SMSEncryption Project Requirements Specification

COS730 - Group 1

Version 3.7
May 11, 2014

History

Date	**Version**	**Description**	**Updater**
5 April	Version 0.1	Document Created	Henko
5 April	Version 0.2	Document layout added	Henko
10 April	Version 0.3	Added other sections	Henko
11 April	Version 0.4	Added parts to Introduction	Henko
12 April	Version 0.5	Added General Description	Henko
21 April	Version 0.6	Modification of current document and IEE compliance adjustment	Luke, Jaco
21 April	Version 0.7	Added to some empty sections	Jaco, Luke
21 April	Version 0.8	Added appendix A	Hein
21 April	Version 0.9	Added appendix B	Hein
21 April	Version 1.0	Added appendix C	Hein
21 April	Version 1.1	Overview of grammar and sentence structure as well as some contextual/ explanative additions	Vincent
26 April	Version 1.2	Added Appendix for Desgin Principles	Jaco
26 April	Version 1.3	Added Appendix for Secure Desgin	Luke
26 April	Version 1.4	Added background verbatim from client	Luke
28 April	Version 1.5	Added iOS guidelines to Appendix Z	Jaco
28 April	Version 1.6	References according to IEEE	Jaco
29 April	Version 1.7	Added FRQ4-11, FRQ1.1 and FRQ1.2	Jaco
29 April	Version 1.8	Updated original Usecase Diagram	Jaco
29 April	Version 1.9	Protocol description for Appendix C	Hein
30 April	Version 2.0	Some grammar checking, added some context	Vincent
2 May	Version 2.1	Aesthetics of document, expanded Overview	Jaco
2 May	Version 2.2	Added state diagram, and general i* diagram	Vincent
2 May	Version 2.3	Updated numbers on usecase and FRQ's	Jaco
2 May	Version 2.4	Split FRQ's into sub functions	Jaco
3 May	Version 2.5	Fixed minor errors to Appendix C	Henko
07 May	Version 2.6	Added Contact's IStar diagram	Vincent
07 May	Version 2.6.1	Added general state diagram to folder only	Vincent
07 May	Version 2.7	FRQ adjustments and general error fixes	Jaco
07 May	Version 2.7.1	Updated usecase diagram	Jaco
07 May	Version 2.8	Added state diagrams: AddContact, AddUser, EditContact, Local Authentication	Luke, Jaco Vincent

Date	Version	Description	Updater
07 May	Version 2.9	Modified FRQ and DC layout	Jaco
07 May	Version 3.0	Made changes to section 5.4	Luke
09 May	Version 3.1	Layout modification	Jaco
10 May	Version 3.2	Updated general state diagram	Vincent
10 May	Version 3.3	Expanded section 2.2 with old info	Jaco
10 May	Version 3.3.1	Added FRQ 12, Synchronise contact	Jaco
10 May	Version 3.3.2	Expanded DC 1 with more specific details	Jaco
10 May	Version 3.3.3	Added standards compliance section	Jaco
10 May	Version 3.3.4	Minor alterations to Software system attributes	Jaco
10 May	Version 3.4	Added to and fixed general state diagram, and general i* diagram	Vincent
10 May	Version 3.4	Added encrypt and decrypt state diagrams	Hein
10 May	Version 3.5	Modified encrypt and decrypt state diagrams	Luke
11 May	Version 3.6	Updated Usecase, added admin i*diagram	Jaco
11 May	Version 3.6.1	Added Appendix F and i* diags to it	Jaco
11 May	Version 3.6.2	Updated state diagrams	Jaco
11 May	Version 3.6.3	Inserted encrypt and decrypt state diagrams	Jaco
11 May	Version 3.6.4	Added mitigation features to Appendix D	Jaco
11 May	Version 3.6.5	Added inputs and outputs to service contracts	Luke
11 May	Version 3.6.6	Add Appendix F to overview Edit FRQ 2 to be: Edit User Account	Jaco
11 May	Version 3.6.7	Added Edit User State Diagram and Remove Contact State Diagram to relevant FRQ	Jaco
11 May	Version 3.7	Document editing, grammar checks, added context for understanding	Vincent

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 requirements are well understood by all stakeholders. It is therefore intended for all stakeholders of the project; including the developers and customers.

1.2 Background

Certain operations within remote parts of South Africa require reliable transmission of data which cannot be achieved through traditional means of data transmission (e.g. GSM, 3G, etc). As an alternative, SMSes are used to transmit these messages. Some of the data which is transmitted is sensitive, and requires some form of encryption. As traditional encryption functions often produce characters which fall outside the character set of modern cellphones, it is necessary to use an encoding scheme to translate these characters back into the cellphone's character set. Most encoding schemes (such as base64) increase the length of the message this may result in the encrypted message exceeding the maximum character allowance for SMS.

1.3 Scope

The goal of this project is to create a mobile application which can be used to encrypt text before sending it via SMS technology, which can be decrypted on the receiving end. The application must be able to be used on more than one platform (i.e. iOS and Android).

By using the SMSEncryption application, the user will be able to encrypt messages which can only be decrypted by using the same application on the receiving end. The application will require local authentication in order to gain access to the appliaction and make use of it's features.

The benefit of this application is that you can use SMS technology to send confidential messages which can only be viewed by you and the desired recipient of the message - who is the only party who can unencrypt the message.

1.4 Definitions, acronyms and abbreviations

- SMSEncryption - The name of the current 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.
- Message - The text intended to be sent from a sender to a receiver or stored once said message has been encrypted via SMSEncryption.
- Plaintext - Is information a sender wishes to transmit to a receiver.
- Ciphertext - Is the result of encryption performed on plaintext using an algorithm, called a cipher.
- Encrypt - To alter the plaintext using an algorithm so as to be unintelligible to unauthorized parties.
- Decrypt - The act of decoding a ciphertext back into the orginal form before encryption took place.
- User - An authorised person who will interact with the application.
- Sender - The person who authored and intends to send a message that has been encrypted via the application.
- Receiver - The intended party who receives a message which has been encrypted via the application.
- SMS - Short Message Service (SMS) is a text messaging service component 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.

- Entropy - The expected value of the information contained in a message.

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

The Overall Description section will provide a background to the reader for the SMSEncryption application, and contains the following sections: Product perspective, Product functions, User characteristics, Constraints, Assumptions and Dependencie as well as Apportioning of requirements.

The Specific Requirements section contains 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.

The sections contained in Specific Requirements include 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 SMSEncryption for each version of the developed application.

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

2 Overall Description

2.1 Product perspective

2.1.1 Description

SMSEncryption is a new product which can be used in conjunction with any mobile text manipulation application, such as the general keyboard input used by the different mobile operating systems, or any other text manipulating application, capable of using the basic GSM character set - should you require encryption and decryption functionality for secure communication between two parties.

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 large SMSes will be expensive.

2.1.2 System interfaces

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 in the agreed upon way.

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 a clipboard on the device to facilitate 'copying' and 'pasting' of texts or ciphertexts.

2.1.6 Memory

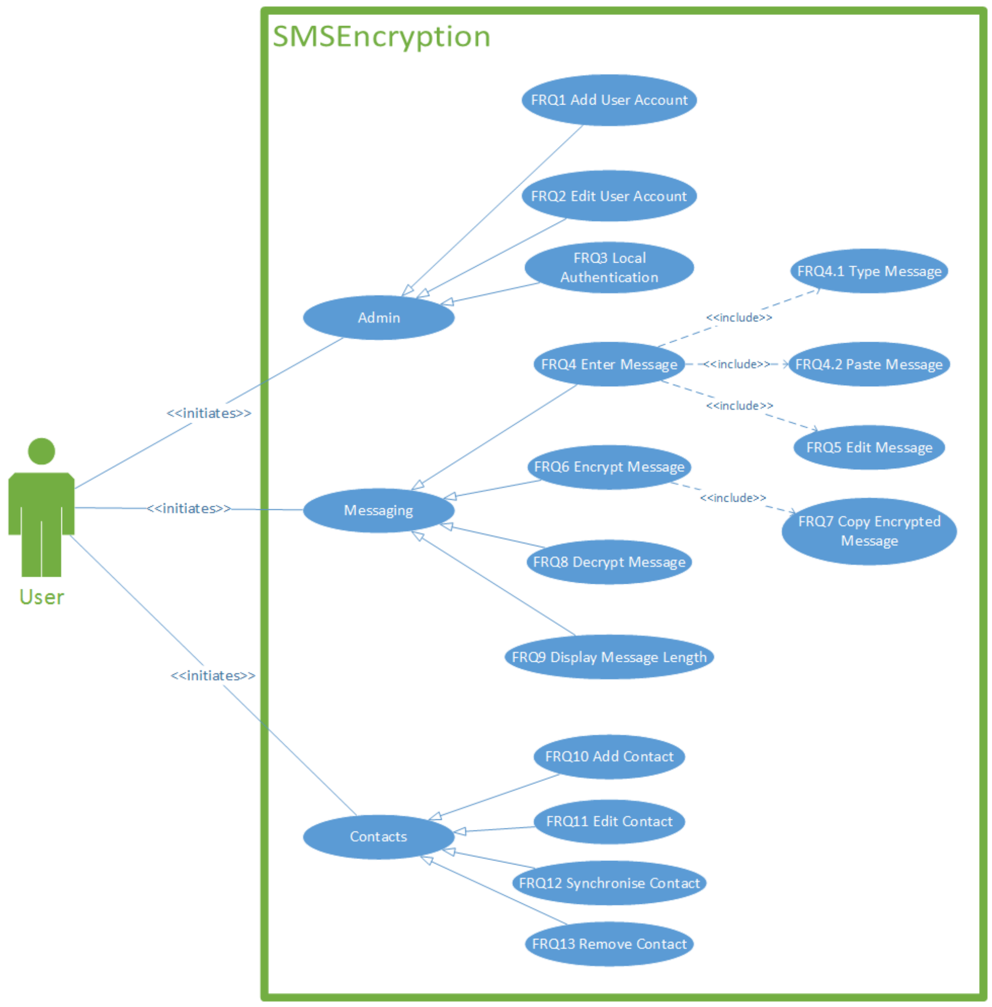
The device needs minimal storage space on the device for a local database and the application is 2mb large.

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

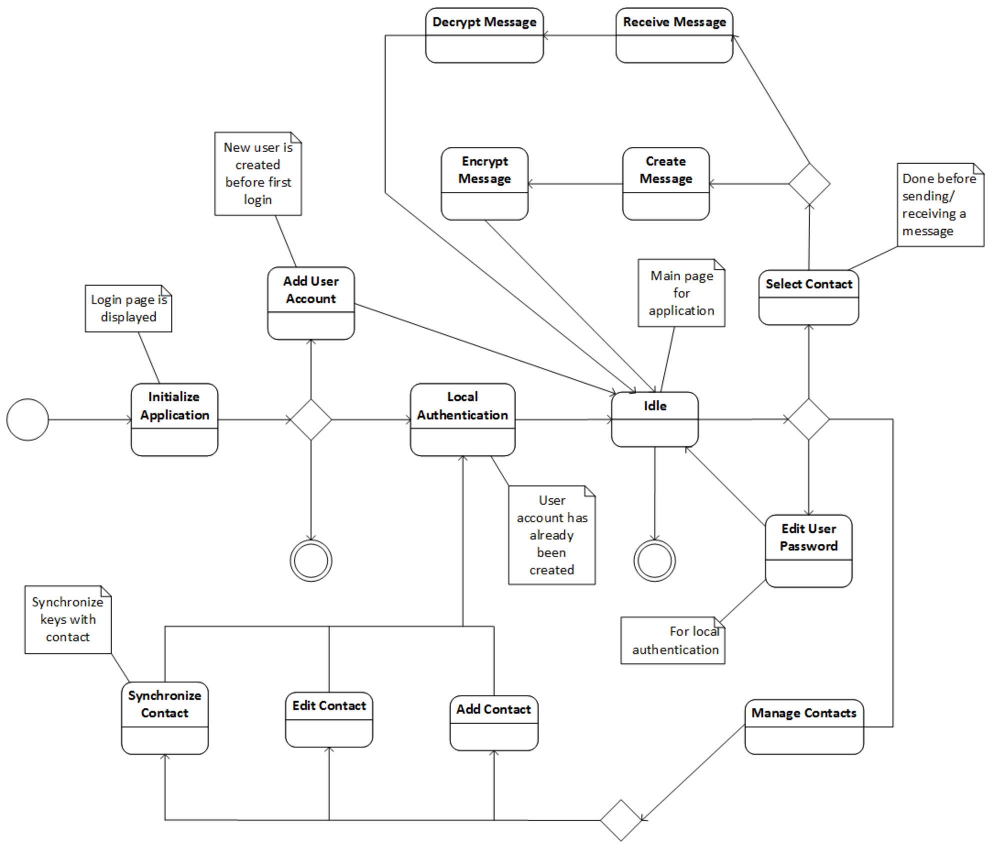
2.1.8 Use Cases

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 a user account in the application.
  - On first use of the application, a username and password (with confirmation of password) must be created that will ensure user authentication for future use of the application.
  - Only a single user account is allowed per device.
- The user shall 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 in.
  - 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.

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 will be used to perform encryption.
- When creating a message within the application, the text can be either typed, or pasted in via the device "clipboard".
- Once a 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 should 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.

  - Before ciphertext can be pasted into the application, the sender of the message must first be selected as a contact. The sender's contact details will be used to decrypt the ciphertext.
  - The desired receiver should be able to decrypt the message back into its original plaintext.

- The message length should be displayed.
  - This is to ensure that message length does not exceed the maximum amount of 144 characters.

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.
    - 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).
- The application must allow a contact to be edited once 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 who will have full access to all the features provided by the application after local authentication, 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.
- 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.

2.4 Constraints

- The encrypted message length is limited to 160 characters, regardless of plaintext message length.
- 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) for text between different interfaces/applicaitons.

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.

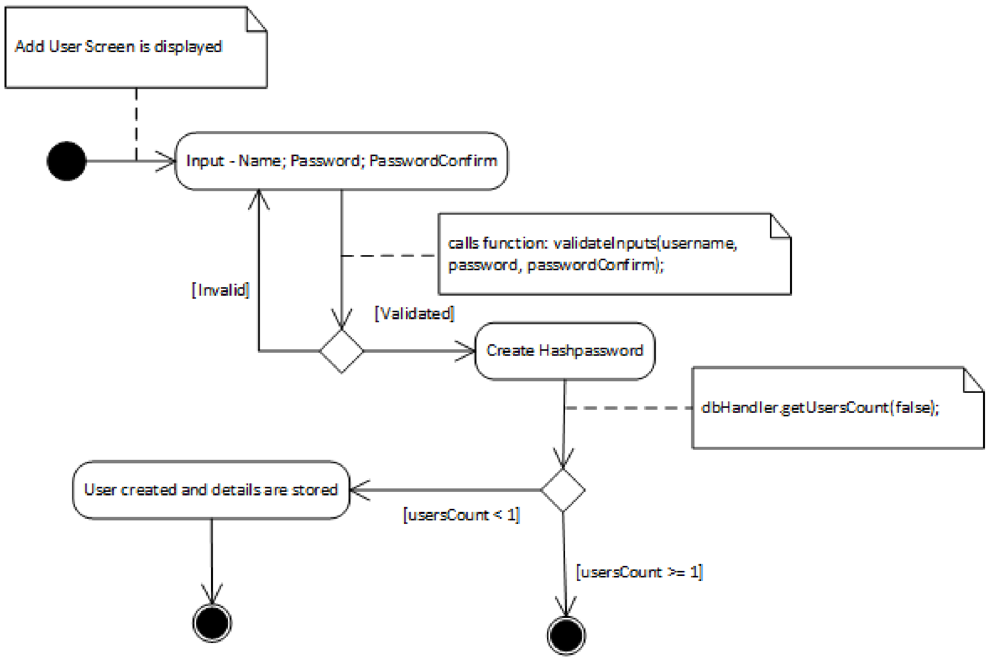
3 Specific requirements

3.1 External interfaces

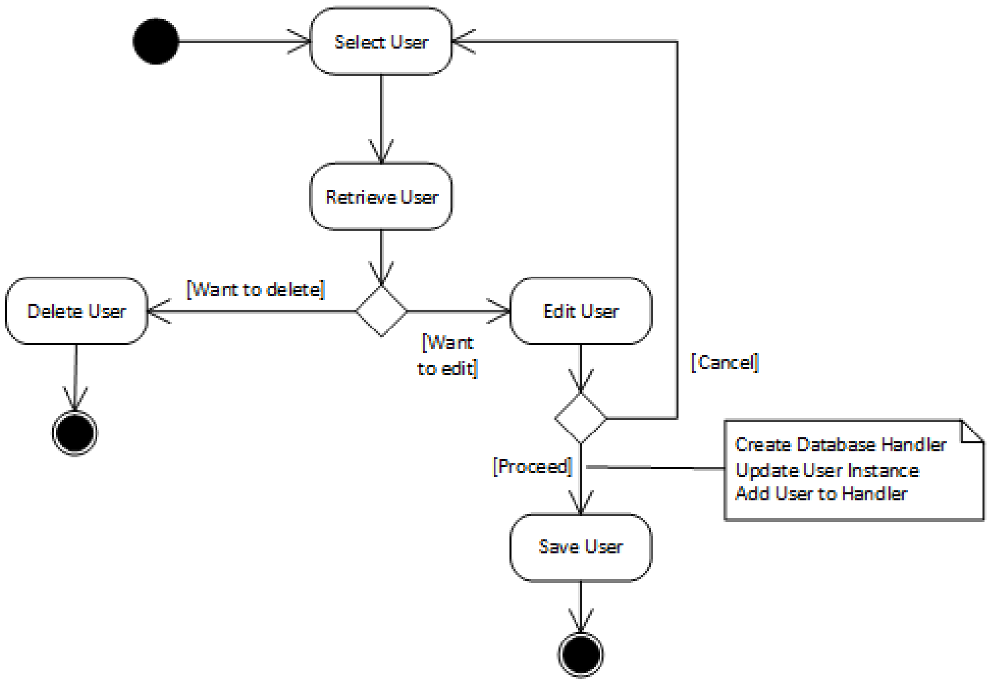
3.2 Functions

3.2.1 Admin Functions

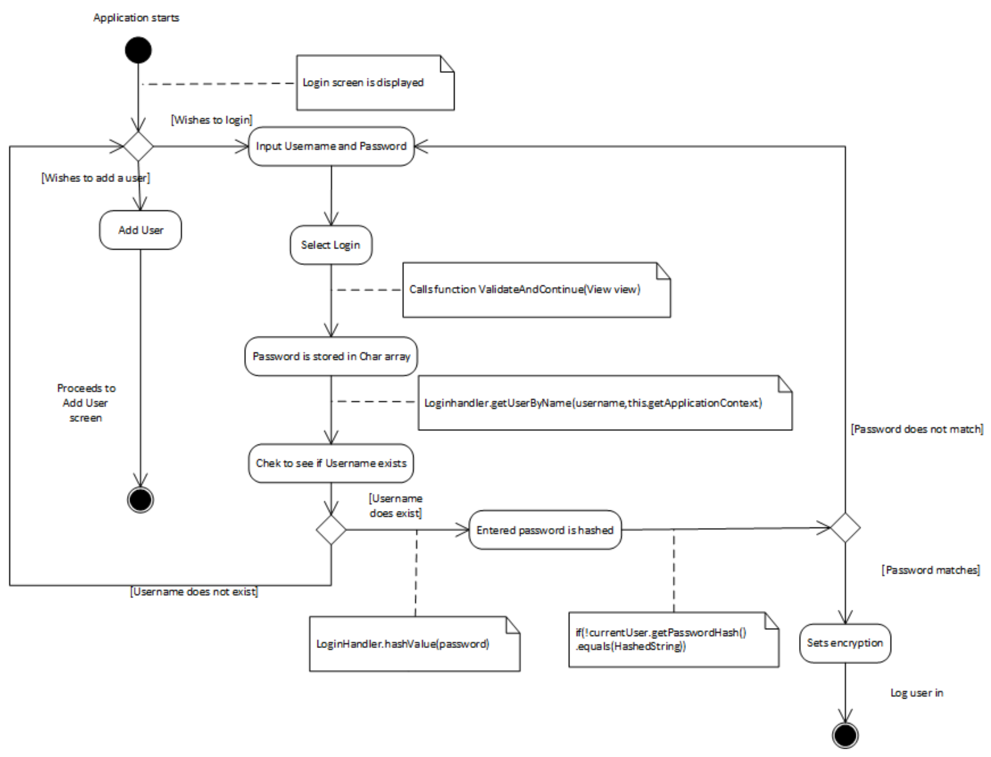
- Add User Account : FRQ1
  (Source: Bernard Wagner, Priority: Medium)
  - A user must be able to create a password protected account for the app.
  - **Inputs:** The user inputs a desired user name , password and fills in the password conformation field.
  - **Outputs:** A user account is either created and stored or an error message is displayed and no account is created.


Add User Screen is displayed
Input - Name; Password; PasswordConfirm
calls function: validateInputs(username, password, passwordConfirm);
[Invalid]
[Validated]
Create Hashpassword
dbHandler.getUsersCount(false);
User created and details are stored
[usersCount < 1]
[usersCount >= 1]

- Edit User Account : FRQ2
  (Source: Group Deliberation, Priority: Medium)
  - The user must be able to edit his/her authentication details.



- Local Authentication : FRQ3
  (Source: Bernard Wagner, Priority: Medium)
  - The application must authenticate a user by requiring a password in order to log into the application, in order to ensure confidentiality.
  - **Inputs:** The user inputs his account username and password.
  - **Outputs:** After the application has checked the users details against the stored details under that username, it will log the user in and show them the main menu or it will display an error message and refuse the inputed details.

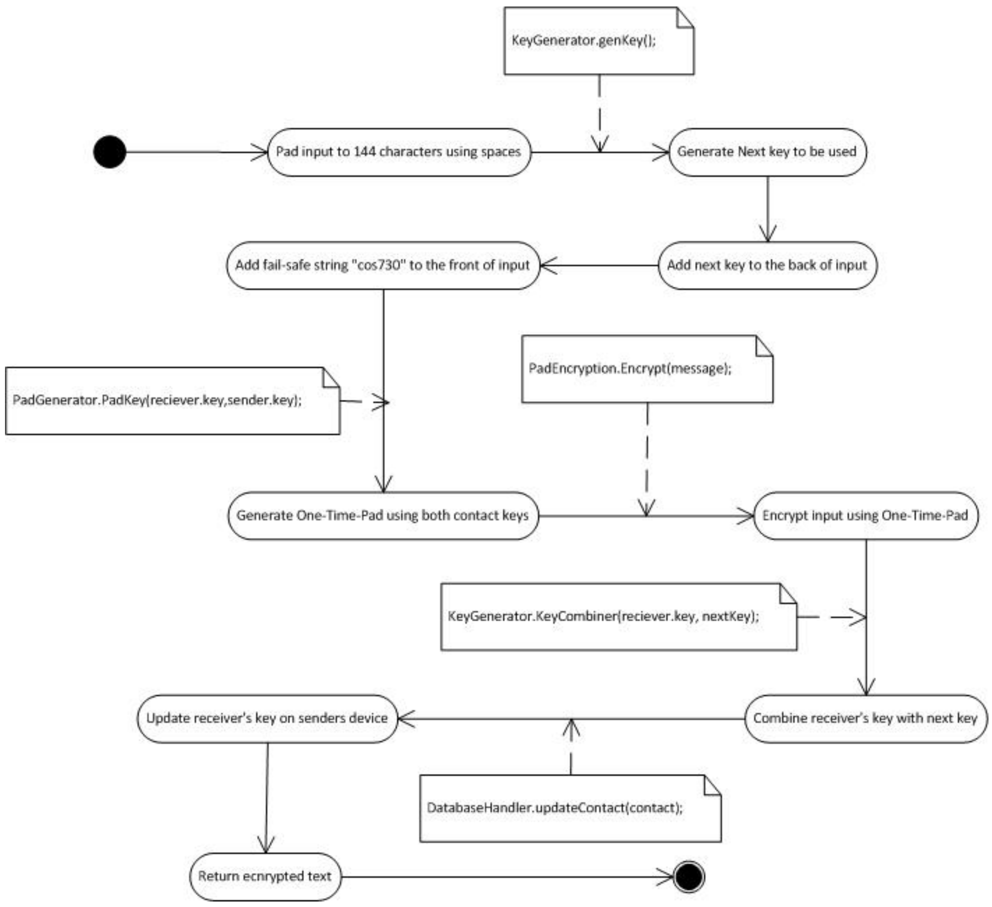

Application starts
Login screen is displayed
[Wishes to login]
Input Username and Password
[Wishes to add a user]
Add User
Select Login
Calls function ValidateAndContinue(View view)
Password is stored in Char array
Loginhandler.getUserByName(username,this.getApplicationContext)
Proceeds to Add User screen
[Password does not match]
Chek to see if Username exists
[Username does exist]
Entered password is hashed
[Password matches]
[Username does not exist]
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.
- Type message : FRQ4.1
  (Source: Bernard Wagner, Priority: High)
  - A user must be able to type text into the application.
- 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 clipboard.
- Edit Message: FRQ5
  (Source: Bernard Wagner, Priority: Medium)

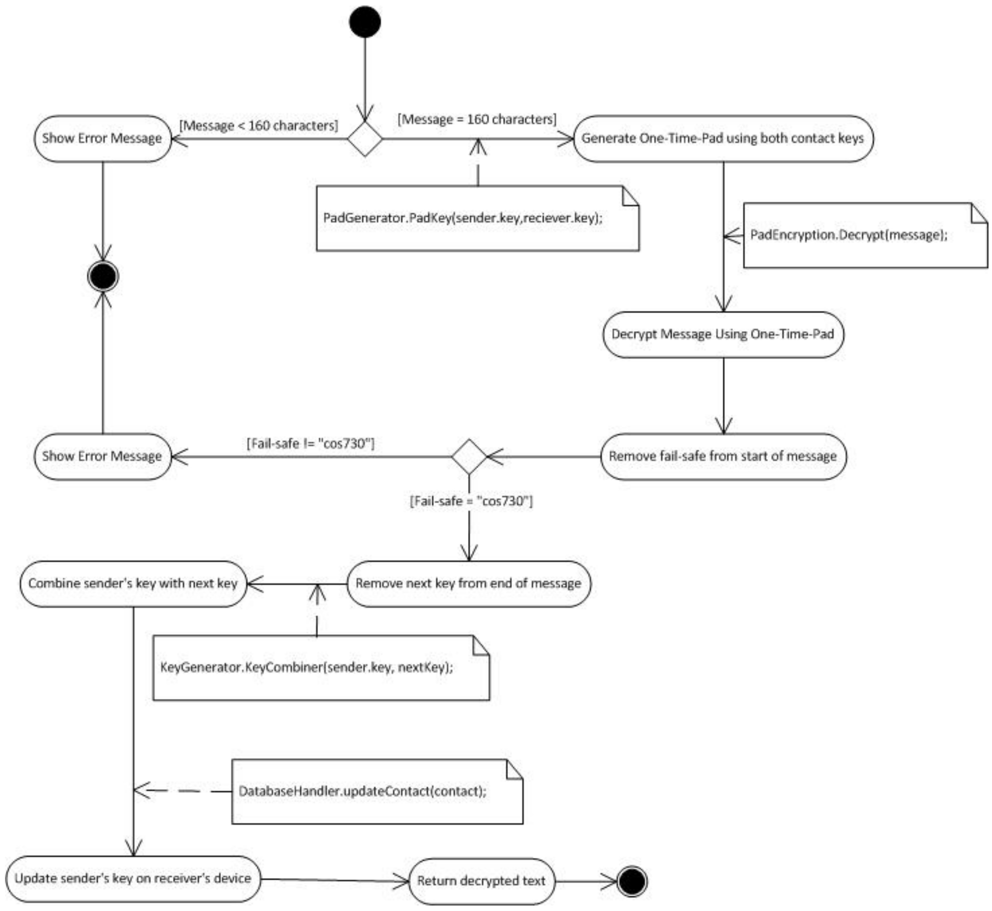
  - The message text must be editable once it has been input into the application by the user.

- Encrypt message : FRQ6
  (Source: Bernard Wagner, Priority: High)
  - The message must be encrypted using a suitable encryption method.


KeyGenerator.genKey();
Pad input to 144 characters using spaces
Generate Next key to be used
Add next key to the back of input
Add fail-safe string "cos730" to the front of input
PadEncryption.Encrypt(message);
PadGenerator.PadKey(reciever.key,sender.key);
Generate One-Time-Pad using both contact keys
Encrypt input using One-Time-Pad
KeyGenerator.KeyCombiner(reciever.key, nextKey);
Update receiver's key on senders device
Combine receiver's key with next key
DatabaseHandler.updateContact(contact);
Return ecnrypted text

- 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 a suitable messaging application.

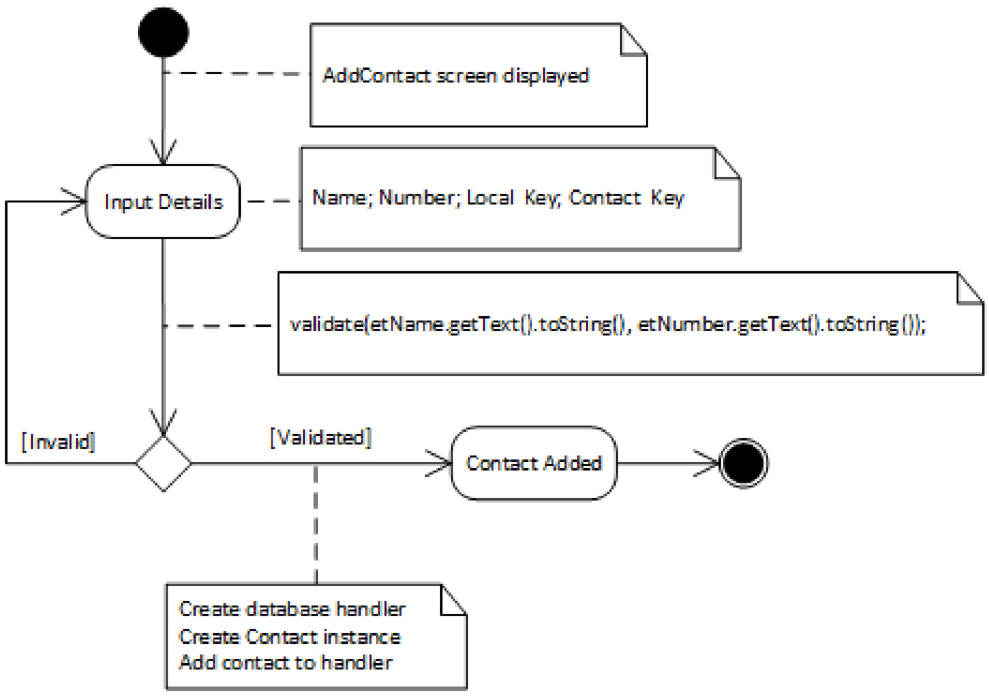
- Decrypt message : FRQ8
  (Source: Bernard Wagner, Priority: High)
  - The application must be able to decrypt the message (on the receiving end) to reveal the original text.


Show Error Message
[Message < 160 characters]
[Message = 160 characters]
Generate One-Time-Pad using both contact keys
PadGenerator.PadKey(sender.key,reciever.key);
PadEncryption.Decrypt(message);
Decrypt Message Using One-Time-Pad
Show Error Message
[Fail-safe != "cos730"]
Remove fail-safe from start of message
[Fail-safe = "cos730"]
Combine sender's key with next key
Remove next key from end of message
KeyGenerator.KeyCombiner(sender.key, nextKey);
DatabaseHandler.updateContact(contact);
Update sender's key on receiver's device
Return decrypted text

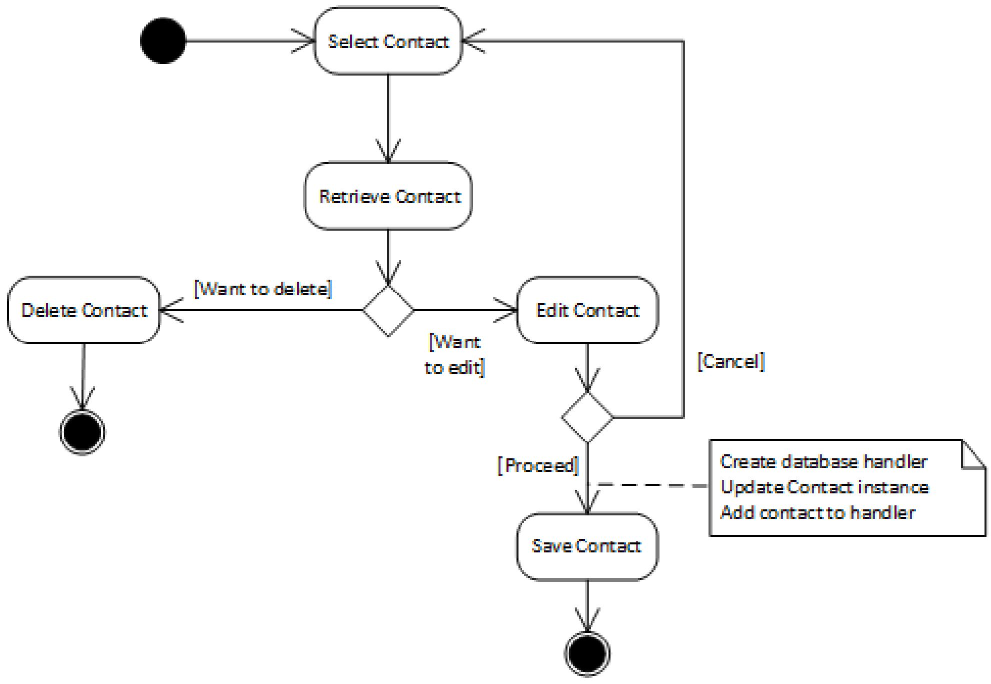
- Display message length : FRQ9
  (Source: Bernard Wagner, Priority: Low)
  - The numbers of characters in the message must be displayed.

3.2.3 Contacts Functions

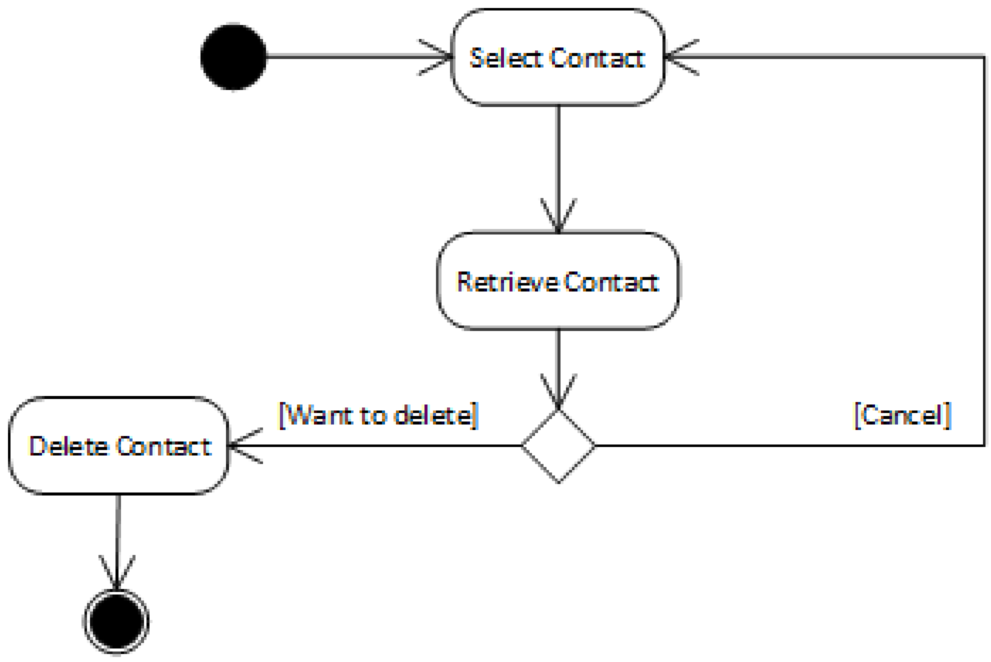
- Add Contact : FRQ10
  (Source: Bernard Wagner, Priority: High)
  - Before communicating with someone, the receiver must be added as a contact, in order to be able to communicate with that user.
  - **Inputs:** The user inputs a contact's Name,number,local key and the contacts key.
  - **Outputs:** The contact is created and stored or the apllication shows an error message if input details were invalid.


AddContact screen displayed
Input Details
Name; Number; Local Key; Contact Key
validate(etName.getText().toString(), etNumber.getText().toString());
[Invalid]
[Validated]
Contact Added
Create database handler
Create Contact instance
Add contact to handler

- Edit Contact : FRQ11
  (Source: Bernard Wagner, Priority: Medium)
  - A contact must be editable once it has been added.
  - **Inputs:** The user selects a contact which they wish to modify the details for or to delete from the application.
  - **Outputs:** The selected contacts details will be modified and stored or the contact will be deleted.



- Synchronise Contact : FRQ12
  (Source: Group Deliberation, Priority: High)
  - The user shall be able to synchronise with a contact at any time after they have been added.
- Remove Contact : FRQ13
  (Source: Bernard Wagner, Priority: High)
  - A user must be able to remove a 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).
  - Expected time the applcaiton should take to decrypt is less than one second.
  - Expected time the applicaiton should take to encrypt is less than one second.
- The encryption method must be secure.
  - The Encyprion method used should have an entropy of less than 1%.
- The applcaiton must be secure.
  - A users password must not be viewable as to prevent unauthorized use of the application.
  - A users password should be encrypted or stored as a hash value.
  - 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

3.5 Design constraints

- Message length : DC1
  (Source: Bernard Wagner, Priority: High)
  - Due to the fact that the primary messaging service which the client intends to use is SMS this limits the input size of the text to 160 characters.
  - The encryption process uses 6 characters for a fail safe and 10 characters to embed the next key.
  - This means plaintext must be limited to 144 characters.
  - To maintain consistency we enforce this as the maximum length of messages which the application can encrypt, regardless of the intended messaging application.
- The usable characters : DC2
  (Source: Bernard Wagner, Priority: High)
  - The usable character which can be encrypted by the application is the GSM character set because the primary intended messaging service that the client wishes to use is SMS.
- Application resource requirementsr : 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 stipuled in Appendix D, Secure design principles.

- The Application must conform to the Android and iOS design principles for each respective platform, 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 in the application should be static and exist as long as the application is open or said information is removed/edited.
- The ciphertext should decrypt into its plaintext.

3.6.2 Availability

- The user should be able to use the application as long as it is running and should not be made to wait while the application performs a function.
- The application should not interfere with any other applications which are running on the device.

3.6.3 Security

- A secure encryption and decryption method will be used for the localised database.

3.6.4 Maintainability

- The source code should be maintainable (simplistic 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 operating systems namely Android and iOS.

4 Appendix A - RSA

Introduction

This appendix is about research done in pursuit of a possible solution to the given problem. It is about RSA and how we researched possible RSA solutions to encrypt an SMS message.

Method

We started by trying to use the build in RSA implementation that is built into Java. After that we did research into the background of RSA, more specifically the maths that make it work. We then attempted numerous combinations of the mathematical principals behind RSA to see if any of them could manage to be used to fulfill the needed requirements.

Result

The build in RSA used keys that would become too large to redistribute, in order to accommodate encrypted text of as close as possible to 160 characters after padding required a 700 bit key. It also limited the amount of characters to about 77 characters before it became larger than 160 characters.

We implemented a custom RSA but it started out weak due to the limits imposed by our character set. We looked into an alternative where 2 encrypted characters represented 1 plain text character. This gave some strength to the encryption but limited the message one could send to 80 characters. The client said that this was not an option.

Discussion

When thinking about modern encryption we think about RSA and how useful it is, the thing we easily forget is behind the scenes large amounts of data is transferred just to enable the encryption and decryption. It is because of the keys being too large to SMS that the build in RSA was disregarded, along with uncontrolled padding in an environment where message length was extremely important. In our custom RSA we could control the length of the key but just like the build in version it limited characters too much.

Conclusion

RSA works well in modern technologies but it only works well where we can transfer large amounts of data relatively easily such as for example data transfer over the internet. We need large keys to make the encryption strong due to the limitations of the character set, but with no way of distributing the key and the limitations to the key length RSA is not the answer to this problem.

5 Appendix B - One time pads

Introduction

This is about 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 apart from the max possible length it could be.

Method

We did some research into one time pads and why it is that they are so strong. After that we implemented a onetime pad algorithm and it looked very promising.

Result

The encryption is very strong, allows for 1 to 1 character encryption thus enabling us to have a plain text message of 160 characters fully utilizing space. It seemed to be the solution to the problem.

Discussion

The first thing that comes to mind when thinking about one time pad encryption is how to distribute the pad. The pad needs to be distributed between the two parties and the must any given moment in time know what the next line that will be used will be, in other words it requires synchronization.

Conclusion

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

6 Appendix C - Encryption Protocol

Introduction

This Appendix discusses the encryption protocol designed after extensive research. It comprises of a combination of a key based encryption protocol along with a one-time pad encryption, it is because of this we refer to the protocol as being a Hybrid encryption. This protocol is very complex and very powerful. Because it uses one time pad encryption the encryption itself is unbreakable unless you have the pad. The pad is generated using two special keys stored on each user representing the users internal key for that contact, and that contacts internal key for himself. Thus the pad is never communicated instead each message send contains the next key to be used. Thus it requires the previous key to get the next key creating a key dependency, increasing encryption strength. However to further increase the strength the key send is used in combination with the internal key to produce the next key to be actually used, because the key changes after a message has been sent, replay attacks are impossible as the message would no longer decrypt.

Diagrams

Work flow diagram

Description

As stated above, the 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 keys with each other. These keys are instantiated when users add each other as "contacts" within the application. When adding a "contact", a user is provided a key that has to be shared with the intended receiver of any future messages. Once both have been provided with a key (that they share with one another; so that each of them has their own key, along with the shared key from the intended contact), both keys are stored on the device as a "contact". This is done for each contact so no keys are shared beyond one contact.

When a user wishes to send a message to a specific contact, the two keys stored for that contact is 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.

Next a new key is generated to be used for the next communication. 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 to 154 characters.

Next we add the failsafe string "cos730" to the beginning of the message, this used by the application to determine the success of the decryption so that local keys are not changed on failed decryptions.

Once the key is added to the message, the message is then encrypted using the special One-Time-Pad.

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

To receive a message would be the opposite: the two internally stored keys for the contact that sent the message would be retrieved, and used in the one-time pad function to produce a one-time pad. This pad is then used to decrypt the 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 success. If it fails it will show error message else it would continue. The new key is then taken from the decrypted message - as it will be the last 10 characters (the special key created by our function).

The key that was taken from the message is used in combination with the internal key for that contact to produce a new combined key, which is used to replace the key the receiver has stored for that contact.

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 attemt to mitigate each threat in SMSEncryption.

- 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 SMSEncryption data is stored encrypted using the entered password of the user as well as the username.

- 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 which you are developing for will handle memory management , this can be exploited by would be attackers by looking for unprotected areas in memory.

In SMSEncryption 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.

In SMSEncryption password login is required to gain access to the application and its content, the 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 implements or use.

In SMSEncryption we are making use of AES encryption for the database.

- Lack of Binary Protections

This is a universal problem as almost all code which is compiled into binaries will be able to be reverse engineered into some form of discernable source code.

With regard to SMSEncryption this means we must ensure the former sections are well taken care of to mitigate the effects of this threat.

8 Appendix E - Design principles

Introduction

This sections contains the design principles available for Android and iOS developers. As the goal is to make SMSEncryption for both these platforms both sets of principles need 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 users content is at its heart.
  - Here are some ways to make sure that your designs elevate functionality and defer to the users content.
    - Take advantage of the whole screen. Reconsider the use of insets and visual frames andinsteadlet the content extend to the edges of the screen. Weather is a great example of this approach: The beautiful, full-screen depiction of a locations 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 elementssuch as Control Centerprovide 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 itgiving the impression of looking through rice paperit 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 colorsuch 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 todays date and perform other calendar tasks.
        * When users select a month, the year view zooms in and reveals the month view. Todays 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 doesnt measure the beauty of an apps artwork or characterize its style; rather, it represents how well an apps appearance and behavior integrates with its function to send a coherent message.
    - People care about whether an app delivers the functionality it promises, but theyre also affected by the apps appearance and behavior in strongsometimes subliminalways. 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 thats intrusive, frivolous, or arbitrary, people might question the apps reliability or trustworthiness.

  - On the other hand, in an app that encourages an immersive tasksuch as a gameusers expect a captivating appearance that promises fun and excitement and encourages discovery. People dont expect to accomplish a serious or productive task in a game, but they expect the games appearance and behavior to integrate with its purpose.

- Consistency
  - Consistency lets people transfer their knowledge and skills from one part of an apps UI to another and from one app to another app. A consistent app isnt a slavish copy of other apps and it isnt 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 its 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 objectsfor 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 peoples 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 andduring operations that last more than a few secondsa 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 shouldnt be the only feedback mechanism because people cant always hear their devices.

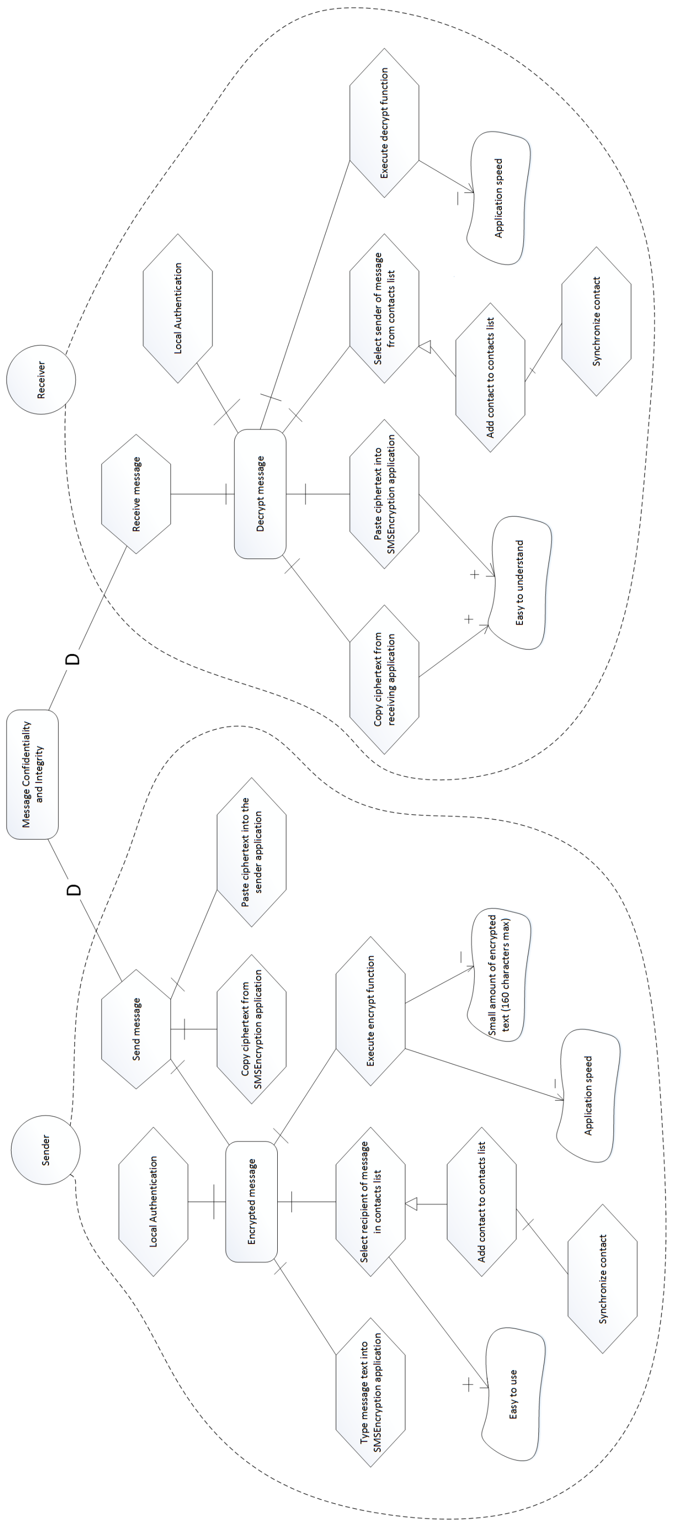
- Metaphors
    - When virtual objects and actions in an app are metaphors for familiar experienceswhether these experiences are rooted in the real world or the digital worldusers quickly grasp how to use the app.
    - Its best when an app uses a metaphor to suggest a usage or experience without letting the metaphor enforce the limitations of the object or action on which its 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 its 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 thats underway.

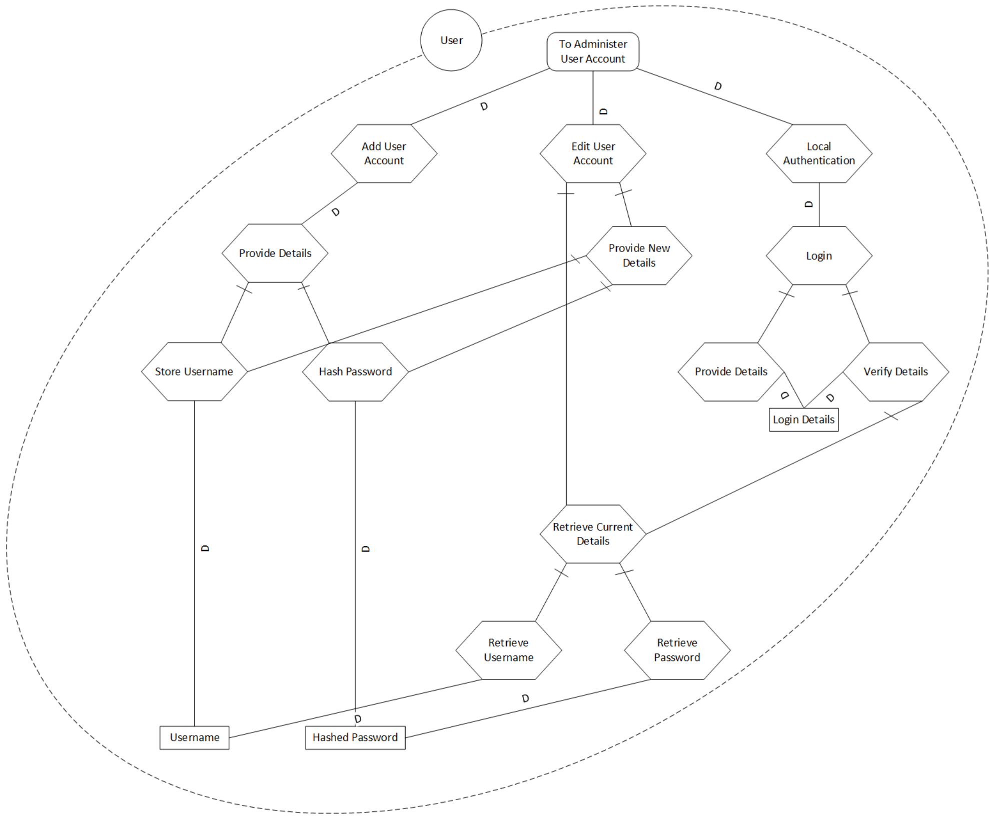
9 Appendix F - i* Diagrams

This sections contains i* diagrams we made during the requirements elicitation process.

General i* Diagram

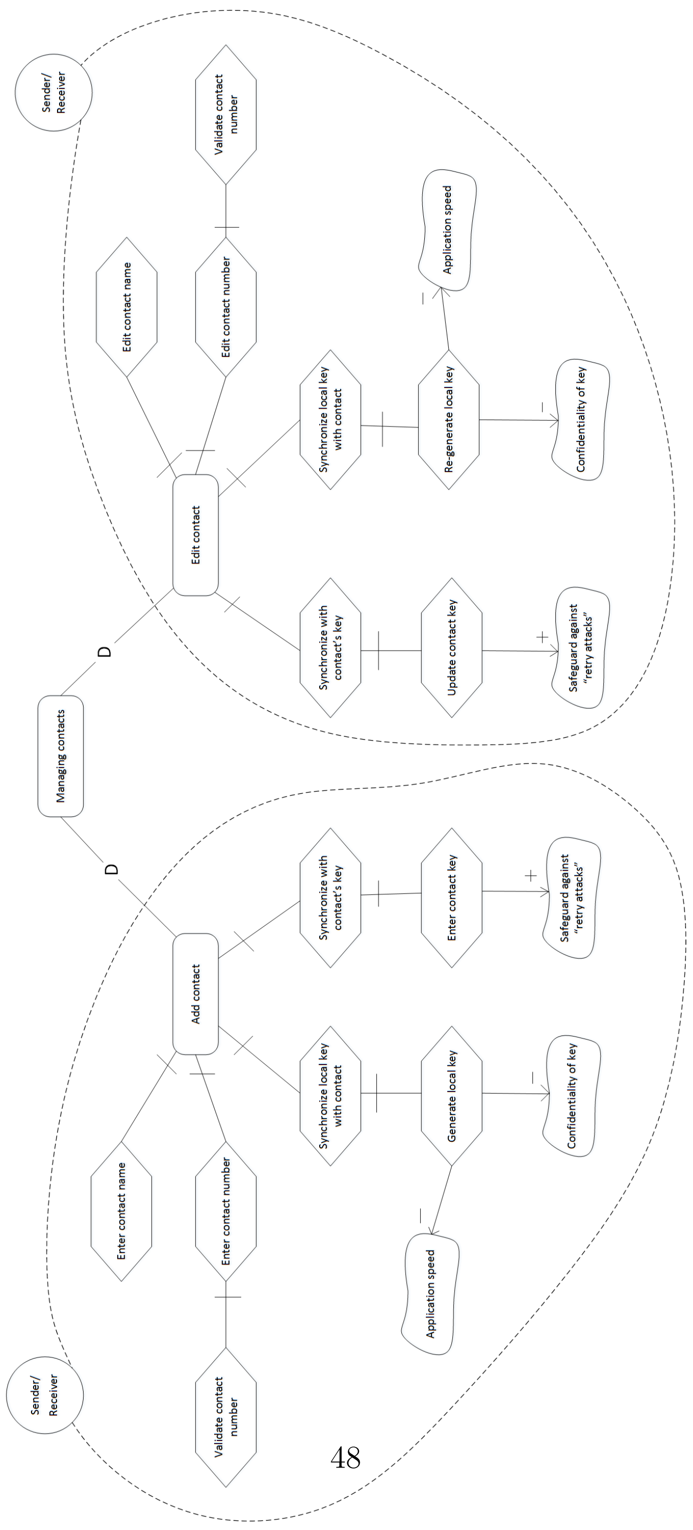

Receiver
Local Authentication
Execute decrypt function
Application speed
Select sender of message from contacts list
Add contact to contacts list
Synchronize contact
Receive message
Decrypt message
Paste ciphertext into SMSEncryption application
Easy to understand
Copy ciphertext from receiving application
D
Message Confidentiality and Integrity
D
Paste ciphertext into the sender application
Send message
Copy ciphertext from SMSEncryption application
Execute encrypt function
Small amount of encrypted text (160 characters max)
Application speed
Sender
Local Authentication
Encrypted message
Select recipient of message in contacts list
Add contact to contacts list
Synchronize contact
Type message text into SMSEncryption application
Easy to use

Admin functionality i* Diagram



Messaging functionality i* Diagram

Contacts functionality i* Diagram


Sender/ Receiver
Validate contact number
Edit contact name
Edit contact number
Application speed
Synchronize local key with contact
Re-generate local key
Confidentiality of key
Edit contact
Synchronize with contact's key
Update contact key
Safeguard against "retry attacks"
D
Managing contacts
D
Synchronize with contact's key
Enter contact key
Safeguard against "retry attacks"
Add contact
Synchronize local key with contact
Generate local key
Confidentiality of key
Enter contact name
Enter contact number
Application speed
Sender/ Receiver
Validate contact number
+
−