Department of Computer Science & Engineering
The University of Texas at Arlington

Project Charter
CSE 4316: Senior Design I
Fall 2023

ESMS
Encrypted SMS

Gilbert Lavin
Jacob Holz
Landon Moon
Nam Huynh
Parker Steach

Revision History

Revision	Date	Author(s)	Description
0.1	9.13.2023	LM	document creation
0.2	9.15.2023	ALL	first rough draft
0.3	9.22.2023	ALL	first complete draft

CONTENTS

List of Figures

1 Problem Statement

In the modern age, people and businesses use messaging apps or text messages in order to communicate with each other. Messaging apps claim to be secure but use internet protocols to send the data through centralized servers. These servers expose a possible security risk to your private messages. If a user does not have internet access but has cell service, SMS messaging is used. The SMS messaging protocol is insecure due to information being unencrypted. We believe people deserve an independent and secure way to message others without the possibility of a third party viewing your messages.

2 Methodology

We will build an app that allows users to message each other with a truly private service. This app will not utilize relay severs we control and will only communicate over SMS. When a user sends a message, the app will encrypt the contents, and then directly send the contents to the recipient with SMS. The recipient will decrypt the contents and read the message normally without the possibility that the message was read by a third party.

3 Value Proposition

Any entity, whether that is a single user or a multi-million-dollar business, may have the need to send information securely. Additionally, in the modern age, there is a greater risk of malicious hackers getting access to your data from the services that you use. These services might save messages on an internet server that has the possibility of being attacked or contain a backdoor created by developers. By using a secure messaging solution, we can eliminate any entry points that would put the company's data at risk. By investing in ESMS, business employees can securely send messages to co-workers containing company secrets without the risk of any 3rd parties accessing the content of their messages.

4 Development Milestones

- Project Charter first draft - September 2023
- System Requirements Specification - October 2023
- Architectural Design Specification - November 2023
- Demonstration of TBD feature - December 2023
- Detailed Design Specification - February 2024
- Demonstration of TBD feature - February 2024 - sprint 1
- Demonstration of TBD feature - March 2024 - sprint 2
- Demonstration of TBD feature - March 2024 - sprint 2
- CoE Innovation Day poster presentation - April 2023
- Demonstration of TBD feature - April 2024 - sprint 3
- Demonstration of TBD feature - May 2024 - sprint 4
- Final Project Demonstration - May 2024

5 Background

Messaging apps send data one of two ways: through SMS which is unencrypted, or through the internet which uses a server or servers to relay messages. Nowadays phones use modern protocols that are encrypted but are sent using internet protocols. These protocols are secure but require the user to have mobile data and trust the relay server owners. If a user is in a remote location, they are less likely to have a consistent internet connection. When this happens, messages will be sent using SMS which is unencrypted and insecure as mentioned before. There are also apps such as WhatsApp and Facebook Messenger that are available but utilize a server to relay information between phones. This exposes two risks to anybody who uses these services. The first is that user information has the possibility of being accessed through intentional means by the service. This could be a backdoor that the company uses to moderate your messages, or for police to see your 'encrypted' messages. Secondly, since your information might be saved on a server, attackers don't need to attack you directly to get your messaging history. Users depend on their messages being either not saved or having no security vulnerabilities. ESMS avoids these risks by using the SMS protocol which can be found on any phone and is a peer-to-peer protocol. Encrypting SMS messages doesn't require our internet server to handle requests and insulates the encrypted data from systems that may have the knowledge required to decrypt the message besides the destination system. This gives security guarantees to the messages that people and businesses send every day.

Businesses, like any other users, need to communicate and send messages. Businesses have a lot of internal communication between coworkers that can contain company secrets. Most companies depend on service providers similar to the ones mentioned above which suffer from the security risk of those providers. Services like Microsoft Teams have become very popular for communication within medium-sized and large organizations. Companies that are focused on security consider all of their attack surfaces which include their internal service communication. Because this information is being sent to a service outside of the business, it adds one more way company information might get leaked. The pitfall of this approach is that conversation history is stored on a central server, SMS on the other hand communicates the information and then the device saves it. SMS does not allow attackers the ability to view unencrypted previously sent messages.

ESMS has the opportunity to provide a single solution to securely send messages between any two phones*. This provides a security guarantee for both individual users and internal communication at a large company.

* Apple does not allow 3rd party SMS applications, so this app will only be available for Android devices.

6 Related Work

Most companies use services such as Microsoft Teams, WhatsApp, and others for their in-house messaging and communication. However, all of these services have issues with them, most of them are security related or malware related. Microsoft Teams has security issues, users can upload malicious files onto Teams channels and there is nothing checking to make sure they are safe [5]. Whatsapp, while secure, collects large amounts of user data, which is then stored company servers. Providing more opportunities for users information to get taken [2]. Groupme may allow users to to have large group chats and is very popular in colleges but it lacks the ability to use End-to-end encryption meaning not only the people sending and receiving the chats have access to the contents of messages but also hackers [1]. Discord has the same issue as Groupme, which is the lack of End-to-end encryption. this means mods can read all of your messages, even in private channels [3]. Skype, has had many security breaches and "refuses to say whether it can eavesdrop on calls and messages. That almost certainly means it can do so. Microsoft has changed Skype to make it easier for states to snoop on users. Skype also gave personal

data about a Wikileaks supporter to another company without any legal obligation to do so" [4]. With all that in mind, our app is looking to improve in all of these areas, where our competitors lack. Our app offers End-to-end encryption, will not steal you data since it will be stored locally, does not use WiFi meaning you cannot get hacked through your internet connection and you get virus sent to you since you need to friend the person before you can even start having a conservation

7 System Overview

Our solution to the problem focuses on using SMS, which can be used by any phone, and encrypting the data that is sent. This way we avoid the perceivable security risks of using our own intermediate servers as relays.

ESMS will work by encrypting the desired data locally and passing the data through the SMS channel to be received by the other phone. If the raw text is visible to the user, it will be unreadable because the data has been effectively scrambled in the encryption process. Once the encrypted data is sent, it will be received by the other phone where it can decrypted and read by the other user. The encryption method that will be used will default to a standardized secure algorithm. Other algorithms will be available but will not be as secure; due to this, the user will need to manually change it and a warning will be given. More focus will be put into conventional quantum-cracked algorithms along with a couple of quantum-safe algorithms. Additionally, other non-secure encryption methods will be available so the user can see the encryption

The structure of the app will be divided into two main sections: the front-end and the back-end. The back-end will provide a simple interface for the front-end code so that the details of sending and receiving messages can be ignored. Additionally, the encryption details will be held in the back-end of the code and the details will not be known to the front-end. This is a very generalized structure but the back-end and front-end will have their own architecture.

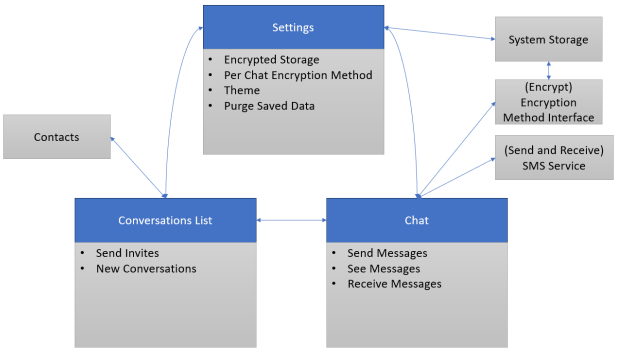

Figure 1: Overview of System by Components

The front end will split into contacts, conversion, and setting settings. The contacts will hold the contact information of the phone along with some extra app-specific data. This includes whether the recipient has the app, and other metadata. If the recipient does not have the app, an option will be available to send a download link to the recipient. The contacts page will also be the place where conversations can start. By clicking on a contact, the user can start a conversation with that recipient. The conversations will work the exact same as a normal messaging app but because it is done through SMS, multimedia can not be sent.

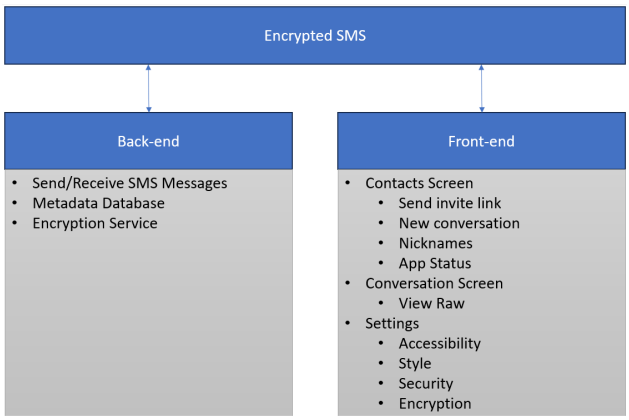

Figure 2: Overview of System by End

Lastly, the settings page will be a page with different sliders and options to change different options. These options include whether or not to have a login screen, the encryption method to be used, and themes. This information will be stored in the back-end and will be able to be accessed by the front-end

The back end will provide a couple of services to the front end. These services are primarily an SMS encryption interface and a local database to store metadata about contacts. These actions can be executed and the information can be queried from a simplified interface. There are two central actions, encrypting/decryption data, and sending messages through SMS.

8 Roles & Responsibilities

The stakeholders of our project are both our sponsor/point of contact, along with any possible user of the app. This can be a single user or a business that plans on using ESMS. The point of contact for this project is primarily the professor of our senior design class. In this fact, anyone can download the app and benefit from the security. Even if the app is not downloaded or known of, the app has achieved its privacy goals due to non-users not being able to access or know of users personal messages. This is all assumed that the world has access to a smartphone that can download and run our app. Even if the public doesn't have a smartphone they are still being affected by the lack of knowledge that has been encrypted thanks to our app. Every day people consisting of family, friends, and classmates will be our source of contacts that will help test and give feed back on our apps creation, along with our sponsor,Shawn N. Gieser, helping implement and provide resources to our team through out the apps creation. The team members are Jacob Holz, Landon Moon, Gilbert Lavin, Nam Huynh, and Parker Steach. The product owner is Jacob with co-ownership being owned by the group. The scrum master will be Gilbert and Parker for the entirety of the project. The project will be split into two teams, the front end and the back end. Gilbert, Nam, and Parker are all working on the front end. The front end is focused on the GUI consisting of the three screens to keep up with the user's chats, people the user knows, and the settings for the app in total. Jacob and Landon are on the back end working on the encryption and the actual SMS structure to connect our users to each other privately and proficiently. All team members will be product testers, along with all members keeping up with documentation and scheduling. The end goal of the project is to have help from some friends, family, or fellow students that will be the first product testers as well to get outside opinions for our app to help lead to releasing our code to the world in the near future. In conclusion, with the combined effort of all team members- and the help of our sponsor and product testers our app will flourish and be able to move to market.

–MAKE HALF A PAGE –MAKE LESS CASUAL

9 Cost Proposal

ESMS will need at least one of our Google accounts to be upgraded to Android Developer Account. If we are unable to get the developer access, we will not be able to publish our application.

9.1 Preliminary Budget

Item	Amount	Sponsor/Company
Department Provided Funding	$800	CSE department
Android Developer Account*5	-$125	Google

Table 1: Overview of highest exposure project risks

9.2 Current & Pending Support

We have access to $800 from the CSE Department. We are unlikely to pursue or receive any other funding at this time.

10 Facilities & Equipment

The app can be created anywhere but for the majority of the time it will be worked on in the senior design lab, ERB, UTA library, WH, UC, and at the given group-mates houses, mainly Gilbert's 55 million dollar Mansion since it will have the most space and entertainment such as a massive saltwater fish tank bigger than most people houses. Nam's box counting, he doesn't have a house. Jacobs virtual reality simulator where he plugs himself in to study and rest. The locations for working on this project are very open and free counting 89 percent of the project is just digital work on a laptop or computer. In retrospect, if there is WiFi available we can use that area to work but preferably our group will stick to regular buildings in UTA to keep things simple and connected. When testing does occur our team plans to download do internal testing and use it among us. Once the app has been tested for a while we will open it to the class, then UTA, then the Dallas/Fort Worth area, then the world. To accomplish this task We will need laptops and computers to run and write the code, maybe some specific libraries to make it easier would help too. Then once the code is functional our group will need phones to text each other. Our team members will also need multiple cellular contracts to actually send each other messages. Along with many willing people to be software testers near the end of our project.

–CLEAN UP/ NO MANSIONS

11 Assumptions

The following list contains critical assumptions related to the implementation and testing of the project.

- Cryptographically secure encryption algorithms should not be implemented manually, so trusted implementations of any algorithms to be used must already be available for our chosen platform.
- SMS is a very insecure data transfer method.
- Distribution of an End-to-End encrypted communication application is and will continue to be legal for the foreseeable future.
- All users have unlimited texts, so inflated character counts will not be an issue.
- Modern SMS-capable devices are sufficiently computationally powerful to encrypt and decrypt messages quickly.

12 Constraints

The following list contains key constraints related to the implementation and testing of the project.

- Final prototype demonstration must be completed by May 1st, 2023
- Mobile devices are less computationally powerful than many static devices, so secure cryptography will introduce a noticeable lag time.
- Total development costs must not exceed $800
- Any stored data must be able to be encrypted at the user's discretion
- Cannot use the internet for sending messages

13 Risks

Risk description	Probability	Loss (days)	Exposure (days)
Lack of specific documentation	0.50	4	2
Unforeseen complexity in detecting receipt of SMS	0.50	10	5
Environment configuration errors	0.90	3	2.7
Global pandemic	0.05	20	1
Vehicular malfunction	0.50	1	0.5
Inconsistent graphics between phones	0.20	0.5	0.1

Table 2: Overview of highest exposure project risks

14 Documentation & Reporting

14.1 Major Documentation Deliverables

14.1.1 Project Charter

At the end of every sprint, we will review our work and confirm we are still following the charter. If any changes have happened, we will discuss and update the charter as needed. The initial version of the project charter will be completed by September 22. We will complete the charter before we begin development on September 25.

14.1.2 System Requirements Specification

After the document is created, we will revisit the document towards the end of each sprint and if any changes have happened, we will discuss and update the charter as needed. The initial version of the SRS will be completed by October 6. We will have a final version of the SRS by October 16.

14.1.3 Architectural Design Specification

After the document is created, we will revisit the document towards the end of each sprint and if any changes have happened, we will discuss and update the charter as needed. The initial version of the ADS will be completed October 30. We will have a final version of the ADS by November sixth.

14.1.4 Detailed Design Specification

After the document is created, we will revisit the document towards the end of each sprint and if any changes have happened, we will discuss and update the charter as needed. The initial version of the DDS will be completed February sixth. We will have a final version of the DDS by February 13.

14.2 Recurring Sprint Items

14.2.1 Product Backlog

Once the SRS has been finalized, the team will take the SRS and turn them into product items. The team will take a group vote to prioritize which items to complete in the sprints. Trello will be used to maintain the product backlog.

14.2.2 Sprint Planning

Sprints will be planned by the scrum master with input from the rest of the team. There will be a total of 8 sprints from September 11 to April 24.

14.2.3 Sprint Goal

Several sprint goals will be determined by our class syllabus. More development-focused sprints will be determined by the scrum master. Three to five days before our sprint begins we will email/meet with our customer and inform them of our sprint goal.

14.2.4 Sprint Backlog

The sprint backlog will be determined by the scrum master, as they plan out each sprint. Similar to the product backlog, our spring backlog will be maintained using Trello.

14.2.5 Task Breakdown

At the start of each sprint we will let individual members claim any specific tasks they want to work on. If there are any disputes or leftover tasks, the scrum master will distribute those tasks. We will use a shared Excel file to track our time.

14.2.6 Sprint Burn Down Charts

Jacob will use Excel to generate our sprint burn down chart. The specific contributions of each member are stored and calculated in the excel document. The format will be a standard Excel chart of the burn down form with the standard colors.

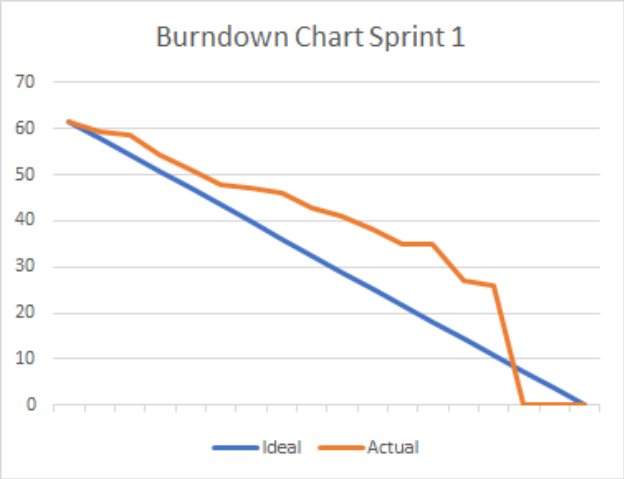

Figure 3: Example sprint burn down chart

14.2.7 Sprint Retrospective

On the last Friday before a sprint ends we will have our sprint retrospective. In these meetings our members will discuss aspects that went well, what could've been improved, and what needs to be changed.

Anything notable, that fits into the three aforementioned categories, will be written down in the group document. Any personal feedback will be recorded by that individual.

14.2.8 Individual Status Reports

We will have group meetings three times a week Mon/Wed/Fri, in these meetings each member will give a status update. This will include progress on current tasks for this sprint, how much estimated man-hours is left before each task is completed, and if they believe assistance is required to complete their tasks.

14.2.9 Engineering Notebooks

As of Fall 2023, we will not be required to keep or maintain an engineering notebook

14.3 Closeout Materials

14.3.1 System Prototype

The final system prototype will be a set of software that will allow people to message each other, securely, through SMS. This will be demonstrated at the end of the Spring 2024 semester, we will provide phones to show the app. We will be able to toggle plain text or show the encrypted messages. We will do a Prototype Acceptance Test with our customer showing functionality.

14.3.2 Project Poster

Our poster will include information about the security issues with modern day messaging services and explain how our app is different and secure. Dimensions (TBD). Our poster will be delivered with out project demonstration at the end of April / early May.

14.3.3 Web Page

Our web page will contain our goal for this project, explain the issue with modern messaging systems, and a visual of how our app is different. This will delivered with our project at the end with our project poster, April 30 (guess).

14.3.4 Demo Video

The demo video will provide a video going over the main use-cases. This will explain how our app works at a very surface level, so anyone can understand how to use it and what it does and does not do. Ideally it would only be 1-2 minutes as the functionality is very simple, but powerful.

14.3.5 Source Code

We will use git as our version control system, and maintain our code using GitHub. The source code will be provided as it will be open source. We will use the MIT license, it will be listed in a Readme file.

14.3.6 Source Code Documentation

Documentation will be provided in the code near the implementation. This documentation will follow a predefined standard that documents function/class purpose, side effects, and context. Large-scale implementation details will be laid out in the architectural design specification.

14.3.7 Hardware Schematics

This project is software-focused and does not require any hardware schematics.

14.3.8 CAD files

This project is software-focused and does not require any CAD files.

14.3.9 Installation Scripts

Our app will preferably be published on the Google Play Store, so installation will be handled automatically. Additionally updates will be executed through the Google Play Store, so the user doesn't have to manually update the app.

14.3.10 User Manual

Our app interface will mimic modern messaging apps, so it will be intuitive for most users, though we will make simple instructions available in context specific information blobs. Any encryption-related information will be handled in the background, abstracted away from the user. There will be some UI documentation, but it should not be required to use the app.

REFERENCES

[1] Kendall Aronson. Groupme vs. whatsapp: Best messaging app? https://screenrant.com/groupme-vs-whatsapp-best-messaging-app/, Jan 2023.

[2] Avoidthehack! Here are 5 reasons to stop using whatsapp. https://avoidthehack.com/stop-using-whatsapp, Aug 2022.

[3] Devin Partida. Are discord messages encrypted? how safe is discord? https://rehack.com/trending/culture/are-discord-messages-encrypted/, Jul 2023.

[4] Richard Stallman. Reasons not to use skype. https://stallman.org/skype.html.

[5] Skskit Team. Find out the advantages and disadvantages of teams. https://www.syskit.com/blog/10-pros-and-cons-of-microsoft-teams/, May 2023.