SIT 302

Analysis and Design Group Report

Natalie Jablonsky, Matthew Leeds, Oliver Necovski

Table of Contents

Introduction

Producing a product with a lot of requirements however contains many limitations is challenging. This analysis report details many aspects of our project and its current state, we explain the problem we are trying to solve and how we solve it. Limitations in the technology we are using creates obstacles which require further action and alterations to our project to allow us continue with our objective.

Problem Defined

Sensitive data can only be sent by insecure means over a mobile network. Currently encrypted messages require a data connection to be sent, this is not available to everyone all the time Network congestion can cause mobile network data to a standstill, this is common at sporting events, music concerts and other crowded places while standard network SMS messages stay active.
As a team we will create a secure system that encrypts and decrypts an SMS over a mobile network without a data connection.

As a team we will create a method of sending an Encrypted SMS over mobile network and have it decrypted on the other end.
We are using the Arduino Microcontroller and various add ons to allow us to connect this to a mobile network using the the use of a PC.
Using the Arduino we will be able to push a message from the computer to the arduino board which can then encrypted and sent to an another awaiting arduino microcontroller setup or another user who may have a decryption key for the message.
This is particularly important where a data connection may not be reliable enough or cannot be found, eg: rural towns, medical services, crowded areas and more.

Systems Analysis & Outcomes

Due to the cost of the hardware, the prototype will be demonstrated by encrypting and sending a message from the arduino, then sending the message back for decryption.

The device was originally planned to use AES encryption, but issues with processing unicode characters meant using a Caesar cypher for now, with the possibility of implementing our own complex encryption method later if time allows.

Additionally, if we went with AES, it would only supports 112 characters of plain text, which should become 152 characters encrypted. SMS has a limit of 160 characters, though the possibility of 'Concatenated SMS' was brought up, along with MMS, but the focus is currently on getting encryption working.

Application Functionality

Our end goal is to be able to provide an easy to use solution for encrypting and decrypting SMS messages. As previously discussed this solution is perfect for the use secure communication where a data connection is not available, our prototype has proven that we are closer to achieving our objective.

With the combination of hardware and software we have successfully built a prototype to encrypt SMS Messages which can be sent to another Arduino Microcontroller to be decrypted. Currently Project ASM is using one Arduino Microcontroller which is able to send the encrypted message to a mobile phone using a Caesar shift cipher.
The prototype is only functional using back end code to write the message and add the desired phone number to send the message.
The Arduino Microcontroller is set to follow a process to send the message and listen for incoming messages.
Once the message has been entered into the appropriate places the code is executed.
The microcontroller will encrypt the message which is displayed on the LCD screen before sending then continues to listen for any incoming messages to decrypt.

Further functionality will be added into the final product which will be implemented into an application. The application will have user friendly and GUI with simplistic navigation.

Database/Data Model Designed

Due to the nature of this project, there was no need to implement a data base as we are creating a secure communications channel for users to send and receive sensitive data. If the project evolves and there is a need to also store the data that is being sent, a database design will be required.

ASM Storage		
Device	***Data stored***	***Storage type***
Arduino Mega 2560	Program: Send/Receive Encrypt/Decrypt	Permanent
Simcard	Message and Details (Time/Date/Sender)	Data is removed after criteria is met. (Messages stored on sim > 3)
Mobile Phone	Message	Optional (Users are free to keep or delete messages that are sent or received on their mobile phones)
Computer	User Interface	Permanent

ASM Program		
Variable	***Type of data***	***Sample Data***
message	String	“This is a cake”
incomingNumber	String	“+61XXXXXXXXX”
date	String	“07/05/16”
incomingString	String	“This is a cake”
stringComplete	Boolean	“true”
plaintext	Char	“This is a cake”
ciphertext	Char	“Wklv#lv#d#fdnh”
destinationNumber	String	“+61XXXXXXXXX”

count	Integer	“0”
buffer	Unsigned Character	“64”

Security Plan

Importance of security:
Security is a major factor that needs to be taken into account when creating or maintaining any IT service or product in current times. If your product or service is not secure, there is almost a guarantee that someone will try to exploit it. Due to these exploits, people often feel unsafe or afraid to use these services to their full potential. To combat this, a secure solution must be created whether it is by implementing encryption algorithms or by setting user specific security options. A great example would be sharing information or photos on facebook. If facebook did not have security options that allowed a user to modify what information would be available to the public on their profile they may not share as much information and would most likely use the service less and less. This would be disastrous for their company as they are offering a service that allows people to connect with one another and share their experiences, thoughts and feelings.

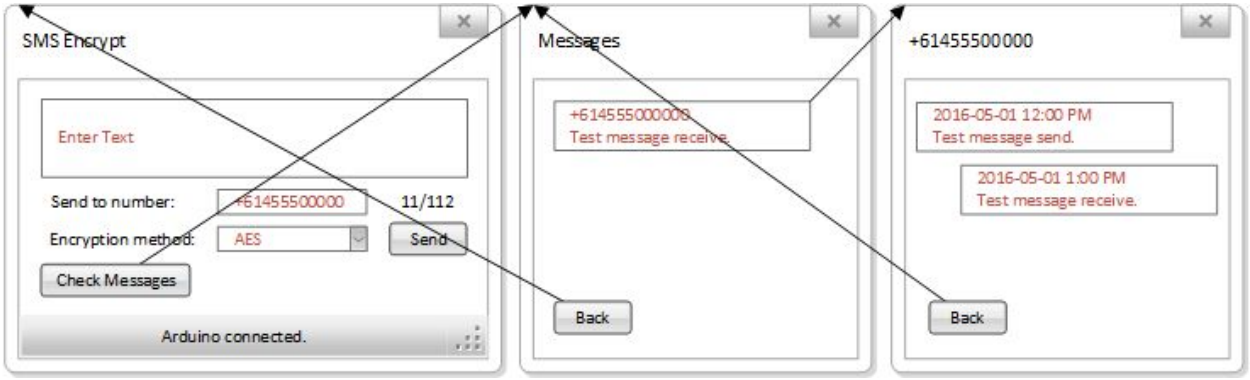
“Security is, I would say, our top priority because for all the exciting things you will be able to do with computers - organizing your lives, staying in touch with people, being creative - if we don't solve these security problems, then people will hold back.” (Bill Gates, 2005)

ASM security:
The goal of our project is to create a means to send data safely over the gsm network. Currently the gsm network does have some forms of encryption in place however, if a message is intercepted it can easily be read. To ensure that the communication between the arduino board and mobile device is kept secure we have implemented an encryption method to prevent attackers from deciphering any messages that may have been intercepted. By adding this layer of encryption, even if a message is intercepted the message will be in an encrypted format so the attacker would have to attempt to decrypt before having any chance of viewing the plaintext.

As of this moment we are using the Caesar cipher encryption method. The method involves shifting characters around by a certain value in order to hide the message. For example, if the letter is ‘A’ and we shift it by the value 3, we are left with the ciphertext ‘D’ . The Caesar cipher is not a method that we would continue using if we were to further develop this project, as it is one of the weaker types of encryption. Originally, we had successfully implemented a stronger encryption method known as AES (Advanced Encryption Standard) however due to limitations with unicode characters in sms we unfortunately could not continue using this method. As we are nearing the end of this trimester, if time allows we would like to revisit the encryption and

create our own method.

User Interface/Style Guide/Input Processes



Since the purpose of the prototype arduino is relatively simple, the GUI won't be too complicated. It'll need a field to input text and a target mobile number, a letter count that dynamically updates with the chosen encryption method, and a drop down to select said encryption method (should more than one be implemented. There should also be a method for checking sent and received messages. A status bar will show the status of the arduino.

Data Flow/Use Case/Object Class

Data Flow of the communication process

The message is sent to the Arduino, which is then encrypted and passed on through the GSM network to the mobile phone. For decryption the process is reversed.

Use Case for Sending & Receiving SMS

	Send Encrypted SMS Use Case
Name	Send Encrypted SMS
Description	Send an encrypted SMS to a specified phone number
Actor	Sender
Pre-Condition	The sender has the Arduino on and the GUI running
Flow	1. Sender enters SMS in the Message Box 2. Sender enters phone number to send the SMS 3. Sender selects encryption method (if applicable) 4. Sender clicks the “Send” button 5. System encrypts message 6. System sends message via Arduino
Post-Condition	The SMS is encrypted and sent to specified phone number
Exceptions	6a. System unable to send SMS 6b. System stops the sending process and notifies the Sender of error “Unable to send SMS from the Arduino”

	Receive Decrypted SMS Use Case
Name	Receive Decrypted SMS
Description	Receive a decrypted SMS from another phone number
Actor	Receiver
Pre-Condition	The receiver has the Arduino on and the GUI running
Flow	1. System receives SMS via Arduino 2. System decrypts SMS 3. Receiver clicks “Check Messages” button 4. Receiver clicks the phone number with a new message 5. Receiver reads the SMS 6. Receiver selects the “Back” button twice to return to the main screen
Post-Condition	Receive a decrypted SMS from a phone number
Alternate Flow	4a. No SMS received 4b. Receiver selects the “Back” button to return to the main screen

Plans

Implementation plan

There are four main areas of implementation which represent our respective roles; software (Nat), hardware (Matt), encryption (Oliver) and user interface (Stephen).

The following is a timeline which describes our contributions and the dates the deliverables were delivered. All software for the Arduino board are developed in Arduino IDE 1.6.7, UI is designed in Visio and developed in Visual Studio. Using the Scrum methodology we implement a component each sprint (which is a week long). Each component is implemented independently to the others so that they can be tested and refined before integrating them into the system.

In implementing the LCD we used the LiquidCrystal-I2C (fdebrabander 2016) library available via GitHub. AES encryption was implemented using the AESLib (Landman 2016) library also available via GitHub.

Person(s)	**Deliverable**	**Date to be provided**
Matt/Nat/Oliver/Stephen	Research	Start of Week 5
Matt	Buy Hardware	End of Week 2
Matt	Assemble Hardware	End of Week 3
Matt/Nat/Oliver	Send SMS from Arduino	End of Week 4
Matt/Nat/Oliver	LCD working on Arduino	End of Week 5
Nat/Oliver	Implement AES on Arduino	End of Week 7
Nat/Oliver	Send encrypted SMS	End of Week 8
Nat/Oliver	Implement Caesar cipher on Arduino	End of Week 8
Matt/Nat/Oliver	Receive SMS	End of Week 9
Stephen	Design UI	End of Week 10
Nat/Stephen	Implement UI	End of Week 11
Matt/Nat/Oliver/Stephen	Combine components to create	End of Week 11

	full system	
Stephen	User Documentation	End of Week 11

As you can see the device is yet to be fully implemented. Many deliverables are dependent upon each other such as being able to send an SMS to the phone before we can implement sending encrypted messages. Although we are on track, complications occurred in sending AES encrypted messages to the phone causing us to change the encryption system.

Testing plan

For testing we have four levels: unit, integration, system, and acceptance (Schwalbe pg 334-335). As part of our implementation plan we sought to develop each main component independently and integrate these components together later to create the completed system. Because of this it allows us to test the device at these four levels effectively.

Unit Testing

Each development area was tested with the following checklists. Furthermore, software components were developed in separate source files.

Hardware

Item	**Notes**	**Y/N**
Does the Arduino board power on?	The board needs to be connected via a usb cable to the computer or 12VDC adaptor.	
Does the GSM Shield power on?	It needs to be connected to the Arduino board using the correct pins.	
Do AT commands work with the GSM Shield?	Can be checked once the shield is connected to the board using the correct pins. If not configured correctly nothing will be sent or received. Commands can be hardcoded into the software.	
Does the LCD screen power on?	It needs to be connected to the Arduino board. Light the screen up with a jumper wire connecting itself at the LED pin.	
GSM Shield Arduino pins	TX1 - 18 RX1 - 19	

LCD Arduino pins	SDA - 20 SCL - 21	

Software

Item	Notes	Y/N
Using the correct version of the IDE?	Arduino IDE 1.6.7	
Correct pins are specified for each component?	The code for communicating with the GSM Shield and LCD must declare the correct pins used by the Arduino.	
Correct settings for the GSM module/SIM900 with AT Commands?	AT+CSCS=GSM - set character set to GSM AT+CMGF=1 - set SIM to text mode AT+CSMP=17,167,0,0 - set text mode parameters (default) AT+CNMI=2,2,0,0,0 - send incoming SMS to serial port These settings can be hardcoded into the software.	
Can the Arduino board upload the program?	Known Issues: Timeout error (StackOverflow 2016) Compilation error Specify correct board (Arduino Mega 2560) and port (platform dependent) in IDE	
Can Arduino send a SMS?	Using AT Commands	
Can Arduino receive a SMS?	Using AT Commands	
Can text be displayed on LCD?	Display text using the LiquidCrystal I2C library.	
Can the mobile phone receive SMS?	Does it receive the full and intended message?	

Cryptography

We have implemented a Caesar cipher of shift size 3. So testing the encryption and decryption is relatively straightforward because the encryption/decryption process for a Caesar cipher has an expected output. The following table shows the corresponding encryption of the characters.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W

U	V	W	X	Y	Z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
X	Y	Z	[	\	]	d	e	f	g	h	i	j	k	l	m	n	o	p	q

o	p	q	r	s	t	u	v	w	x	y	z	1	2	3	4	5	6	7	8
r	s	t	u	v	w	x	y	z	{	|	}	4	5	6	7	8	9	:	;

9	0	`	,	.	/	;	‘	[	]	-	=	\	~	!	@	#	$	%	^
<	3	c	/	1	2	>	*	^	¿	0	@	_	!	$	C	&	‘	(	a

&	*	(	)	_	+	<	>	?	:	“	{	}	_	+	|		\r
)	-	+	,	b	.	?	A	B	=	%	~		b	.	à	#	P

Encryption Tests

This table shows the expected outputs when encrypting messages.

Input	Output
this is a text	wklv#lv#d#wh(w
Hello, how are you?	Khoor/#krz#duh#|rxB
ID #: 1234567	LG#&=#456789
!@#$%^&*()-=+	$C&’(a)-+,0@.

Decryption Tests

This table shows the expected outputs when decrypting messages.

Input	Output

wklv#lv#d#wh(w	this is a text
Lw*v#gdqjhurxv#wr#jr#dorqh$#Wdnh#wklv1	It’s dangerous to go alone! Take this.
DUGXLQR	ARDUINO
Wkh#wlph#lv#49=73	The time is 16:40

Invalid Inputs

Since the message is stored as a string using \, ‘ or “ without using their equivalent escape sequences will result in incorrectly encrypting/decrypting these characters.

Security Tests

We can test our current protocol against common cryptographic attacks. Ideally we want to be unable to break our protocol with the following attacks. In our current system, a Caesar cipher is a weak encrypting system.

Attack	**Successful (Y/N)**
Brute Force - Try every possible key combination	
Letter Frequency Analysis - work out substitutions based on letter frequency	
Slide Attack - Does the number of rounds of encryption make it harder to decrypt?	
Known Plaintext - Can we work out the algorithm with the plaintext and ciphertext?	

User Interface

Item	**Notes**	**Y/N**
Can the user write a message?		
Is the user’s input suitable for encryption (length, characters etc.)?		
Can the user receive a message?		
Can the UI communicate with the Arduino?	Is the UI able to pass the data to the Arduino? Are error messages implemented?	

Integration Tests

This checklist verifies if two or more components work together.

Item	Y/N
Can we send a SMS and display the outgoing SMS on the LCD?	
Can we receive a SMS and display the incoming SMS on the LCD?	
Can we send an encrypted SMS?	
Can we decrypt a received SMS?	
Can we send an encrypted SMS and display on LCD?	
Can we decrypt a received SMS and display on LCD?	
Can we send a SMS via the UI?	
Can we receive a SMS via the UI?	
Can we send an encrypted SMS via the UI?	
Can we decrypt a received SMS via the UI?	

System Tests

This checklist verifies that the components work together for the given functions.

Item	Notes	Y/N
Can we send an encrypted SMS via the UI and display the SMS on the LCD?	Sending a SMS	
Can we receive an encrypted SMS via the UI and display the decrypted SMS on the LCD?	Receive a SMS	

Acceptance Test

This checklist verifies that the device meets the needs of the client/end-user.

Item	Notes	Y/N
Can Jemal run the device?	Is the interface intuitive? Can an end-user use the device effectively? We want the system to be easy to use.	
Does it meet Jemal's needs?	Are all the functions available?	

Deployment plans

Release

Because of potential applications for this device in the medical, corporate and military industries, the source code for the device will be closed source. The source code in its current state is not suitable to be released to the public and could compromise the communication system it uses. The messaging application which provides an interface to the Arduino will be provided in CD or USB form along with the Arduino hardware.

Install

The Arduino software with its dependencies will be preloaded on to the Arduino boards and shipped to the users including a USB cable. Therefore the device itself will ideally be installed by connecting it to the computer via a USB cable. For the messaging application it will be installed via CD or USB with an installation wizard.

Activation

The messaging application is activated when selected by the user. In order for proper execution the Arduino needs to be connected also, otherwise the user will be unable to use the device (send or receive messages).

Updates

Updates will be provided via an in-built updater. The built-in updater prompts the user when a new update is in and will automatically update the messenger application and Arduino software. This allows us to provide the user with the most to date encryption algorithm maintaining the highest security possible.

Uninstall

The messaging application can be uninstalled via the Control Panel (Windows) whilst the Arduino software can be uninstalled by uploading a new program via the Arduino IDE.

Retire

We will retire the device when Arduino no longer supports Mega2560 or the security of the device is compromised.

Maintenance plan

In a real life scenario our software will be maintained through patches and updates. These updates will most likely occur occur when there are improvements to the encryption protocol, bug fixes and fix issues that may arise when the Arduino software is updated. Support for the user will be provided through end-user documentation, detailing the setup of the board and the usage of the user interface. Furthermore, a bug tracking could be implemented in a real life situation to help identify issues in the device. Given the sensitive nature of code and the fact that it won't be open sourced there will be no developer documentation to ensure the security of the device. There could be potential in the future to provide APIs into the system to allow developers to modify the system.

Functioning Prototype

Please View our prototype video available on youtube.

https://youtu.be/gLgeRV-jqhs

A statement from Jemal. Please check email.

Conclusion

This document captures our plans and processes in developing our project. We are developing an Arduino device that can send and receive encrypted messages via the GSM network. The system will send a SMS via a graphical user interface which communicates with an Arduino plugged into the computer. We initially set out to implement AES encryption on the Arduino, however due to issues of sending unicode characters to mobile phones we had to change our encryption method to a Caesar cipher. In summary, we have documented our data models, security plans, our UI design as well as Use-Case/Data-flow diagrams and plans for our project.

References

ABC News 2016, "One-on-One with Bill Gates", Wash Redmond 2005, retrieved 5 May 2016, <http://abcnews.go.com/WNT/CEOProfiles/story?id=506354&page=1>

StackOverflow (2016), “Arduino: Upload Timeout error”, Stackoverflow.com, retrieved 8 May 2016, <http://stackoverflow.com/questions/21582172/arduino-upload-timeout-error>

Landman (2016), “DavyLandman/AESLib” , retrieved 8 May 2016, <https://github.com/DavyLandman/AESLib>

fdebrabander (2016), “fdebrabander/Arduino-LiquidCrystal-I2C-library”, retrieved 8 May 2016, <https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>

Schwalbe K 2014, Information Technology Project Management , 7th edn, Course Technology, Boston, USA.