Encrypted SMS

Senior Design

Jacob Holz, Landon Moon, Parker Steach, Gilbert Lavin, Nam Huynh

UTA Department of Computer Science and Engineering

Background

Most modern messaging apps today utilize the standardized RCS protocol or a proprietary protocol to send and receive messages. While these protocols are mathematically sound, the messages are sent though intermediary servers that are owned by the same developers the made the application. This poses the risk of a built in backdoor to view your secure messages. These backdoors can then be used by the developers to view your messages either for themselves or for law enforcement. Additionally, due to the possibility of your messages being stored on these servers, bad actors can attack these servers more easily than your personal device.

The SMS protocol came before the RCS protocol and is unencrypted in its nature. Additionally, the SMS protocol was built on a separate network of telecommunication servers than the internet. This is why SMS can be used even if internet service is not available.

ESMS tries to remedy these security flaws in other messaging apps by utilizing the SMS protocol. By wrapping the unsecure SMS protocol with a local encryption layer on the user's phone, ESMS is fundamentally unable to implement any backdoor or store any data about the message contents on external servers.

Key Requirements

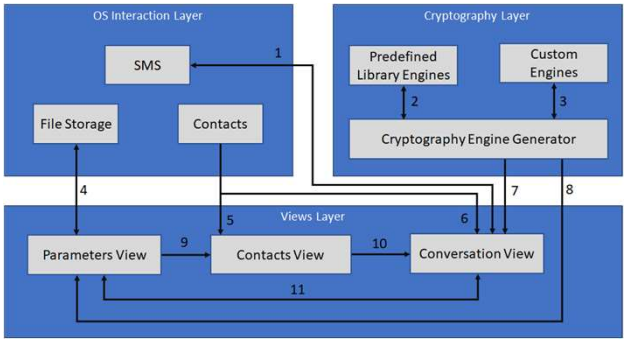
In our System Requirements Specification document, we outlined many requirements. Listed below are the key requirements found in that document:

1. ESMS shall be available for Android systems through a downloadable APK and the Google Play Store.
2. ESMS shall have a UI/UX design that follows the industry standard for messaging applications.
   - Requires a 'Contact view'
   - Requires a 'Conversation view'
   - Requires a 'settings view'
3. ESMS shall have 'good' performance and have quick loading times.
   - Requires message delays are less than 10 seconds.
   - Requires app startup takes less than 5 seconds.
4. ESMS shall implement multiple secure and unsecure encryption algorithms.
   - Requires the AES algorithm.
   - Requires the DES algorithm.
   - Requires the Caesar Cipher Algorithm.
5. ESMS shall exclusively use the SMS protocol to send and receive messages. Additionally, ESMS shall not use internet protocols by any mean.

Architectural Design

The internal structure of ESMS is divided into three layers to facilitate interaction with the Android API, cryptography algorithms, and the user. The details are shown to the right in the Architectural Design Data Flow diagram.

The OS Interaction layer's purpose is to provide a clean and simple service that provides the rest of ESMS with the required functions. Retrieving SMS, Contact, and File Storage information from the Android API has many complex options that are not required be ESMS. This layer reduces the feature set from the Android API such that ESMS development is easier and more structured. This layer also provides no access to internet services.



The Views layer was designed to fulfill the 2nd key requirement. Each of these views are very common among commercial message apps. Each view represents a screen which requires relevant services from the other layers.

The Cryptography layer was designed to fulfill the 4th key requirement. Each encryption algorithm must be able to encrypt and decrypt strings of characters; this functionality is provided the Cryptography Engine Generator sub-section. This subsection has two sources of encryption algorithms from the other sub-sections, predefined and custom engines.

Implementation Details and Test Plan

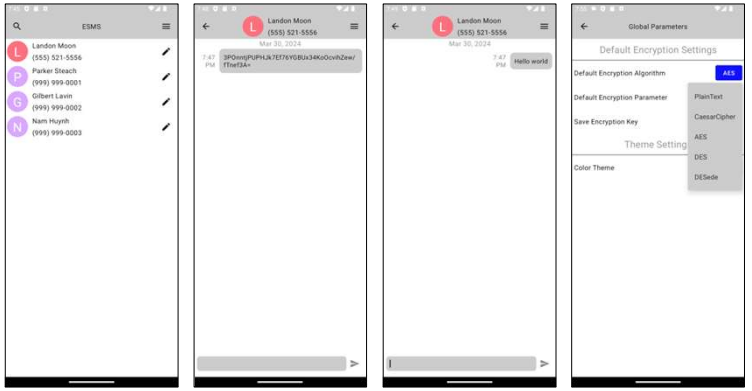
Android Development

ESMS was developed using the Jetpack Compose framework within the Android Studio IDE. The initial development focused on basic navigation and defining the interfaces between each of the layers and their sub-sections. ESMS follows the MVVM (Model, View, ViewModel) architecture due to the structure of Jetpack Compose.

The most important implementation detail of ESMS is how the cryptography engines were defined. Jetpack Compose is a framework for Kotlin which is an extension of Java. Java provides the AES and DES encryption schemes in the javax.crypto package. Custom implementations are all considered unsecure and simply fulfill the encrypt() decrypt() requirement from the Cryptography Engine Generator interface

Testing

Testing of the app was done using the Android Studio debugger and live use on our personal phones. Additional testing and feedback were performed by the amazing beta testers for ESMS.



Conclusions and Future Work

- ESMS provides a truly secure way to send messages between two phones which doesn't not depend on possibly backdoored internet servers.
- Currently ESMS has all the required features outlined in our SRS and follows the design outlined in our ADS and DDS.

Future Work

- Further research of asymmetric encryption implementation
- Research of technical details of MMS and its implementation

References

1. Avoidthehack! Here are 5 reasons to stop using whatsapp. https://avoidthehack.com/stop-using-whatsapp, Aug 2022.
2. Developer policy center. https://play.google.com/about/developer-content-policy/
3. Devin Partida. Are discord messages encrypted? how safe is discord? https://rehack.com/trending/culture/are-discord-messages-encrypted/, Jul 2023.
4. Kendall Aronson. Groupme vs. whatsapp: Best messaging app? https://screenrant.com/groupme-vs-whatsapp-best-messaging-app/, Jan 2023.
5. Richard Stallman. Reasons not to use skype. https://stallman.org/skype.html.
6. Skskit Team. Find out the advantages and disadvantages of teams. https://www.syskit.com/blog/10-pros-and-cons-of-microsoft-teams/, May 2023.

October 2023

Charts

Feel free to add your data to these charts and add them to your poster.