ChaCha20 Private Messenger Translator

Software Project Management Plan

Dominic Oertel

Towson University

April 2021

Welcome to the SPMP of the CC20 Private Messenger-Translator. The purpose of this document will be to walk the reader through the various elements of the software development process, organizational attributes, and offer a general preview of the finished product.

This software is being developed with the understanding that we are living in a world where surveillance has increasingly become the norm of everyday life and true privacy is a dying concept. Digital technology has afforded us many benefits since the beginning of the information age, and it is the view of some, including the developer of this project, that as citizens of a free republic we have a fundamental right to confidentiality, integrity, and accessibility in our technology.

Tables of contents

Executive Summary

The ChaCha20 Private Messenger Translator will not in itself be a messenger application. Instead, it is designed to be used in tandem with insecure channels of communication i.e. messenger apps such as iMessage, WhatsApp, Facebook Messenger, etc. Users may use this software to encrypt their messages in a quick and safe manner, making it very difficult for owners of conventional messengers to be able to snoop in on their data. Intuitively, this software is also required on the receiving end of messages to perform decryption.

A GUI will be supplied to the user that will walk them through each process, making the software very simple to use while offering some insight into basic cryptographic concepts. Users will only need Python 3, its standard libraries, as well as one dependency: A library by the name of pycryptodome, which will supply the backend program with the ChaCha20 encryption scheme.

Deliverables

Project Diagrams – 3/31/2021

SPMP – 4/09/2021

Back-end Translator and Key Establishment program – 4/23/2021

Front-end GUI program – 5/7/2021

Final Testing and Submission – 5/14/2021

Glossary

Developer – The author of this SPMP and project, Dominic Oertel

Supervisor – The professor who mandated this project, Rebecca Broadwater

CC20 PMT – ChaCha20 Private Messenger-Translator, the subject of this report

Security – the inability of third parties to intrude on user data

En- Decryption – translation of readable messages into almost unbreakable code and back

Channels of communication – messengers, SMS, etc.

ChaCha20 – Encryption algorithm used by Secure Messenger Application

SPMP – Software Project Management Plan

GUI – Graphical User Interface

Project Organization

The project contains two main programs. First, the backend encryption/decryption code, which will serve the software's primary purpose of taking user input, performing numerous complex bit-by-bit operations on it, and returning the corresponding output. The backend program will also be responsible for key establishment, an advanced cryptographical procedure using a well-known process called the Diffie Helman Protocol. In other words, this first program will be the one performing all mathematical operations as well as ensuring that data generated for the end user will be secure and safe to send over insecure channels.

Second, to implement this translation program, there will be a front-end GUI program that will be responsible for collecting user input and displaying corresponding output. The GUI will act as the sole middleman between the user and backend code. All interactions between the user and CC20 Messenger Application will be done inside of this UI. Thus, it is essential that this front-end program will be simple, ergonomic, and quickly responsive to user input. Thus, in essence, the project is organized into two distinct halves. Most functional aspects will be handled by the backend, the non-functional ones by the front.

All responsibilities for this project will be handled by the sole developer. Responsibilities are comprised of designing the program, determining its organizational layout, compiling adequate documentation and diagrams, completing source code for the front- and backend Python code, debugging to ensure the software operates as outlined, and most importantly, the developer will be responsible for ensuring total and inalienable privacy for the end user of the CC20 Messenger. Absolutely no data will be collected from users of the end-product, as that would defeat the software's purpose.

Managerial Process

The philosophy going into this project was to emphasize convenience for the end user as much as possible. This means that users must be able to install it and its dependencies quickly, and that they are able to quickly grasp its intuitive design. Naturally, functionality is the top objective of development. The software must function properly as advertised with minimal to no bugs or interrupts. After functionality, convenience for the end user as described is the next highest objective.

This project's development is based on the assumption that there is demand for truly private and secure communication between end users. Although far less convenient than using secure messengers as they are, this application seeks to put the means of unquestionable security and confidentiality back into the hands of the consumer. It does this by walking users through each step of the process of key establishment, which is the make-or-break aspect of confidentiality. When using messengers such as SMS, WhatsApp, Snapchat, etc., even under the assumption that the companies in control of these applications are not secretly collecting and storing messages from their users, there is still the problem that the keys they are using are not in the hands of the user alone. Despite their promises, corporations are responsible for user rights and privacy, which have been infringed upon on numerous occasions in the past. The consumer must be the only one with access to their private decryption key to be absolutely certain that there is no chance of it being misused.

Risks of note include the following: bugs, development falling behind schedule, and user irresponsibility. First and foremost, there is always a chance that bugs may slip through testing and may end up in the final product. In the event of this happening, the problems must be made clear to the developer as soon as possible. Depending on the nature of the glitch that occurred, it may be fixed in anywhere from a couple minutes to a few days. In any case, problems with the software will be handled immediately if they are discovered and made known.

Furthermore, in the unlikely event of software development falling behind its planned schedule, counter measures will be taken by the developer to ensure that the finished product will be released on the specified date and time. This may mean working “overtime” in other words more than the standard daily amount allotted to development of the project, or it may mean initializing communication between the developer and supervisor to come to some sort of agreement on how to handle the issue.

Finally, user irresponsibility is the most complex problem to solve, and possibly the only one out of the hands of the developer. This software includes a built-in default key for the user to use if they wish, however, it is not recommended as the cryptanalysts generally agree a different key should be used for each session of use.

Technical Process

This project will be programmed using the Python 3 programming language for all source code. Almost all functions and modules will rely on standard libraries included in Python. The one exception will be the encryption scheme library, pycryptodome. ChaCha20 is a module of that library, and it will be the one used to encrypt and decrypt user messages. Key establishment will be done using standard python mathematical functions. Meanwhile, the GUI will use Tkinter, the most popular and standard python GUI programming library.

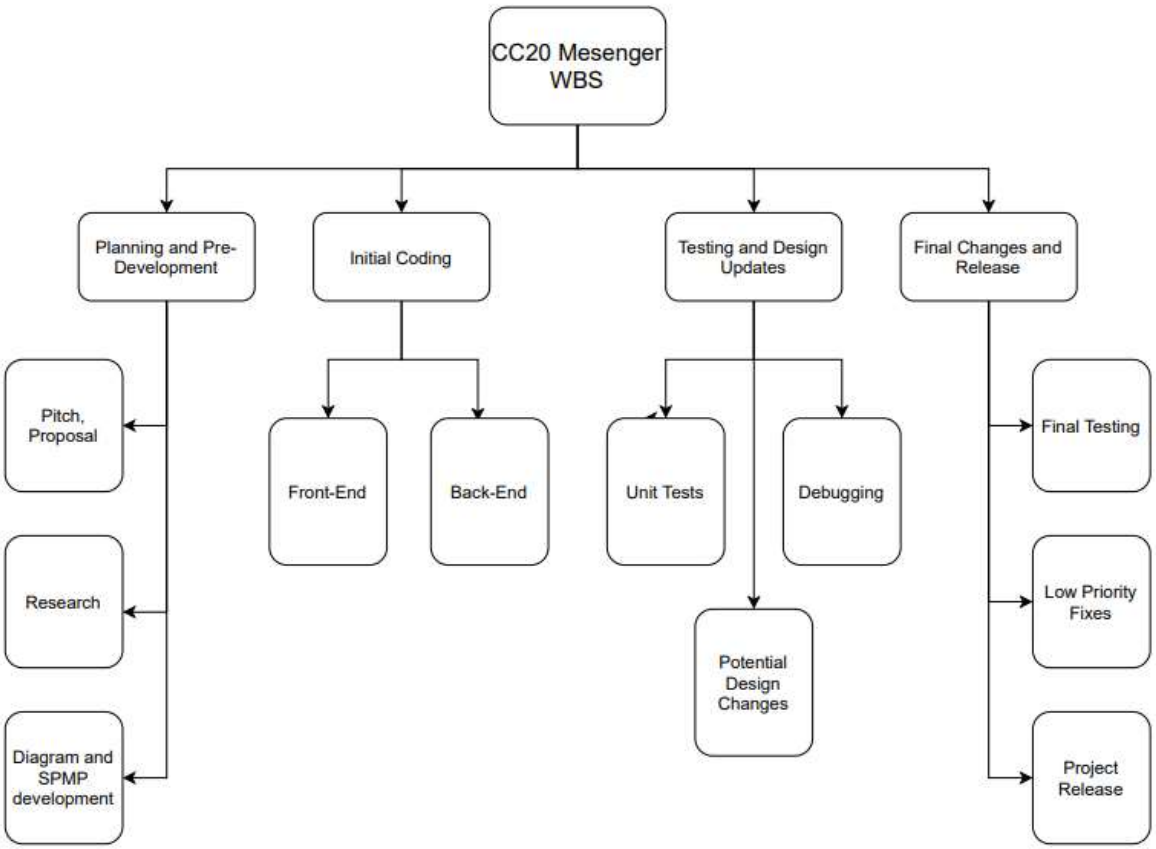
Planning and research of the CC20 PMT development process began in early March of 2021. It was decided early on that the end product would be an encryption – decryption program that is open for anyone to use and offers accessibility to those not familiar with more complex encryption programs, which often need knowledge of the command line to use. Intuitively, to offer the greatest ease of access to users, it was decided shortly after that a GUI would need to be developed in tandem with the backend program. It took several weeks of research and initial testing to determine whether Python 3, Tkinter, and pycryptodome would be used for this project, or Java 8, JavaFX, and java.security. In the end, Python won out because it offers more powerful and simplified encryption schemes, being a newer language with the same cross-platform aspects that Java has boasted for the past 25 years.

It was decided that the Messenger-Translator would offer three primary functions and would take user input via two different methods. The functions are key establishment, ensuring total security on behalf of the user, encryption of user input, as well as decryption of encrypted data. The two channels of user input and output are as follows: Standard text input from the keyboard or clipboard, as well as files from the user as input, namely binary (.bin) files. When messages are encrypted, the operations done on them are in binary. Thus, keeping the input or output in this form offers certain advantages.

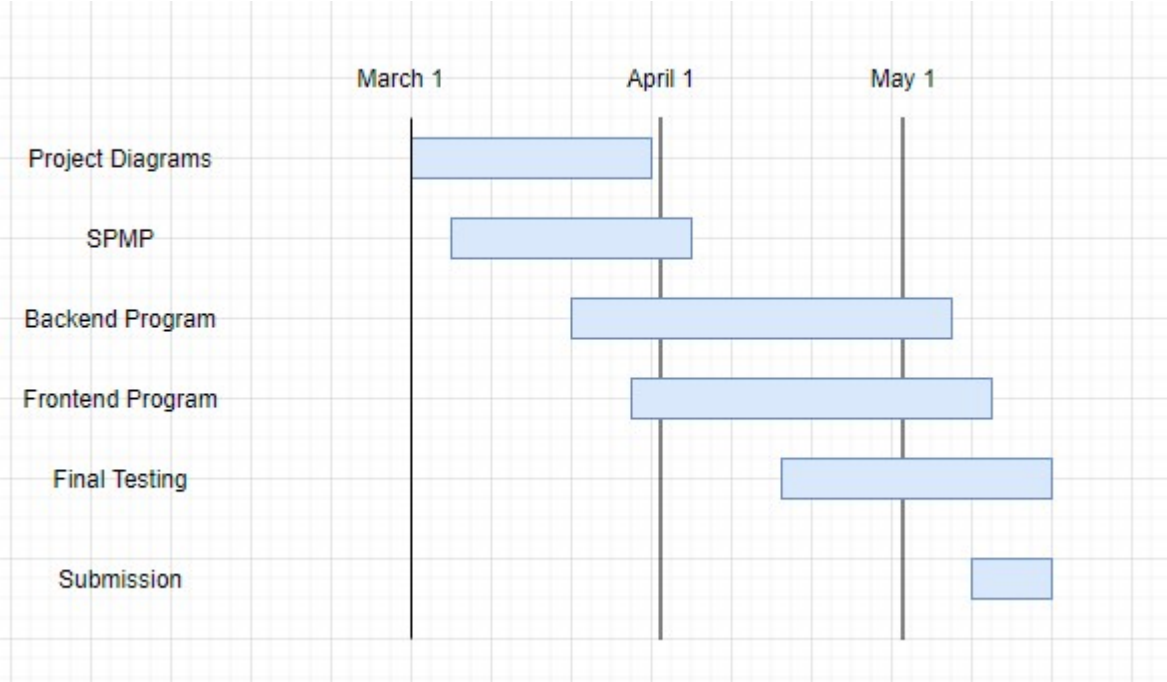
The first coding for this project took place near the end of March. It was decided that back-end programming was first and foremost the top priority and should thus be completed first. Furthermore, the GUI would be heavily dependent on what the back end could and could not offer. Thus, while development of the encryption and decryption mechanisms is well underway, the GUI programming is still in its early stages as of the end of the first week of April.

All Python programming is being done using the JetBrains PyCharm Integrated Development Environment. JetBrains offers all students enrolled in university a license to their software packages free of charge. PyCharm offers an extremely ergonomic experience, providing users with code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supported web development via Django and data science via Anaconda. Diagrams for this project were created primarily using draw.io and documents such as this one were made in Microsoft Office's Word 2019.

Work Elements



Schedule


March 1
April 1
May 1
Project Diagrams
SPMP
Backend Program
Frontend Program
Final Testing
Submission