SMSEncryption Project Requirements Specification

COS730 - Group 1

Version 2.0
May 27, 2014

History

Date	Version	Contributors
05 April	Version 0.1	Henko
10 April	Version 0.2	Henko
11 April	Version 0.3	Henko
12 April	Version 0.4	Henko
21 April	Version 0.5	Hein, Jaco, Luke, Vincent
26 April	Version 0.6	Jaco, Luke
28 April	Version 0.7	Jaco
29 April	Version 0.8	Hein, Jaco
30 April	Version 0.9	Vincent
02 May	Version 1.0	Jaco, Vincent
03 May	Version 1.1	Henko
07 May	Version 1.2	Jaco, Luke, Vincent
09 May	Version 1.3	Jaco
10 May	Version 1.4	Hein, Jaco, Luke, Vincent
11 May	Version 1.5	Jaco, Luke, Vincent
12 May	Version 1.6	Jaco, Luke
14 May	Version 1.7	Henko, Jaco, Luke
15 May	Version 1.8	Jaco, Luke
21 May	Version 1.9	Vincent
22 May	Version 1.9.2	Vincent
26 May	Version 2.0	Jaco, Luke

Group members

Vincent Buitendach	11199963
Luke Lubbe	11156342
Jaco Swanepoel	11016354
Henko van Koesveld	11009315
Hein Vermaak	11051567

Contents

1 Introduction

1.1 Purpose

This document describes the software requirements and specifications for the SMSEncryption mobile application.

The document will be used to ensure that requirements are well understood by all intended stakeholders of this project - such as developers and customers.

1.2 Background

Certain operations within remote parts of South Africa require reliable and confidential transmission of information(text) which cannot be achieved through the more modern means of data transmission, such as GSM or 3G, as these wireless technologies are unreliable in certain remote parts of South Africa. This unreliability is caused by weak, or even no, signal of modern wireless technologies; this unreliability forces the users in these areas to rely on an older message transmission technology - the SMS. These users require some form of encryption to ensure safe transmission of their sensitive data. As traditional encryption functions often produce characters which fall outside the character set that the SMS uses, it is necessary to make use of an encoding scheme that can translate the intended message text into ciphertext that can remain within the limits of the SMS character set. Most encoding schemes (such as base64) increase the length of the message drastically: this may result in the ciphertext exceeding the maximum amount of characters allowed per SMS.

1.3 Scope

The goal of this project is to create a mobile application that can translate the message text into an acceptable ciphertext that can be sent per SMS without exceeding the length that is allowed per SMS. When the message(ciphertext) is received on the opposite end of the 2-way communication, the application should be able to decrypt the received ciphertext back into its original plaintext (which will allow the receiver of the message to read the message, without loss of confidentiality). The application must be able to function on more than one platform (i.e. iOS and Android), as well as work cross-platform.

By using the SMSEncryption application, the user will be able to encrypt message text, which can then only be decrypted by using the same application on the receiving end (after a sync process has been completed by each user in the 2-way communication - see below for more information on the sync process). The application will require local authentication in order to gain access to the application and make use of its features.

The benefit of using this application would be that users can send encrypted messages via the SMS technology - ensuring that messages remain confidential between the

sender/receiver of the message, and that messages uphold their integrity (i.e. has not been read/modified by an unknown third party).

1.4 Definitions, acronyms and abbreviations

- SMSEncryption - The name of the project which will allow users to encrypt and decrypt text, with the main purpose of it being sent as an SMS, or via other messaging applications such as WhatsApp, WeChat, Facebook chat etc, if available.

- Message - The text that will be communicated between two parties.

- Plaintext - The unmodified text a sender wishes to privately communicate to a receiver.

- Ciphertext - The result of an encryption algorithm(cipher) performed on the message plaintext.

- Cipher - The algorithm used to translate plaintext into ciphertext.

- Encrypt - The act of converting plaintext, using a cipher, into unintelligible data that cannot be read by any unauthorized parties.

- Decrypt - The act of decoding a ciphertext back into its original form(plaintext), so that the message can be read by the intended receiver.

- Unauthorized - A person/party that is not allowed to view a message sent between two, private, parties.

- User - An authorised user that will interact with the SMSEncryption application.

- Sender - The person/party that authored, and intends to send, a message that will be encrypted via the SMSEncryption application.

- Receiver - The intended person/party that will receive an encrypted message(ciphertext) which can be decrypted by the SMSEncryption application.

- SMS - Short Message Service (SMS) is a text messaging service component a of phone, web, or mobile communication systems. This allows for short messages to be sent to other devices over a network which is not controlled by the sender or receiver.

- SMSC - Short Message Service Centre (SMSC) is a network element in the mobile telephone network. Its purpose is to store, forward, convert and deliver SMS messages.

- GSM - Global System for Mobile Communications (GSM) is a second generation standard for protocols used on mobile devices.

- 3G - Is a third generation mobile communications standard that allows mobile phones, computers, and other portable electronic devices to access the Internet wirelessly.

- Entropy - The randomness contained within ciphertext to ensure that no pattern will emerge when studying the ciphertext.

- Contact - Data stored on the device to represent a person, their mobile phone number, and other data to ensure reliable communication via encryption.

- Synchronisation - The process whereby contact data is matched on more than one device in order to allow a private communications channel via encryption.

- Desyncrhonisation - The event that occurs when contact data, between two devices, is no longer in sync with one another.

- Resynchronisation - The synchronisation process is repeated to ensure synchronization between two contacts.

1.5   Document Conventions

- Documentation formulation: LaTeX

- Naming convention: Crows Foot Notation

- Largely compliant to IEEE 830-1998 standard

1.6   References

- Kyle Riley - MWR Info Security

    - face-to-face meeting
    - email

- Bernard Wagner - MWR Info Security

    - face-to-face meeting
    - email

- Electronic, M., n.d. *One Time Pad Encryption, The unbreakable encryption method.* s.l.:mils electronic gesmbh & cokg.

- Kaliski, B., n.d. *The Mathematics of the RSA Public-Key Cryptosystem.* s.l.:RSA Laboratories.

- *OWASP Mobile Security Project, 2014.* Available from: `<https://www.owasp.org/index.php/OWASP_Mobile_Security_Project>`. [23 April 2014].

- *Design Principles, 2014.* Available from: `<https://developer.android.com/design/get-started/principles.html>`. [23 April 2014].

- *Design Principles, 2014.* Available from: `<https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/Principles.html>`. [23 April 2014].

- Pohl, K. (2010). *Requirements Engineering: Fundamentals, Principles, and Techniques.* 1st ed. Heidelberg: Springer.

- IEEE-SA Standards Board, 1998. *IEEE Recommended Practice for Software Requirements Specifications.* IEEE Std 830-1998

1.7   Overview

The rest of this document will be organized to include the following sections: General Description and Specific Requirements for the SMSEncryption application.

The General Description section will provide an overall background to the reader of this document, for the SMSEncryption application. It contains the following sections: Product perspective, Product functions, User characteristics, Constraints, Assumptions and Dependencies, as well as Apportioning of Requirements.

The Specific Requirements section contains all requirements for the SMSEncryption application, and is organised by application features. This is done in such a way that it will highlight the functions of the application. It contains the following sections: External Interfaces, Functions, Performance Requirements, Logical Database, Design Constraints, and Software System Attributes.

The appendices A, B, and C contain research on encryption conducted by the development team.

Appendix D contains a set of secure design principles relevant to the application.

Appendix E contains a set of design principles from both Android, and iOS, that must be present in the SMSEncryption application for both mobile operating systems.

Appendix F contains i* diagrams we drew during the requirements elicitation process.

Appendix G contains the detailed description of changes to each version of the document.

2 Overall Description

2.1 Product perspective

2.1.1 Description

The SMSEncryption application is a new product that can accommodate the encryption/decryption of text to be sent via SMS, or any other text communications service, such as WhatsApp, that can make use of more modern wireless data communication technologies - if it is available. Message text can be manipulated via any text manipulation application, such as the default Android/iOS keyboard, capable of using the basic GSM character set , that will be used by our application. Any third party keyboard applications such as SwiftKey can also be used.

The GSM character set contains a limited amount of characters, which will, in turn, limit the encryption methods we can make use of, as many encryption algorithms greatly increase both the size of the message, and the number of different characters used. Making use of these encryption algorithms that generate large amounts of characters will be infeasible, as sending a large amount of text via SMS will be expensive - as users generally pay a fixed amount per 160 characters.

2.1.2 System interfaces

The application is usable on both a tablet and mobile phone. Below are some screenshots of the application on both these:
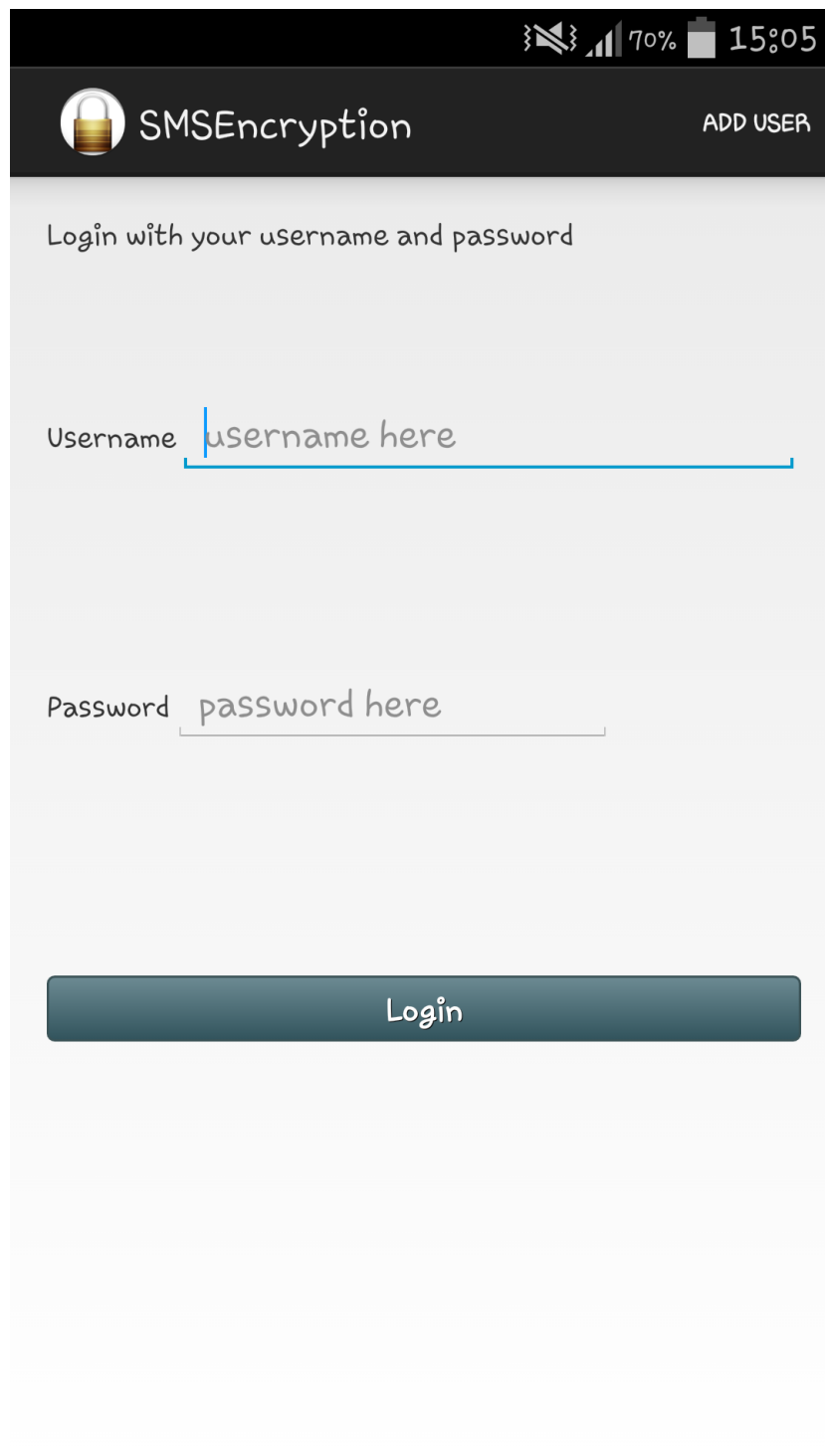
This is the Login Screen which users see when they first open the application.

After selecting "Add user" the user is taken to the Add User screen. The Add User screen is used to add new user accounts to the application.

The user can enter their login credentials to an created account to log into the application.

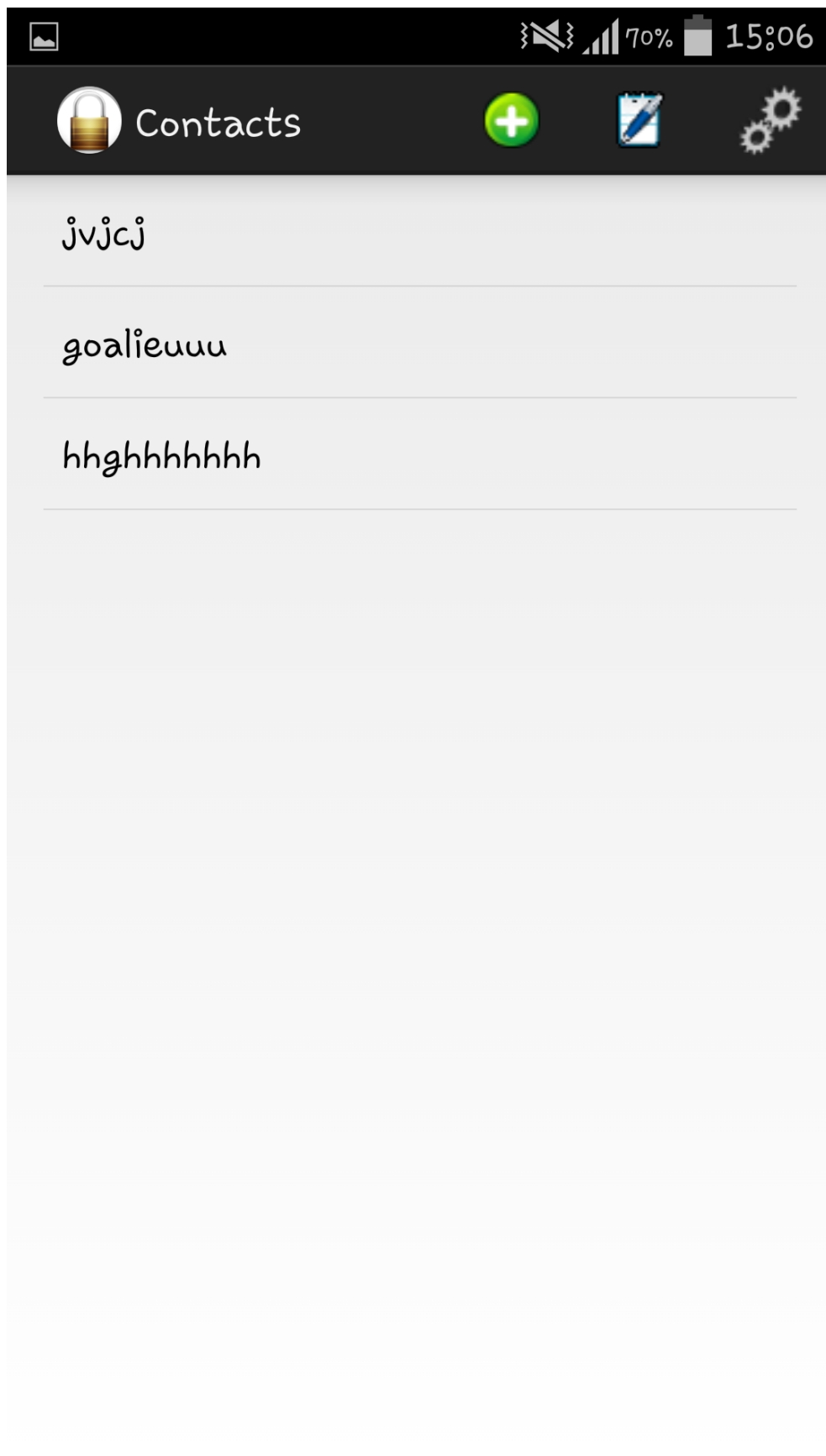After having logged in the user sees the Contacts menu.

After selecting the "plus" symbol the user is taken to the Add Contact screen where they can add a contact.

🔒 Edit Contact   ⬅

Select the contact and edit fields

ȷvȷcȷ   ◢

Contact Name   ȷvȷcȷ

Contact Number   🔷   (optional)

My Key

(188')0OJ>   🔄

Contacts Key

!V:9"*R?(0

📝   Save changes

❌   Delete contact

After selecting the pen and paper icon the user is taken to the edit contact screen where by using the drop down box they can select which contact they wish to edit details for.

After selecting the gear icon a user is taken to the settings page where they can change the logged in accounts username and password.
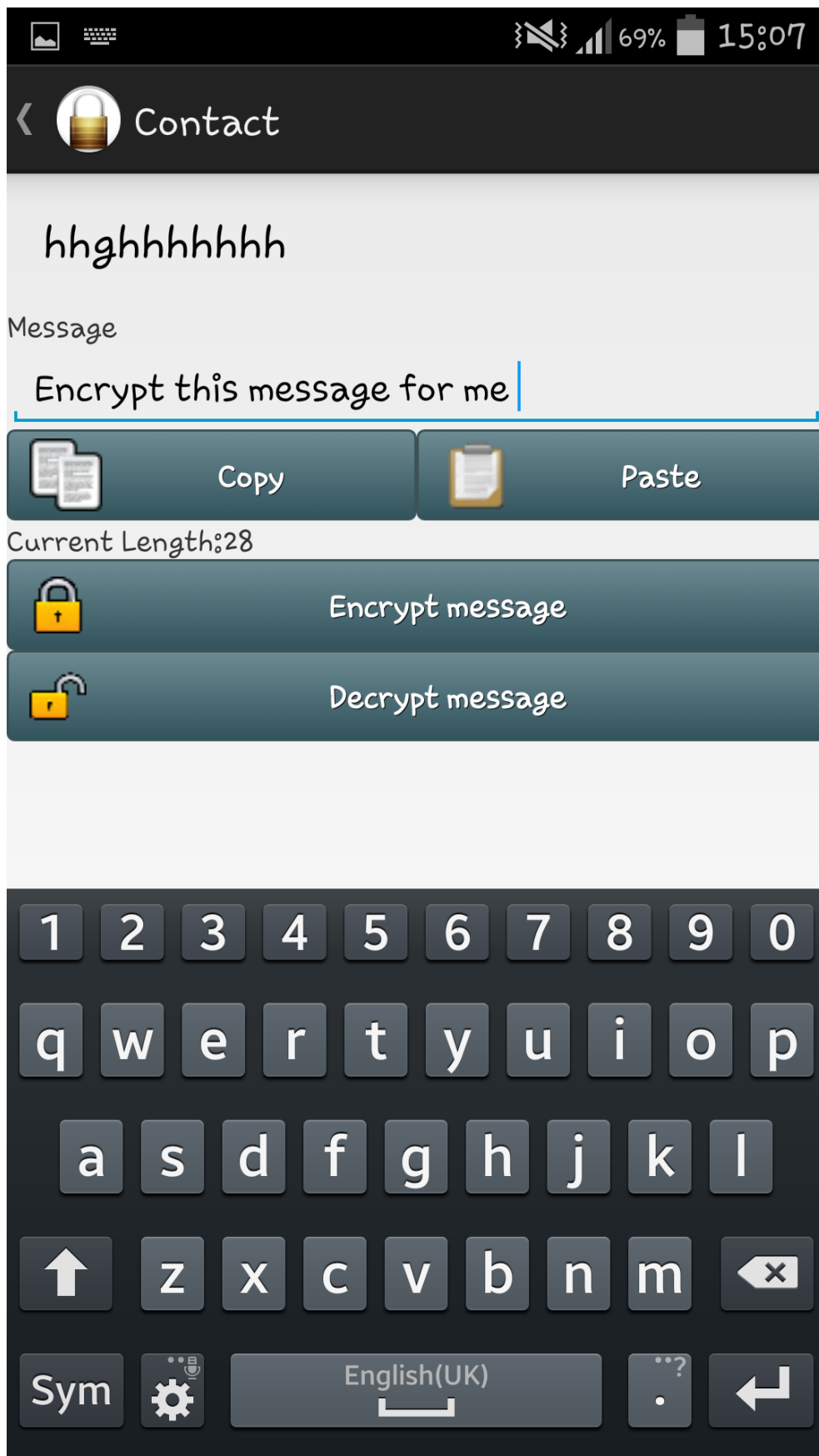
If a user selects a contact from the main menu they are shown the contact messaging menu where they can encrypt a message to/from that contact.
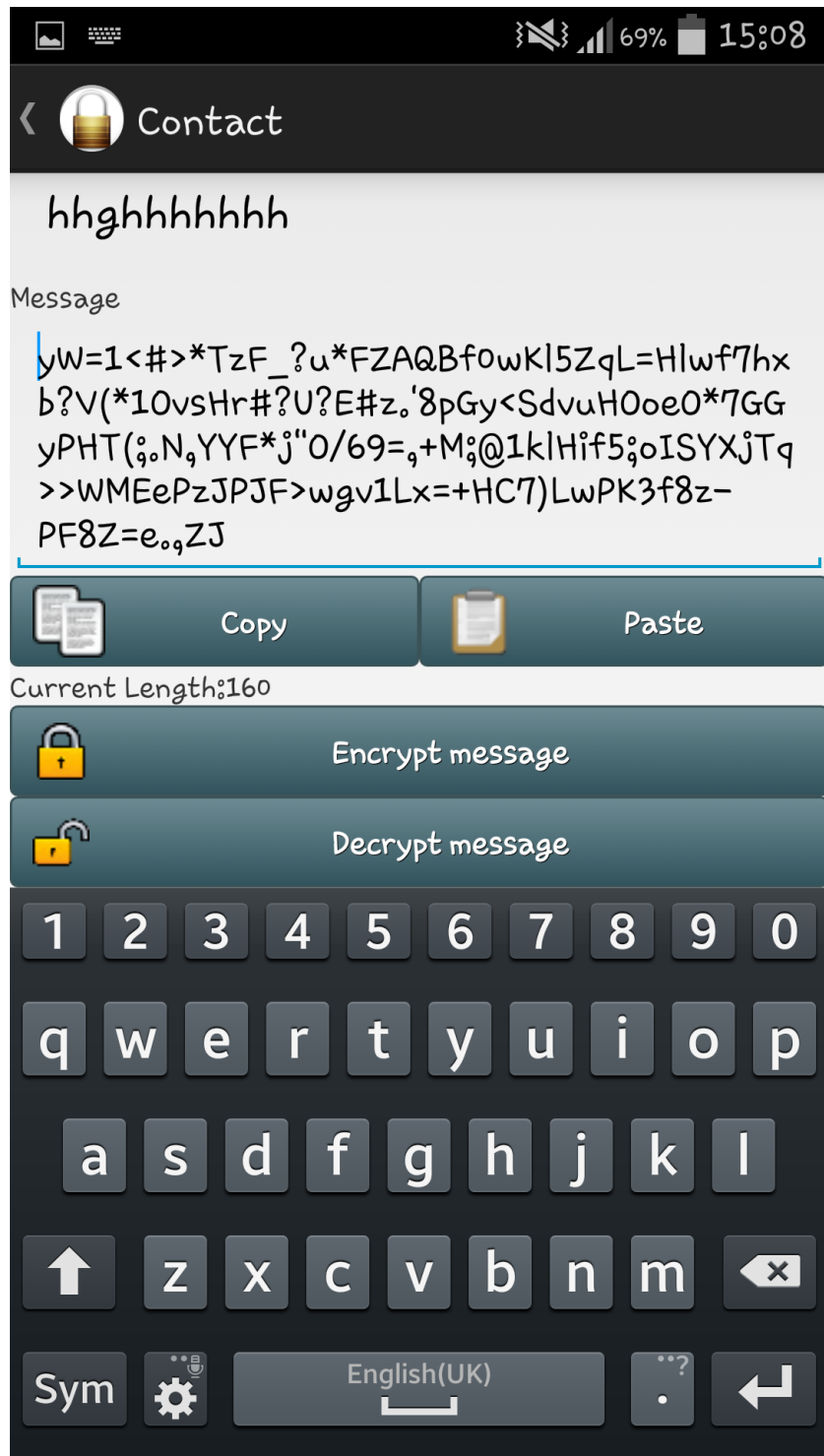
After having entered a message they wish to encrypt the user can select encrypt and the ciphertext for that message will be displayed which they can then send to that contact using SMS, WhatsApp or any text messaging service.

2.1.3 User interfaces

The user interface is what will allow the user to type a message, encrypt it, copy the ciphertext, and paste it into the application that will send the message. On the receiving end, the message received will be copied, and pasted into the SMSEncryption application, which will be used to decrypt the received message. This ensures integrity of the message, as only users of the application will be able to encrypt/decrypt the message if they have synchronized each other as contacts within the application.

2.1.4 Hardware interfaces

The software will run on a mobile device that allows user interaction and text manipulation.

2.1.5 Software interface

The software interface will make use of operating system features, such as the 'clipboard' on the device to facilitate 'copying' and 'pasting' of texts or ciphertexts.

2.1.6 Memory

The device needs minimal storage space, for the application, and the database it creates on the device, uses approximately 2mb.

2.1.7 Operations

- User-initiated operations:
  - Create user account
  - Edit user account
  - Add contact
  - Edit contact
  - Remove contact
  - Enter message
  - Encrypt message
  - Decrypt message
  - Synchronise contact

2.1.8 Use Cases

SMSEncryption Use Case Diagram
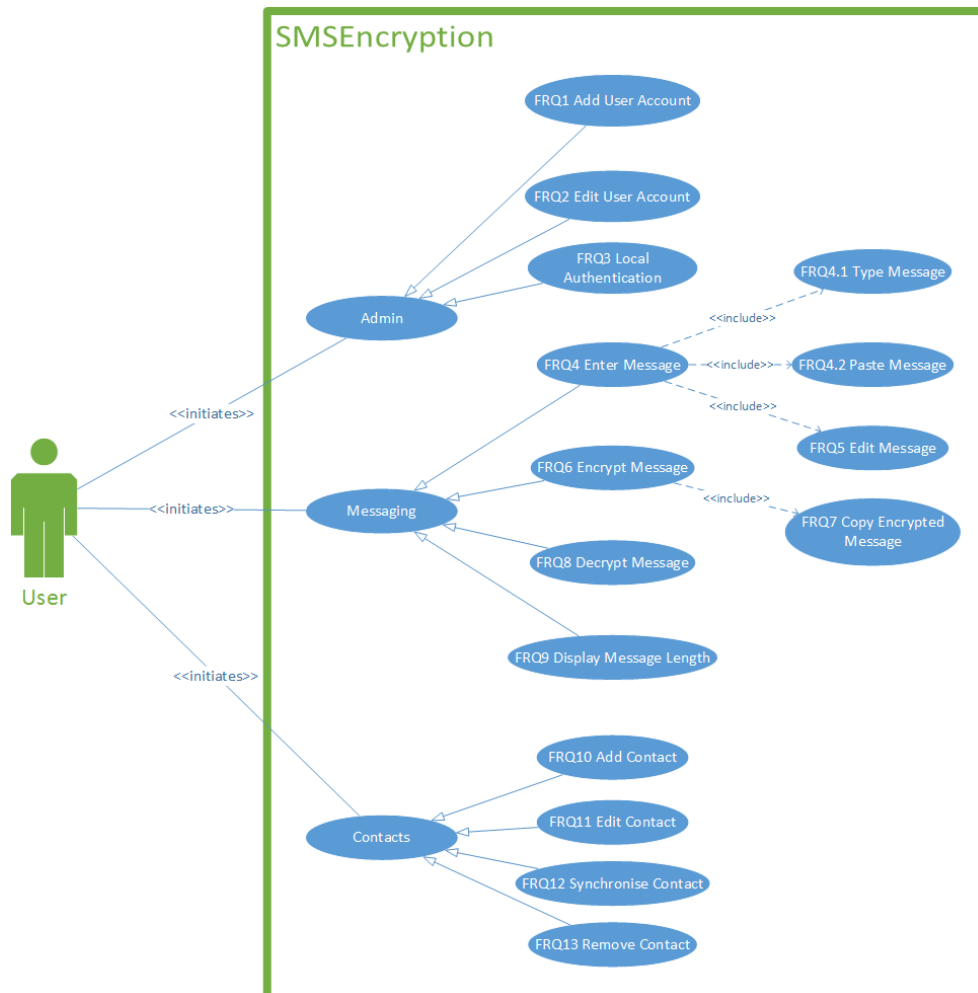
2.1.9 State Diagram

SMSEncryption State Diagram
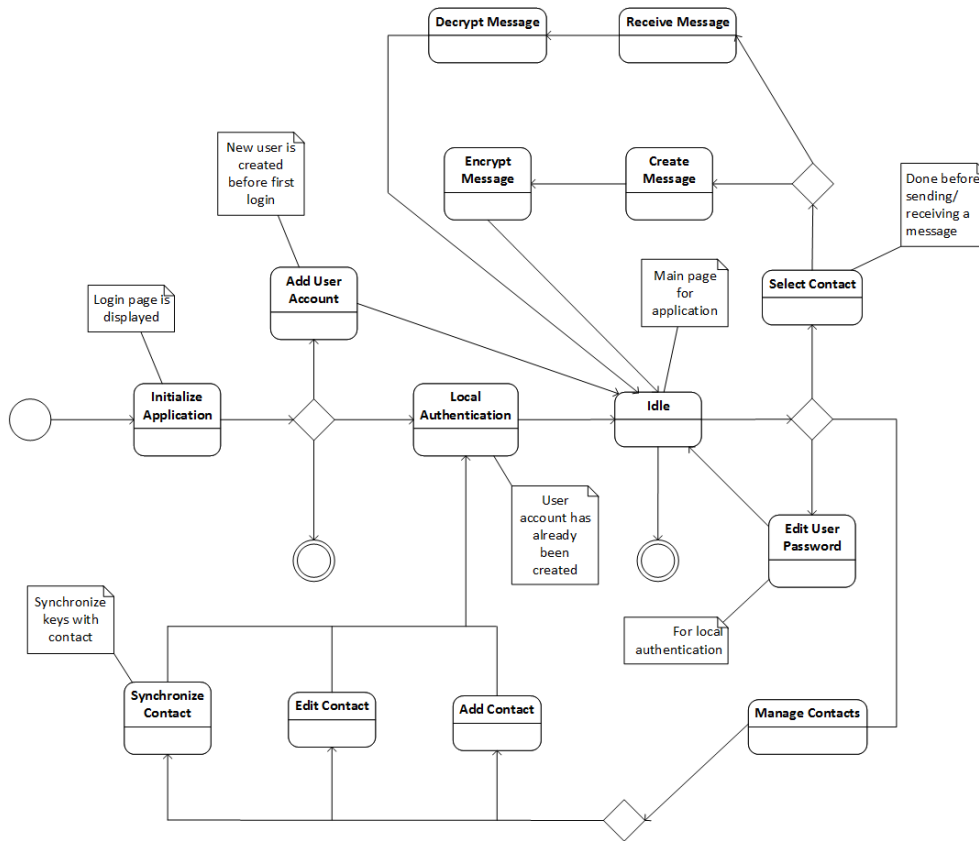
2.2 Product functions

The application is divided into 3 core functionalities: Admin, Messaging, and Contacts.

2.2.1 Admin

- The user should be able to create an initial (compulsory) user account within the application.

    - On first use of the application, a username and password (with confirmation of password) must be created that will ensure that user authentication will take place for future use of the application.

    - The application only allows a single user account per device.

- The user must be able to edit his/her user account.

    - Should it be required, once authentication has validated the user (i.e. he/she is "logged-in"), the user can edit his/her account details.

- The application will make use of local authentication to verify the user before logging him/her into the application, and granting access to all of the application's features.

    - Every time a user wants to use the application, the set password must be provided along with the login details.

- If the provided password (and related details) are entered correctly, the user gains access to the application.

- If the password provided is found to be incorrect, three times in a row, the application will lock for a specified amount of time - preventing access from an unauthorised user, as well as, so-called "brute force" attacks.

2.2.2   Messaging

- Before creating a message within the application, the user must select the intended contact/receiver of the message.

  - The selected contact's details (see below for a more elaborate explanation of the "details" used) will be used to perform the encryption.

- When creating a message within the application, the text can be either typed in, or pasted in via the device "clipboard".

- Once the message text has been entered into the application (via any method listed above), it can be edited within the application.

- Once editing the message text has been finalized, the message can be encrypted.

- The application must allow the encrypted message to be copied onto the device clipboard.

  - The encrypted message (ciphertext), can then, by the user, be copied and pasted into an application that will send the ciphertext to the desired receiver; any messaging application can be used.

- If an encrypted message is received (and pasted into the application), the application must be able to decrypt the ciphertext - if the correct contact (sender of the message) is selected, as the "details" for the selected contact is used to perform decryption.

  - The receiver of the ciphertext should be able to decrypt the message back into its original plaintext, and thus read the intended message.

- The message length should be displayed in the editing process (before encrypting/copying the message from the application).

  - This is to ensure that the message length does not exceed the maximum amount of 144 characters (the remaining 16 characters of the 160 characters are used for synchronization - see below for more details).

2.2.3   Contacts

- A user must be able to add a contact to the application.

  - In order for communication to take place between two devices, i.e. two contacts, they first need to be synchronized.

– A user adds what is called a "contact". The application will require the following: the name of the contact, a locally generated unique word/key (to be provided to the other user with which messages will be exchanged), and the contact's unique generated word/key (which has to be received from the contact with which communication will take place).

– Before the two contacts can be synchronized with one another, both users must provide their unique key to each other.

  * This will synchronize communication between the devices once all contact information has been correctly saved on both devices.

  * If both users add each other as a contact at relatively the same time, synchronization can take place at a greater speed (as the synchronization process on both devices are dependent on one another, and have to be manually done by the users of the application).

- The application must allow a contact to be edited once it has been added.

- The application must allow removal of a contact that has been added.

- The application must allow resynchronization of contacts, should desynchronization occur.

2.3 User characteristics

- There will be only one user "class" (no distinction will be made between normal users, admin users, etc.) that will have full access to all the features provided by the application after local authentication has taken place, since the application only allows for a single user account to be created per device.

- It is assumed that the user has proficient knowledge on how to use the "clipboard" of their device; i.e. copy message text from applications, such as the standard SMS application, and paste it within this application, or vice versa.

- It is also assumed that users perform the device synchronization phase correctly; before exchanging messages, as there is no way for the device to detect errors in synchronization - such as an incorrect, or older (already used) key. The synchronization process remains manual.

2.4 Constraints

- The encrypted message length is limited to 160 characters, regardless of plaintext message length.

- Message text below the 144 character limit (plus a reserved 14 characters for synchronization) will be encrypted into 160 characters.

- The application must make use of the basic GSM character set.

2.5   Assumptions and dependencies

- It is assumed that the amount of characters in the basic GSM character set is 128 for the 7-bit encoding used in GSM.

- It is assumed that the devices being used allows has clipboard functionality (copy/paste functionality) between different interfaces/applications on the device itself.

2.6   Apportioning of requirements

The following are possibilities that can be added to future versions of the sytem:

- SMS capability from within the application.

- Compatibility for other operating systems, such as iOS or Windows Phone.

3 Specific requirements

3.1 External interfaces

Found in requirement	Add User Account : FRQ1
Name of item	Username
Description of purpose	The desired Username which the application authenticates the user
Source of input	User
Destination of output	Application database
Valid range, accuracy, and/or tolerance	Standard GSM 03.38 character set
Relationships to other inputs/outputs	A user account will be bound to the desired username provided by the user

Found in requirement	Add User Account : FRQ1
Name of item	Password
Description of purpose	The desired password which will be associated with the specific username created by the user
Source of input	User
Destination of output	Application database
Valid range, accuracy, and/or tolerance	Standard GSM 03.38 character set
Relationships to other inputs/outputs	A user account will have the password associated with the provided username set by the user to this value

Found in requirement	Add User Account : FRQ1
Name of item	Password Confirmation
Description of purpose	A prompt to re-enter the password provided by the user
Source of input	User
Destination of output	N/A
Valid range, accuracy, and/or tolerance	Standard GSM 03.38 character set
Relationships to other inputs/outputs	The value submitted is used to confirm the correctness of the desired password by the user in order to accommodate for human error

Found in requirement	Add User Account : FRQ1
Name of item	User Account
Description of purpose	The user account that has been created and has been associated with the provided username and password submitted by the user
Source of input	Application
Destination of output	Application database
Valid range, accuracy, and/or tolerance	Checked during creation process
Relationships to other inputs/outputs	The username and password entered by the user in order to create the sole account linked to the application

Found in requirement	Add User Account : FRQ1
Name of item	Error Message
Description of purpose	This message indicates that an error has been found during account creation, and that, accordingly, no account has been created
Source of input	Application error checking
Destination of output	User interface
Valid range, accuracy, and/or tolerance	N/A
Relationships to other inputs/outputs	The details provided during account creation was invalid for one or more reasons

Found in requirement	Local Authentication : FRQ3
Name of item	Username
Description of purpose	The username associated with the user account that the user logs into the application with
Source of input	User
Destination of output	N/A
Valid range, accuracy, and/or tolerance	Standard GSM 03.38 character set
Relationships to other inputs/outputs	The username that was used to create the user account

Found in requirement	Local Authentication : FRQ3
Name of item	Password
Description of purpose	The password associated with the user account that the user logs into the application with
Source of input	User
Destination of output	N/A
Valid range, accuracy, and/or tolerance	Standard GSM 03.38 character set
Relationships to other inputs/outputs	The password that was provided by the user in order to create the user account

Found in requirement	Local Authentication : FRQ3
Name of item	Error Message
Description of purpose	The error message informs the user that the authentication process failed
Source of input	Application logic
Destination of output	User interface
Valid range, accuracy, and/or tolerance	N/A
Relationships to other inputs/outputs	The username and password provided during the don't correlate with the user account that has been created on the device

Found in requirement	Add Contact : FRQ10
Name of item	Contact Name
Description of purpose	The name a user wishes to have associated with the intended receiver of future messages
Source of input	User
Destination of output	User interface
Valid range, accuracy, and/or tolerance	Any character data
Relationships to other inputs/outputs	The text entered by the user to represent a specifi contact on the device

Found in requirement	Add Contact : FRQ10
Name of item	Contact Number
Description of purpose	The mobile phone number associated with the con being added to the device
Source of input	Contact
Destination of output	N/A
Valid range, accuracy, and/or tolerance	Standard mobile phone numbers with or without the international dialing prefix
Relationships to other inputs/outputs	The value entered by the user to represent a contact's mobile phone number , and is associated with the provided contact name

Found in requirement	Add Contact : FRQ10
Name of item	Local Key
Description of purpose	The key generated locally which will be used by the application encryption algorithm
Source of input	Application generated; user initiated
Destination of output	N/A
Valid range, accuracy, and/or tolerance	Standard GSM 03.38 character set
Relationships to other inputs/outputs	This key will be used in conjunction with the key generated by the contact to ensure synchr and therefore secure communication, between the two synchronized devices

Found in requirement	Add Contact : FRQ10
Name of item	Contact Key
Description of purpose	The key generated locally by the contact which will be used by the application encryption algorith
Source of input	Contact
Destination of output	N/A
Valid range, accuracy, and/or tolerance	Standard GSM 03.38 character set
Relationships to other inputs/outputs	This key will be used in conjunction with the locally generated key to ensure synchronizatio and therefore secure communication, between the two synchronized devices

Found in requirement	Add Contact : FRQ10
Name of item	Contact
Description of purpose	The contact that has been created
Source of input	Application
Destination of output	Application database
Valid range, accuracy, and/or tolerance	N/A

28

Found in requirement	Add Contact : FRQ10
Name of item	Error Message
Description of purpose	The error message informs the user that the contact could not be added to the database
Source of input	Application logic
Destination of output	Application interface
Valid range, accuracy, and/or tolerance	N/A
Relationships to other inputs/outputs	The contact details that had been entered by the user was found invalid

Found in requirement	Edit Contact : FRQ11
Name of item	Contact
Description of purpose	The selected contact to be edited by the user
Source of input	User
Destination of output	N/A
Valid range, accuracy, and/or tolerance	N/A
Relationships to other inputs/outputs	The selected contact's details stored in the application database

Found in requirement	Edit Contact : FRQ11
Name of item	Local Key
Description of purpose	The key generated locally for device resynchronization
Source of input	Application generated; user initiated
Destination of output	N/A
Valid range, accuracy, and/or tolerance	
Relationships to other inputs/outputs	This key will be used in conjunction with the key generated by the contact to ensure resync and therefore secure communication, between the two synchronized devices

Found in requirement	Edit Contact : FRQ11
Name of item	Contact Key
Description of purpose	The key generated locally by the contact for device resynchronization
Source of input	Contact
Destination of output	N/A
Valid range, accuracy, and/or tolerance	
Relationships to other inputs/outputs	This key will be used in conjunction with the locally generated key to ensure resynchronizat and therefore secure communication, between the two synchronized devices

Found in requirement	Edit Contact : FRQ11
Name of item	Contact
Description of purpose	The contact that has been updated
Source of input	N/A
Destination of output	Application database
Valid range, accuracy, and/or tolerance	N/A
Relationships to other inputs/outputs	All details related to the specific contact

Found in requirement	Edit Contact : FRQ11
Name of item	Error Message
Description of purpose	The error message informs the user that saving the contact's edited data has been unsuccessful
Source of input	Application logic
Destination of output	Application interface
Valid range, accuracy, and/or tolerance	N/A
Relationships to other inputs/outputs	The contact details that had been entered by the user was found to be invalid

3.2   Functions

3.2.1   Admin Functions

- Add User Account : FRQ1
  (Source: Bernard Wagner, Priority: Medium)

  – A user must be able to create a password protected account for the application

  – **Inputs:** The user enters a desired username, and password (as well as the password confirmation)

  – **Outputs:** A user account is either created and stored in the application database, or an error message is displayed and no account is created
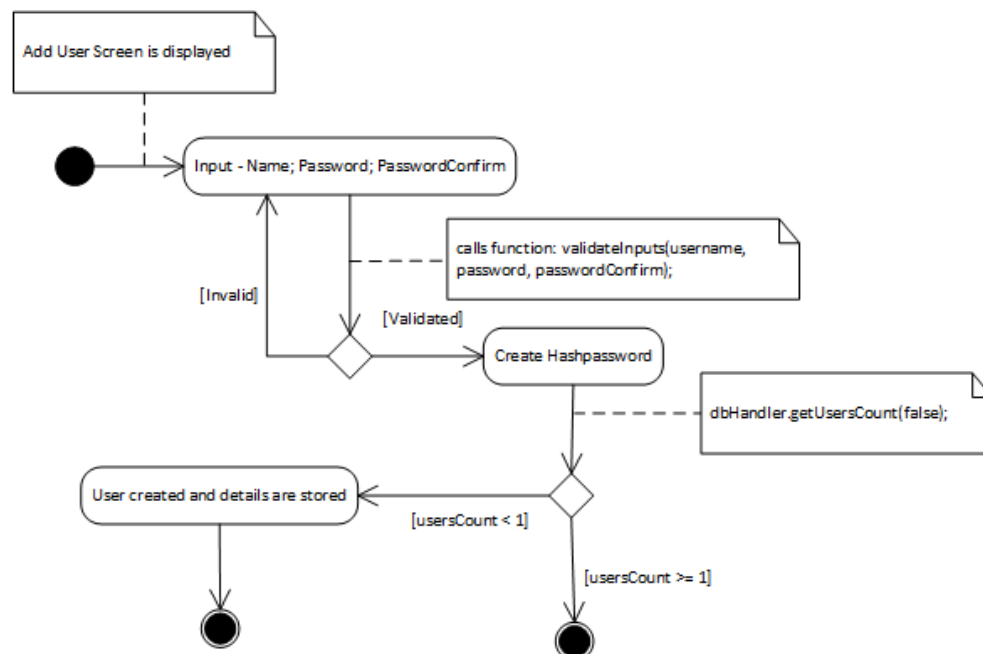
- Edit User Account : FRQ2
  (Source: Group Deliberation, Priority: Medium)

  – The user must be able to edit his/her user account details
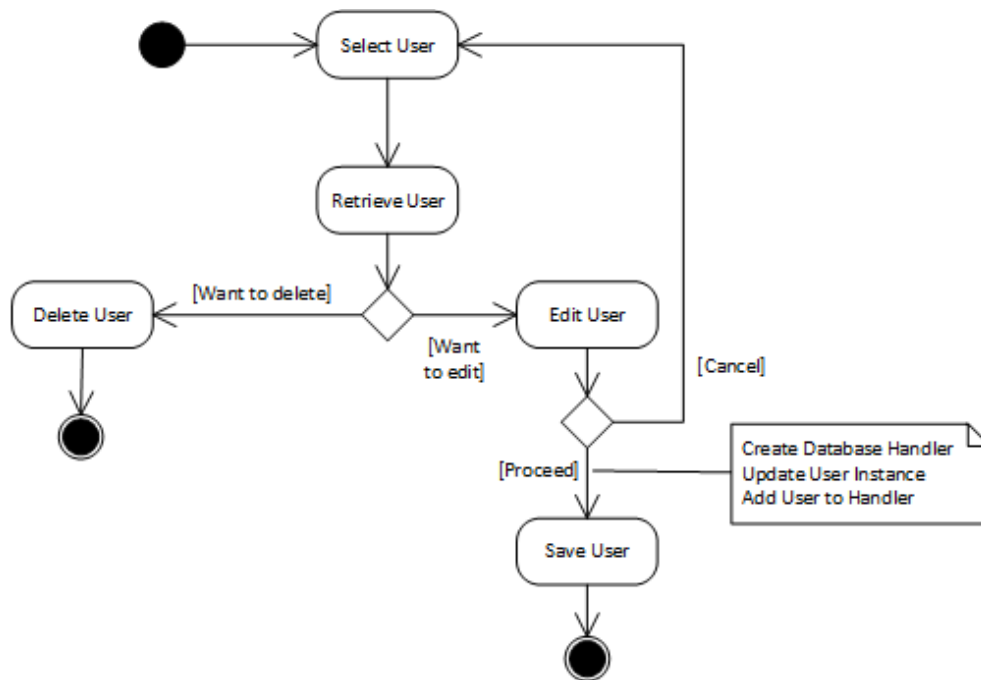
- Local Authentication : FRQ3
  (Source: Bernard Wagner, Priority: Medium)

  - To ensure confidentiality, the application must authenticate a user by requiring a password before granting access to the application features

  - **Inputs:** The user enters his/her username and password

  - **Outputs:** After user authentication has been successful (the username/password provided had been found correct), access is granted to all application features. If authentication fails(and incorrect username/password was entered), and error message is displayed. If authentication fails repeatedly for a specific amount of times; within a specific time slot, the application is locked for a specific amount of time

Application starts

Login screen is displayed

[Wishes to login]

Input Username and Password

[Wishes to add a user]

Add User

Select Login

Calls function ValidateAndContinue(View view)

Proceeds to
Add User
screen

Password is stored in Char array

Loginhandler.getUserByName(username,this.getApplicationContext)

[Password does not match]

Chek to see if Username exists

[Username
does exist]

Entered password is hashed

[Username does not exist]

[Password matches]

LoginHandler.hashValue(password)

if(!currentUser.getPasswordHash()
.equals(HashedString))

Sets encryption

Log user in

3.2.2 Messaging Functions

- Enter message : FRQ4
  (Source: Bernard Wagner, Priority: High)

  - A user must be able to input text into the application

Entered Message

[Manual]

Show Empty Message

[Not Done]

[Automatic]

Paste Message

[Done]

[Not Done]

[Done]
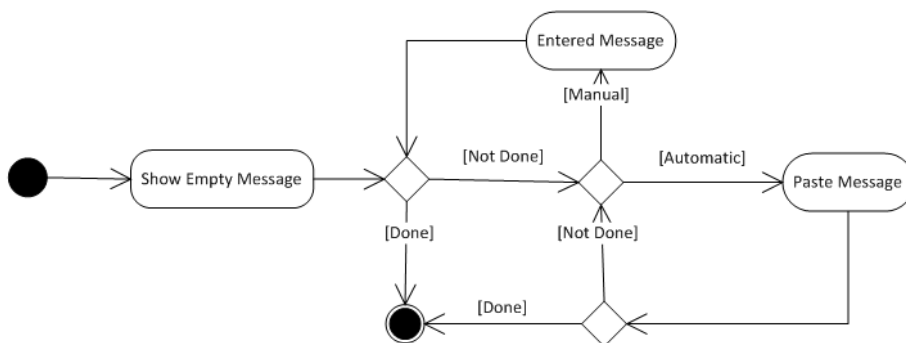
- Type message : FRQ4.1
  (Source: Bernard Wagner, Priority: High)

  - A user must be able to type text into the application directly

- Paste message : FRQ4.2
  (Source: Bernard Wagner, Priority: High)

  - A user must be able to paste an already constructed message into the application, using the device clipboard

32

- Edit Message: FRQ5
  (Source: Bernard Wagner, Priority: Medium)

  – The message text must be editable once it has been input into the application
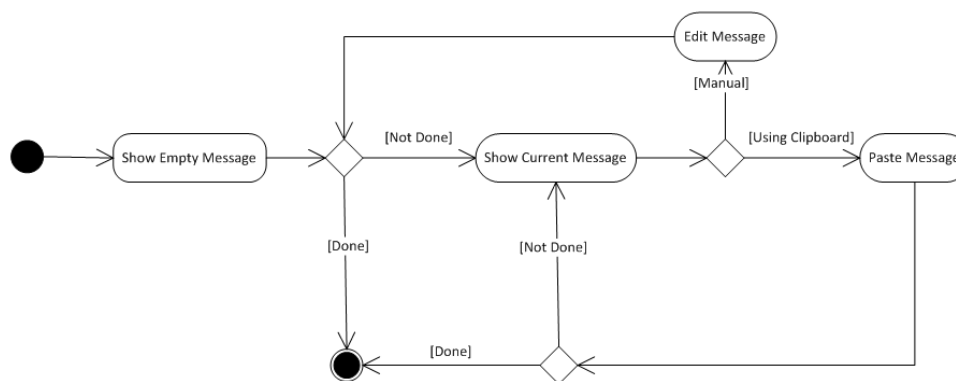
- Encrypt message : FRQ6
  (Source: Bernard Wagner, Priority: High)

  – The message must be encrypted using a suitable encryption method
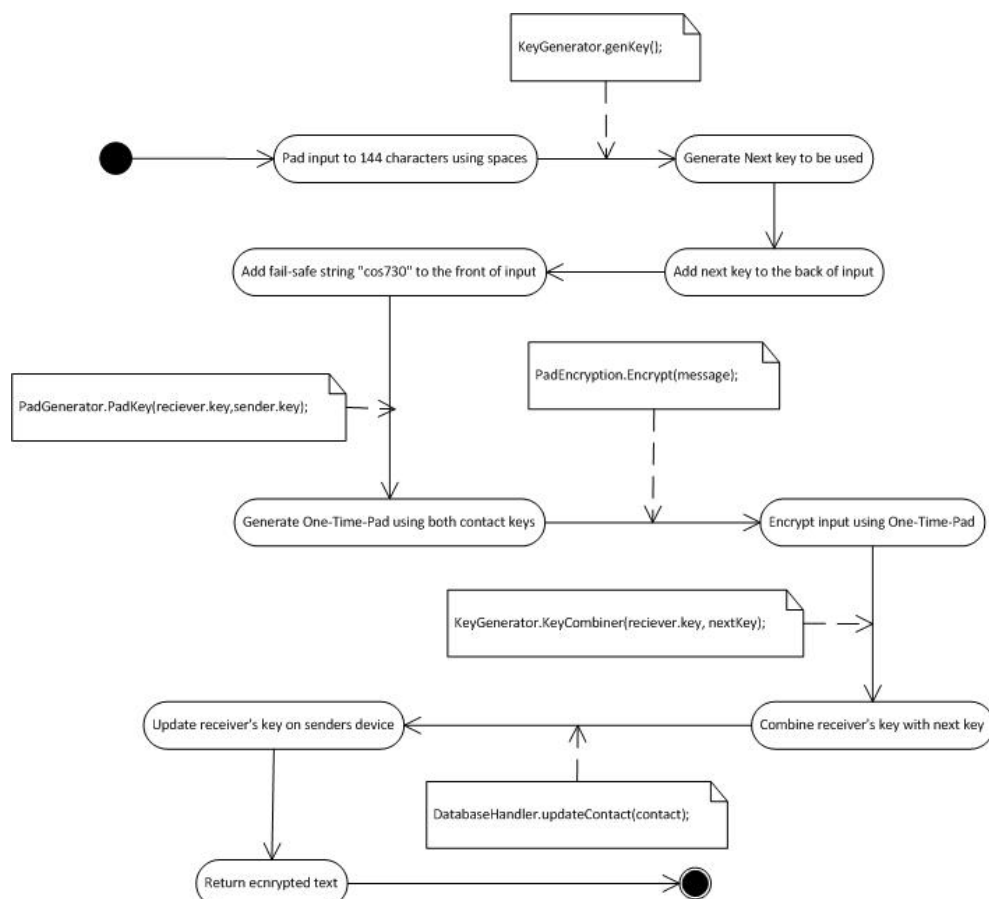
- Copy Encrypted Message : FRQ7
  (Source: Bernard Wagner, Priority: High)

  – Once a message has been encrypted, a user must be able to copy the ciphertext, and paste it into any selected messaging application

- Decrypt message : FRQ8
  (Source: Bernard Wagner, Priority: High)

  – The application must be able to decrypt received messages to reveal their original plaintext - if the sender of the message's contact is provided and synchronized
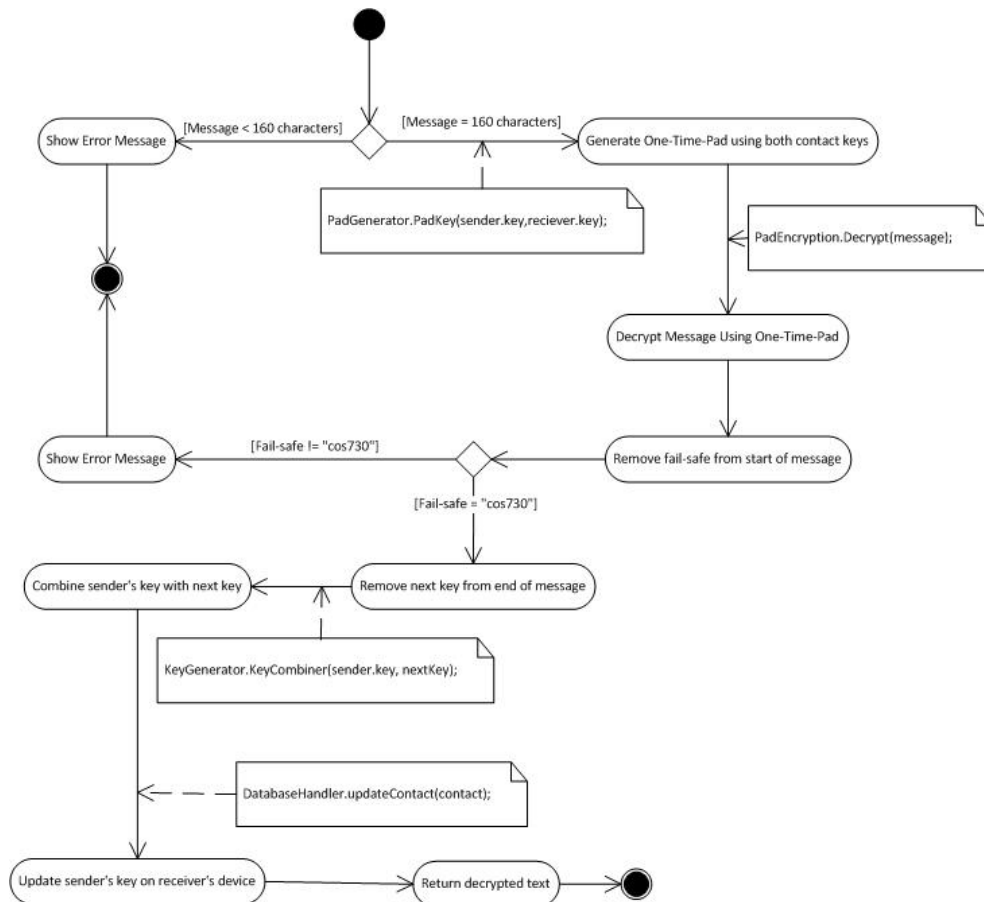
- Display message length : FRQ9
  (Source: Bernard Wagner, Priority: Low)

  – The numbers of characters in a message must be displayed within the application
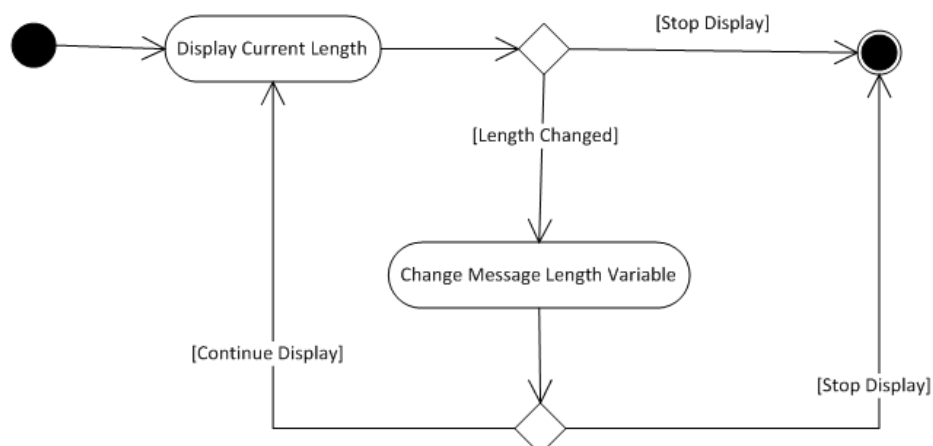
3.2.3 Contacts Functions

- Add Contact : FRQ10
  (Source: Bernard Wagner, Priority: High)

  – Before communicating with another user, the receiver of these (future) messages must be added as a contact on the local device, in order to be able to communicate with that user

  – **Inputs:** The receiver's name, mobile number, and locally generated key must be provided (along with the local device's generated key)

  – **Outputs:** The contact is created and stored in the application database, or the application displays an error message if the contact's details entered was found incorrect
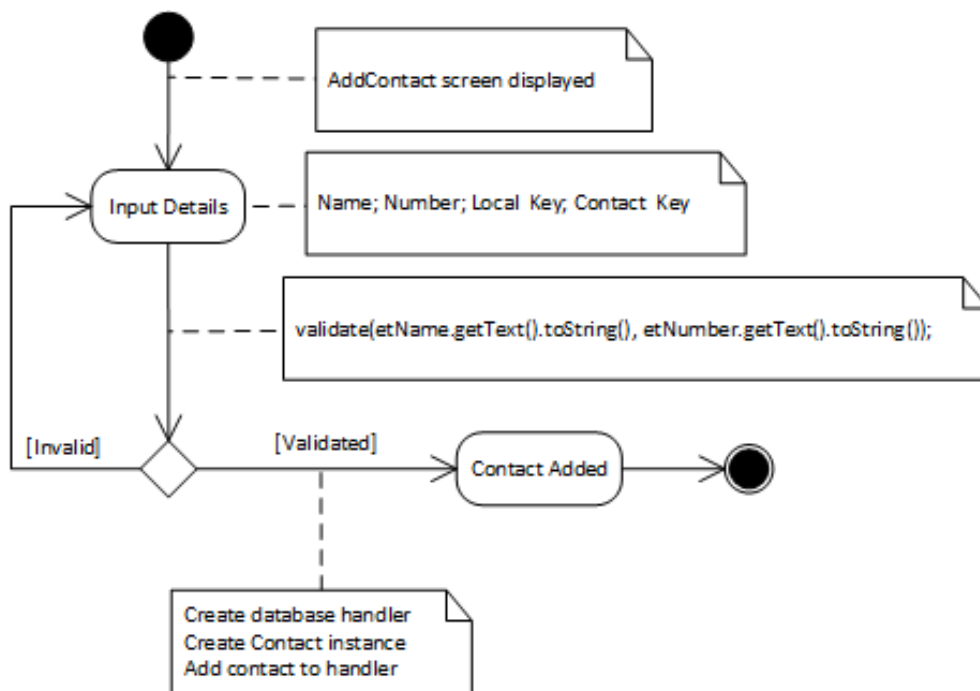
- Edit Contact : FRQ11
  (Source: Bernard Wagner, Priority: Medium)

  – A contact must be editable/deletable once it has been added

  – **Inputs:** The user selects a contact which they wish to modify, or delete, within the application

  – **Outputs:** The selected contact's details will be modified in the application database, or the contact will be removed from the application database
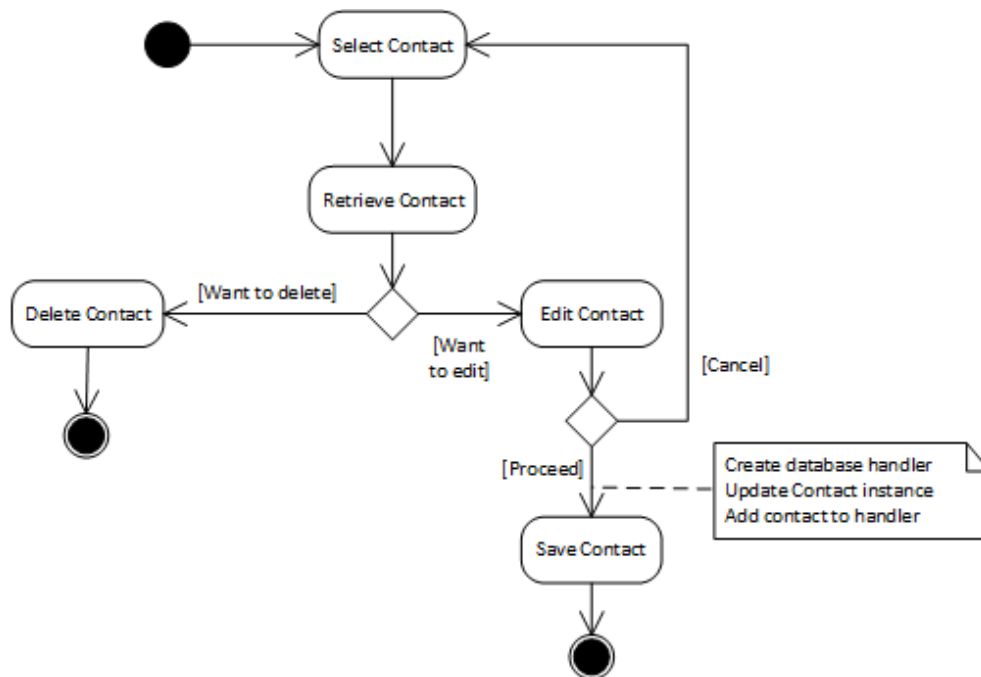
- Synchronise Contact : FRQ12
  (Source: Group Deliberation, Priority: High)

  – The user must be able to synchronize with a contact, at any time, after the contact has been added to the application
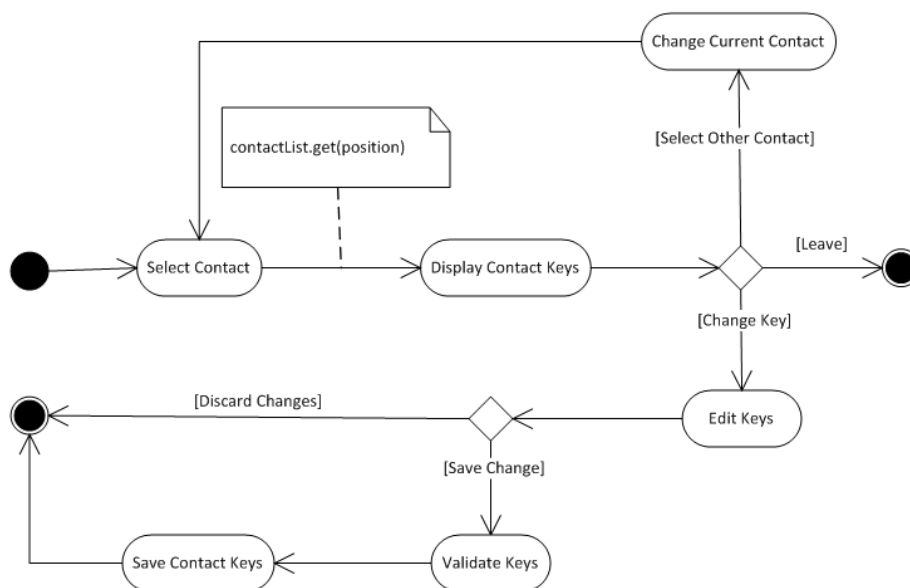
- Remove Contact : FRQ13
  (Source: Bernard Wagner, Priority: High)

  – A user must be able to remove an already added contact
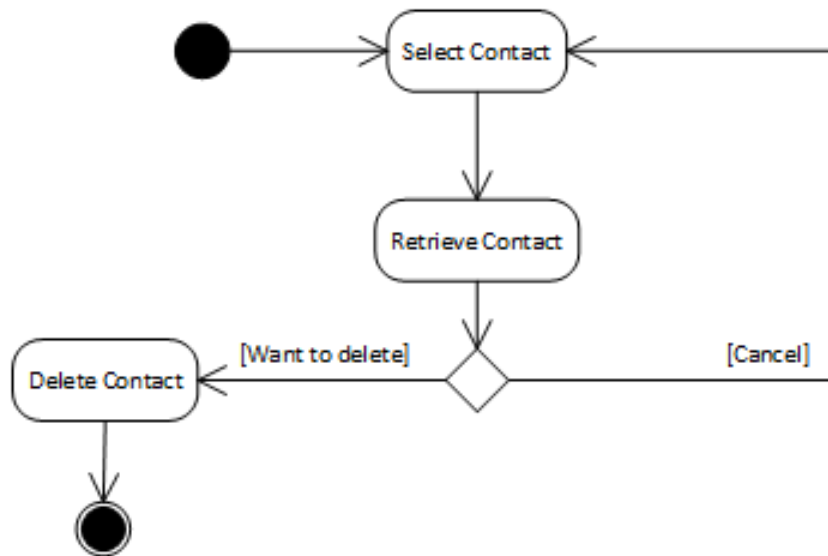
3.3 Performance requirements

- The application should operate in a timely manner. The user should not be made to wait an unreasonable amount of time (this variable can be affected by the system environment e.g. resource availability)

  - The application is expected to encrypt a message in less than one second
  - The application is expected to decrypt a message in less than one second

- The encryption method used must ensure secure communication via compromised communication methods

  - The encyption method used should have an entropy value of less than 1%

- The application must make use of local authentication

  - A user's password must always remain masked; when logging in, or even when editing the user account details
  - A user's password should be encrypted or stored as a hash value within the application database
  - Contact information stored by the application must not be obtainable by unauthorized users
  - Contact information should be encrypted to ensure that it is not readable by unauthorized users

3.4 Logical database requirements

- The user's password must be hashed using a SHA-256 hash function
- The data stored by the application should only be viewable by an authorised user

37

- Contact data must be encrypted to ensure confidentiality - using an applicable hash function

- Every user has zero or more contacts associated with it

- Once a user account is deleted, all data associated with that account, including all saved contacts, will be deleted
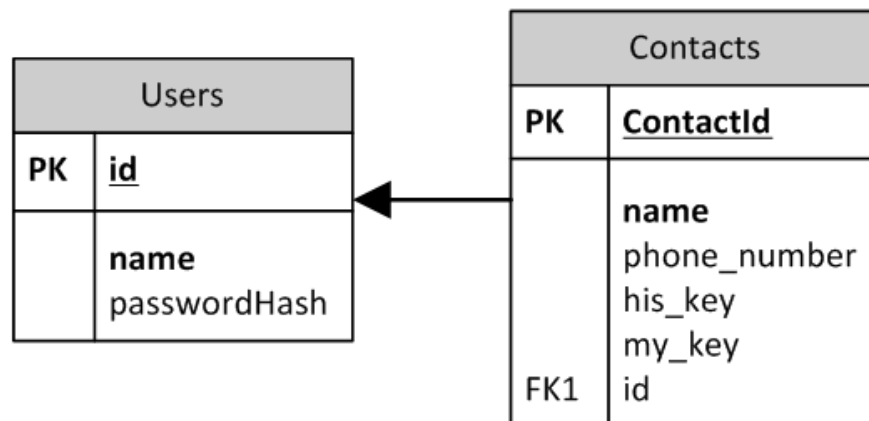
3.5   Design constraints

- Message length : DC1
  (Source: Bernard Wagner, Priority: High)

  – Due to the fact that the primary messaging service that the client intends to use is SMS, the amount of characters per message is limited to 160 characters in total

  – The encryption process uses 6 characters for a failsafe, and 10 characters to embed the next key to be used for future communication

  – This means that plaintext entered by the user is limited to 144 characters

  – To maintain consistency, we enforce this as the maximum length of messages which the application can encrypt, regardless of the messaging application used to send the message

- The usable characters : DC2
  (Source: Bernard Wagner, Priority: High)

  – The usable characters that can be encrypted by the application is limited to the GSM character set, because the primary messaging service that this application is developed for is SMS.

- Application resource requirements : DC3
  (Source: Bernard Wagner, Priority: High)

– The application should function efficiently, with the least amount of resource usage

3.5.1 Standards compliance

- The application must be secure, as is stipulated in Appendix D: Secure Design Principles

- The application must conform to the Android and iOS respective platform design principles; these can be found in Appendix E, Design principles

3.6 Software system attributes

3.6.1 Reliability

- The application should run until the user closes it

- Any information stored by the application should be static and exist as long as the application is open or said information is edited/removed

- The ciphertext should decrypt back into its original plaintext

3.6.2 Availability

- The user should be able to use the application regardless of runtime, and should not be made to wait while the application performs a function

- The application should not interfere with any other applications that could be running simultaneously on the device

3.6.3 Security

- A secure encryption (and decryption) method will be used for the localised application database

3.6.4 Maintainability

- The source code should be simplistic, maintainable, and readable/documented

- The application should not act in unpredictable ways

3.6.5 Portability

- The client has requested that different versions of the application be developed to execute on different mobile operating systems: namely Android and iOS

4   Appendix A - RSA

Introduction

This appendix contains research done in pursuit of a possible solution to the given problem - encryption over an unsecure network. The research done includes RSA, and how it can be used as a possible solution to encrypt an SMS message.

Method

We started by trying to use the built-in RSA implementation that is included in the default Java library. After that we did research into the background of RSA, or, more specifically, the mathematics behind the RSA algorithm. We then attempted numerous combinations of the mathematical principals behind RSA to see if any of them could be used to fulfill the requirements for this project.

Result

The built-in RSA algorithm requires generating keys that would be too large to send as an SMS, thus making redistribution of keys difficult. For example, to fulfill the purposes of this project, a message may only contain, at most, 160 characters. Using RSA to generate a key on a message of 160 characters (after it has encrypted with a one-time pad - that also forms part of our algorithm), it would generate a 700 character key for the message, which is more than four times larger than the message itself. Using the RSA within our application would limit text messages to less than 77 characters (which would already generate a key of more than 160 characters). This approach would be inefficient, and was discarded as a means of encryption for our application.

Instead, we attempted to implemented a custom RSA algorithm, but it started out weak due to the limits imposed by our character set. We looked into an alternative where 2 encrypted characters could represent 1 plaintext character. This gave some strength to the encryption but limited the message one could send to 80 characters (i.e. 2 x 80 = 160 characters). Our client declined this suggestion, and said that it is not an option to be considered.

Discussion

While thinking about modern encryption, we thought about RSA and how useful it is. It is easy to forget that, behind the scenes, large amounts of data is transferred just to enable the encryption and decryption functionality. It is because of the keys being too large to send within a single SMS that the built-in RSA was disregarded, along with uncontrolled padding in an environment where message length was extremely important.

In our custom RSA algorithm we could control the length of the keys somewhat, but just like the built-in version it still limited the amount of characters to half the size of full SMS.

Conclusion

RSA works well with modern technologies, where large messages can be sent by only using a small amount of data; be it via a more modern wireless network, such as 3G. Encryption can be made stronger with the use of larger keys, but for the intents and purposes of this project, along with the fact that message sent are limited to 160 characters, our custom RSA algorithm was a good suggestion, but still not the answer we were looking for.

5 Appendix B - One time pads

Introduction

This document contains research done into one-time pads, an encryption technique that, if used correctly, is unbreakable. It also provides the person attempting to decrypt the message with no information about the plaintext whatsoever, except from the maximum possible length that it could be.

Method

We did some research into one-time pads, and what it is that makes it so strong. This inspired us to develop a one-time pad algorithm that can be used within our application, and the results looked very promising.

Result

The encryption is very strong, and allows for a 1:1 character ratio after encryption has taken place on the message. This allows us to fully utilize the 160 characters per SMS, as the ciphertext length does not grow beyond the plaintext length. This was the solution to the problem we had before, when considering an RSA algorithm.

Discussion

The first thing that comes to mind when thinking about one-time pad encryption is how to distribute the pad itself. The pad needs to be distributed between the two parties who intend to communicate with each other. At any point during communication between these parties, the next line from the one-time pad must be known in advance. This requires some form of synchronization between the two parties before secure communication can take place.

Conclusion

In terms of message length and encryption strength, this solution is perfect, but with no way of distributing the one-time pad securely, we had to disregard this solution.

6 Appendix C - Encryption Protocol

Introduction

This Appendix discusses the encryption protocol that we designed after some extensive research. It comprises of a combination of a key-based encryption, and one-time pad encryption. Because of this combination of encryption protocols, the protocol we developed is referred to as a hybrid encryption protocol. This hybrid protocol is very complex, but very powerful. Because it uses one-time pad encryption, the ciphertext itself is unbreakable - unless you have access to the pad. The pad is generated using two special keys stored on each user's device: one key generated locally on each device, and another key that represents the other user's (intended receiver of messages) locally generated key (which the users have to share with one another via any means of communication). Thus, the pad used is never communicated as messages are exchanged between two parties, instead, each message sent contains the next key to be used from the pad. This creates a key dependency, as a previously sent/received key is used to determine the next key to be used. This will strengthen the encryption, as it is not possible for an attacker to know what key will be used next, even if a message is compromised somehow. However, to further increase the encryption strength, the key sent is used in combination with the local(internal) key to produce the next key to be used with future communication between these two parties. Because the key changes after a message has been sent, replay attacks are impossible as the message can only be decrypted once before the keys change.

Description

As stated above, this protocol is a hybrid between key-based encryption and one-time pad encryption. The only way two users can communicate with each other via the SMSEncryption application is when both users have shared their local keys with each other. These keys are instantiated when users add each other as "contacts" within the application. When adding a contact, a key is generated locally on the device that has to be shared with the intended receiver of any future messages. Once both users have shared their local keys with one another (so that each of them have two keys in total: their locally generated key, as well as their contact's locally generated key), both keys are stored on the device, along with other relevant data, as a contact. This is done for each contact created so that no keys are shared beyond one contact. This also limits compromised communication between two parties, as the compromised key cannot be used on messages sent between any other two parties.

Please note that messages sent with this algorithm are limited to 144 plaintext characters. The remaining 16 characters are used for synchronization purposes. Read on for more information regarding the synchronization process.

When a user wishes to send a message to a specific contact, the two keys stored for that contact are retrieved, and fed into a special function that produces a one-time pad. Before the message is encrypted, it is padded to 144 characters using spaces. This ensures a constant ciphertext length, so that any potential attackers cannot determine

the plaintext message length.

Next, a new key is generated to be used for the next communication that could take place between these two users. This key is encoded using our special character set to represent a 10 character string. This key is then added to the end of the encrypted message, bringing the total amount of characters thus far to 154 characters.

Next, we add the failsafe string "cos730" to the beginning of the message. This is used by the application to determine the success of the decryption so that local keys are not changed on failed decryptions, and wrongfully used in future communication. If the message decryption fails for some reason, resynchronization has to take place between these two users.

Once the new key is added to the message to be sent, the message is then encrypted using the special one-time pad. Local keys are also updated to maintain synchronization between the two contacts.

Once the message is sent, the newly produced key is stored internally; it is also used along with the stored key from the contact. These two keys are fed into a special key-combining function to produce yet another new key. The stored key for the contact is then replaced with this new key.

To receive a message would be the opposite: the two internally stored keys, (one local key, and the sender of the message's (contact's) key) are retrieved, and used as parameters in the function that generates the one-time pads. This produces another one-time pad, which is then used to decrypt the received message (which will produce the original plaintext, plus the appended special 10 character key, and 6 character failsafe).

The failsafe is extracted from the message and compared to the string "cos730" to check for successful decryption. If it fails, an error message will be displayed.

If the failsafe test succeeds, the newly received key is then used in combination with the locally generated key, and fed into the special key-combining function to produce yet another new key. This combined key is then stored as this user's local key.

7 Appendix D - Secure design principles

As specified by the OWASP mobile security project, the following are the most prevalent mobile threats as of 2014 (which are applicable to the SMSEncryption project).

Below, the description of each problem is a short statement on how we attempt to mitigate each threat in the SMSEncryption application.

- Insecure Data Storage

The security of data the application stores is of the utmost importance as it could store the public and private keys of users or the OTP . Therefore we must consider threats to the data which is stored by the application.

In the SMSEncryption application, data stored is first encrypted using both the password of the user account linked to the application, as well as the username of the user of the application.

- Unintended Data Leakage

Data leakage is a viable threat which demonstrates the lack of control developers have when developing on mobile applications. For instance, the OS (Operating System) that you are developing for will manage memory. This can be exploited by would-be attackers by attempting to gain access to unprotected areas in memory.

In the SMSEncryption application, sensitive data, such as the password, is encrypted.

- Poor Authorization and Authentication

Poor authorization and authentication is relative to this project as we have to consider the consequences of the application being accessed and used by unauthorized personnel.

To gain access to the SMSEncryption application, a password is required in order to log into the application and gain access to the application features and its content(database).

- Broken Cryptography

We have to ensure that we make use of a suitable encryption method so that it can not be easily decrypted by attackers, and that it does not require a disproportionate amount of resources to implement or use.

In the SMSEncryption application we make use of AES encryption for the local database on each device.

- Lack of Binary Protections

This is a universal problem, as almost all code which is compiled into binary files could be reverse engineered into some form of discernable source code.

With regards to the SMSEncryption application, this means that we must ensure that all former sections are well taken care of in order to mitigate the effects of this threat.

8 Appendix E - Design principles

Introduction

This section contains the design principles available for Android and iOS developers. As one of the goals of this project is to make the SMSEncryption application available for both of these platforms, both sets of principles require due consideration.

Android

The android design principles were developed with user experience in mind, and are as follows:

- Enchant Me.
  - Delight me in surprising ways.
    * A beautiful surface, a carefully-placed animation, or a well-timed sound effect is a joy to experience. Subtle effects contribute to a feeling of effortlessness and a sense that a powerful force is at hand.
  - Real objects are more fun than buttons and menus.
    * Allow people to directly touch and manipulate objects in your app. It reduces the cognitive effort needed to perform a task while making it more emotionally satisfying.
  - Let me make it mine.
    * People love to add personal touches because it helps them feel at home and in control. Provide sensible, beautiful defaults, but also consider fun, optional customizations that don't hinder primary tasks.
  - Get to know me.
    * Learn peoples' preferences over time. Rather than asking them to make the same choices over and over, place previous choices within easy reach.

- Simplify My Life.

  - Keep it brief.

    * Use short phrases with simple words. People are likely to skip sentences if they're long. Pictures are faster than words. Consider using pictures to explain ideas. They get people's attention and can be much more efficient than words.

  - Decide for me but let me have the final say.

    * Take your best guess and act rather than asking first. Too many choices and decisions make people unhappy. Just in case you get it wrong, allow for 'undo'.

  - Only show what I need when I need it.

    * People get overwhelmed when they see too much at once. Break tasks and information into small, digestible chunks. Hide options that aren't essential at the moment, and teach people as they go.

  - I should always know where I am.

    * Give people confidence that they know their way around. Make places in your app look distinct and use transitions to show relationships among screens. Provide feedback on tasks in progress.

  - Never lose my stuff.

    * Save what people took time to create and let them access it from anywhere. Remember settings, personal touches, and creations across phones, tablets, and computers. It makes upgrading the easiest thing in the world.

  - If it looks the same, it should act the same.

    * Help people discern functional differences by making them visually distinct rather than subtle. Avoid modes, which are places that look similar but act differently on the same input.

  - Only interrupt me if it's important.

    * Like a good personal assistant, shield people from unimportant minutiae. People want to stay focused, and unless it's critical and time-sensitive, an interruption can be taxing and frustrating.

- Make Me Amazing.

  - Give me tricks that work everywhere.
    - ∗ People feel great when they figure things out for themselves. Make your app easier to learn by leveraging visual patterns and muscle memory from other Android apps. For example, the swipe gesture may be a good navigational shortcut.

  - It's not my fault.
    - ∗ Be gentle in how you prompt people to make corrections. They want to feel smart when they use your app. If something goes wrong, give clear recovery instructions but spare them the technical details. If you can fix it behind the scenes, even better.

  - Sprinkle encouragement.
    - ∗ Break complex tasks into smaller steps that can be easily accomplished. Give feedback on actions, even if it's just a subtle glow.

  - Do the heavy lifting for me.
    - ∗ Make novices feel like experts by enabling them to do things they never thought they could. For example, shortcuts that combine multiple photo effects can make amateur photographs look amazing in only a few steps.

  - Make important things fast.
    - ∗ Not all actions are equal. Decide what's most important in your app and make it easy to find and fast to use, like the shutter button in a camera, or the pause button in a music player.

iOS Human Interface Guidelines

The Apple Developer page specifies various principles under their Human Interface Guidelines.

Designing for iOS 7

iOS 7 embodies the following themes:

- Deference. The UI helps users understand and interact with the content, but never competes with it.

  - Although crisp, beautiful UI and fluid motion are highlights of the iOS 7 experience, the user's content is at its heart.

  - Here are some ways to make sure that your designs elevate functionality and defer to the user's content.

    * Take advantage of the whole screen. Reconsider the use of insets and visual frames and instead let the content extend to the edges of the screen. Weather is a great example of this approach: The beautiful, full-screen depiction of a location's current weather instantly conveys the most important information, with room to spare for hourly data.

    * Reconsider visual indicators of physicality and realism. Bezels, gradients, and drop shadows sometimes lead to heavier UI elements that can overpower or compete with the content. Instead, focus on the content and let the UI play a supporting role.

    * Let translucent UI elements hint at the content behind them. Translucent elements such as Control Center provide context, help users see that more content is available, and can signal transience. In iOS 7, a translucent element blurs only the content directly behind it, giving the impression of looking through rice paper, it doesnt blur the rest of the screen.

- Clarity. Text is legible at every size, icons are precise and lucid, adornments are subtle and appropriate, and a sharpened focus on functionality motivates the design.

  - How to provide clarity

    * Providing clarity is another way to ensure that content is paramount in your app. Here are some ways to make the most important content and functionality clear and easy to interact with.
    * Use plenty of negative space. Negative space makes important content and functionality more noticeable and easier to understand. Negative space can also impart a sense of calm and tranquility, and it can make an app look more focused and efficient.
    * Let color simplify the UI. A key color, such as yellow in Notes, highlights important state and subtly indicates interactivity. It also gives an app a consistent visual theme. The built-in apps use a family of pure, clean system colors that look good at every tint and on both dark and light backgrounds.
    * Ensure legibility by using the system fonts. iOS 7 system fonts automatically adjust letter spacing and line height so that text is easy to read and looks great at every size. Whether you use system or custom fonts, be sure to adopt Dynamic Type so your app can respond when the user chooses a different text size.
    * Embrace borderless buttons. In iOS 7, all bar buttons are borderless. In content areas, a borderless button uses context, color, and a call-to-action title to indicate interactivity. And when it makes sense, a content-area button can display a thin border or tinted background that makes it distinctive.

- Depth. Visual layers and realistic motion impart vitality and heighten users' delight and understanding.

  - Use Depth to Communicate

    * iOS 7 often displays content in distinct layers that convey hierarchy and position, and that help users understand the relationships among onscreen objects.
    * By using a translucent background and appearing to float above the Home screen, folders separate their content from the rest of the screen.
    * Reminders displays lists in layers, as shown here. When users work with one list, the other lists are collected together at the bottom of the screen.
    * Calendar uses enhanced transitions to give users a sense of hierarchy and depth as they move between viewing years, months, and days. In the scrolling year view shown here, users can instantly see today's date and perform other calendar tasks.
    * When users select a month, the year view zooms in and reveals the month view. Today's date remains highlighted and the year appears in the back button, so users know exactly where they are, where they came from, and how to get back.

         * A similar transition happens when users select a day: The month view appears to split apart, pushing the current week to the top of the screen and revealing the hourly view of the selected day. With each transition, Calendar reinforces the hierarchical relationship between years, months, and days.

Apple Design Principles

- Aesthetic Integrity

  - Aesthetic integrity doesn't measure the beauty of an app's artwork or characterize its style; rather, it represents how well an app's appearance and behavior integrates with its function to send a coherent message.

  - People care about whether an app delivers the functionality it promises, but they're also affected by the app's appearance and behavior in strong, sometimes subliminal, ways. For example, an app that helps people perform a serious task can put the focus on the task by keeping decorative elements subtle and unobtrusive and by using standard controls and predictable behaviors. This app sends a clear, unified message about its purpose and its identity that helps people trust it. But if the app sends mixed signals by presenting the task in a UI that's intrusive, frivolous, or arbitrary, people might question the app's reliability or trustworthiness.

- On the other hand, in an app that encourages an immersive task, such as a game, users expect a captivating appearance that promises fun and excitement and encourages discovery. People don't expect to accomplish a serious or productive task in a game, but they expect the game's appearance and behavior to integrate with its purpose.

- Consistency

  - Consistency lets people transfer their knowledge and skills from one part of an app's UI to another and from one app to another app. A consistent app isn't a slavish copy of other apps and it isn't stylistically stagnant; rather, it pays attention to the standards and paradigms people are comfortable with and it provides an internally consistent experience.

  - To determine whether an iOS app follows the principle of consistency, think about these questions:

    * Is the app consistent with iOS standards? Does it use system-provided controls, views, and icons correctly? Does it incorporate device features in ways that users expect?
    * Is the app consistent within itself? Does text use uniform terminology and style? Do the same icons always mean the same thing? Can people predict what will happen when they perform the same action in different places? Do custom UI elements look and behave the same throughout the app?
    * Within reason, is the app consistent with its earlier versions? Have the terms and meanings remained the same? Are the fundamental concepts and primary functionality essentially unchanged?

- Direct Manipulation

  - When people directly manipulate onscreen objects instead of using separate controls to manipulate them, they're more engaged with their task and it's easier for them to understand the results of their actions.

  - Using the Multi-Touch interface, people can pinch to directly expand or contract an image or content area. And in a game, players move and interact directly with onscreen objects, for example, a game might display a combination lock that users can spin to open.

- In an iOS app, people experience direct manipulation when they:
  * Rotate or otherwise move the device to affect onscreen objects
  * Use gestures to manipulate onscreen objects
  * Can see that their actions have immediate, visible results

- Feedback

  - Feedback acknowledges people's actions, shows them the results, and updates them on the progress of their task.

  - The built-in iOS apps provide perceptible feedback in response to every user action. List items and controls highlight briefly when people tap them and, during operations that last more than a few seconds, a control shows elapsing progress.

  - Subtle animation can give people meaningful feedback that helps clarify the results of their actions. For example, lists can animate the addition of a new row to help people track the change visually.

  - Sound can also give people useful feedback, but it shouldn't be the only feedback mechanism because people can't always hear their devices.

- Metaphors

  - When virtual objects and actions in an app are metaphors for familiar experiences, whether these experiences are rooted in the real world or the digital world, users quickly grasp how to use the app.

  - It's best when an app uses a metaphor to suggest a usage or experience without letting the metaphor enforce the limitations of the object or action on which it's based.

  - iOS apps have great scope for metaphors because people physically interact with the screen. Metaphors in iOS include:
    * Moving layered views to expose content beneath them
    * Dragging, flicking, or swiping objects in a game
    * Tapping switches, sliding sliders, and spinning pickers
   * Flicking through pages of a book or magazine

- User Control

    - People, not apps, should initiate and control actions. An app can suggest a course of action or warn about dangerous consequences, but it's usually a mistake for the app to take decision-making away from the user. The best apps find the correct balance between giving people the capabilities they need while helping them avoid unwanted outcomes.

    - Users feel more in control of an app when behaviors and controls are familiar and predictable. And when actions are simple and straightforward, users can easily understand and remember them.

    - People expect to have ample opportunity to cancel an operation before it begins, and they expect to get a chance to confirm their intention to perform a potentially destructive action. Finally, people expect to be able to gracefully stop an operation that's underway.

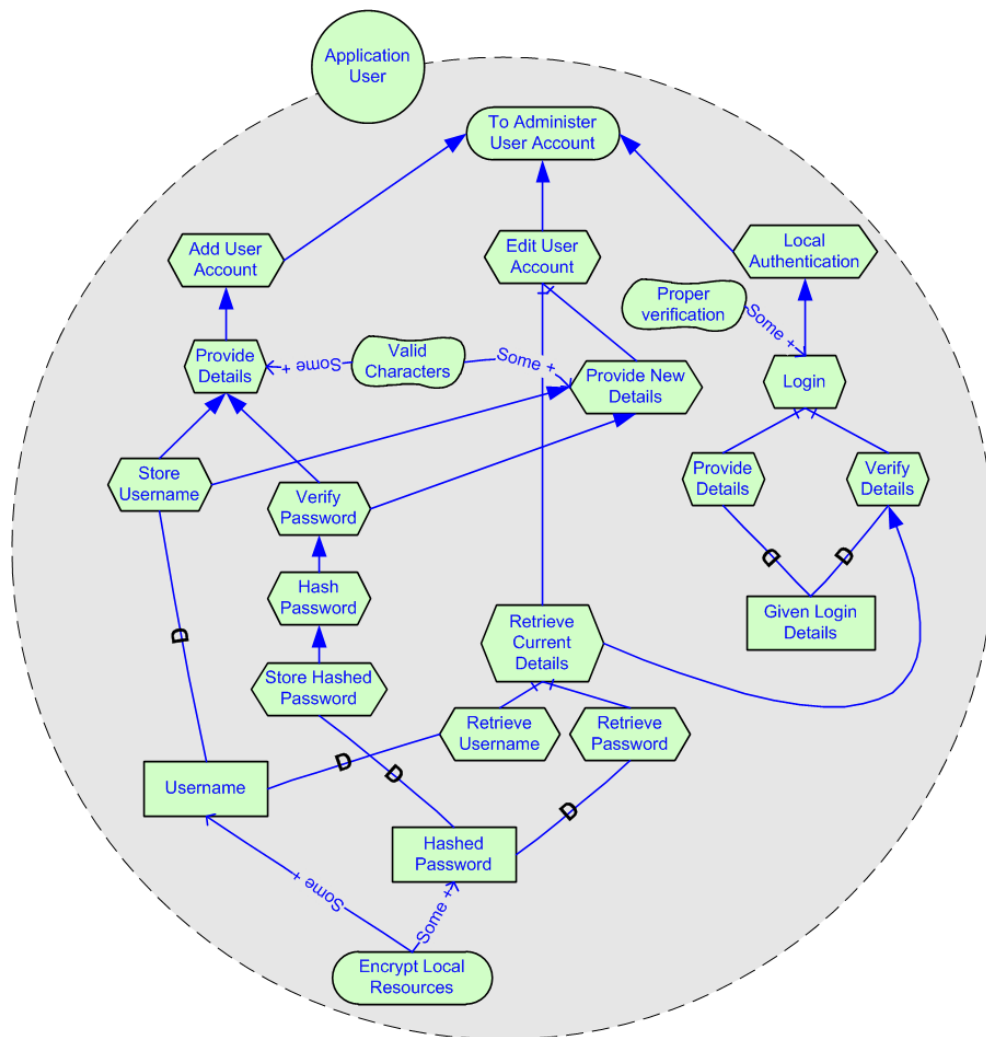# 9 Appendix F - i* Diagrams

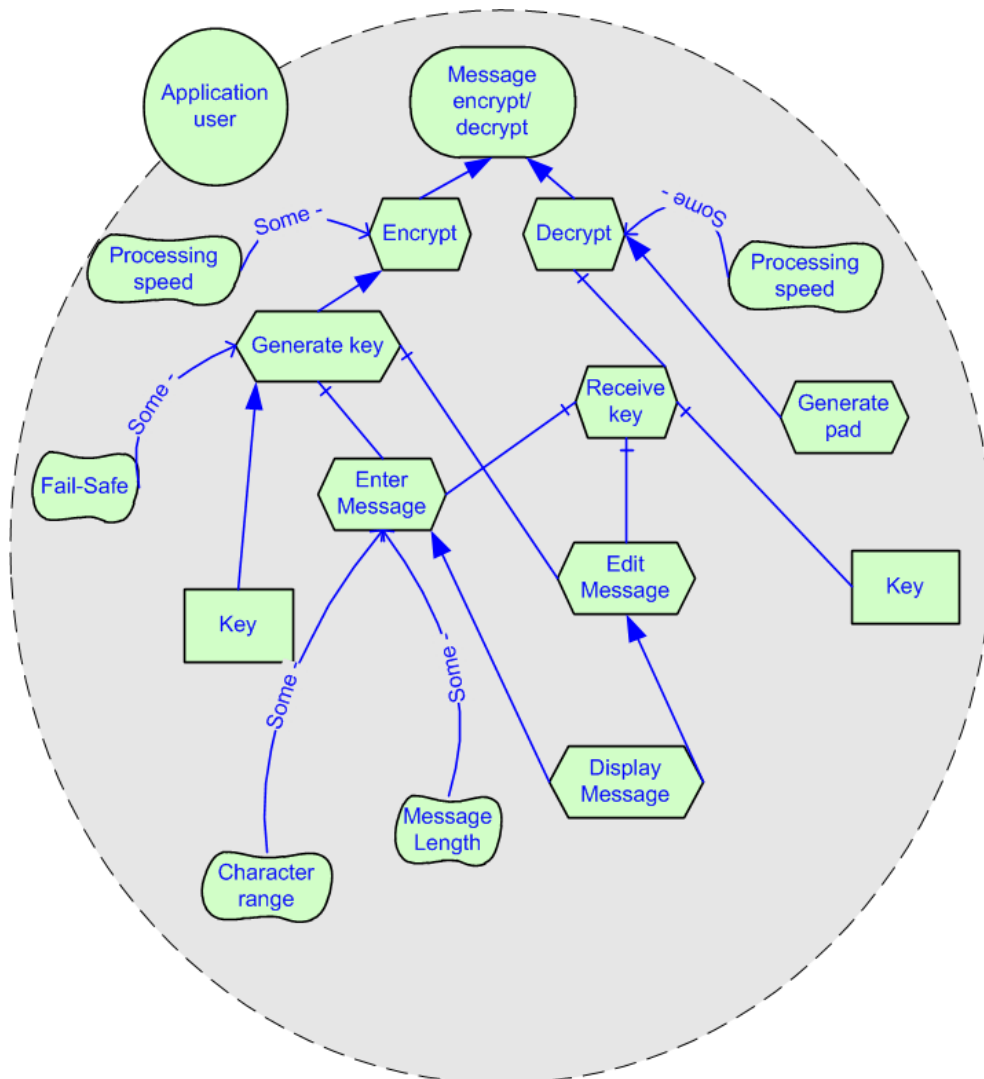This sections contains i* diagrams we made during the requirements elicitation process.
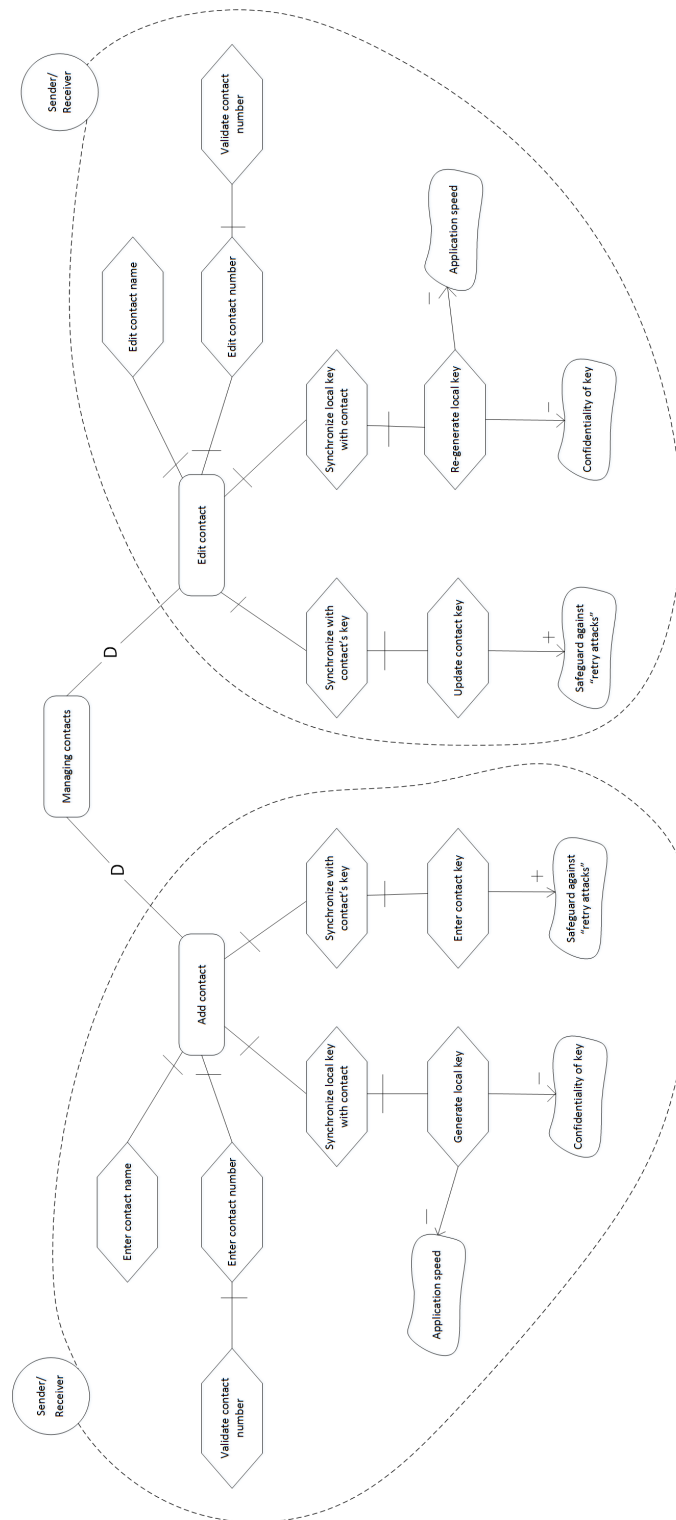
General i* Diagram
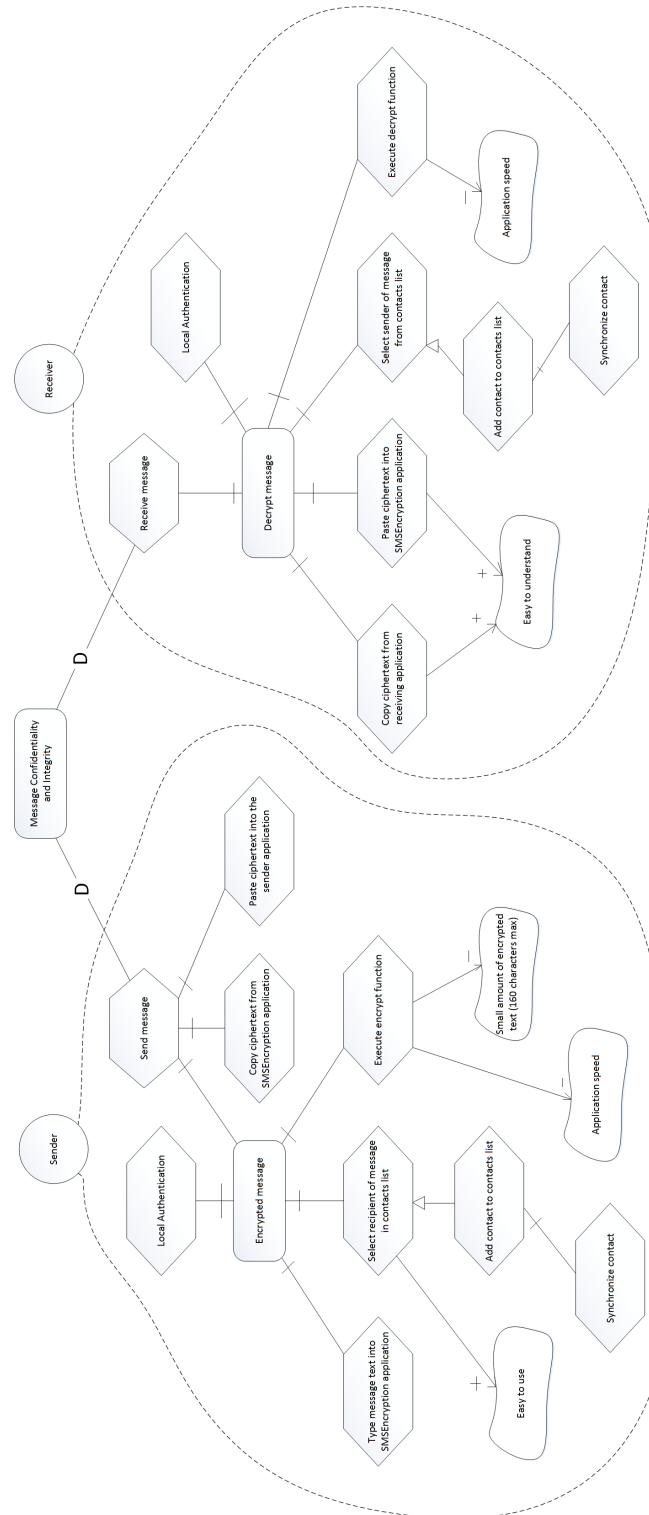
Admin functionality i* Diagram

Messaging functionality i* Diagram

Contacts functionality i* Diagram

10 Appendix G - Version History

Version	Description	Updater
Version 0.1	Document Created	Henko
Version 0.1	Document layout added	Henko
Version 0.2	Added other sections	Henko
Version 0.3	Added parts to Introduction	Henko
Version 0.4	Added General Description	Henko
Version 0.5	Made document IEEE compliant	Jaco, Luke
Version 0.5	Added to some empty sections	Jaco, Luke
Version 0.5	Added appendix A	Hein
Version 0.5	Added appendix B	Hein
Version 0.5	Added appendix C	Hein
Version 0.5	Overview of grammar and sentence structure	Vincent
Version 0.5	Some contextual/explanative additions	Vincent
Version 0.6	Added Appendix for Desgin Principles	Jaco
Version 0.6	Added Appendix for Secure Desgin	Luke
Version 0.6	Added background verbatim from client	Luke
Version 0.7	Added iOS guidelines to Appendix Z	Jaco
Version 0.7	References according to IEEE	Jaco
Version 0.8	Added FRQ4-11, FRQ1.1 and FRQ1.2	Jaco
Version 0.8	Updated original Usecase Diagram	Jaco
Version 0.8	Protocol description for Appendix C	Hein
Version 0.9	Some grammar checking, added some context	Vincent
Version 1.0	Aesthetics of document, expanded Overview	Jaco
Version 1.0	Added state diagram, and general i* diagram	Vincent
Version 1.0	Updated numbers on usecase and FRQ's	Jaco
Version 1.0	Split FRQ's into sub functions	Jaco
Version 1.1	Fixed minor errors to Appendix C	Henko
Version 1.2	Added Contact's IStar diagram	Vincent
Version 1.2	Added general state diagram to folder only	Vincent
Version 1.2	FRQ adjustments and general error fixes	Jaco
Version 1.2	Updated usecase diagram	Jaco
Version 1.2	Added AddContact state diagram	Jaco, Luke, Vincent

Version	Description	Updater
Version 1.2	Added AddUser state diagram	Jaco, Luke, Vincent
Version 1.2	Added EditContact state diagram	Jaco. Luke, Vincent
Version 1.2	Added Local Authentication state diagram	Jaco, Luke, Vincent
Version 1.2	Modified FRQ and DC layout	Jaco
Version 1.2	Made changes to section 5.4	Luke
Version 1.3	Layout modification	Jaco
Version 1.4	Updated general state diagram	Vincent
Version 1.4	Expanded section 2.2 with old info	Jaco
Version 1.4	Added FRQ 12, Synchronise contact	Jaco
Version 1.4	Expanded DC 1 with more specific details	Jaco
Version 1.4	Added standards compliance section	Jaco
Version 1.4	Minor alterations to Software system attributes	Jaco
Version 1.4	Added to and fixed general state diagram,	Vincent
Version 1.4	Added to and fixed general i* diagram	Vincent
Version 1.4	Added encrypt and decrypt state diagrams	Hein
Version 1.4	Modified encrypt and decrypt state diagrams	Luke
Version 1.5	Updated Usecase, added admin i*diagram	Jaco
Version 1.5	Added Appendix F and i* diags to it	Jaco
Version 1.5	Updated state diagrams	Jaco
Version 1.5	Inserted encrypt and decrypt state diagrams	Jaco
Version 1.5	Added mitigation features to Appendix D	Jaco
Version 1.5	Added inputs and outputs to service contracts	Luke
Version 1.5	Add Appendix F to overview	Jaco
Version 1.5	Edit FRQ 2 to be: Edit User Account	Jaco
Version 1.5	Added Edit User State Diagram to FRQ	Jaco
Version 1.5	Added RemoveContact State Diagram to FRQ	Jaco
Version 1.5	Document editing and grammar checks	Vincent
Version 1.5	Added context for understanding	Vincent
Version 1.6	Added Appendix G	Jaco
Version 1.6	Added Content to Section 3.1	Jaco
Version 1.6	Worked on Section 3.1	Luke
Version 1.7	Added FRQ 4 and 5 Diagrams	Henko
Version 1.7	Added FRQ 9 and 12 Diagrams	Henko
Version 1.7	Added content to 3.4 and 1.4	Henko
Version 1.7	Worked on section 3.1	Luke, Jaco
Version 1.7	Added last 2 iStar diagrams	Luke, Jaco
Version 1.8	Completed Section 3.1	Luke
Version 1.8	General fixes on format of 3.1	Jaco
Version 1.9	Context/grammar/spellchecking	Vincent
Version 1.9.2	Context/grammar/spellchecking	Vincent
Version 2.0	Added Screenshots	Jaco
Version 2.0	Added Screenshot descriptions	Luke