methodology

When the user when they open the app is prompted by a "register" or "login" option. Chances are the end user is new to the application and will choose to create an account with GSEM.

Currently the app is named Firebase secure chat app.

On user registration they will be asked for a valid email address and their desired password. All input provided by the user will be hashed using hashmaps, nothing is stored in plaintext in the database (avoiding facebook's mistakes). From there the user can edit their profile page and add their full name, phone number, profile image, and cover image. Once that is complete within the profile page, the user can generate a public and private key pair. This is important as after looking around on the web, it is not secure to have the server generate these keys and send it to the clients. Therfore, I solved this by allowing the user to create the key pair on their local android device, and then the public key on creation would be sent to the firebase database. Where it would be stored in the User table in the column labeled public key with that particular user.

Currently I am using Firebase as the backend for the application as it fits perfectly for this kind of application.

Once the user has his / her account all setup, they may proceed to the people's tab to look up other users by their name or email addresses. From there they can send a friend request which if accepted will allow the two individuals to exchange public keys which would be stored in the friends table for each respective user using the uid (user identification number).

Once the users have accepted eachother's friend requests they will be able to send eachother messages, files, images, videos, etc. All of this will be secured. For the case that someone manages to inflitrate or tamper with any of the messages going back and forth between clients, I am in the process of incorporating digital signatures to validate that the messages have not been touched. If the message has been tampered with it would be deleted and the recipeient would not see it. Both parties would have to regenerate their public and private keys and update the database.

4. Experiments

I have tested various designs to implement this. Currently the chat application and RSA are not yet integreated together as I just finished writing the RSA implementation and chat java files.

Online I found various githubs about peer to peer chat apps using java socket programming. I looked at those for ideas on the logic for two clients being able to communicate with eachother. The only difference with those and what I currently have, is peer to peer does not typically (usually) involve a server. My current implmenation does for security reasons.

5. Discussion / Analysis / Conclusion

Items Completed:

- Created Firebase backend servic
- User Registration and Login is fully functional
- Images, and other media files are permitted on the server
- All data within the database is hashed, nothing is in plaintext
- If the user is not signed in, they will be redirected to the welcome screen as they do not have access to the internal functionality without an account
- App uses SMTP protocol to send recovery password email to the user granted that the email address is valid.
- All fields have checks to ensure that everything is formatted properly. Will prevent an user from registering or logging in if an error exists
- Configured / integrated realtime database
- Configured / integrated cloud storage access for users
- Specified the routes to each activity to prevent an user from accessing something they are not.

Todo List:

- Rename project to GSEM, (current title was from one of my many sandbox environments I created).
- Add Instant messaging module into Android Studio Project
- Add the RSA and AES key pair generaator implementation java files to preojct
- Store the Private Key into a keystore
- Send the public key to the Firebase database.
- Process Images using Digital Imaging Processing (found a book online from tutorials point that explains how to scramble and rearrange pixels in a secure fashion).
- Remove / Delete friend,
    - Which will delete any public key relation to that friend so the two can no longer communicate nor see their past messages.

If time is remaining todo list:

- Incorporate SMS authentication by validating user is real by prompting for a verification code that is sent by text message.

References

Babic, Filip. "Using Firebase to Create a 'Simple' Chat Application in Android." Medium. COBE, August 17, 2017. https://medium.cobeisfresh.com/using-firebase-to-create-a-simple-chat-application-in-android-4b32fdbf565e.

"Cryptography : Android Developers." Android Developers. Accessed April 13, 2020. https://developer.android.com/guide/topics/security/cryptography.

DigitalOcean. "SSH Essentials: Working with SSH Servers, Clients, and Keys." DigitalOcean. DigitalOcean, January 12, 2017. https://www.digitalocean.com/community/tutorials/ssh-essentials-working-with-ssh-servers-clients-and-keys.

"Generate Public and Private Keys." Generate Public and Private Keys (The Java™ Tutorials > Security Features in Java SE > Generating and Verifying Signatures). Accessed April 13, 2020. https://docs.oracle.com/javase/tutorial/security/apisign/step2.html.

"GNU Handbook - Distributing Keys." Distributing keys. Accessed April 13, 2020. https://www.gnupg.org/gph/en/manual/x457.html.

Instructables. "Creating a Chat Server Using Java." Instructables. Instructables, September 30, 2017. https://www.instructables.com/id/Creating-a-Chat-Server-Using-Java/.

"Java Cryptography - Encrypting Data." Tutorialspoint. Accessed April 13, 2020. https://www.tutorialspoint.com/java_cryptography/java_cryptography_encrypting_data.htm.

"Java Digital Image Processing Tutorial." Tutorialspoint. Accessed April 13, 2020. https://www.tutorialspoint.com/java_dip/.

Kaustav, Kaustav, and Chanda. "Creating an Asynchronous Multithreaded Chat Application in Java." GeeksforGeeks, April 1, 2019. https://www.geeksforgeeks.org/creating-an-asynchronous-multithreaded-chat-application-in-java/?ref=rp.

"Multi-Threaded Chat Application in Java: Set 2 (Client Side Programming)." GeeksforGeeks, June 17, 2017. https://www.geeksforgeeks.org/multi-threaded-chat-application-set-2/?ref=rp.

Payán, Abel Suviri. "Create RSA Key on Android for Sign and Verify." Medium. Medium,

March 26, 2019. https://medium.com/@abel.suviri.payan/create-rsa-key-on-android-for-sign-and-verify-9debbb566541.

"Public Key Encryption." Tutorialspoint. Accessed April 13, 2020. https://www.tutorialspoint.com/cryptography/public_key_encryption.htm.

RichardCypherRichardCypher 12366 bronze badges, and Marvin PintoMarvin Pinto 16333 silver badges1212 bronze badges. "Implementing RSA into an Android Messaging App." Software Engineering Stack Exchange, December 1, 1961. https://softwareengineering.stackexchange.com/questions/137664/implementing-rsa-into-an-android-messaging-app.

Somnium, SomniumSomnium 2, G. H. FaustG. H. Faust 69655 silver badges44 bronze badges, edc65edc65 31.1k33 gold badges2828 silver badges8787 bronze badges, Todd LehmanTodd Lehman 1, trichoplaxtrichoplax 10.5k66 gold badges4444 silver badges7676 bronze badges, DenDenDoDenDenDo 2, et al. "Rearrange Pixels in Image so It Can't Be Recognized and Then Get It Back." Code Golf Stack Exchange, May 1, 1964. https://codegolf.stackexchange.com/questions/35005/rearrange-pixels-in-image-so-it-cant-be-recognized-and-then-get-it-back.

Stanoyevitch, Alexander. *Introduction to Cryptography with Mathematical Foundations and Computer Implementations*. Boca Raton: Chapman & Hall/CRC, 2013.

"Store RSA Public and Private Key to Mysql Database." Store RSA public and private key to mysql database (Security forum at Coderanch). Accessed April 13, 2020. https://coderanch.com/t/651828/engineering/Store-RSA-public-private-key.

"Understanding RSA Algorithm." Tutorialspoint. Accessed April 13, 2020. https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_understanding_rsa_algorithm.htm.