Functional Specification

SecureYAC Application

|  |  |
| --- | --- |
| Student Name: | Liucija Paulina Adomaviciute |
| Student ID: | 21790411 |
| Student Name: | Eryk Zygmunt Styczynski |
| Student ID: | 21753851 |
| Date of Submission: | /11/2024 |

**Table of Contents**

[1. Introduction 3](#_Toc182047721)

[1.1. Overview 3](#_Toc182047722)

[1.2. Glossary 3](#_Toc182047723)

[2. General Description 3](#_Toc182047724)

[2.1. Product / System Functions 3](#_Toc182047725)

[2.2. User Characteristics and Objectives 3](#_Toc182047726)

[2.2.1. User Characteristics 3](#_Toc182047727)

[2.2.2. User Objectives 3](#_Toc182047728)

[2.3. Operational Scenarios 4](#_Toc182047729)

[2.4. Constraints 4](#_Toc182047730)

[3. Functional Requirements 4](#_Toc182047731)

[3.1. Identity Verification Using X3DH Key Agreement Protocol 4](#_Toc182047732)

[3.2. Message and File Encryption Using The Double Ratchet Algorithm 5](#_Toc182047733)

[3.3. File Compression Using Huffman Coding Algorithm 6](#_Toc182047734)

[3.4. Placeholder 6](#_Toc182047735)

[4. System Architecture 6](#_Toc182047736)

[5. High-Level Design 6](#_Toc182047737)

[6. Preliminary Schedule 6](#_Toc182047738)

[7. Appendices 7](#_Toc182047739)

[7.1. Bibliography 7](#_Toc182047740)

# Introduction

## Overview

The main goal of this application is secure messaging. With Double Ratchet protocol combined with Extended Triple Diffie-Hellman key agreement protocol, this application offers a secure, peer-to-peer messaging platform with a clean and easy-to-use graphical user interface that does not rely on the security of third parties, such as servers or the service provider. The goal is not to obscure the fact that the two users are communicating but to secure the contents of the conversation.

## Glossary

# General Description

## Product / System Functions

1. Identity verification using X3DH key exchange protocol
2. Message and file encryption using The Double Ratchet Algorithm
3. File compression using Huffman Coding Algorithm
4. Message and file sending using Peer-to-Peer protocol

## User Characteristics and Objectives

### User Characteristics

This application is aimed at security and privacy-conscious users who have at least minimal computer skills. The user’s goal is to have secure and private conversations with other users without relying on servers or service providers. The typical customer will use other privacy-enhancing features such as secure browsing and limiting data sharing with third parties. They will carefully select applications they use based on how much data they collect, and share and who has access to that data. The main challenge for the user is to find reliable, trustworthy and secure open-source applications to use for daily activities such as conversations and file sharing. To achieve more privacy and security these users are willing to sacrifice some functionality, speed and convenience.

### User Objectives

**Authentication** – The system should authenticate users based on X3DH key exchange.

**File Compression and Encryption** – The system should compress and encrypt files before sending them.

**File Transfer** – The system should be able to send files between two users.

**Identity** – The system should generate a new identity on user’s request.

**Key Bundles** – The system should allow user to export key bundles that other users can use to initiate messaging with them.

**Messaging** – The system should send messages between two users after they have authenticated each other.

**Peer-to-Peer Connection** – The system should allow direct peer-to-peer connection between users.

**Security** – The system should encrypt all outbound and decrypt all inbound messages.

**User Interface** – The system’s user interface should provide an easy way to generate new identity, export and import key bundles, send and receive messages and files.

## Operational Scenarios

1. Log in
2. Identity verification between two actors
3. Sending message between two actors
4. Sending a file between two actors
5. Identity verification between multiple actors
6. Sending message between multiple actors
7. Sending a file between multiple actors

## Constraints

# Functional Requirements

## Identity Verification Using X3DH Key Agreement Protocol

#### Description

X3DH (Extended Triple Diffie-Hellman) is an asymmetric key protocol that establishes a shared secret key between two parties who mutually authenticate each other based on public keys. Due to the server-less nature of the application, instead of the protocol involving three parties, as defined in the whitepaper, it is modified to involve only two – Alice and Bob.

X3DH has three phases:

1. Bob gives his “prekey bundle” (identity key, signed prekey, prekey signature, one-time prekey) to Alice.
2. Alice uses Bob’s “prekey bundle” to send an initial message to Bob.
3. Bob receives and processes Alice’s initial message.

**Publishing Keys**

Identity needs to be generated only once. New signed prekeys and prekey signatures will need to be generated at some interval (once a week or once a month). The new values will replace the previous ones.

Bob creates a set of elliptic curve public keys.

After sharing a new signed prekey, the private key corresponding to the previous signed prekey will be deleted. One-time prekey private keys will be deleted as Bob exports sets containing them.

**Sending the Initial Message**

To perform an X3DH key agreement with Bob, Alice uses the “prekey bundle” to send Bob the initial message.

**Receiving the Initial Message**

Upon receiving Alice’s initial message, Bob retrieves Alice’s identity key and ephemeral key from the message.

If the initial decryption is successful, the protocol is complete and Bob deletes any one-time prekey private key that was used for forward secrecy. Bob then continues to use SK and keys derived from SK for further communication with Alice.

#### Criticality

Essential.

#### Technical issues

**Implementation**

No Java libraries available for the protocol – it will need to be implemented from scratch.

**Key Exchange**

The parties must exchange the initial key bundle through a secure channel.

**Key Reuse**  
Post-X3DH protocol must randomize the encryption key before Bob sends encrypted data. Failure to do so may cause key reuse and reduced security.

#### Dependencies with other requirements

None.

## Message and File Encryption Using The Double Ratchet Algorithm

#### Description

The Double Ratchet algorithm is used by two parties (Alice and Bob) to exchange encrypted messages using a shared secret key. To prevent the decryption of future messages due to the theft of one party’s keys, the symmetric-key ratchet is combined with the Diffie-Hellman ratchet, which updates the chain keys based on the Diffie-Hellman output.

The parties derive new keys with Diffie-Hellman calculation results for every message. Because of this, the later keys cannot be calculated using the earlier ones.  If the key is unknown, the output data is indistinguishable from random. Diffie-Hellman public values are attached to the message.

In Double Ratchet between Alice and Bob, Alice initializes with Bob’s public key and the initial root key (shared secret). As part of the initialization, Alice generates a new ratchet key pair using X3DH and feeds the output to the root KDF to calculate a new root key and send a chain key. The old root key may be deleted. When Alice sends her first message, she applies a symmetric-key ratchet to her sending chain key. The output is a new message key.

When Alice receives a message from Bob, it will contain a new ratchet public key. Alice applies a Diffie-Hellman ratchet step to derive new receiving and sending chain keys. Then, a symmetric-key ratchet step is applied to the receiving chain to get the message key for the received message.

#### Criticality

Essential.

#### Technical issues

No Java libraries available for the algorithm – it will need to be implemented from scratch.

#### Dependencies with other requirements

The Double Ratchet algorithm depends on the output of X3DH.

## File Compression Using Huffman Coding Algorithm

#### Description

Huffman coding is an algorithm for data compression with the goal of reducing its size without the loss of details (lossless compression). The algorithm uses character frequency analysis to assign each character a code with the most frequent characters being assigned the shortest codes.

This algorithm can be used to compress various types of files that can then be sent using peer-to-peer protocol,

#### Criticality

#### High.

#### Technical issues

The efficiency on the algorithm depends on having frequent characters in the file.

#### Dependencies with other requirements

None

## Placeholder

#### Description

#### Criticality – Essential

#### Technical issues

#### Dependencies with other requirements

#### Others as appropriate

# System Architecture

# High-Level Design

# Preliminary Schedule

# Appendices

## Bibliography

[1] M. Marlinspike, “The X3DH Key Agreement Protocol,” Signal, Nov. 04, 2016. https://signal.org/docs/specifications/x3dh (accessed Nov. 05, 2024).

[2] M. Marlinspike, “The Double Ratchet Algorithm,” Signal, Nov. 20, 2016. https://signal.org/docs/specifications/doubleratchet/ (accessed Nov. 05, 2024).

[3] M. J. Golin and G. Rote, “A Dynamic Programming Algorithm for Constructing Optimal Prefix-Free Codes with Unequal Letter Costs,” IEEE Transactions on Information Theory, vol. 44, no. 5, pp. 1770–1781, Jan. 1998, doi: https://doi.org/10.1109/18.705558.

[4] D. Huffman, “A Method for the Construction of Minimum-Redundancy Codes,” Proceedings of the IRE, vol. 40, no. 9, pp. 1098–1101, Sep. 1952, doi: https://doi.org/10.1109/jrproc.1952.273898..