USING SMARTPHONE TO PREVENT KEYLOGGING AND SHOULDER SURFING

by

RAHIL KARIM ALI

Submitted in partial fulfilment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2017

I dedicate this thesis work to

My beloved parents Karim Ali and Mehrbano Ali for their unconditional love,

My Supervisor Dr. Srinivas Sampalli for his guidance, support and motivation throughout the research work,

My Siblings Shoaib Ali and Sanam Ali who believed in me

And

To my best friend Sehrish Khawaja who stood by my side.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Intruders have developed many methods to obtain sensitive information- some of the information is private and confidential such as username and password. Although strong cryptographic algorithms and authentication schemes have been developed by other researchers, the credentials can be easily cracked through attacks such as brute-force, dictionary, shoulder surfing, and keylogging. This thesis presents a new approach to prevent two attacks, namely, keylogging and shoulder surfing. We propose a technique to login users into a secure account without entering their usernames and passwords on a physical or virtual keyboard. The usernames and passwords are stored in the smartphone and can be transferred to the system using Wi-Fi, NFC (Near-field communication) or Bluetooth technologies. Furthermore, the usernames and passwords are encrypted using AES-128 encryption algorithm. Since AES-128 encryption algorithm requires a secure key to encrypt data, we have used Diffie-Hellman key exchange algorithm to generate the secure key. Moreover, the secure key is verified using the one-way hash function SHA-256 as Diffie-Hellman is susceptible to man-in-the-middle attacks. A proof of concept prototype has been implemented and tested using Wireshark and USBlyzer to analyze the network traffic and to ensure that the credentials are transferred to the desktop application in an encrypted form.

LIST OF ABBREVIATIONS USED

WORA	Write Once Run Anywhere
SDK	Software Development Kit
NFC	Near-Field Communication
IDE	Integrated Development Environment
API	Application Programming Interface
NDEF	NFC Data Exchange Format
USB	Universal Serial Bus
MITMA	Man In The Middle Attack
RAT	Remote Administration Trojan horse
CCTV	Closed Circuit Television
PC	Personal Computer
ATM	Automated Teller Machine
DH	Diffie Hellman
Wi-Fi	Wireless Fidelity
SSL	Secure Sockets Layer

SSH	Secure Shell
IPSec	Internet Protocol security
AES	Advanced Encryption Standard

ACKNOWLEDGEMENTS

I would first like to thank my supervisor Dr. Srinivas Sampalli for his guidance and support. This work would not have been possible without his support.

I wish to express my sincere thanks to my family, my parent Karim Ali and Mehrbano Ali for their unconditional love and support in all aspect of my life, my siblings Shoaib Ali and Sanam Ali for their support, trust and love.

I would also like to thank my best friend Sehrish Khawaja who was always there for helping me in my difficult times, for all the emotional support, for believing in me and for motivating me in all my ups and downs.

Last but not the least I would like to thank all teachers of my entire academic career for sharing their knowledge and wisdom throughout my entire life and for making me capable to achieve this milestone.

CHAPTER 1 INTRODUCTION

We will begin our journey of this thesis by providing a brief introduction about various terms and concepts that are used in different chapters. We will then discuss the research problem followed by an overview of the proposed solution and the work done for preventing keylogging and shoulder surfing.

1.1 BRIEF INTRODUCTION OF THE TERMS AND CONCEPTS

1.1.1 PASSWORD ATTACKS

Passwords are the key component to authenticate the user to any system. Depending on the authentication scheme the type of password can be in text, graphical, or biometric form. Passwords play a vital role to authenticate users to various type of applications such as ATM machines, online accounts, computer logins etc. As the password protects the system from non-legitimate users, cybercriminals develop techniques to perform malicious activities such as stealing data, identity, and passwords. Intruders make the system vulnerable by accessing the important information or by using different software programs and can steal sensitive information from the system. In this section, we will briefly discuss the various types of attacks through which passwords and other information can be stolen.

a) **Brute force attack:**

For cracking short passwords, Brute force attack is very a fast technique. It usually works for short passwords and is not very useful for longer passwords. It is an attack in which attacker uses all possible combinations to retrieve password or PIN. Therefore, it is recommended to use strong and longer passwords with the combination of alpha, numeric and special characters.

b) Dictionary attack:

Users commonly uses their names, date of births or simple words taken from the dictionary which can be easily judged by the attacker [8]. Attackers make a list of the most commonly used words and apply all these words to crack the password, which is also referred to as a dictionary attack. This attack is sometimes faster than brute force attack [7].

c) Shoulder surfing:

Shoulder surfing is done by observation technique. The attacker can spy over victim's shoulder to get the password, PINs, and other sensitive information while the victim is typing it in his/her personal or public computer.

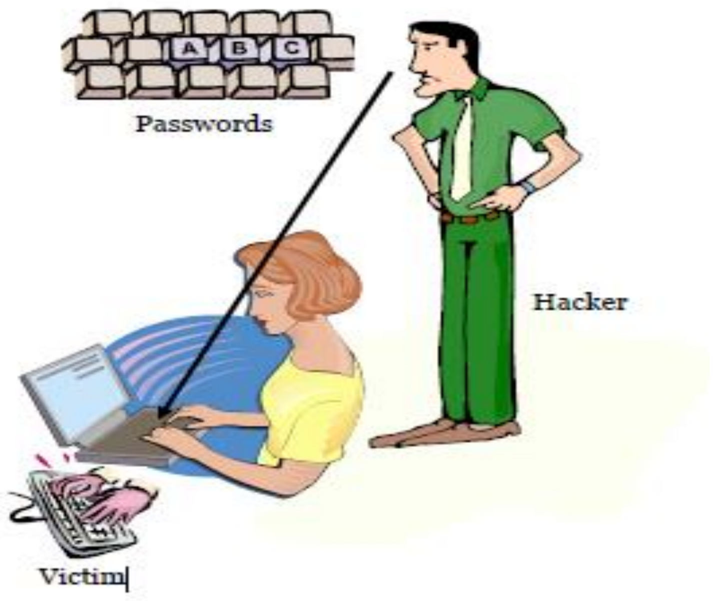


Figure 1: Shoulder surfing [37]

Usually, this attack takes place in public places where there are many people around the user such as at a library's computer, an ATM machine, a public internet cyber cafe, or a shopping mall. There are various techniques for the shoulder surfing which includes observing the hidden CCTV Camera or observing the number of keys pressed by the user. The attacker then uses all the possibilities to crack that password [8][9]. Our thesis work focuses on the prevention of this password attack technique.

d) **Keyloggers:**

Keylogging is a technique of recording all the keys pressed by the user on a physical or virtual keyboard. There are many keylogging software and hardware tools available to perform this attack. Keylogging software can be installed on any computer which makes a log file of the keys pressed and sends it to the attacker's computer or email address. The attacker will then get the information of all the keys pressed by the user [8][9]. This attack is usually done on public computers, as it is easy for attacker to access the public computers to install the malware before another user access that computer. Our thesis work also focuses on the prevention of this password attack.

e) **Phishing attack:**

This attack is basically the web-based attack, in which attacker redirects the user to the fake website which is very similar to the original website. Suppose, the user wants to access www.gmail.com. The attacker redirects the user to the fake webpage which can be www.gmale.com. If the user doesn't notice the change in the website address, he/she will enter the username and password. The attacker will immediately steal the username and password and have access to the user's sensitive information [7].

1.1.2 DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM

The Diffie-Hellman key exchange algorithm is used to share a secret key between two parties. After exchanging the secret key, the two parties can communicate with each other on any channel. While exchanging the key there is no authentication mechanism available so the algorithm is easily attacked by the man-in-the-middle attack.

In our thesis, we have implemented the Diffie-Hellman for exchanging the key between the mobile and the desktop application [15] [24]. We have considered the limitation of the Diffie Hellman and improved the key exchanging mechanism to prevent man-in-the-middle attack.

1.1.3 HASHING

Hashing refers to the conversion of data into a fixed size smaller value that represents the original data, but cannot be used to retrieve the original data. Hashing is done by one way hash function which maps the fixed size data in the arbitrary size. The output of the hash functions is called hash values, digests, hash codes, and hashes. The advantage of using hash is, it cannot be revert back as it not an encryption process but a one-way cryptographic function [16] [17]. We have used the hashing technique in our project to improve the Diffie-Hellman key exchange algorithm. The hashing algorithm which we have used is SHA 256 which generates 256-bit (32-byte) unique signature for text.

1.2 RESEARCH PROBLEM

The internet has become the basic need for the modern world and people are using it for almost everything such as mobile banking, confidential information sharing, chatting, and social networking. A user is not just connected with the internet but connected with the whole world via internet. Nowadays, internet users are facing many kind of attacks, which aim to steal the user's ID and password. Passwords play very important role while doing different computing tasks. They provide security against unwanted access to our personal resources [3]. Passwords are used in ATM machines, for accessing sensitive information, logging into the computer system, and mobile authentication [2]. Text passwords are the most common type of passwords and are considered as insecure because there are many attacking techniques developed to steal them. It is always recommended that text password should be complicated and consist of alphanumeric characters [3]. However, they should be easily remembered by the user but difficult to guess by pretender [4]. Users who choose easy and short password can be the target of shoulder surfing, dictionary attack and brute force attack [5][6]. Other than this, there are many malware used to make the system vulnerable and to collect the sensitive data [1]. Some malicious software also steals the passwords such as keystroke loggers [3]. Our research work aims to mitigate these type of attacks for which we have considered the following research questions:

1. How to protect the passwords while using it in the public or shared computer.
2. How to prevent keylogging and shoulder surfing attacks while using passwords in public places.

This thesis provides the proposed solution and an in-depth research study done for solving the research problem.

1.3 Introduction to the proposed approach

Our aim is to solve the research problem which is to prevent from keylogging and shoulder surfing. To solve the problem we have proposed a system through which user can access the username and password secretly. Our approach uses smartphones to access the username and password and send it to the computer, rather than typing it on the physical or virtual keyboard.

The user have to install our desktop and mobile application. The mobile application will store user's credentials (User name and password) of all the websites which the user needs to access often on the public or personal computer. The user then just need to open the desired webpage and use our mobile application to send the credentials required for logging into the website. Therefore, the user do not need to type the username and password on the physical or virtual keyboard to login. This is how we can prevent shoulder surfing and keylogging because an attacker cannot see the password over the shoulder and none of the keylogging software or tools can record the keys as they are not pressed. The user's credentials can be sent to the desktop application through three different mediums which are Wi-Fi, Bluetooth, and NFC. The user can select any of the methods to send the credentials from the mobile to the desktop application. The data will be sent in an encrypted form by a private key cryptosystem or symmetric key cryptosystem. To encrypt the data, we have used AES 128 encryption algorithm. The key required for encryption will be exchanged through the Diffie-Hellman key exchange algorithm. As the Diffie-Hellman key exchange algorithm is vulnerable to man-in-the-middle attack, we have used an improved version of the Diffie-Hellman which uses SHAH 256 hashing algorithm.

The proposed approach also works as a password manager as all the passwords are saved and can be update using the mobile application. All the passwords are stored locally in the mobile device and cannot be accessed or managed through any other system or webpage.

1.4 OUTLINE OF THESIS

The rest of the thesis is organized as follows: Chapter 1 introduces the various terms and concepts that are used in the report later. It also provides a brief description of password attacks and an introduction of the proposed approach. Chapter 2 provides the background details of keylogging, SHA-256, symmetric key cryptography, AES-128 and the Diffie-Hellman key exchange algorithm. In chapter 3, we have provided a literature review which describes the work done by other researchers on the Keylogging and shoulder surfing attacks and different approaches to mitigate these attacks. Moreover, it also discusses the security issues in Diffie-Hellman key exchange algorithm and different methods to overcome these issues. Chapter 4 presents the proposed approach to prevent keylogging and shoulder surfing attacks. Chapter 5 discusses the implementation and technical details of the proposed approach. Chapter 6 provides an in-depth description of the experiment conducted to test the proposed approach. Finally, the report ends with the conclusion (Chapter 7) of our thesis work and discusses the limitations and possible enhancements that can be done in future. Appendix A contains the screenshots of the desktop application and Appendix B contains the screenshots of the mobile application.

CHAPTER 2 BACKGROUND

2.1 KEYLOGGING

Keyloggers are also called keystroke loggers or system monitors. It can be a hardware device or a software, which monitors the key strokes pressed in any specific computer in which the keylogger is installed. It can be used for negative and positive both purposes. Parents use the keylogger to monitor their children's activities on computer, private investigators use it for evidence, companies use it for employee monitoring, analysts and developers use it for studying human interaction with the system. However, there are many unlawful uses of the keyloggers. Cybercriminals use keyloggers to steal the confidential information, username and passwords, identities, and banking information. Most of the time attackers do not require physical access to the victim's computer. They trick the user to download a spyware and execute that as a RAT Trojan horse. The software will have two files which get installed in the same directory. The first file is the dynamic link library (DDL) file, which is responsible for recording the keys pressed and the second file is an executable file (.EXE) which is responsible for installing the DDL file and triggers it to work [10]. The keylogger software will record the keystrokes pressed by the user and send the information to the attacker via the internet. Different keylogging techniques are now explored by attackers in which the two main techniques are software keyloggers and hardware keyloggers.

2.1.1 SOFTWARE KEYLOGGERS

Monitoring keys through software keyloggers is based on operating systems [11][12] because the information of keystrokes is being passed between the computer keyboard interface and the operating system OS. Keylogger software uses the hooking mechanism through which it can capture the data from the keyboard. The data is then sent to the attacker by copying it on a hard drive or by sending it via the internet.

2.1.2 HARDWARE KEYLOGGERS

A hardware keylogger is a circuit which is located between the computer and keyboard. Figure 2 shows the two types of PS/2 keyloggers.

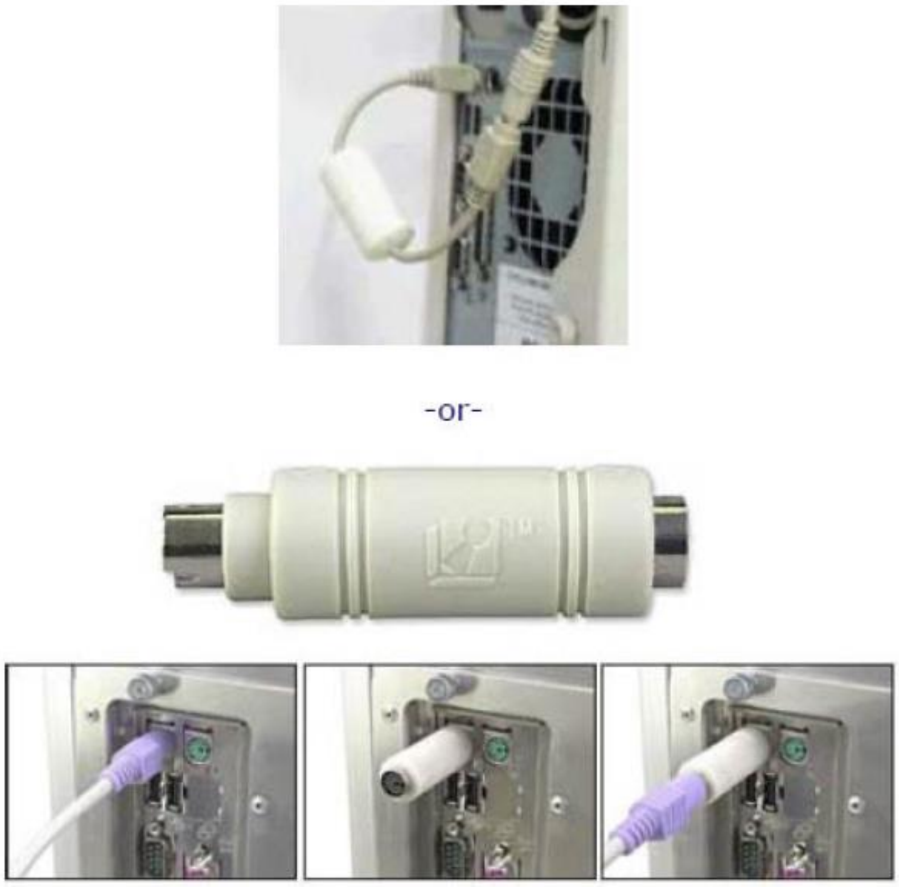


Figure 2: PS/2 keyloggers [36].

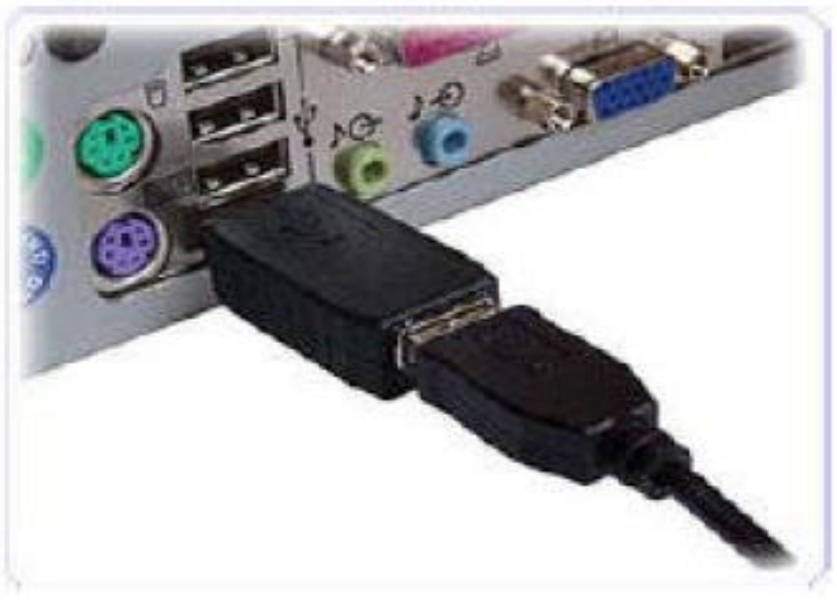


Figure 3: USB keyloggers [36].

Figure 2 and figure 3 shows that the keyloggers are directly connected to the computer. The other way to install the hardware keylogger is to install it in a standard keyboard.

2.1.3 WORKING OF KEYLOGGERS

The Keyloggers are active during the time when the key is pressed, and the pressed key displayed on the monitor. Figure 4 shows the working of keylogger spyware attacks. The three users are using the different internet services. The attacker is present between them, making sure that somehow the user installs the spyware in the computer. Software seems to be legitimate but actually it is to fool users. When the user downloads and installs it, all the key strokes of the keyboard are recorded and saved in the log file. Then the log file is sent to the attacker periodically via email. The red arrows in figure 4 shows the entry of the keylogger.

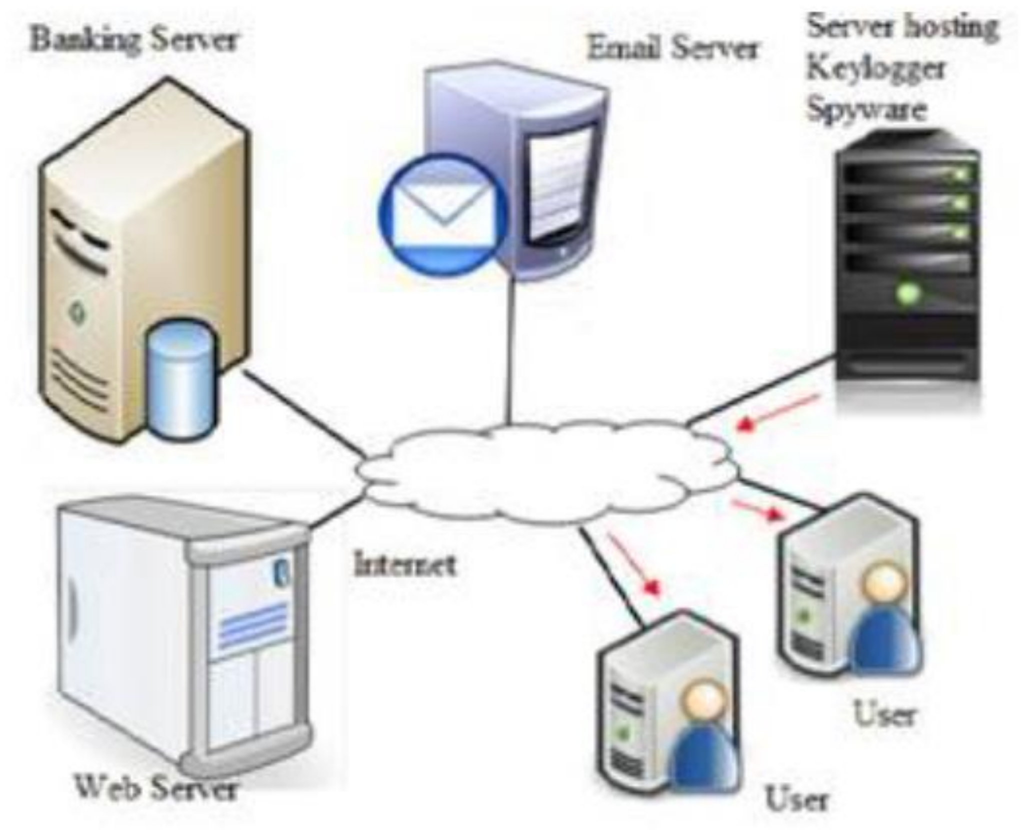


Figure 4: Working of keylogger [9]

Figure 5 shows that as soon as the log file is created, it is sent to the intruder. The blue lines shows the transfer of email containing log file and confidential information from user’s computer. If this act is done continuously then it can cause a huge loss to the user.

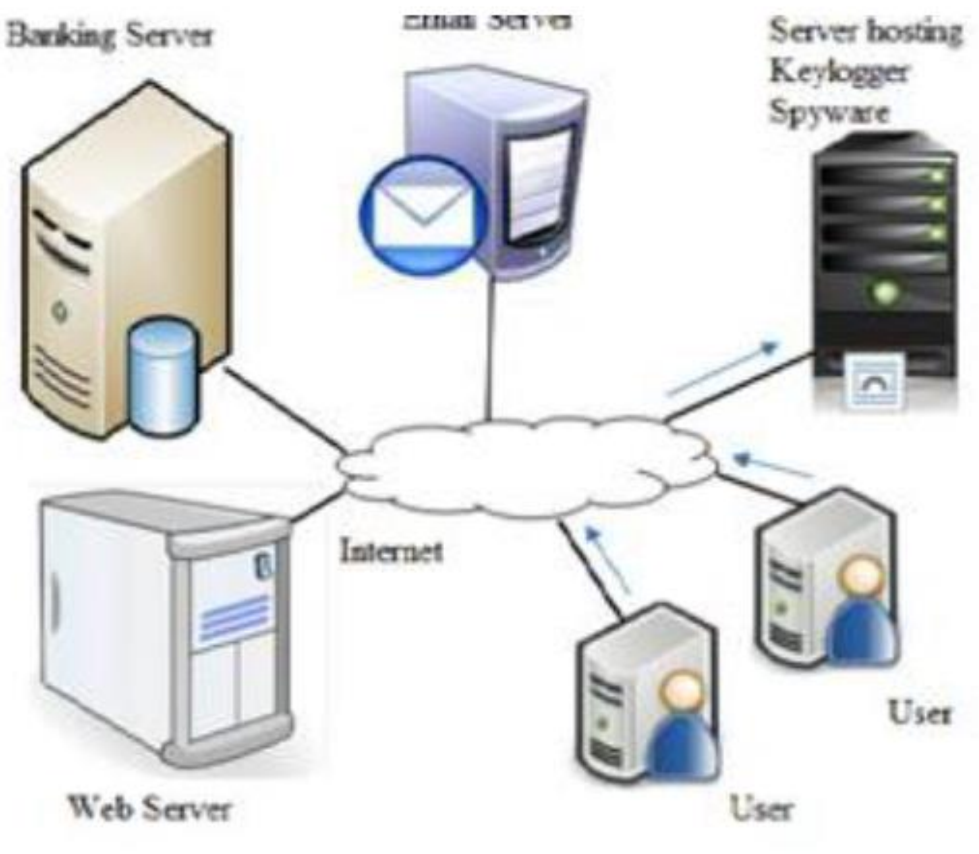


Figure 5: Sending log file to the intruder [9].

2.2 SHA-256

There are a number of algorithms developed for hashing one of which is SHA-256, which we have used to verify the secret key generated by the Diffie-Hellman key exchange algorithm. SHA stands for secure hashing Algorithm, which was developed by the National Institute of Standards and Technology (NIST). The newer versions SHA-256, SHAH-384, SHAH-512 (numbers represent the length of bits) were published in 2002 [18].

The two-different version of this algorithm are SHA-1 and SHA-2. They are different in construction and bit-length of signature. SHA-2 is the successor to the SHA-1 as it was an improved version of the algorithm. SHA-1 is 160-bit hash and SHA-2 comes in a variety of lengths in which the most popular hashing algorithm is SHA-256. Hashing algorithm creates the unique hashes for every possible input, which is irreversible. There are two possible values of bit which are 0 and 1. So the number of possible combinations generated can be expressed as the number of possible values raised to the number of bits. So, the SHA-256 will have 2^{256} possible combinations. The result of 2^{256} is a huge number and it has a less chance that two values will generate the same hash [19].

2.3 SYMMETRIC KEY CRYPTOGRAPHY

Cryptography is a modern encryption technique, which was first designed to secure the communication of military. In this era, the internet has grown rapidly and cryptographic techniques are required to secure the communication over the internet. Cryptography can be classified into two types: Symmetric Key cryptography and Asymmetric key cryptography. The Symmetric-key cryptography is also known as single-key or private-key cryptography as it uses one private key to encrypt and decrypt the information. Some popular symmetric key algorithms are shown in figure 6.

Asymmetric-key cryptography is also known as public key cryptography, which uses two different keys to encrypt and decrypt the information; one is the public key and other is the private key. Figure 6 also shows some well-known Asymmetric key algorithms [20] [21]

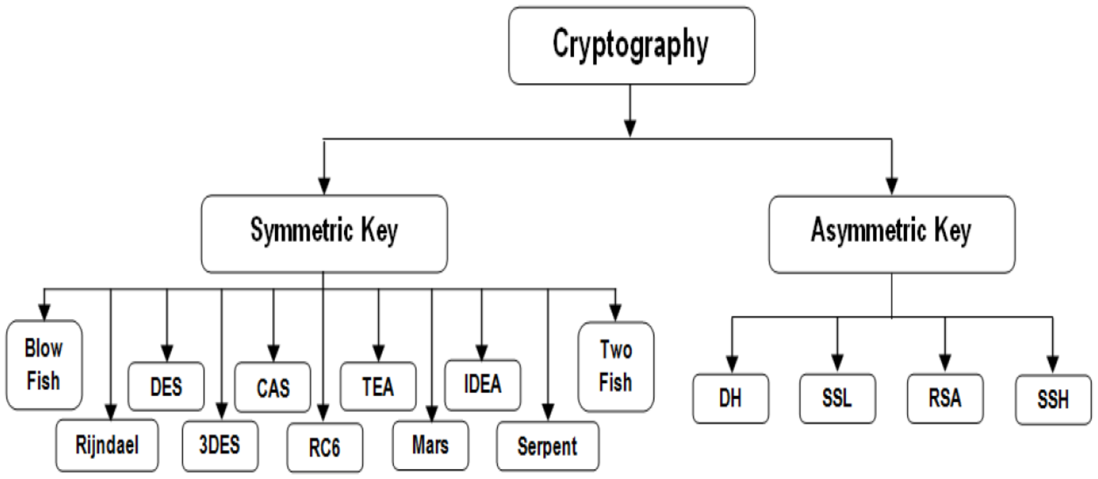


Figure 6: Cryptographic algorithms [21]

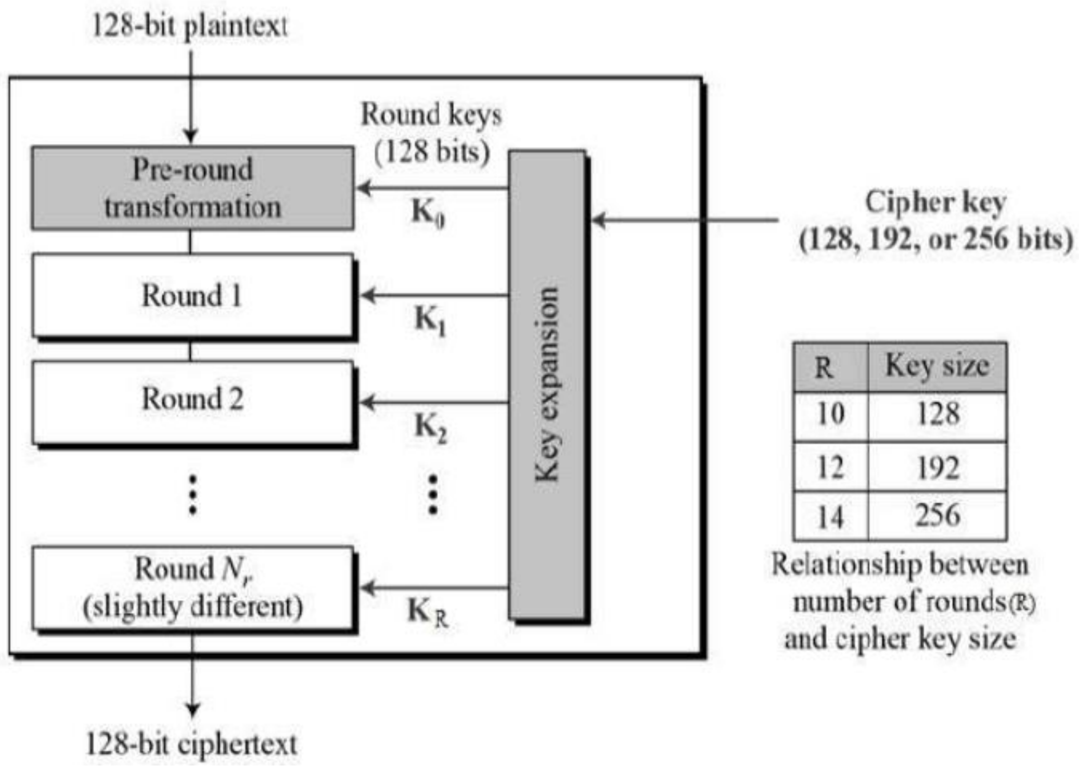
We have used Symmetric key cryptography in our thesis work. AES 128 is used to encrypt and decrypt the user credentials when they are sent from mobile to desktop application.

2.4 AES-128

AES (Advanced Encryption Standard) -128 is also known as Rijndael, which is use in the encryption of an electronic data. It was established by the U.S NIST (National Institute of standards and technology) in 2001. It is widely adopted and most popular symmetric algorithm and six times faster than triple DES. Triple DES was designed to overcome the vulnerability against exhaustive key search attack but it was found slow.

2.4.1 WORKING OF AES

AES is based on ‘substitution-permutation network’. It is based on a series of linked operations, some operations replace inputs by specific outputs which is called substitution. However, other operation shuffle the bits around, which is called permutations. The number of rounds in AES depends on the length of key such as 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Every round uses a different 128-bit round key and it is derived from the original AES key [22]. Figure 7 shows the schematic of AES structure.



R	Key size
10	128
12	192
14	256

Figure 7: Schematic of AES structure [22]

2.4.2 AES ENCRYPTION PROCESS

Each round in the encryption process consist of four sub-processes. The process of first round is shown in Figure 8.

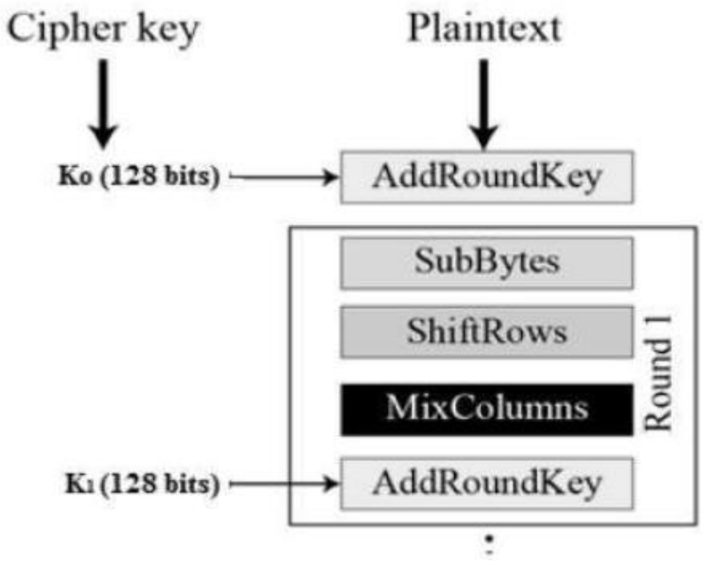


Figure 8: AES encryption process [22].

a) **Byte substitution:** In this step, each byte is replaced with another byte according to the lookup table. Figure 9 shows the byte substitution.[23]

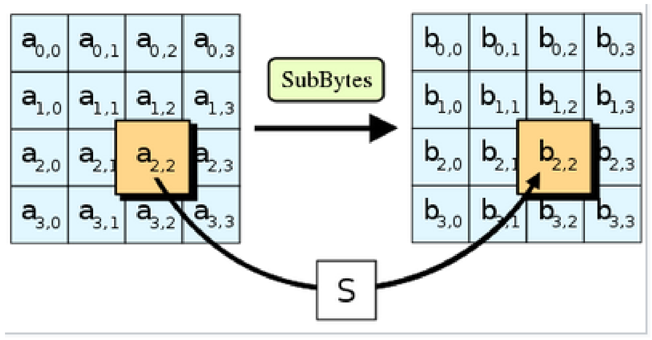


Figure 9: Byte substitution [23].

b) Shift rows:

All the four rows of the matrix is shifted to the left side. The entries which fall off are re-inserted from the right side of the row. It works as follows: The first row is not shifted, second row is shifted one byte to the left, third row is shifted two positions to the left, fourth row is shifted three positions to the left and at the end the result will be new matrix which consist of the same 16 bytes [22]. Figure 10 shows the shift rows.

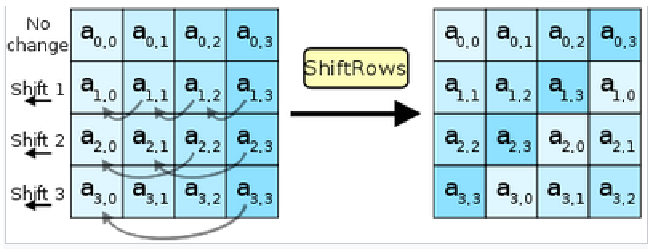

No change
Shift 1
Shift 2
Shift 3
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

Figure 10: Shift rows [23]

c) Mix columns:

It takes the four byte of one column as an input which are transformed using a special mathematical function. It will then result in four new bytes which are replaced by the original column. [22]. Figure 11 shows the mix column [23]

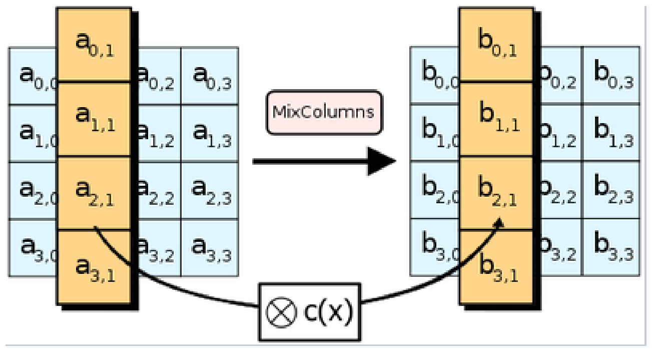

MixColumns
$\otimes c(x)$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

Figure 11: Mix column [23].

d) Add round key:

The 16 bytes of the matrix are then considered as 128 bits which is then XORed to the round key of 128 bits. The output of last round is considered as cipher text [22].

2.4.2 AES DECRYPTION PROCESS

The AES decryption process is similar to the encryption process but it is in the reverse order. Each round have four processes: [22]

1. Add round key
2. Mix columns
3. Shift rows
4. Byte substitutions

All these process are discussed earlier in section 2.4.1.

2.5 DIFFIE-HELLMAN

The Diffie-Hellman key exchange algorithm is used to exchange the cryptographic keys securely over a public network between two parties. It was the first public-key protocol which was conceptualized by RALPH Merkle and it is an earliest practical example of public key exchange in the field of cryptography. The Diffie-Hellman key exchange algorithm allows two parties to establish a shared secret key which is used for encryption. [24][25]

1. Alice and bob choose two prime numbers g and p which are not meant to keep secret.
2. Alice and Bob chooses a secret number (a) and (b) respectively, but they don't tell anyone.

3. Now, Alice compute g^a mod p and result will be called A.
4. Bob will compute g^b mod p and result will be called B.
5. Alice and Bob exchange their results A and B with each other.
6. Now, Alice will take the number send by Bob and do the exact same operation with it. So the operation will be B^a mod p.
7. Bob will also do the same operation with the result send by Alice. Which will be A^b mod p.
8. The magic here is the answer Bob will get at the end will be the same answer which Alice got.

Figure 12 shows the general illustration of the Diffie-Hellman key exchange algorithm. [25]

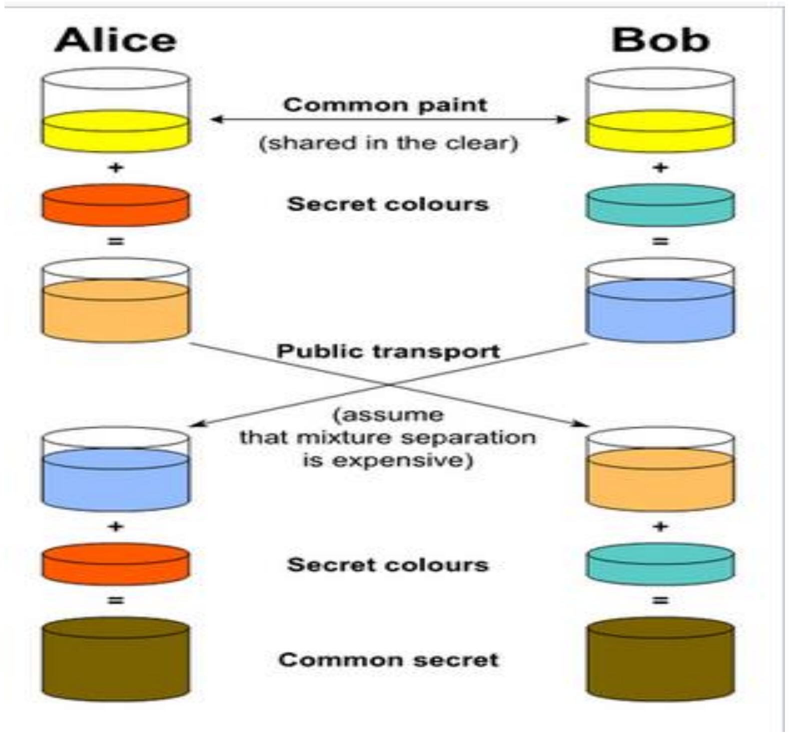


Figure 12: Diffie-Hellman key exchange algorithm [25].

At the end, Alice and Bob have the same value which will be use as a secret key. Alice and Bob get the same value because of the property of modulo exponents, which is:[24][25]

$(g^a \bmod p)^b \bmod p = g^{ab} \bmod p$

$(g^b \bmod p)^a \bmod p = g^{ba} \bmod p$

Alice and Bob will get same answer no matter in which order they do the exponentiation. Alice will do in one order and Bob will do in other order. They both will never know the secret number they used to get the results but they arrive at the same result at the end. [24][25]

In this thesis project we have used the Diffie-Hellman key exchange algorithm to share the secret key between the mobile and desktop application. The Drawback of Diffie-Hellman is the man-in-the-middle attack. The section 2.5.1 briefly describes Man-in-the-middle attack.

2.5.1 MAN-IN-THE-MIDDLE ATTACK:

The vulnerability in the Diffie-Hellman key exchange is man-in-the-middle attack. The following points explain how man-in-the-middle attack takes place in Diffie-Hellman key exchange algorithm.

1. In the attack the attacker intercepts Alice's public number and change it with new number and sends to Bob.
2. When Bob will transmit his public value, the attacker substitutes it with new number and sends it to Alice.

3. Attacker and Alice agree on the same shared secret key
4. Attacker and Bob also agree on the same shared secret key.
5. Attacker can decrypt any messages sent out by Bob or Alice.

The vulnerability is because of lack of authentication while exchange the numbers [26].

Figure 12 shows the illustration of man-in-the-middle attack

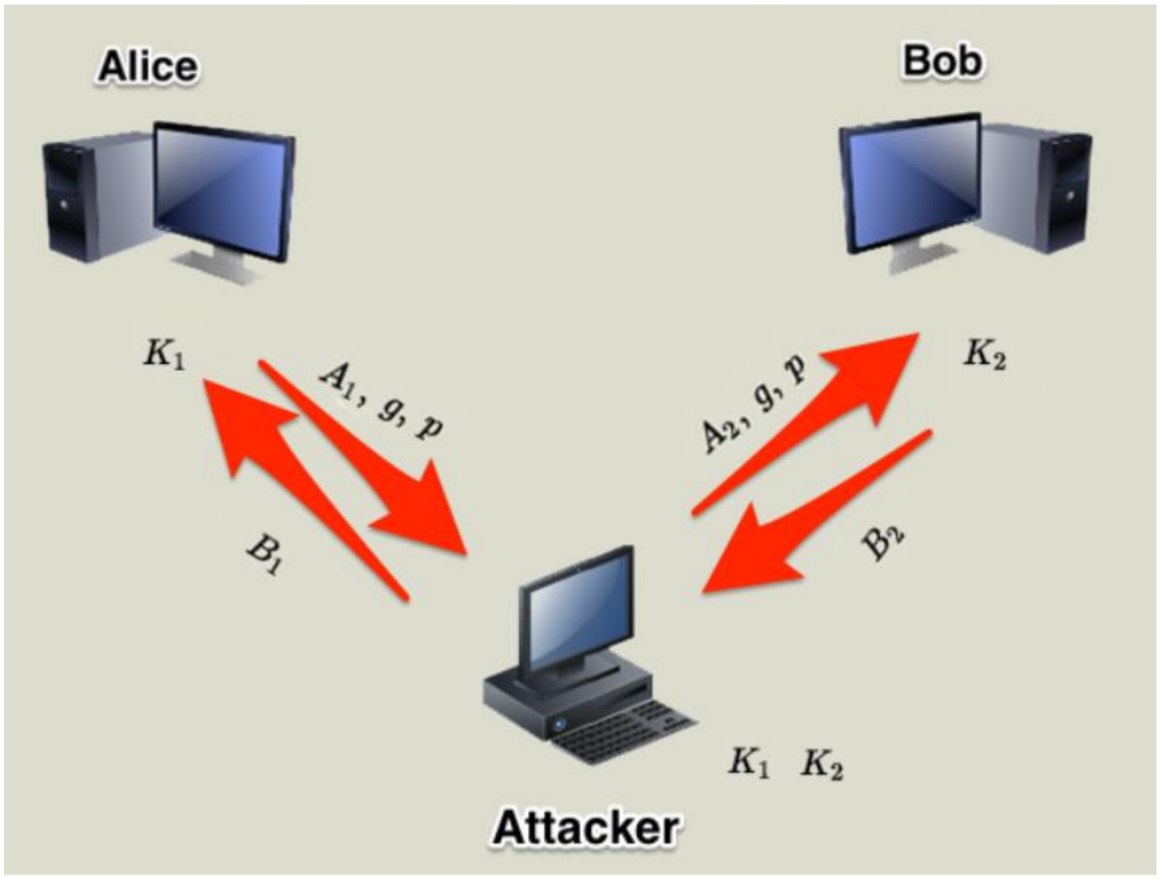


Figure 13: MITM attack in DH [26].

CHAPTER 3 RELATED WORK

We are interested to answer our research questions stated in section 1.2. Our aim is to prevent the two main password attacks which are keylogging and shoulder surfing. To start our research work, we have studied some research papers which describes the technique and approaches developed by other researchers to overcome the password attacks.

This section includes the work done to prevent shoulder surfing and keylogging, some limitations of these proposed approaches and how we have make our approach better by overcoming these limitations. This section also discusses some work done on Diffie-Hellman key exchange algorithm.

3.1 LITERATURE SURVEY ON KEYLOGGING AND SHOULDER SURFING

There is a lot of work done by other researchers to prevent keylogging and shoulder surfing. Our related work on these attacks describes the approach by other researchers.

3.1.1 KEYLOGGING

In section 2.1, we have discussed some ways through which keyloggers are stealing the sensitive information. There are several ways of mitigating the keylogging described by different researchers. Researcher Parekh et al [35] have described the concept of virtual keyboard. Which is basically a software that is used to mitigate the password attack by Trojans. Virtual keyboard is an on-screen keyboard which uses mouse to enter sensitive data like credit card details or passwords. Virtual keyboards has a limitation as Trojans are becoming advanced which takes the screenshots on the mouse click event. The screenshots are then sent to hackers at later time. Moreover, virtual keyboards are also susceptible to shoulder surfing as attackers can observe the monitor screen and mouse clicks over the shoulder. The researchers have further described their proposed technique related to the virtual keyboard, which is anti-screenshot virtual keyboard. Researcher have proposed a keyboard on which, when mouse curser move to any key, all the keys in that row of the keyboard are changed to some special symbol like a hash (#) or an asterisk(*). Figure 14 shows that when the mouse cursor moves on an Anti-Screenshot Virtual Keyboard the top row has changed the key values to asterisk(*) [35].

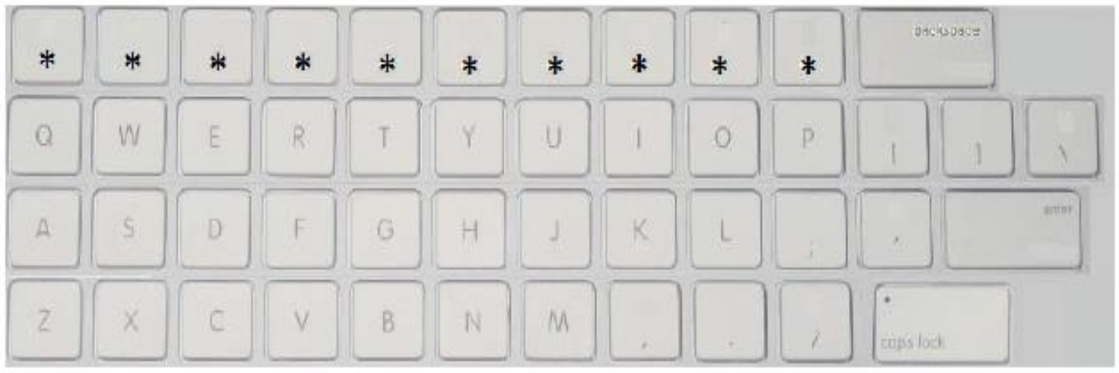

* * * * * * * * * *
backspace
Q W E R T Y U I O P [ ] \
A S D F G H J K L ; ' enter
Z X C V B N M , . / caps lock

Figure 14: Virtual Keyboard [35].

When any particular key is pressed, all the keyboard keys will change to asterisk. As a result, the Trojan capturing the screenshot on mouse click will no longer be useful for the attacker. Figure 15 shows that all the keys are changed by asterisks [35]

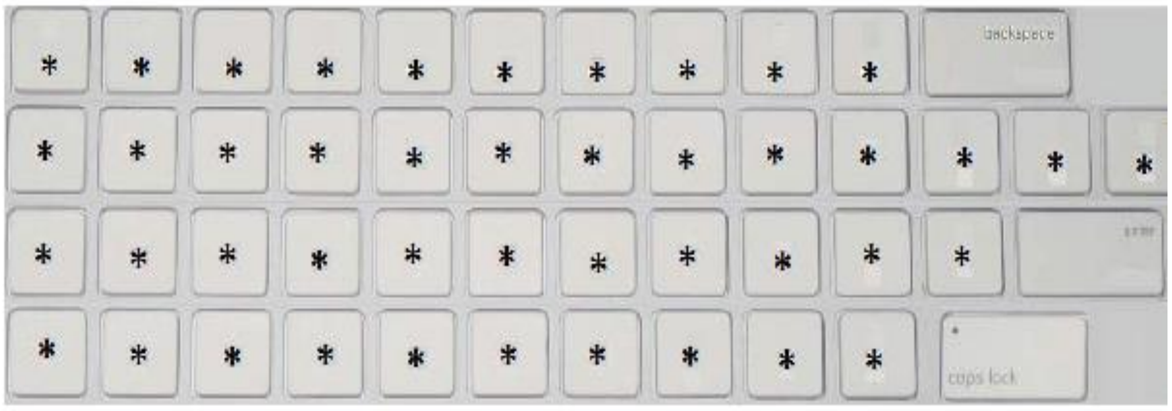


Figure 15: Virtual Keyboard changing all values [35].

The keyboard proposed by researcher is divided into different sections as shown in figure 16. Each time the key is pressed or the page is loaded, the order of the keys is changes in each section. As a result, Trojan cannot find the input key, even if Trojan have captured any picture of the virtual keyboard. Figure 16 shows the areas of which the keys are changed [35]

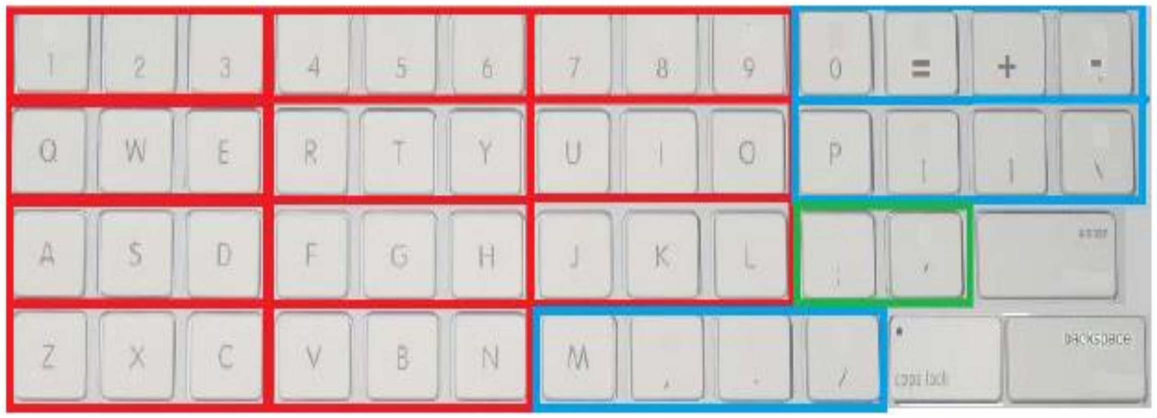


Figure 16: Virtual keyboard area division [35].

Assume that if user has 8-bit password which only consist of characters, then the Trojan will need approximately two million attempts to find the password and if user wants to enter bank information then bank account usually get block after three unsuccessful attempts. Anti-screenshot virtual keyboard provides high level of security.

Tom Olzak [36] describes some of the defensive measures that can be taken to prevent keyloggers, which includes:

1. Lock the computer when not in use
2. Implement the physical security controls
3. Block the access to malicious websites
4. Apply security patches
5. Use keylogger detection software
6. Use screen-based virtual keyboards rather than giving input through the physical keyboard.
7. Use automatic form filler software
8. There are some software solutions like GuardedID, which encrypts the keyboard input so that keyloggers cannot detect the input.

3.1.2 SHOULDER SURFING

There is a lot of work done for the prevention of shoulder surfing such as the implementation of graphical passwords, two way authentication, and virtual keyboards. Virtual keyboard provides the protection against the keyloggers but still vulnerable for shoulder surfing. There are many recent attempts made to improve the virtual keyboards by loading and changing the layout dynamically to confuse the attacker. There are many graphical password techniques introduced to authenticate the user but still they are vulnerable to the attacks such as screenshot capturing or observing CCTV camera.

Md.Haque et al [38] have also proposed an approach of graphical passwords. In their proposed approach the graphical password replaces the text passwords.

There are two phases for proposed approach. First is registration phase and second is login phase.

REGISTRATION PHASE:

1. User will create his/her profile by giving proper information including username.
2. The user will be presented with the 25 images from which user will select any number of image, user can also select one image or choose an image more than once. The selection of images will be considered as password. This graphical password will work as a first step of authentication. Figure 17 shows the set of images which are presented to user on the registration phase.

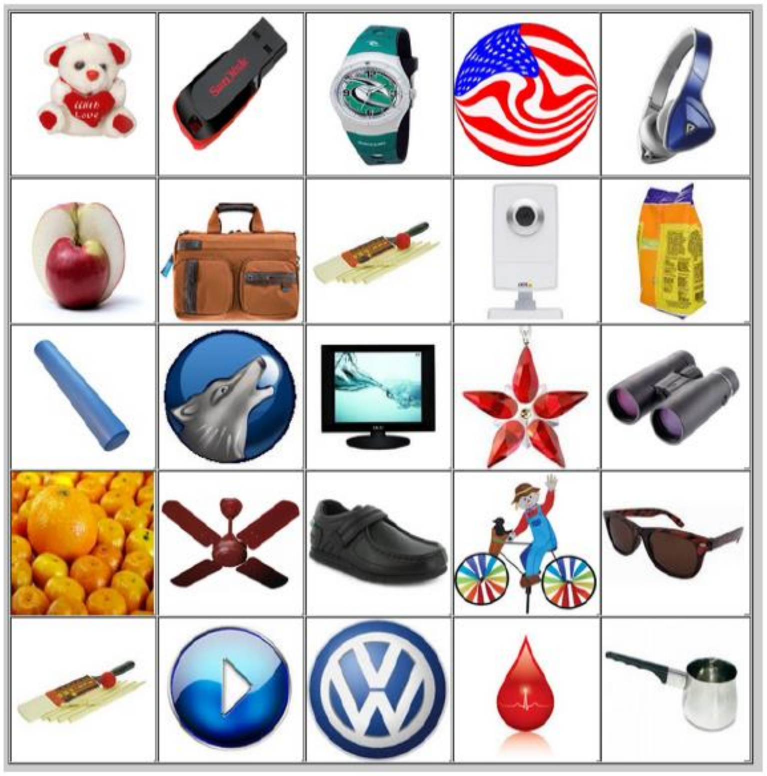


Figure 17: Set of images [38]

3. After selecting the images, user will be asked to choose any one picture from that for the further authentication process.
4. After that user will be presented with the set of different questions and for answering each question user must point a region on the image. If there are three questions, so user will point three regions on the image which will be called region-of-answer (ROA). Figure 18 shows the selection of region on an image.

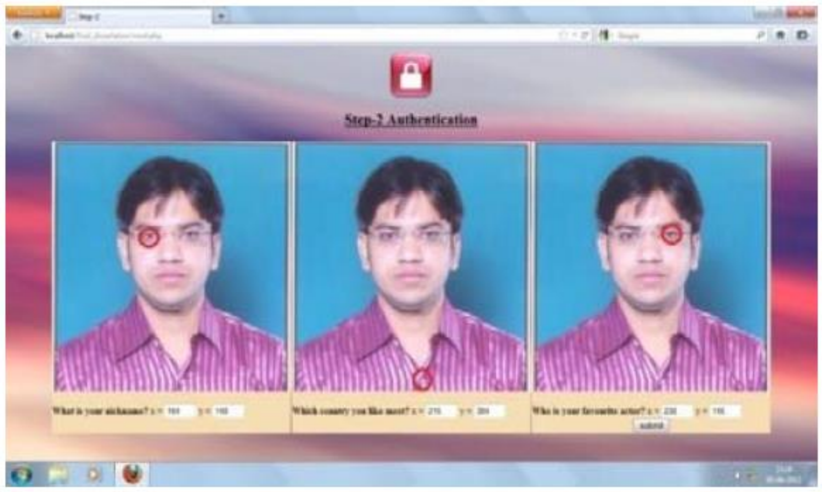

Step-2 Authentication
What is your nickname? x = y =
Which country you like most? x = y =
Who is your favourite actor? x = y =
submit

Figure 18: Selection region on an image [38].

LOGIN PHASE:

1. For logging into account, the system will ask the two-way authentication which user has setup while registering.
2. In the first step, user will be asked for his/her username and graphical password which was selected in the registration phase.
3. The order of the images provided to the user will be random each time user logs in.
4. After providing the correct images, the user will be provided with questions to authenticate him/her second time.

5. Questions will also be random and to answer the question user have to choose the ROA on the image.
6. If the authentication is successful, then user will be allowed to access the account.

There is a couple of limitations which we have considered for this approach which are as follows:

1. The user has to remember a lot of images and regions to authenticate every time when he/she tries to login. If the user has multiple accounts, then user needs to remember the images and their ROA for each account to login. Our proposed approach gives the better solution of this limitation as users don’t need to remember the any password to authenticate themselves.
2. If the attacker is observing the user via CCTV while the user is selecting the images and their ROA, then there is still a possibility of shoulder surfing attack. Our proposed approach also gives the better solution as user don’t need to enter any password to authenticate themselves.

Researcher Nand et al [63] have also considered these limitations and proposed an approach named ‘PassBoard’ to tackle these limitation. PassBoard uses dynamic layout that does not have any pattern therefore it is difficult for attacker to memorize or guess next layout of the PassBoard. It is an extension in a google chrome browser which have a randomized virtual keyboard containing keyboard keys in a square matrix format. There are two separate matrices for alphanumeric and special characters. Alphanumeric is in first block and special characters are in second block. Both have a same input method. To select the desire character user will perform two button clicks. The first click (will be for row click) can be any key in the same row of the desired character in the PassBoard. The second click (will be for column click) can be any key in the same column of the desired character in the PassBoard. This way user can input the whole password without pressing any characters of his/her password. After the user is done, he/she can press the 'Done' button through which the password is copied into the clipboard and paste it in the password field where it is required. This will also prevent the keyloggers to capture the keystrokes and it would be difficult for an attacker to shoulder surf. Figure 19 shows the working of PassBoard.

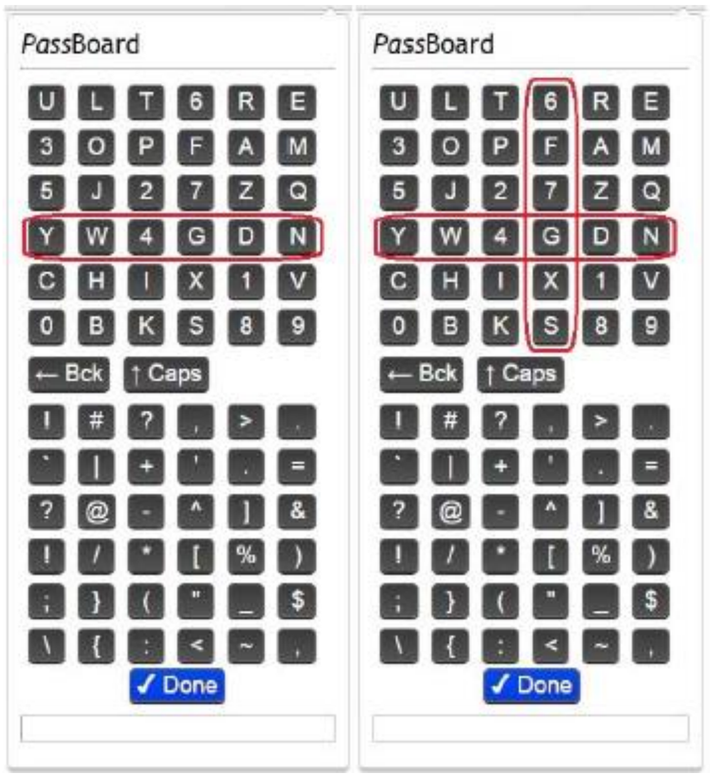


Figure 19: Working of PassBoard. [63]

3.2 LITERATURE SURVEY ON DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM

In any type of network, the communication is very important part of it. It is important that the data transfer from one node to another is always secure. Security is becoming very important for the computer users and military. When the data is transferred between web users and web servers or between mobile application and web server, it is necessary to ensure the confidentiality, authenticity and integrity. The transfer of users' credentials is also very risky when it is transferred on an insecure channel. It is important to send it in an encrypted form and communication between both the parties is always secure. Our approach uses the Diffie-Hellman key exchange algorithm which shares the secret key in a secure manner on an insecure channel. The shared secret key is important for both the parties who may not have communicated with each other ever previously. With the help of key, both the parties can encrypt their communication on an insecure channel. There are many protocols which also uses the Diffie-Hellman to share the secret key such as SSL, SSH and IPSec. [64] Michel Abdalla [65] have proposed a Diffie-Hellman based encryption scheme which is also known as DHIES and DHAES and used in many draft standards. DHIES is based on Diffie-Hellman which combines (a) symmetric encryption method, (b) a message authentication code, and (c) a hash function. DHIES is chosen to provide the security against cipher text attacks.

Furthermore, authors Chirstoph et al [30] uses some variations of Diffie-Hellman key exchange algorithm to solve the secure key distribution problem in their system. Authors further mentioned that they are using key authentication center (KAC) for authenticating all public keys which was a major problem. In this approach, the facility only wants the name of the partner that the user wants to communicate and a public key signed by KAC. In our application, we have used Diffie-Hellman to do a security handshake between two devices to develop a secure communication medium for exchange of credential. Diffie-Hellman can further be extended for security purposes with cryptography as mentioned by Victor Boyco et al [31] that they are designing a password protected key exchange protocol using cryptographically secured keys which should be kept secret between the users to avoid the attacker from launching offline dictionary attacks. Authors have also discussed an extended protocol called PAK-X in which the user has the plaintext version of the passwords and server just has the verifier of the password, so even when the server is compromised, the information is still safe.

The Diffie-Hellman itself is not secure as discussed by Nan Li [32]. Author says that the Diffie-Hellman protocol can be easily attacked by the man-in-the-middle attack and the attacker may pretend to be the authentic person for fraud. The Diffie-Hellman is a non-authenticated key exchange protocol and does not offers authentication for the communicating users and hence it is vulnerable to the man-in-the-middle attack. An attacker in the middle can generate two distinctive Diffie-Hellman key exchanges with both Alice and Bob and pretend to be Alice against Bob and vice versa, which will allow the attacker to decrypt the message, read it and then re-encrypt the message and pass it along. As we are aware of this limitation of man-in-the-middle attack, we have used the hashing technique to overcome this limitation. We have used hashing to hash our secret key and then send it to the other party so that attacker in the middle cannot decrypt that key.

Researcher Bao et al [34] further discusses various problems of the Diffie-Hellman with computational and decisional means. Authors are trying to find the relationship between the variation of the Diffie-Hellman problem, which also includes the computational and decisional cases. By finding the relation between them they could obtain reduction that are valuable and that advantage could be used against all other variation as well. Authors quote three main variation in their words as “We show that all three variations of computational Diffie-Hellman problem: square Diffie-Hellman problem, inverse Diffie-Hellman problem and divisible Diffie-Hellman problem, are equivalent with optimal reduction” [34]. The authors used polynomial reduction and transformation for relating the complexities. They didn’t find any appropriate solution for these problems but they left these interesting problems for future researchers. With all these problems in the Diffie-Hellman key exchange protocol and the research done on them I found that using just Diffie-Hellman would not be enough, so in my application, I have used hashing to further secure Diffie-Hellman where the user has created a hash function and the server match that has function with the one that user has for authentication purpose.

CHAPTER 4 METHODOLOGY

4.1 Brief Overview of Proposed Approach

In this section, we will discuss a brief overview of the proposed approach. The proposed approach consists of two components the mobile application that runs on a smartphone, and the desktop application that runs on the client machine. The mobile application stores all the credentials (username and password) related to a particular URL and transfers it to the desktop application.

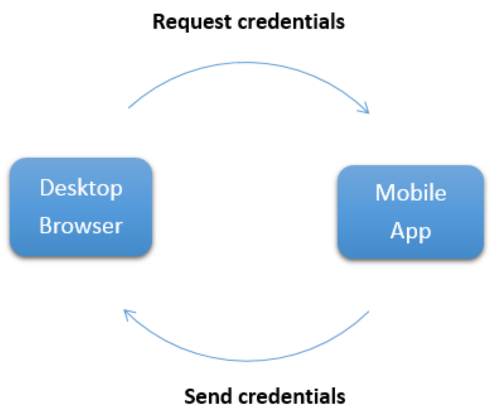


Figure 20: Overview of the proposed approach.

We have divided our proposed approach into four major phases as follows, all these phases are discussed in detail in section 4.2.

1- Creating connections between two devices
2- Diffie-Hellman key exchange
3- Verify Secret Key

4- Encrypt and Decrypt data.

4.2. Detailed Description of proposed approach

In this section, we will discuss the proposed approach phase by phase in detail. Various tasks are performed by the desktop and the mobile application. Figure 20 and 21 gives an overview of the tasks performed by each application.

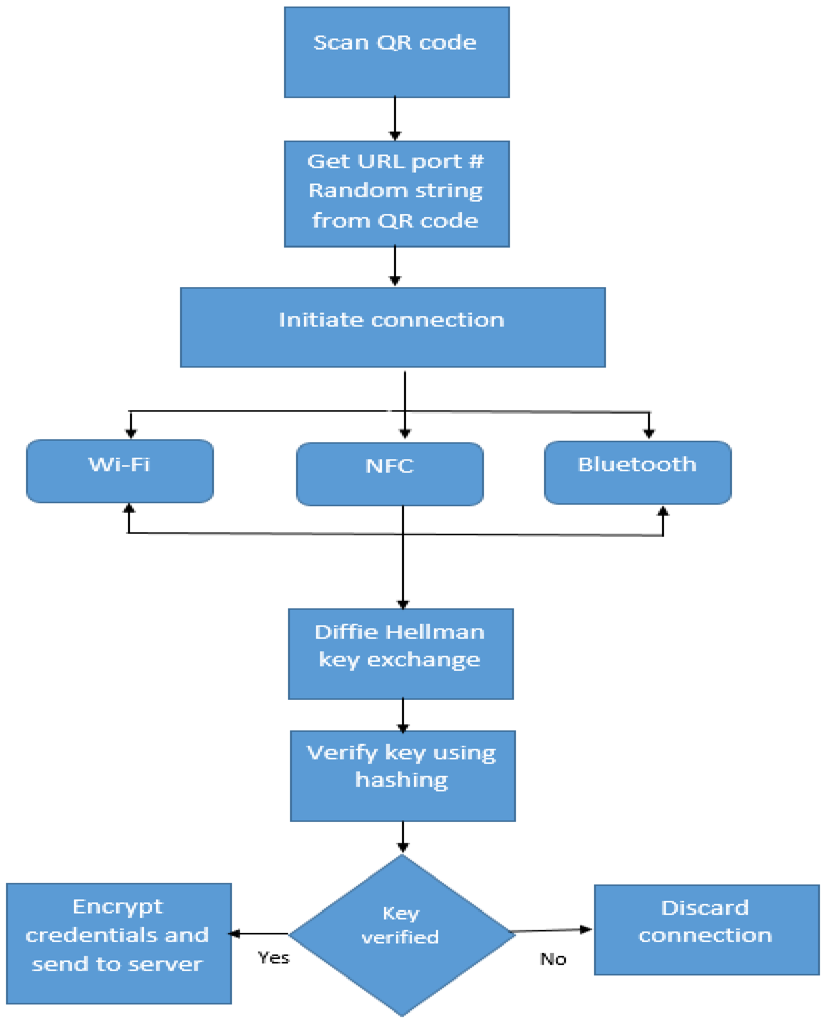
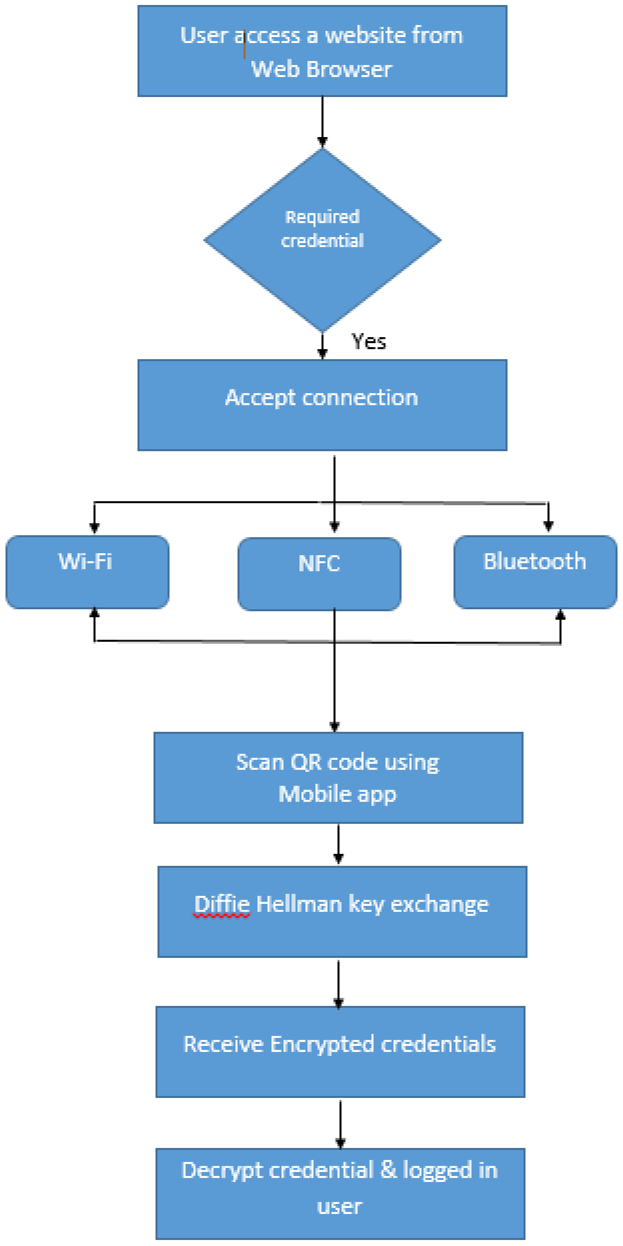


Figure 21: Overview of the mobile application.

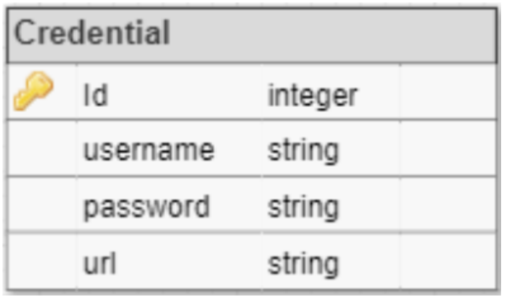
The mobile application performs various tasks including the QR code scanning, initiating a connection, computing secret key using the Diffie-Hellman key exchange algorithm, verifying the secret key, and encryption using the cryptographic algorithm.



User access a website from Web Browser
Required credential
Yes
Accept connection
Wi-Fi
NFC
Bluetooth
Scan QR code using Mobile app
Diffie Hellman key exchange
Receive Encrypted credentials
Decrypt credential & logged in user

Figure 22: Overview of the desktop application.

The desktop and the mobile application works together to get the desired outcome. To better understand the approach we will consider an example of a user that wants to login to a website such as (facebook.com, gmail.com, etc.) the traditional method is to type the username and password using a keyboard, and the user is easily logged into that website. However, it can lead to some attacks such as shoulder surfing and keylogging. If any keylogger is installed on the system, the credentials can be easily compromised. Our approach proposed that instead of typing the credentials using a keyboard the user can transfer the credentials from the smartphone using our mobile application. The user can save all the credentials in the mobile application that he wants to access in public places. The following table structure shows a snapshot of how the data can be stored in the mobile application.

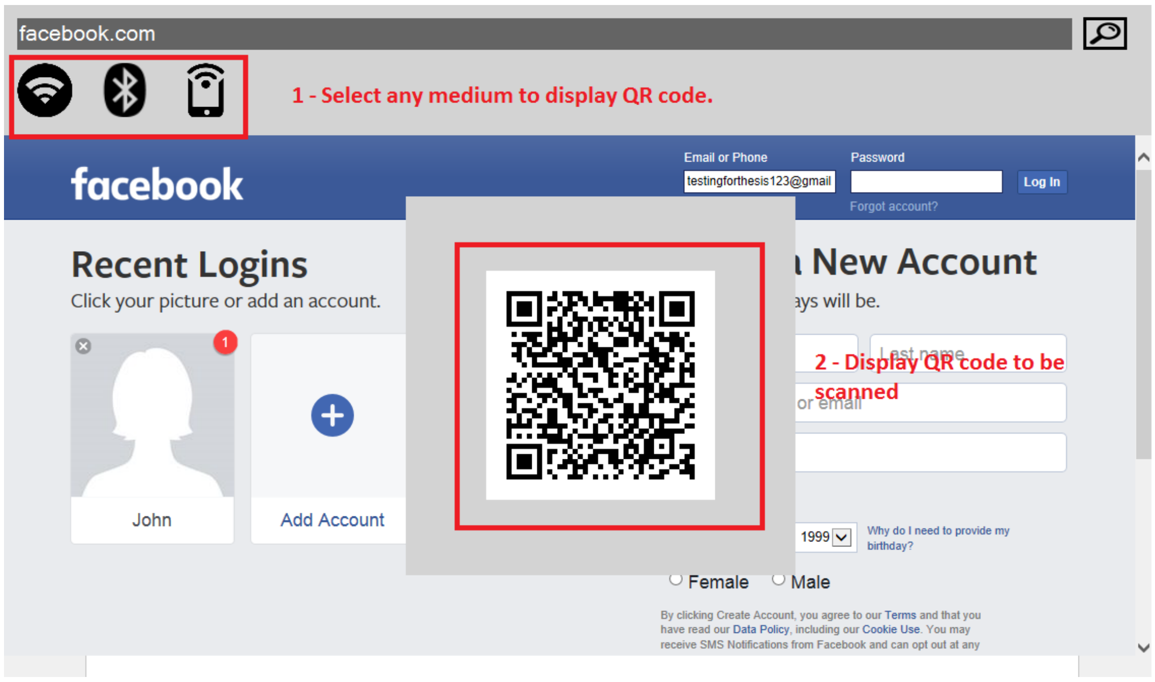


Credential		
🔑	Id	integer
	username	string
	password	string
	url	string

Figure 23: Database table structure

The mobile application provides the feature to insert, view and update credentials, so a user can perform these tasks using the mobile application.

When a user wants to login to a website using our desktop application, he/she has to scan the QR code through our mobile application. The QR code will be displayed on the desktop application as the user selects a medium (NFC, Bluetooth, or Wi-Fi). The following figure shows the screen shot of the desktop application and how the QR code will be displayed.


facebook.com
1 - Select any medium to display QR code.
facebook
Email or Phone
Password
Log In
Forgot account?
Recent Logins
Click your picture or add an account.
John
Add Account
New Account
ays will be.
Last name
2 - Display QR code to be scanned
or email
1999
Why do I need to provide my birthday?
Female
Male

testingforthesis123@gmail

By clicking Create Account, you agree to our Terms and that you have read our Data Policy, including our Cookie Use. You may receive SMS Notifications from Facebook and can opt out at any

Figure 24: Screenshot of the desktop application

Once the QR code is scanned, the mobile application will extract the information from the QR code, get the credentials from the database, encrypts it and transfer it to the desktop application using the selected medium. The desktop application will decrypt the credentials and automatically login user to the website. During this process, there are some significant tasks that are performed in the background which is discussed in detail in the following subsections.

4.2.1 CREATING CONNECTIONS BETWEEN TWO DEVICES

There are three mediums (Wi-Fi, Bluetooth, and NFC) through which both the devices (the client machine on which the desktop application is installed and the smartphone on which the mobile application is installed) can connect. If the user selects the Wi-Fi medium on both ends, the desktop application will generate and display a QR code. The QR code contains the IP address of the client machine which is captured on runtime, the port number on which the client machine is listening for any incoming connection and a randomly generated string. The value of this string is different and random each time when the user selects a medium. To make a connection over Wi-Fi the IP-address and port number are the essential components.

When the mobile application scans the QR code it extracts the encoded information and gets the IP-Address of the client machine, the port number on which the client machine is ready to accept the connection and a random string (the use of this randomly generated string is discussed in detail in section 4.2.3). Using the extracted IP-address and the port number the mobile application initiates a connection and wait for the client machine to accept the connection. Once the connection is established between both devices the applications move to the next phase. The following diagram shows the tasks performed by both the applications when the user selects the Wi-Fi medium.

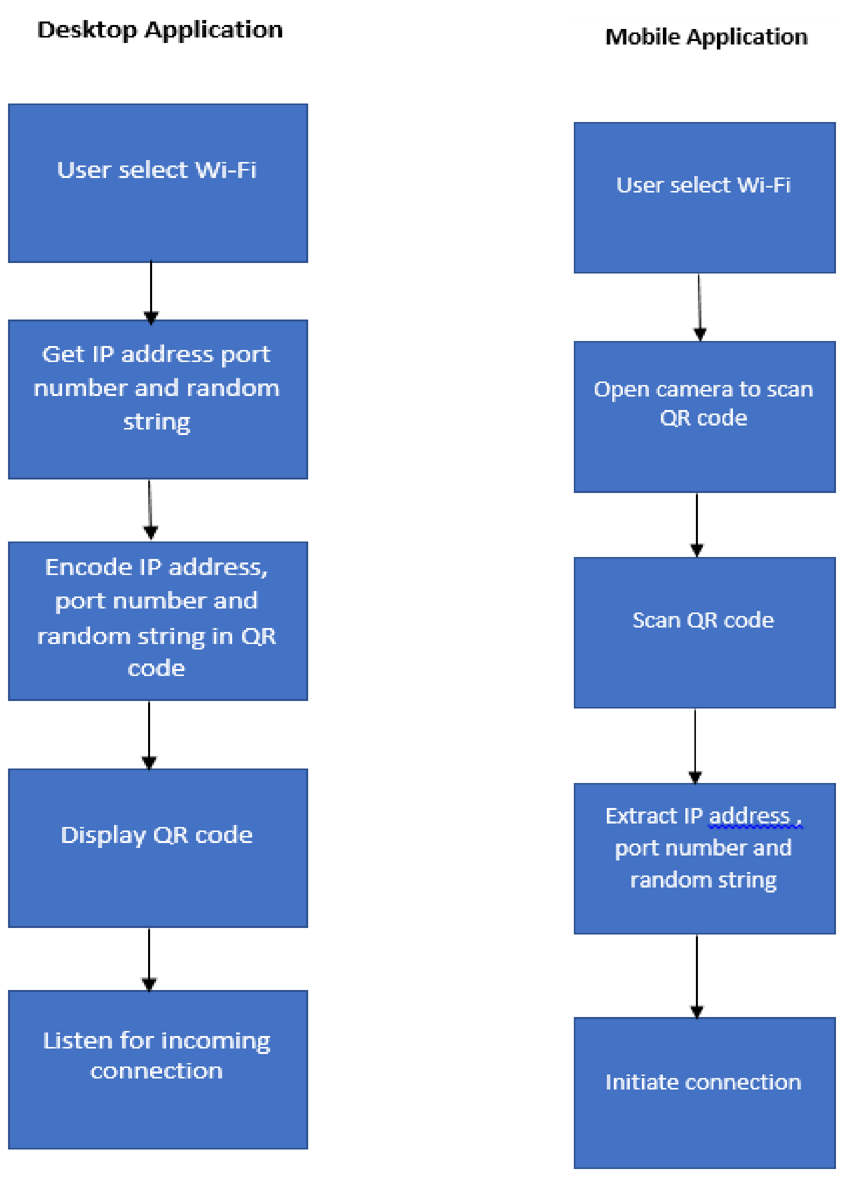

Desktop Application
Mobile Application
User select Wi-Fi
Get IP address port number and random string
Encode IP address, port number and random string in QR code
Display QR code
Listen for incoming connection
User select Wi-Fi
Open camera to scan QR code
Scan QR code
Extract IP address , port number and random string
Initiate connection

Figure 25: Initiating connection through Wi-Fi

Instead of the Wi-Fi connection, if the user selects the Bluetooth connection on both ends. The desktop application will display a list of all the devices that are discoverable via Bluetooth and are in range. From the list of devices the user needs to selects the device that has the mobile application installed on it. If the device is not previously paired with the desktop application it will initiate the pairing process. The pairing is not needed if both the application is already paired. After that the desktop application will display the QR Code that only contains the random string as for the Bluetooth connection the device do not need IP-address or port number. The mobile application will then scan the QR Code and continue to the next phase of the proposed approach. The following diagram shows the tasks performed by both the applications when the user selects the Bluetooth medium.

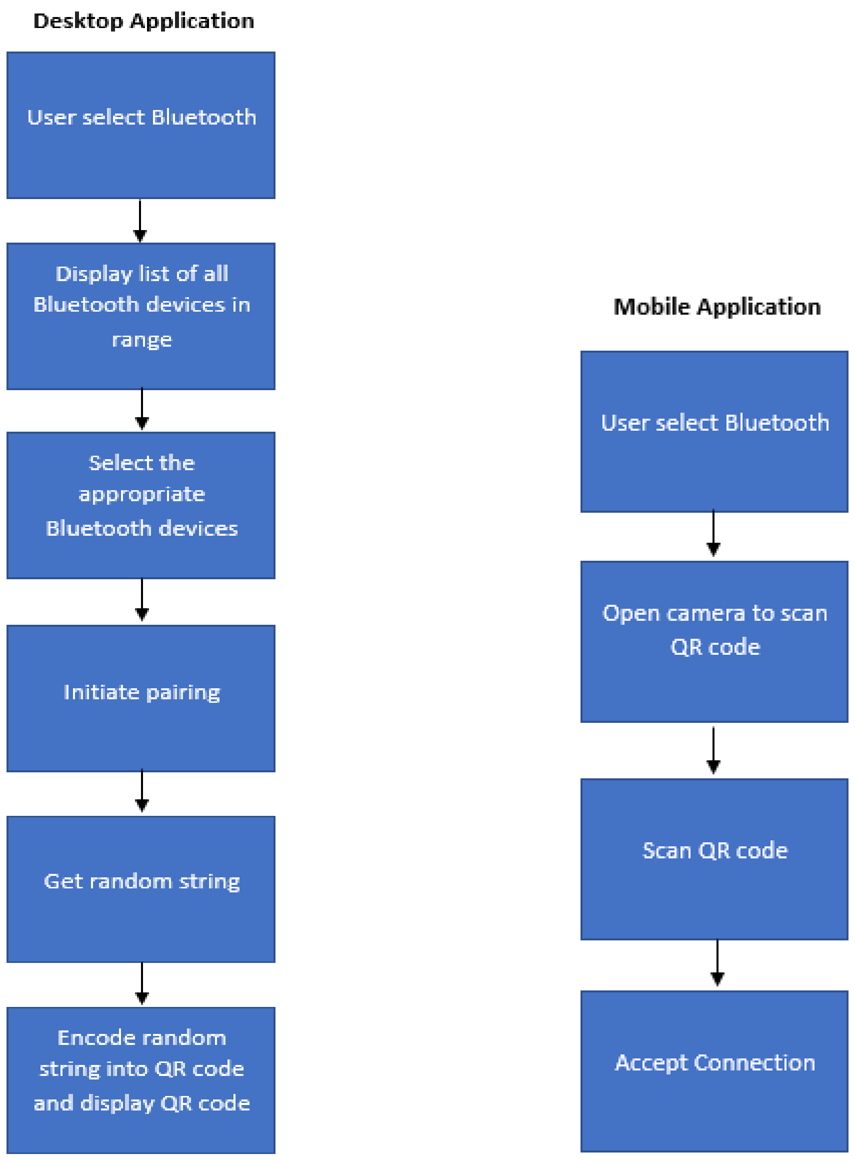

Desktop Application
User select Bluetooth
Display list of all Bluetooth devices in range
Select the appropriate Bluetooth devices
Initiate pairing
Get random string
Encode random string into QR code and display QR code
Mobile Application
User select Bluetooth
Open camera to scan QR code
Scan QR code
Accept Connection

Figure 26: Initiating connection through Bluetooth

The NFC approach is slightly different from the W-Fi and the Bluetooth approach. The smartphone on which the application is installed should be NFC enabled. For the desktop application to receive data via NFC either an NFC reader should be attached to the machine or the machine should be NFC enabled. The NFC reader reads the NFC data and transfers it to the USB port. We have used an NFC enabled device (phone or tablet) and developed an application that reads the NFC data and passes it through the USB port; this will act as a NFC reader.

If NFC is selected on both the applications, the user has to bring the smartphone and the NFC reader closer to send data via NFC. After that, the desktop application will wait until it receives the NFC message that contains some information. It will then display a QR code that needs to be scanned by the mobile application which contains some Diffie-Hellman key exchange parameters, random string, and the initialization vector. We will discuss all these parameters in detail in the following subsections. When the mobile application scans the QR code it extracts the information and proceed to the next phase. The following diagram shows the tasks performed by both the applications when the user selects the NFC medium.

Our proposed approach is designed in such a way that user needs to select a medium (Wi-Fi, Bluetooth or NFC) in order to transfer the credentials. The approach can be extended to make a default connection type so that user do not have to select a medium each time when the user needs to login to a web account. The advantage of implementing the default connection type is user do not have to perform any additional action on the desktop application. The desktop application will automatically show the QR code whenever there is a web page that requires credentials. The mobile application will then scan the QR code to proceed to next phases. There will not be a significant change in the mobile application as the user needs to perform an action to open the camera for scanning the QR code. The default connection type will always show a QR code on each page containing some login fields; and if the user wants to manually login to the web account it will be annoying for the user. The type of the connection can also be encode in the QR code due to which the user will not have to manually select the medium from the mobile application.

Eavesdropping, data modification and interception are some type of the attacks that can be performed on all of the three mediums. Although to transfer the NFC data the devices need to be close to each other, but the attacker can use some equipment such as antenna to perform these types of attacks. Furthermore, Bluejacking, Bluesnarfing and Bluebugging are some Bluetooth attacks that can be done on the Bluetooth devices. Considering all these types of attacks we need to transfer the credentials in a secure way. We have used AES-128 encryption to encrypt the credentials before sending it out on any medium. The encryption process is discuss in section 4.2.4.

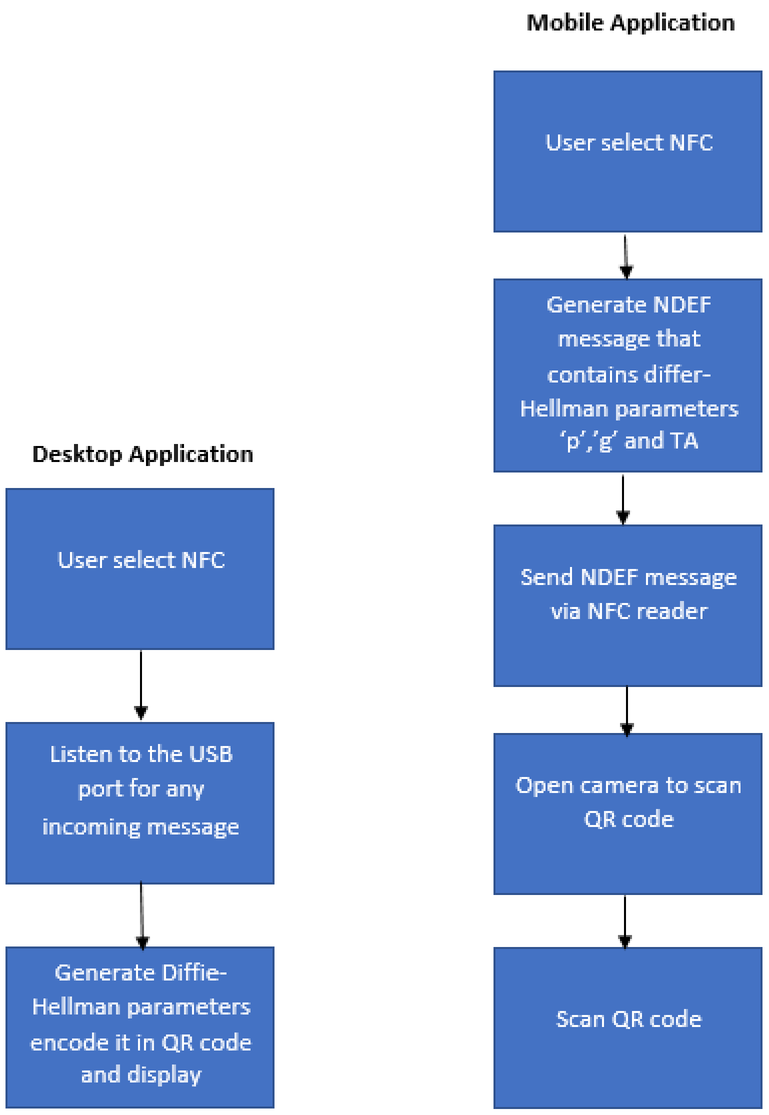

Mobile Application
User select NFC
Generate NDEF message that contains differ-Hellman parameters 'p','g' and TA
Send NDEF message via NFC reader
Open camera to scan QR code
Scan QR code
Desktop Application
User select NFC
Listen to the USB port for any incoming message
Generate Diffie-Hellman parameters encode it in QR code and display

Figure 27: Initiating connection through NFC

4.2.2 DIFFIE-HELLMAN KEY EXCHANGE

After the connection is made between the two devices, both the applications need a secret key to encrypt and decrypt credentials. The secret key is generated using Diffie-Hellman key exchange algorithm. The mobile application generates two random prime number 'p' and 'g', generates a random secret integer and computes TA. Except for the secret integer, all these values are transmitted to the desktop application using the selected medium. The desktop application acknowledges for each value received. The desktop application then generates a random secret integer and compute the TA using the received ‘p’ and ‘g’. The desktop application then sends the computed TA to the mobile application. Both the applications have ‘p’, ‘g’ and TAs, the secret integer is not transmitted and kept secret from others. Both the devices have all the parameters to compute the secret key that will be used to encrypt and decrypt data.

4.2.3 Verify Secret Key

Once the secret key is generated it needs to be verified as there is a chance of MITM attack. The Diffie-Hellman key exchange algorithm and how the secret key can be compromised through MITM attack is discussed in detail in section 2.3. To verify that the secret key is not compromised through MITM attack both the devices must have same secret key.

The one-way hash function and the random string that was generated in section 4.2.1 is used to verify the secret key. Both the application uses the randomly generated string and the one-way hash function (SHA-256) to compute a message digest. The desktop application sends the message digest to the mobile application. The mobile application verifies whether the received message digest is equal to the message digest generated by the mobile application. If both the message digest are equal, that means both the devices have the same secret key, and the key is not compromised via MITM attack. The following diagram shows the overview of the Diffie-Hellman and the key verification process.

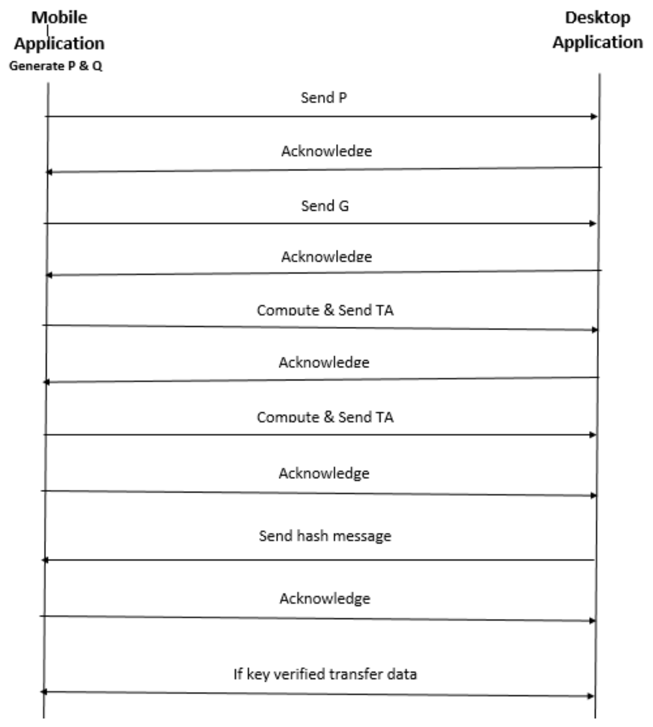


Figure 28: The Diffie-Hellman timing diagram.

4.2.4 ENCRYPT AND DECRYPT DATA

Now using the secret key both the applications can encrypt and decrypt data. The proposed approach uses symmetric key AES-128 encryption algorithm. The algorithm needs a secret key and an initialization vector to perform the cryptographic operations. The initialization vector is generated by the desktop application and transferred it to the mobile application. The mobile application uses the same initialization vector to perform encryption and decryption. The following figure shows the overall encryption and decryption phase.

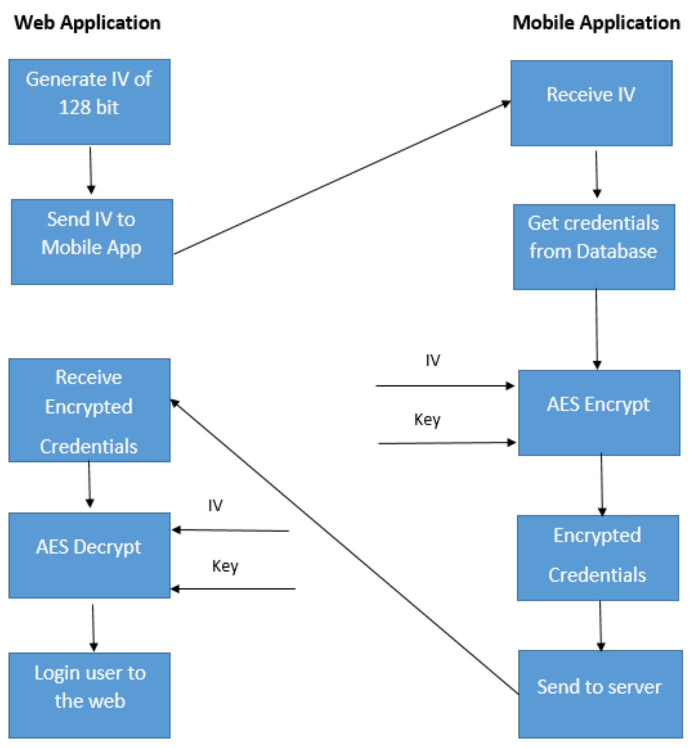


Figure 29: Encryption and decryption process.

Once the mobile application receives the initialization vector, it queries the database and gets the desired credentials. The mobile application then encrypts the credentials using the initialization vector and the secret key and sends it to the desktop application. Upon receiving the encrypted credentials, the desktop application decrypts the credentials and automatically login user on that website.

The overall proposed approach is useful when the user needs to login to any web account using the desktop system. Although most of the users use smartphone for performing their daily computational task such as for banking related activities, using social media or checking or replying to emails; but there are some cases where the user have no option other than to use desktop systems. If the user wants to access some legacy system or wants to login to a remote location the user is bound to use desktop system as smart phones are not capable enough to perform such tasks. Also, organizational accounts are easy and convenient to use in a desktop system rather than on a smartphone such as accessing a printer or project management tools.

CHAPTER 5 IMPLEMENTATION

5.1 DEVELOPMENT ENVIRONMENT

The proposed approach consists of two applications the desktop application and the mobile application. The desktop application was developed using C# on a Windows 8.1 operating system. The following table shows the development environment used for developing the Desktop application.

Operating System	Windows 8.1
IDE	Microsoft visual studio 2017
Programming language	C#
Processor	Intel® core™ i5-5200U CPU @ 2.20GHz
Installed Memory (RAM)	4.00 GB

Table 1: Development environment for the Desktop Application.

The Android mobile application was also developed on the same machine using the IDE android studio 2.3.3. The programing language JAVA was used for developing the Android application. Table 3 and 4 shows the development and testing environment for the mobile app.

Operating System	Windows 8.1

IDE	Android Studio 2.3.3
Programming language	Java 1.8.0_112
Minimum Android Support Version	4.0.3 (IceCreamSandwich)
Processor	Intel® core™ i5-5200U CPU @ 2.20GHz
Installed Memory (RAM)	4.00 GB

Table 2: Development environment for the Mobile Application.

Device	Samsung Galaxy S4
Android	4.3

Table 3: Mobile Device Used for testing.

5.1.1 PROGRAM ARCHITECTURE

To implement the proposed approach, we have used the peer-to-peer architecture. We have developed two applications a mobile app and a Desktop app. These two applications can communicate peer-to-peer with each other using WIFI, Bluetooth or NFC. The confidential data is sent in an encrypted form and is decrypted on the other hand.

5.1.2 JAVA ENVIRONMENT AND LIBRARIES USED

Java is an object-oriented computer programming language which is designed to have minimum implementation dependencies. Java gains popularity due to its WORA (write once run anywhere) feature which means that once the java code is compiled it lets the application to run on any machine without recompiling. Java was initially developed at Sun Microsystems by James Gosling, but it is now acquired by Oracle. As our approach consists of a mobile application, we have developed the application for Android, which is an open source mobile operating system and Java is used as the fundamental component for developing Android applications [49]. There are many libraries and packages available which are responsible for performing various tasks. We have used some of the packages in our application which are discussed below.

i. **java.security:** The package java.security contains many classes that provide various cryptographic operations. In our proposed approach we have used this package for generating message digest using SHA-256 [44].

ii. **java.crypto:** The package java.Crypto contain classes that are responsible for the cryptographic operations such as encryption, generating keys, and generating Message Authentication Code (MAC) [46]. We have used this package to secure the communication between the desktop and the mobile app. We have used AES-128 encryption to encrypt and decrypt user credentials.

iii. **java.net.Socket:** This java.net package provides classes for implementing network related tasks. The package provides the high and low-level API to deal with sockets, addresses, interfaces, connections, URLs, and URIs [47]. We have used Socket class of the java.net package for endpoint communication between the desktop and the mobile app.

iv. **android.nfc:** The android.nfc was added in API level 9 which is responsible for providing access to the NFC for reading and writing NDEF messages. The NfcManager class of the android.nfc API represents the high-level manager to get the device NFC adapter. The NfcAdapter class represents the NFC adapter of the device which is responsible for performing NFC operations. We can get an instance of the NFC adapter by ‘getDefaultAdapter()’ method and can use the NdefMessage message class to encapsulate data in NDEF format.[]

v. **android.bluetooth:** The android.bluetooth API was added in API level 5 which provides functionality including Bluetooth device scanning, making a connection between devices and transferring data between connected devices.

vi. **Zxing:** Zxing is an open source library for Java to scan barcodes/QR codes [50]. We have used this library in our approach to scan QRCode that is displayed on the Desktop application and extract the information from it.

Other packages used for the development of the Android application are java.io used for input and output operations on particular sockets, java.math used for performing mathematical operations including exponents, power and modulus, and java.util used for collections objects including mutable array lists.

5.1.3 C# ENVIRONMENT AND LIBRARIES USED

C# is a general purpose object-oriented programming language developed by Microsoft [51]. C# can be used to develop a wide variety of applications including Windows Form applications, client-server applications, and database applications. [52]. We have used C# to develop the desktop app and used the following libraries to implement various tasks.

i. **System.Net.Sockets:** The socket API provides classes to implement network related tasks. The API is used to create sockets, accept incoming connections, send and receive data over the network. We have used Socket API in our desktop app to perform socket creation and data related task over local WIFI network.

ii. **32Feet.Net:** The 32Feet is shared source project to make Bluetooth and Infrared technologies easily accessible with the C# code [53]. We have used this project in our desktop app to discover and pair with the Bluetooth devices and to transfer and receive data from the connected Bluetooth device

iii. **System.Security.Cryptography:** The security library provides services related to cryptography such as secure encryption and decryption of data, random number generation, hashing, and message authentication [54]. We have used this library in our desktop app for AES encryption and decryption and to generating message digest using SHA-256.

iv. **System.Threading.Tasks:** The threading library provides classes to perform concurrent and asynchronous tasks [55]. We have used this library in our approach to listen for any incoming connection asynchronously.

v. **QRCoder:** The QRCoder is responsible for generating QR Codes. The library is written in C# which has no other dependencies [56]. We have used this library to encode useful information such as IP-address and port number into a QR Code, which will be scanned by the mobile application to retrieve such information.

Other libraries used for the development of the desktop app are System.Numerics used to perform mathematical operation including exponents, modulus and power, and System.Drawing used for displaying QR Codes.

5.2 IMPLEMENTATION DETAILS OF THE PROPOSED APPROACH

This section describes the implementation details of the proposed approach. The step by step process between the mobile and the desktop app is discussed in detail in chapter 4. In this section, we will discuss the implementation details of each step. Following are the details of the major classes and their functionality of the desktop application.

i. **MainForm.cs:** This class plays a vital role in the desktop application as it contains the fundamental logic of the entire application. This class is responsible for all the primary operation including the main UI design. All the callbacks such as button clicks, RunWorkerCompletedEventHandler were implemented in this class. This class is also responsible for the WIFI, Bluetooth, and NFC socket connections.

ii. **AESEncryption.cs:** As the name suggests, this class is responsible for the encryption and decryption of the data using the AES algorithm. The 'EncryptStringToBytes_Aes' method takes data, secret key, and the initialization vector as input parameters and returns the encrypted bytes. Similarly, the 'DecryptStringFromBytes_Aes' method takes the ciphertext along with the secret key and initialization vector as input parameters and returns the decrypted text.

iii. **DiffieHellman.cs:** This class is generating and computing all the values that are used in Diffie-Hellman key exchange. For example generating secret integer, computing 'TA' based on the value of 'p' and 'g' and deriving secret key based on the value of 'TA1' (sent by the mobile device), 'P' and 'g'.

Unlike the desktop application which is a single page application, the mobile application consists of multiple pages called Activity. Each functionality such as WIFI communication, Bluetooth communication, NFC communication, insert and view credentials are implemented in separate activities. Each activity is tightly integrated with the java and an XML file. The java file contains the core logic of the class whereas the XML file only deals with the UI related task. Some of the important classes and their details are explained below.

i. **NfcActivity.java:** This class contains the basic logic for sending data via NFC. The basic functionality of this class is to create and push NDEF message via

NFC. This class also implements the ‘onNdefPushComplete’ call back which is triggered when the NDEF message is successfully sent.

ii. **BluetoothActivity.java:** The class contains basic logic for the Bluetooth feature. It is responsible for requesting Bluetooth permission, listening for any incoming Bluetooth connections, and sending and receiving data over the Bluetooth channel.

iii. **WifiActivity.java:** Similar to the Nfc and Bluetooth activity class, this class also contains the basic logic for communication but over a wifi medium.

iv. **DiffieHellman.java:** This class gives similar functionality as of DiffieHellman.cs except for it is written in Java for the mobile application.

v. **AESEncryption.java:** This class is identical to the AESEncryption.cs class except for it contribute to the mobile application rather than the desktop application.

All these classes work closely with each other to give the desired functionality. We will discuss the major operations that are performed in the proposed approach and how they are implemented in the respective mobile and desktop application.

5.2.1 RANDOM STRING GENERATION:

The desktop application generates a 128-bit random string which is encoded in the form of QR Code along with some other information (such as IP address and port number). The QR Code is then displayed on the screen to be decoded by the mobile application. A different random string is generated each time when the user selects a medium, or a Diffie-Hellman key exchange is done. The following code snippet is used to generate the random string.

```
Random random = new Random();
return randomString = new string(Enumerable.Repeat(chars,
stringLength)
.Select(s => s[random.Next(s.Length)]).ToArray());
```

5.2.2 QRCODE GENERATION:

The QRCoder library is used to generate the QR Codes from the given data. The QR code contains the IP-address, port number, the URL for which we need the credentials and the random string generated above. All these values are concatenated using a predefined string which will be parsed on the mobile end. The QRCode class takes the data as input and returns the QR Code in the Bitmap format which is then displayed in the picture box. The following code snippet shows the generation of QR code.

```
QRCodeGenerator qrGenerator = new QRCodeGenerator();
QRCodeData qrCodeData =
qrGenerator.CreateQrCode(GetLocalIPAddress() + "$$" +
PORTNUMBER.ToString() + "$$" + textBox1.Text + "$$" +
diffieHellman.getRandomString(), QRCodeGenerator.ECCLevel.Q);
```

```
QRCode qrCode = new QRCode(qrCodeData);
Bitmap qrCodeImage = qrCode.GetGraphic(pixels);
```

5.2.3 SOCKET CONNECTIONS:

Sockets are used for communication between the two applications. Data is transferred and received using the input and output stream. Following code is used to listen to any incoming connection on the desktop application.

```
TcpListener serverSocket = new TcpListener(PORTNUMBER);
TcpClient clientSocket = default(TcpClient);
serverSocket.Start();
clientSocket = serverSocket.AcceptTcpClient();
```

On the other hand, the mobile application initiates the connection using the following code snippet.

```
Socket socket = new Socket(ipAddtess, port);
```

Once the connection is successfully established between the two applications, the data is sent and read using the BufferedReader and DataOutputStream class. The object of the BufferedReader class is created by passing the input stream of the socket to the constructor, and the method readLine is used to read any incoming data. The following code snippet shows how the data is read and write on the mobile end.

```
BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
```

```
dataReceived = bufferedReader.readLine();
```

The data is written to the socket using the DataOutputStream class, the constructor takes output stream of the socket as the input parameter to create the instance, and the writeBytes method is used to write any data to the socket.

```
DataOutputStream dOut = new
DataOutputStream(socket.getOutputStream());
dOut.writeBytes(data);
dOut.flush();
```

On the desktop app, ‘NetworkStream’ class is used to read and write data over the sockets. Following code snippet shows how the data is read and write on the desktop app.

```
NetworkStream stream = socket.GetStream();
byte[] myReadBuffer = new byte[byteLenght];
stream.Read(myReadBuffer, 0, byteLenght);

stream.Write(sendBytes, 0, sendBytes.Length);
stream.Flush();
```

5.2.4 GENERATING ‘p’ ‘g’ AND ‘TA’:

For the Diffie-Hellman key exchange, both the desktop and mobile app should agree upon a common value of p and g. The mobile app generates a 1024 bit random prime number for the value of ‘p’ and ‘g’ along with a 256-bit secret integer to compute the value of ‘TA’. Following code snippet shows how these values are generated.

```
Random rand = new Random();
p = new BigInteger(bitLength, certinity, rand);
g = new BigInteger(bitLength, certinity, rand);
TA = g.modPow(secretInteger,p);
```

Once these values are generated, it is transferred to the desktop application. The desktop application then computes the TA and the secret key through which it can encrypt and decrypt data. The code snippet shows how the values of TA and the secret key is computed on the desktop application.

```
TA = BigInteger.ModPow(g, getSecretNumber(), p);
secretKey = BigInteger.ModPow(mobileTA, getSecretNumber(), p);
```

5.2.5 GENERATING RANDOM SECRET INTEGER:

The desktop and the mobile app generates a random integer which is kept secret. These random integers are used to compute the values of TA and the secret key. On the desktop app, the random integer is generated using the following code snippet.

```
byte[] buffer = new byte[Length];
Random rand = new Random();
rand.NextBytes(buffer);
secretNumber = new BigInteger(buffer);
```

The mobile app uses the following code snippet to generate the random integer.

```
Random rand = new Random();
secretInteger = new BigInteger(bitLength, rand);
```

5.2.6 EXCHANGE ‘P’ ‘G’ AND ‘TA’:

The value of ‘p’ and ‘g’ is generated on the mobile side, and these values along with the ‘TA’ is transferred to the desktop app. The desktop app acknowledges for each value received. The following code snippet shows how the values of ‘p’, ‘g’ and ‘TA’ is sent from the mobile application.

```
dataOutStream.writeBytes(getG().toString());
dataOutStream.flush();
String acknowledgment = bufferedReader.readLine();

dataOutStream.writeBytes(getP().toString());
dataOutStream.flush();
acknowledgment = bufferedReader.readLine();

dataOutStream.writeBytes(getTA().toString());
dataOutStream.flush();
acknowledgment = bufferedReader.readLine();
```

We have used the following code snippet to read these values, send the acknowledgments and to send the computed ‘TA’ to the mobile app.

```
String message = readNetworkStream(networkStream, bufferSize);
BigInteger.TryParse(message, out diffieHellman.g);
writeNetworkStream(networkStream, ack);
```

```
message = readNetworkStream(networkStream, bufferSize);
BigInteger.TryParse(message, out diffieHellman.p);
writeNetworkStream(networkStream, ack);

message = readNetworkStream(networkStream, bufferSize);
BigInteger.TryParse(message, out diffieHellman.mobileTA);
writeNetworkStream(networkStream,
diffieHellman.getTA().ToString());
```

5.2.7 VERIFY SECRET KEY USING HASHING:

Both the applications have computed the secret key using the power, modulus and exponential operations. However, there is a chance that the secret key is compromised and is different on each end due to the MITM attack, so the key needs to be verified. On the mobile end, java.security package provides MessageDigest class for hashing. The 'getInstance()' method of the MessageDigest class is used to get the specific instance (such as SHA-256, MD5, etc.). The following code snippet shows an instance of the SHA-256 message digest.

```
MessageDigest digest = MessageDigest.getInstance("SHA-256");
```

The desktop app uses 'SHA256Managed' class to generate the digest. Following code snippet is used to create an instance of the 'SHA256Managed' class and generate the digest of the random string that was generated in section 5.2.1 and the secret key.

```
SHA256Managed hashstring = new SHA256Managed();
byte[] hash = hashstring.ComputeHash(bytes);
```

The mobile app generates a new digest using the random string that was scanned and decoded from the QR Code and the secret key. It then matches the new digest with the digest received from the desktop app. If the two digests are same, then both the applications have same keys. We have used the following code snippet to generate and compare message digest.

```
MessageDigest digest = MessageDigest.getInstance("SHA-256");
byte[] hash = digest.digest(str.getBytes());
String newHash = String.format(flag, new
java.math.BigInteger(hash));

if(serverHash.equal(newHash))
    verifiedKEY = true;
else
    verifiedKEY = false;
```

5.2.8 ENCRYPT AND DECRYPT DATA:

Once the secret key is generated and verified, the mobile application gets the user credentials against the URL. The credential is then encrypted using the AES encryption algorithm and send to the desktop application. The AES encryption algorithm needs initialization vector to perform the encryption and decryption operations. The initialization vector is generated by the desktop application using the following code and sent it to the mobile application.

```
using (AesManaged myAes = new AesManaged())
```

```
{
      return myAes.IV;
}
```

The mobile application uses the initialization vector to encrypt credential and sent it to the desktop application.

```
Cipher AesCipher = Cipher.getInstance("AES");
SecretKeySpec skeySpec = new SecretKeySpec(secKey.getBytes("UTF-
8"), "AES");

IvParameterSpec ivParameterSpec = new
IvParameterSpec(initializationVector);
AesCipher.init(Cipher.ENCRYPT_MODE, skeySpec, ivParameterSpec);
encryptedBytes = AesCipher.doFinal(plainText.getBytes());
```

The desktop application then decrypts the credential and automatically logged in user. The code snippet used for decrypting the credentials on the desktop app is as follows.

```
using (AesManaged aesAlg = new AesManaged())
{
      aesAlg.Key = Key;
      aesAlg.IV = IV;
      ICryptoTransform decryptor =
      aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);
```

```
        using (MemoryStream msDecrypt = new
        MemoryStream(encryptedBytes))
        {
                using (CryptoStream csDecrypt = new
                CryptoStream(msDecrypt, decryptor,
                CryptoStreamMode.Read))
                {
                        using (StreamReader srDecrypt = new
                        StreamReader(csDecrypt))
                        {
                                plaintext = srDecrypt.ReadToEnd();
                        }
                }
        }
}
```

5.2.8 DATABASE ACCESS AND QUERIES

The mobile application uses a database to store user credentials. The ‘DatabaseHandler’ class is created to perform the database related tasks including insert, get and delete operations. The ‘getCredential’ method of the DatabaseHandler class takes the URL as input and returns the credential. Following code snippet is used to get credentials.

```
DatabaseHandler databaseHandler = new DatabaseHandler(this);
Credential credential = databaseHandler.getCredential(url);
```

5.2.9 PERMISSIONS REQUIRED FOR ANDROID APPLICATION:

As our mobile application accesses some of the hardware such as NFC, Bluetooth, camera, and Wifi. The application needs user permission to access all these hardware from the application. Each Android application contains a manifest file that defines all the permissions needed for that application. The following code is used to define permissions for our mobile application.

```
<uses-permission android:name = "android.permission.INTERNET" />
<uses-permission android:name =
"android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name =
"android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name = "android.permission.BLUETOOTH"/>
<uses-permission android:name =
"android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name = "android.permission.CAMERA"/>
<uses-permission android:name = "android.permission.NFC" />
<uses-feature android:name = "android.hardware.nfc"
android:required="true" />
```

Other than permission we also have to define an action when we received an NDEF message in our manifest file.

```
<activity android:name=".NfcActivity">
     <intent-filter>
```

```
         <action android:name =
"android.nfc.action.NDEF_DISCOVERED" />
         <category android:name =
"android.intent.category.DEFAULT"/>
         <data android:mimeType = "text/plain" />
     </intent-filter>
</activity>
```

CHAPTER 6 EXPERIMENTAL RESULTS AND ANALYSIS

In this chapter, we have discussed the experimental environment along with the results. The main objective of the proposed approach is to prevent keylogging and shoulder surfing. We have proposed an approach through which we can prevent these type of attacks. As the proposed approach consists of the mobile and desktop application the data transferred between these two applications should be in encrypted form. The following sections of this chapter discusses the experimental scenario and results. In section 6.1 we have discussed the experimental setup, the device and environment used to test the proposed approach. Later in section 6.2, we have discussed the evaluation of the approach by performing traffic analysis using Wireshark.

6.1 EXPERIMENTAL SETUP

To test the proposed approach the desktop application was installed on a Windows based computer with the following specifications.

- Operating System: Windows OS 8.1
- RAM: 4GB
- Hard Disk Storage Capacity: 500GB
- Processor: Intel® Core™ i5-5200U CPU @ 2.20GHz 2.20GHz
- System Type: 64-bit Operating System, x64-based processor.
- Visual Studio 2017

The mobile application was installed on the android based smart phone which has the following specifications.

- Device Model: Samsung Galaxy S4
- Android Version: Android 5.0.1 (Lollypop)
- RAM: 2GB
- Processor: ARMv7 Quad-core 1.9 GHz
- Internal Storage: 16GB

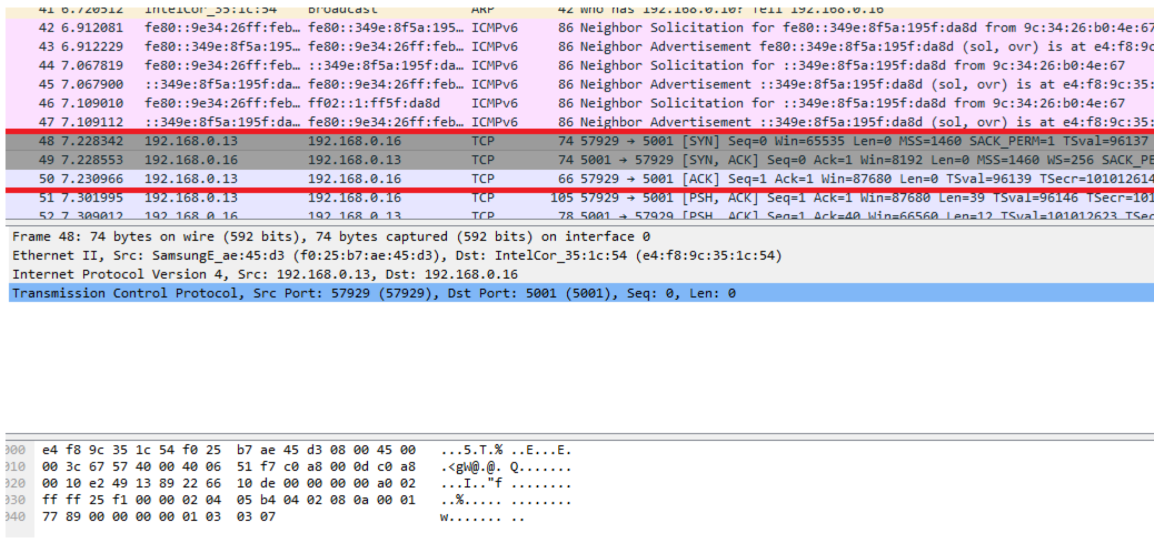
6.2 EVALUATION OF THE PROPOSED APPROACH

To test the proposed approach we have used Wireshark and UBlyzer to capture and analyze network traffic. We have divided the evaluation phase in three major category.

1. Testing over the Wi-Fi medium.
2. Testing over the Bluetooth medium.
3. Testing over the NFC medium.

6.2.1 Testing over the Wi-Fi medium.

We have used Wireshark Version 2.0.3 to capture network packets over the Wi-Fi medium. We have created testing accounts for website such as Facebook, Gmail and Amazon to analyze the proposed approach. The password for these websites are stored in the mobile application so that we can automatically login to the website through the desktop application. When the user selects the Wi-Fi medium on both the applications and scanned the QR code, the mobile application initiate a TCP connection as shown in the following figure.



```
41 6.720512  IntelCor_35:1c:54   Broadcast           ARP      42 Who has 192.168.0.10? Tell 192.168.0.16
42 6.912081  fe80::9e34:26ff:feb… fe80::349e:8f5a:195… ICMPv6   86 Neighbor Solicitation for fe80::349e:8f5a:195f:da8d from 9c:34:26:b0:4e:67
43 6.912229  fe80::349e:8f5a:195… fe80::9e34:26ff:feb… ICMPv6   86 Neighbor Advertisement fe80::349e:8f5a:195f:da8d (sol, ovr) is at e4:f8:9c
44 7.067819  fe80::9e34:26ff:feb… ::349e:8f5a:195f:da… ICMPv6   86 Neighbor Solicitation for ::349e:8f5a:195f:da8d from 9c:34:26:b0:4e:67
45 7.067900  ::349e:8f5a:195f:da… fe80::9e34:26ff:feb… ICMPv6   86 Neighbor Advertisement ::349e:8f5a:195f:da8d (sol, ovr) is at e4:f8:9c:35:
46 7.109010  fe80::9e34:26ff:feb… ff02::1:ff5f:da8d   ICMPv6   86 Neighbor Solicitation for ::349e:8f5a:195f:da8d from 9c:34:26:b0:4e:67
47 7.109112  ::349e:8f5a:195f:da… fe80::9e34:26ff:feb… ICMPv6   86 Neighbor Advertisement ::349e:8f5a:195f:da8d (sol, ovr) is at e4:f8:9c:35:
48 7.228342  192.168.0.13        192.168.0.16        TCP      74 57929 → 5001 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=96137
49 7.228553  192.168.0.16        192.168.0.13        TCP      74 5001 → 57929 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PE
50 7.230966  192.168.0.13        192.168.0.16        TCP      66 57929 → 5001 [ACK] Seq=1 Ack=1 Win=87680 Len=0 TSval=96139 TSecr=101012614
51 7.301995  192.168.0.13        192.168.0.16        TCP     105 57929 → 5001 [PSH, ACK] Seq=1 Ack=1 Win=87680 Len=39 TSval=96146 TSecr=101
52 7.309012  192.168.0.16        192.168.0.13        TCP      78 5001 → 57929 [PSH, ACK] Seq=1 Ack=40 Win=66560 Len=12 TSval=101012623 TSec

Frame 48: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Ethernet II, Src: SamsungE_ae:45:d3 (f0:25:b7:ae:45:d3), Dst: IntelCor_35:1c:54 (e4:f8:9c:35:1c:54)
Internet Protocol Version 4, Src: 192.168.0.13, Dst: 192.168.0.16
Transmission Control Protocol, Src Port: 57929 (57929), Dst Port: 5001 (5001), Seq: 0, Len: 0

000  e4 f8 9c 35 1c 54 f0 25  b7 ae 45 d3 08 00 45 00   ...5.T.% ..E...E.
010  00 3c 67 57 40 00 40 06  51 f7 c0 a8 00 0d c0 a8   .<gW@.@. Q.......
020  00 10 e2 49 13 89 22 66  10 de 00 00 00 00 a0 02   ...I.."f ........
030  ff ff 25 f1 00 00 02 04  05 b4 04 02 08 0a 00 01   ..%..... ........
040  77 89 00 00 00 00 01 03  03 07                     w....... ..
```

Figure 30: Initiating Wi-Fi connection between Desktop and Mobile App.

The above figure shows the packets captured through Wireshark when both the applications are establishing a TCP connection. The desktop application has the IP address 192.168.0.16 and the mobile application has the IP address 192.168.0.13. The mobile application initiate the connection and once the desktop application accepts the connection both the applications exchange the Diffie-Hellman parameters. The following figures show the value of ‘P’, ‘G’ and ‘TA’ transferred from Mobile application to desktop application. Note that these values are sent in plain text and not in encrypted form as they do not need to be kept secret.

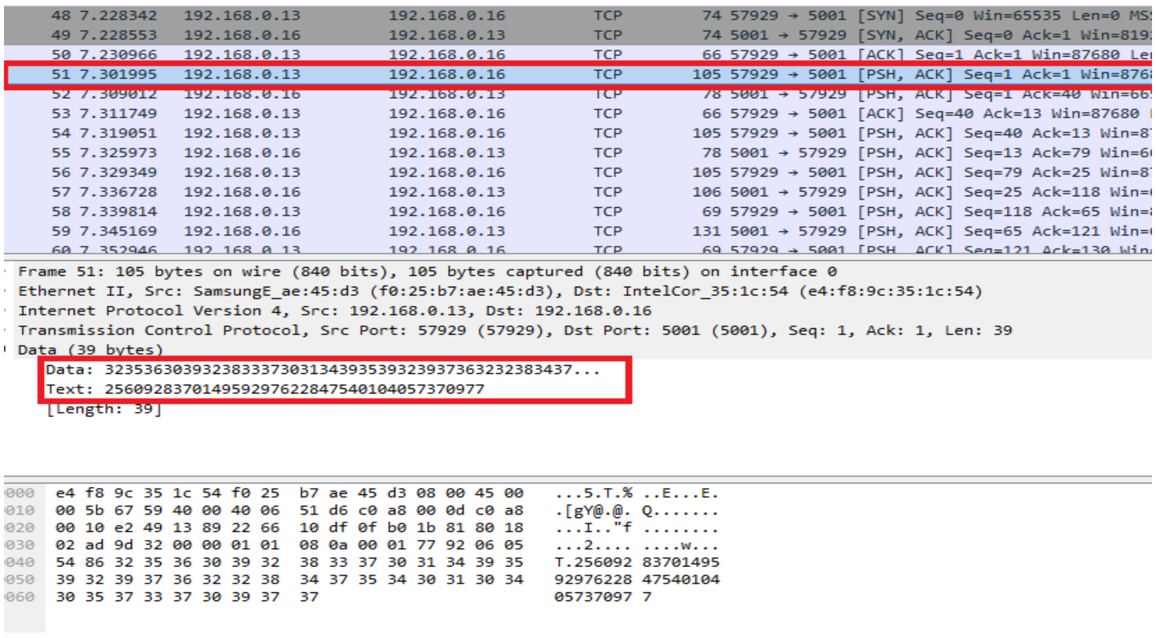

```
48 7.228342   192.168.0.13        192.168.0.16        TCP        74 57929 → 5001 [SYN] Seq=0 Win=65535 Len=0 MS
49 7.228553   192.168.0.16        192.168.0.13        TCP        74 5001 → 57929 [SYN, ACK] Seq=0 Ack=1 Win=819
50 7.230966   192.168.0.13        192.168.0.16        TCP        66 57929 → 5001 [ACK] Seq=1 Ack=1 Win=87680 Le
51 7.301995   192.168.0.13        192.168.0.16        TCP       105 57929 → 5001 [PSH, ACK] Seq=1 Ack=1 Win=876
52 7.309012   192.168.0.16        192.168.0.13        TCP        78 5001 → 57929 [PSH, ACK] Seq=1 Ack=40 Win=66
53 7.311749   192.168.0.13        192.168.0.16        TCP        66 57929 → 5001 [ACK] Seq=40 Ack=13 Win=87680
54 7.319051   192.168.0.13        192.168.0.16        TCP       105 57929 → 5001 [PSH, ACK] Seq=40 Ack=13 Win=8
55 7.325973   192.168.0.16        192.168.0.13        TCP        78 5001 → 57929 [PSH, ACK] Seq=13 Ack=79 Win=6
56 7.329349   192.168.0.13        192.168.0.16        TCP       105 57929 → 5001 [PSH, ACK] Seq=79 Ack=25 Win=8
57 7.336728   192.168.0.16        192.168.0.13        TCP       106 5001 → 57929 [PSH, ACK] Seq=25 Ack=118 Win=
58 7.339814   192.168.0.13        192.168.0.16        TCP        69 57929 → 5001 [PSH, ACK] Seq=118 Ack=65 Win=
59 7.345169   192.168.0.16        192.168.0.13        TCP       131 5001 → 57929 [PSH, ACK] Seq=65 Ack=121 Win=
60 7.352946   192.168.0.13        192.168.0.16        TCP        69 57929 → 5001 [PSH, ACK] Seq=121 Ack=130 Win
Frame 51: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface 0
Ethernet II, Src: SamsungE_ae:45:d3 (f0:25:b7:ae:45:d3), Dst: IntelCor_35:1c:54 (e4:f8:9c:35:1c:54)
Internet Protocol Version 4, Src: 192.168.0.13, Dst: 192.168.0.16
Transmission Control Protocol, Src Port: 57929 (57929), Dst Port: 5001 (5001), Seq: 1, Ack: 1, Len: 39
Data (39 bytes)
    Data: 323536303932383337303134393539323937363232383437...
    Text: 256092837014959297622847540104057370977
    [Length: 39]

000  e4 f8 9c 35 1c 54 f0 25  b7 ae 45 d3 08 00 45 00   ...5.T.% ..E...E.
010  00 5b 67 59 40 00 40 06  51 d6 c0 a8 00 0d c0 a8   .[gY@.@. Q.......
020  00 10 e2 49 13 89 22 66  10 df 0f b0 1b 81 80 18   ...I.."f ........
030  02 ad 9d 32 00 00 01 01  08 0a 00 01 77 92 06 05   ...2.... ....w...
040  54 86 32 35 36 30 39 32  38 33 37 30 31 34 39 35   T.256092 83701495
050  39 32 39 37 36 32 32 38  34 37 35 34 30 31 30 34   92976228 47540104
060  30 35 37 33 37 30 39 37  37                        05737097 7
```

Figure 31: Sending Diffie-Hellman parameter P via Wi-Fi

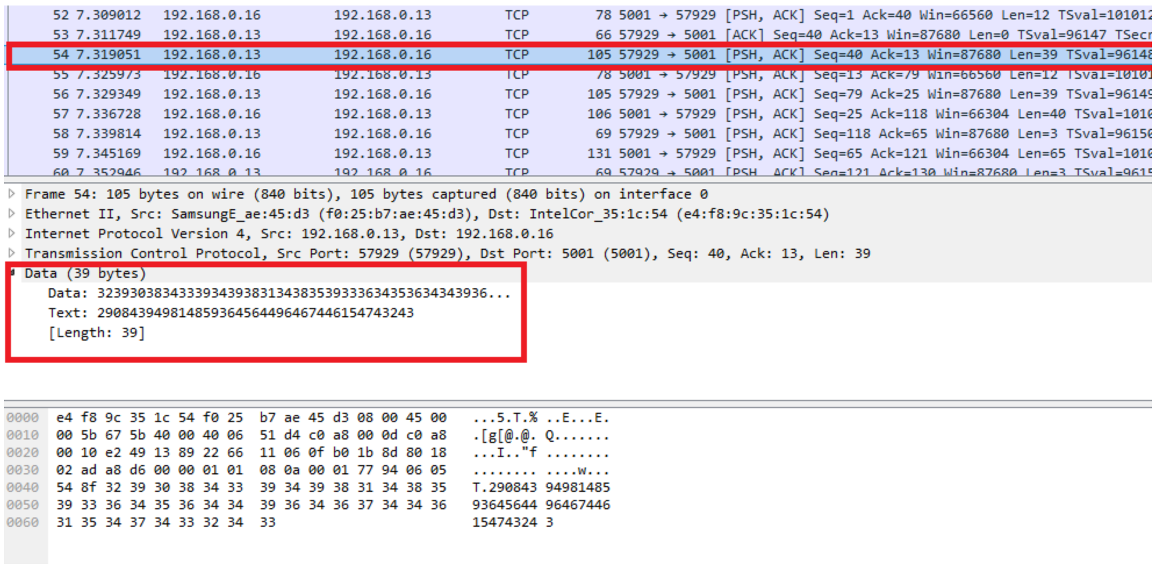
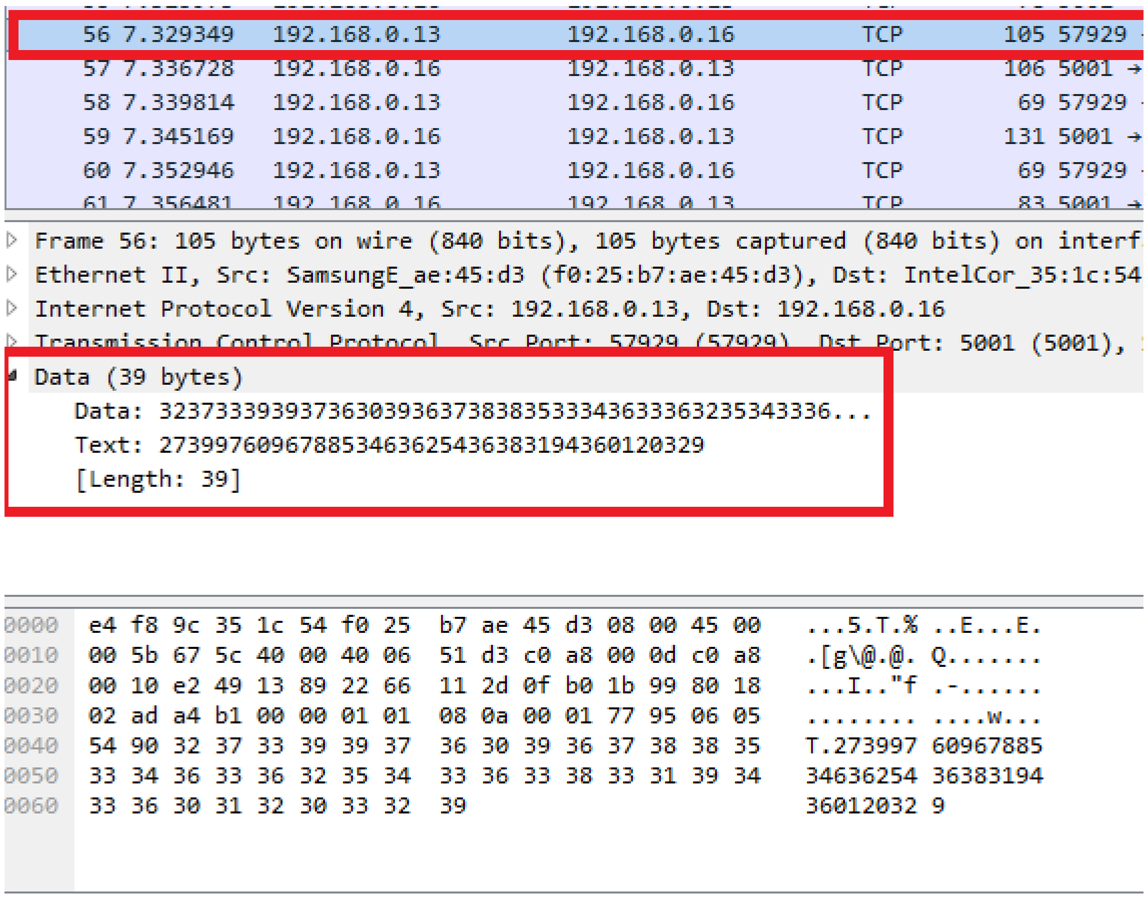

```
52 7.309012   192.168.0.16       192.168.0.13       TCP      78 5001 → 57929 [PSH, ACK] Seq=1 Ack=40 Win=66560 Len=12 TSval=10101
53 7.311749   192.168.0.13       192.168.0.16       TCP      66 57929 → 5001 [ACK] Seq=40 Ack=13 Win=87680 Len=0 TSval=96147 TSec
54 7.319051   192.168.0.13       192.168.0.16       TCP     105 57929 → 5001 [PSH, ACK] Seq=40 Ack=13 Win=87680 Len=39 TSval=9614
55 7.325973   192.168.0.16       192.168.0.13       TCP      78 5001 → 57929 [PSH, ACK] Seq=13 Ack=79 Win=66560 Len=12 TSval=1010
56 7.329349   192.168.0.13       192.168.0.16       TCP     105 57929 → 5001 [PSH, ACK] Seq=79 Ack=25 Win=87680 Len=39 TSval=9614
57 7.336728   192.168.0.16       192.168.0.13       TCP     106 5001 → 57929 [PSH, ACK] Seq=25 Ack=118 Win=66304 Len=40 TSval=101
58 7.339814   192.168.0.13       192.168.0.16       TCP      69 57929 → 5001 [PSH, ACK] Seq=118 Ack=65 Win=87680 Len=3 TSval=9615
59 7.345169   192.168.0.16       192.168.0.13       TCP     131 5001 → 57929 [PSH, ACK] Seq=65 Ack=121 Win=66304 Len=65 TSval=101
60 7.352946   192.168.0.13       192.168.0.16       TCP      69 57929 → 5001 [PSH, ACK] Seq=121 Ack=130 Win=87680 Len=3 TSval=961
▷ Frame 54: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface 0
▷ Ethernet II, Src: SamsungE_ae:45:d3 (f0:25:b7:ae:45:d3), Dst: IntelCor_35:1c:54 (e4:f8:9c:35:1c:54)
▷ Internet Protocol Version 4, Src: 192.168.0.13, Dst: 192.168.0.16
▷ Transmission Control Protocol, Src Port: 57929 (57929), Dst Port: 5001 (5001), Seq: 40, Ack: 13, Len: 39
  Data (39 bytes)
    Data: 323930383433393439383134383539333634353634343936...
    Text: 290843949814859364564496467446154743243
    [Length: 39]

0000  e4 f8 9c 35 1c 54 f0 25  b7 ae 45 d3 08 00 45 00   ...5.T.% ..E...E.
0010  00 5b 67 5b 40 00 40 06  51 d4 c0 a8 00 0d c0 a8   .[g[@.@. Q.......
0020  00 10 e2 49 13 89 22 66  11 06 0f b0 1b 8d 80 18   ...I.."f ........
0030  02 ad a8 d6 00 00 01 01  08 0a 00 01 77 94 06 05   ........ ....w...
0040  54 8f 32 39 30 38 34 33  39 34 39 38 31 34 38 35   T.290843 94981485
0050  39 33 36 34 35 36 34 34  39 36 34 36 37 34 34 36   93645644 96467446
0060  31 35 34 37 34 33 32 34  33                        15474324 3
```

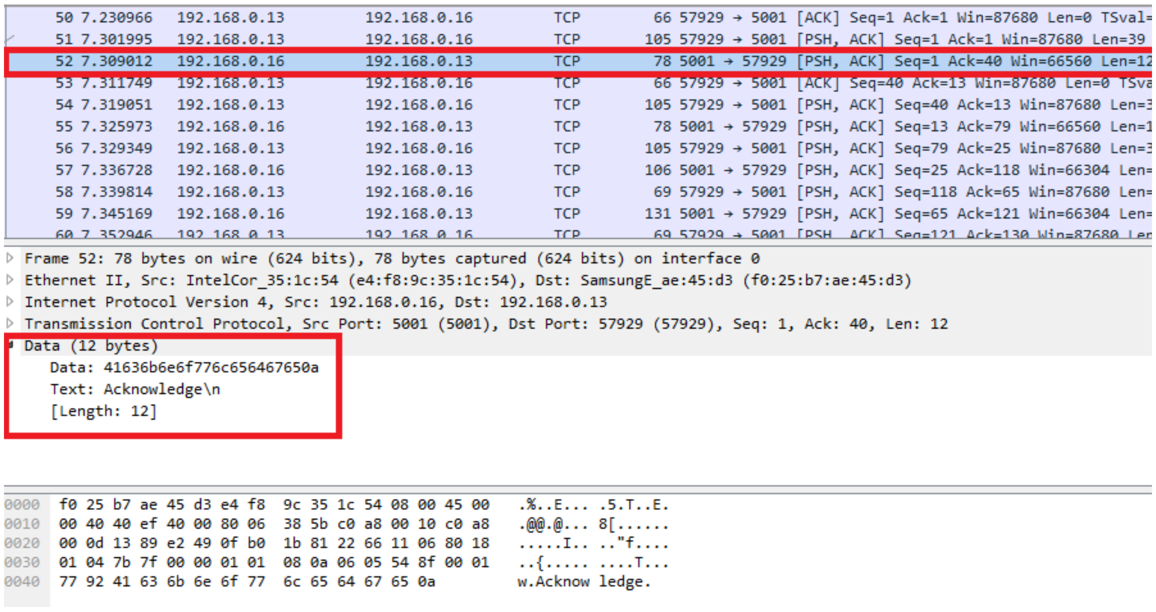
Figure 32: Sending Diffie-Hellman parameter G via Wi-Fi



```
   56 7.329349   192.168.0.13          192.168.0.16          TCP        105 57929
   57 7.336728   192.168.0.16          192.168.0.13          TCP        106 5001 →
   58 7.339814   192.168.0.13          192.168.0.16          TCP         69 57929
   59 7.345169   192.168.0.16          192.168.0.13          TCP        131 5001 →
   60 7.352946   192.168.0.13          192.168.0.16          TCP         69 57929
   61 7.356481   192.168.0.16          192.168.0.13          TCP         83 5001 →

▷ Frame 56: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interf
▷ Ethernet II, Src: SamsungE_ae:45:d3 (f0:25:b7:ae:45:d3), Dst: IntelCor_35:1c:54
▷ Internet Protocol Version 4, Src: 192.168.0.13, Dst: 192.168.0.16
▷ Transmission Control Protocol, Src Port: 57929 (57929), Dst Port: 5001 (5001),
◢ Data (39 bytes)
     Data: 32373339393736303936373838353334363336323534336...
     Text: 273997609678853463625436383194360120329
     [Length: 39]

0000  e4 f8 9c 35 1c 54 f0 25  b7 ae 45 d3 08 00 45 00   ...5.T.% ..E...E.
0010  00 5b 67 5c 40 00 40 06  51 d3 c0 a8 00 0d c0 a8   .[g\@.@. Q.......
0020  00 10 e2 49 13 89 22 66  11 2d 0f b0 1b 99 80 18   ...I.."f .-......
0030  02 ad a4 b1 00 00 01 01  08 0a 00 01 77 95 06 05   ........ ....w...
0040  54 90 32 37 33 39 39 37  36 30 39 36 37 38 38 35   T.273997 60967885
0050  33 34 36 33 36 32 35 34  33 36 33 38 33 31 39 34   34636254 36383194
0060  33 36 30 31 32 30 33 32  39                        36012032 9
```

Figure 33: Sending Diffie-Hellman parameter TA via Wi-Fi.

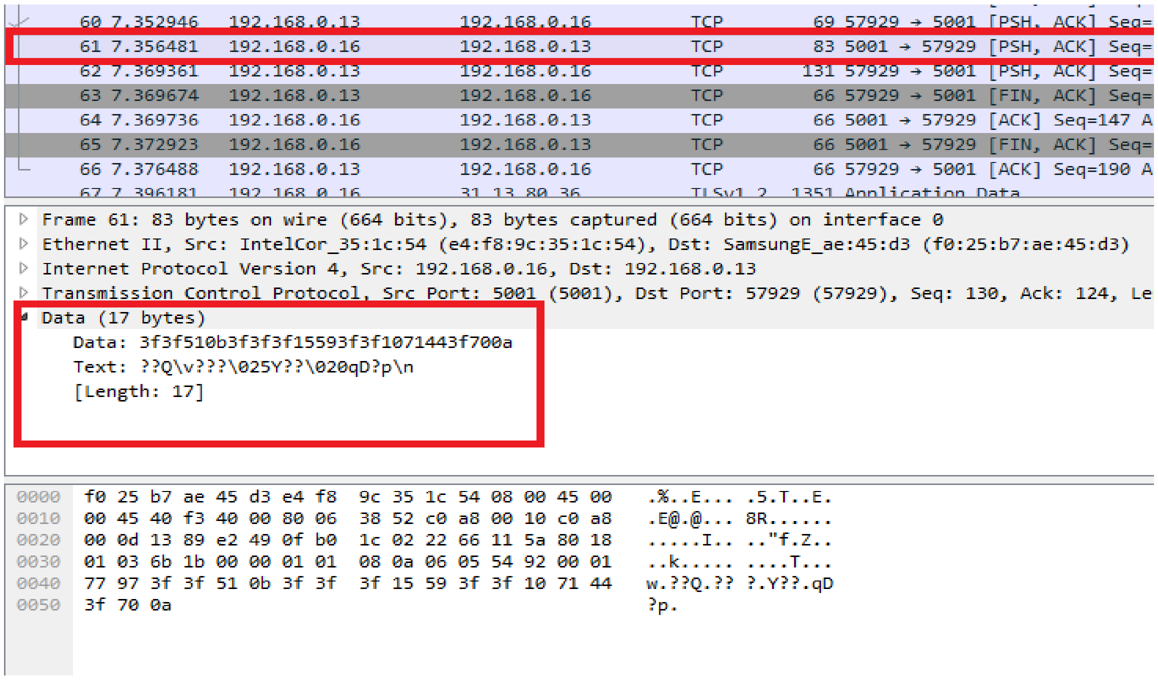
After receiving each Diffie-Hellman parameter the desktop application send an acknowledgment packet to ensure that the parameter is received on the other end. The following figure show the acknowledgment packet sent from desktop application to the mobile application.



```
50 7.230966   192.168.0.13   192.168.0.16   TCP   66 57929 → 5001 [ACK] Seq=1 Ack=1 Win=87680 Len=0 TSval=
51 7.301995   192.168.0.13   192.168.0.16   TCP  105 57929 → 5001 [PSH, ACK] Seq=1 Ack=1 Win=87680 Len=39
52 7.309012   192.168.0.16   192.168.0.13   TCP   78 5001 → 57929 [PSH, ACK] Seq=1 Ack=40 Win=66560 Len=12
53 7.311749   192.168.0.13   192.168.0.16   TCP   66 57929 → 5001 [ACK] Seq=40 Ack=13 Win=87680 Len=0 TSva
54 7.319051   192.168.0.13   192.168.0.16   TCP  105 57929 → 5001 [PSH, ACK] Seq=40 Ack=13 Win=87680 Len=3
55 7.325973   192.168.0.16   192.168.0.13   TCP   78 5001 → 57929 [PSH, ACK] Seq=13 Ack=79 Win=66560 Len=1
56 7.329349   192.168.0.13   192.168.0.16   TCP  105 57929 → 5001 [PSH, ACK] Seq=79 Ack=25 Win=87680 Len=3
57 7.336728   192.168.0.16   192.168.0.13   TCP  106 5001 → 57929 [PSH, ACK] Seq=25 Ack=118 Win=66304 Len=
58 7.339814   192.168.0.13   192.168.0.16   TCP   69 57929 → 5001 [PSH, ACK] Seq=118 Ack=65 Win=87680 Len=
59 7.345169   192.168.0.16   192.168.0.13   TCP  131 5001 → 57929 [PSH, ACK] Seq=65 Ack=121 Win=66304 Len=
60 7.352946   192.168.0.13   192.168.0.16   TCP   69 57929 → 5001 [PSH, ACK] Seq=121 Ack=130 Win=87680 Len

Frame 52: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0
Ethernet II, Src: IntelCor_35:1c:54 (e4:f8:9c:35:1c:54), Dst: SamsungE_ae:45:d3 (f0:25:b7:ae:45:d3)
Internet Protocol Version 4, Src: 192.168.0.16, Dst: 192.168.0.13
Transmission Control Protocol, Src Port: 5001 (5001), Dst Port: 57929 (57929), Seq: 1, Ack: 40, Len: 12
Data (12 bytes)
   Data: 41636b6e6f776c656467650a
   Text: Acknowledge\n
   [Length: 12]

0000  f0 25 b7 ae 45 d3 e4 f8  9c 35 1c 54 08 00 45 00   .%..E... .5.T..E.
0010  00 40 40 ef 40 00 80 06  38 5b c0 a8 00 10 c0 a8   .@@.@... 8[......
0020  00 0d 13 89 e2 49 0f b0  1b 81 22 66 11 06 80 18   .....I.. .."f....
0030  01 04 7b 7f 00 00 01 01  08 0a 06 05 54 8f 00 01   ..{..... ....T...
0040  77 92 41 63 6b 6e 6f 77  6c 65 64 67 65 0a         w.Acknow ledge.
```

Figure 34: Acknowledgment packet.

Once both the devices have received the Diffie-Hellman parameters they compute the secret key. The Desktop application sends a message digest of random string and the secret key using SHA-256 so that the mobile application can verify that the secret key is authentic. The following figure shows the message digest sent to the mobile application.



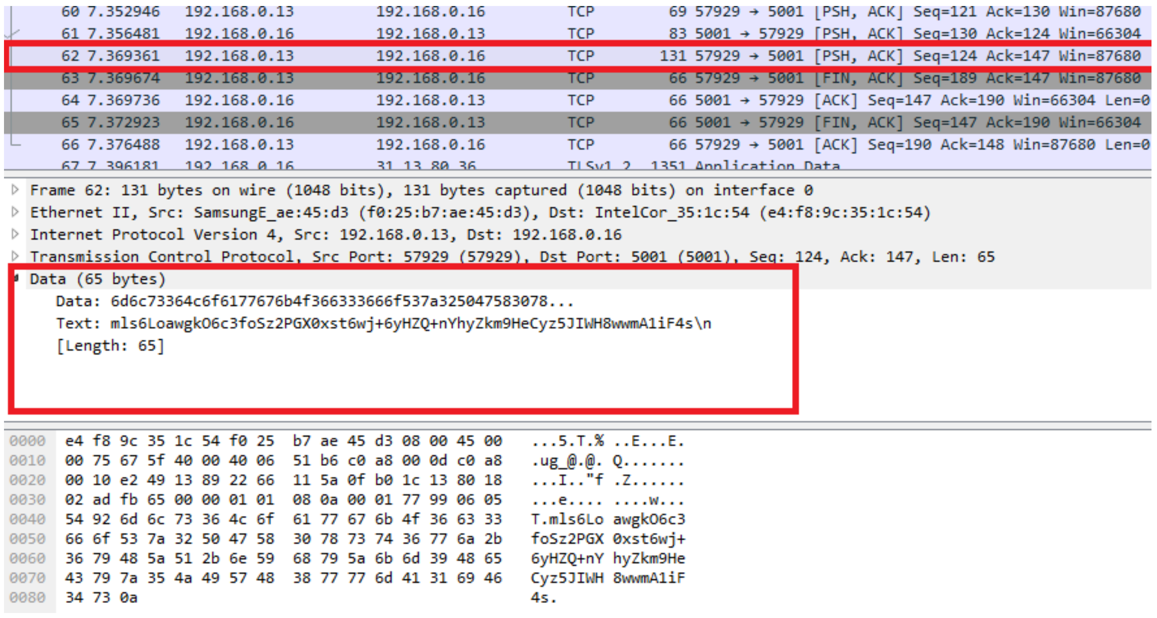
```
60 7.352946   192.168.0.13      192.168.0.16      TCP       69 57929 → 5001 [PSH, ACK] Seq=
61 7.356481   192.168.0.16      192.168.0.13      TCP       83 5001 → 57929 [PSH, ACK] Seq=
62 7.369361   192.168.0.13      192.168.0.16      TCP      131 57929 → 5001 [PSH, ACK] Seq=
63 7.369674   192.168.0.13      192.168.0.16      TCP       66 57929 → 5001 [FIN, ACK] Seq=
64 7.369736   192.168.0.16      192.168.0.13      TCP       66 5001 → 57929 [ACK] Seq=147 A
65 7.372923   192.168.0.16      192.168.0.13      TCP       66 5001 → 57929 [FIN, ACK] Seq=
66 7.376488   192.168.0.13      192.168.0.16      TCP       66 57929 → 5001 [ACK] Seq=190 A
67 7.396181   192.168.0.16      31.13.80.36       TLSv1.2 1351 Application Data
```

```
▷ Frame 61: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
▷ Ethernet II, Src: IntelCor_35:1c:54 (e4:f8:9c:35:1c:54), Dst: SamsungE_ae:45:d3 (f0:25:b7:ae:45:d3)
▷ Internet Protocol Version 4, Src: 192.168.0.16, Dst: 192.168.0.13
▷ Transmission Control Protocol, Src Port: 5001 (5001), Dst Port: 57929 (57929), Seq: 130, Ack: 124, Le
◢ Data (17 bytes)
    Data: 3f3f510b3f3f3f15593f3f1071443f700a
    Text: ??Q\v???\025Y??\020qD?p\n
    [Length: 17]
```

```
0000  f0 25 b7 ae 45 d3 e4 f8  9c 35 1c 54 08 00 45 00   .%..E... .5.T..E.
0010  00 45 40 f3 40 00 80 06  38 52 c0 a8 00 10 c0 a8   .E@.@... 8R......
0020  00 0d 13 89 e2 49 0f b0  1c 02 22 66 11 5a 80 18   .....I.. .."f.Z..
0030  01 03 6b 1b 00 00 01 01  08 0a 06 05 54 92 00 01   ..k..... ....T...
0040  77 97 3f 3f 51 0b 3f 3f  3f 15 59 3f 3f 10 71 44   w.??Q.?? ?.Y??.qD
0050  3f 70 0a                                           ?p.
```

Figure 35: Sending message digest.

After the secret key is verified the mobile application encrypts the credentials and transferred it to the desktop application. The following figure shows that the credentials are sent in encrypted form.



```
60 7.352946   192.168.0.13   192.168.0.16   TCP       69 57929 → 5001 [PSH, ACK] Seq=121 Ack=130 Win=87680
61 7.356481   192.168.0.16   192.168.0.13   TCP       83 5001 → 57929 [PSH, ACK] Seq=130 Ack=124 Win=66304
62 7.369361   192.168.0.13   192.168.0.16   TCP      131 57929 → 5001 [PSH, ACK] Seq=124 Ack=147 Win=87680
63 7.369674   192.168.0.13   192.168.0.16   TCP       66 57929 → 5001 [FIN, ACK] Seq=189 Ack=147 Win=87680
64 7.369736   192.168.0.16   192.168.0.13   TCP       66 5001 → 57929 [ACK] Seq=147 Ack=190 Win=66304 Len=0
65 7.372923   192.168.0.16   192.168.0.13   TCP       66 5001 → 57929 [FIN, ACK] Seq=147 Ack=190 Win=66304
66 7.376488   192.168.0.13   192.168.0.16   TCP       66 57929 → 5001 [ACK] Seq=190 Ack=148 Win=87680 Len=0
67 7.396181   192.168.0.16   31.13.80.36    TLSv1.2 1351 Application Data
```

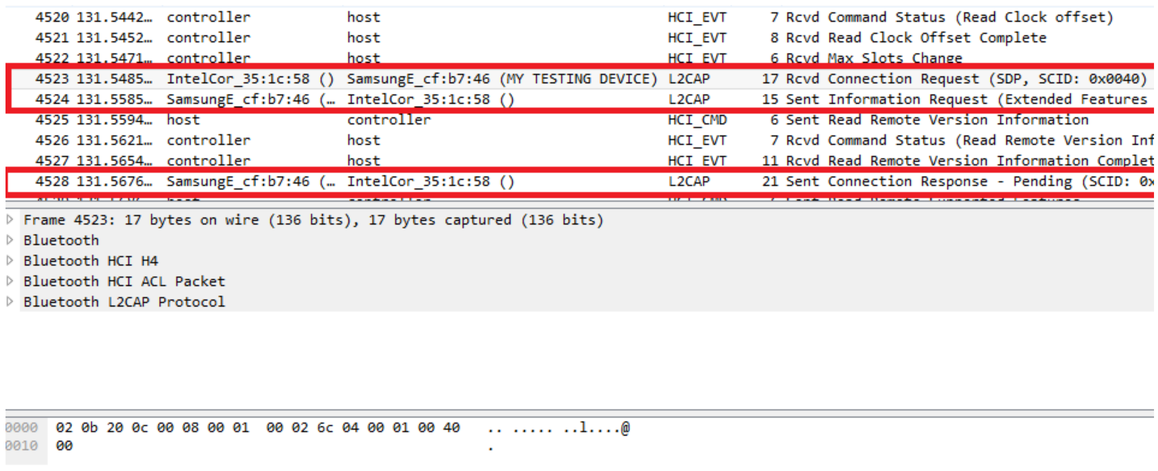
```
▷ Frame 62: 131 bytes on wire (1048 bits), 131 bytes captured (1048 bits) on interface 0
▷ Ethernet II, Src: SamsungE_ae:45:d3 (f0:25:b7:ae:45:d3), Dst: IntelCor_35:1c:54 (e4:f8:9c:35:1c:54)
▷ Internet Protocol Version 4, Src: 192.168.0.13, Dst: 192.168.0.16
▷ Transmission Control Protocol, Src Port: 57929 (57929), Dst Port: 5001 (5001), Seq: 124, Ack: 147, Len: 65
◢ Data (65 bytes)
    Data: 6d6c73364c6f6177676b4f366333666f537a325047583078...
    Text: mls6LoawgkO6c3foSz2PGX0xst6wj+6yHZQ+nYhyZkm9HeCyz5JIWH8wwmA1iF4s\n
    [Length: 65]
```

```
0000  e4 f8 9c 35 1c 54 f0 25  b7 ae 45 d3 08 00 45 00   ...5.T.% ..E...E.
0010  00 75 67 5f 40 00 40 06  51 b6 c0 a8 00 0d c0 a8   .ug_@.@. Q.......
0020  00 10 e2 49 13 89 22 66  11 5a 0f b0 1c 13 80 18   ...I.."f .Z......
0030  02 ad fb 65 00 00 01 01  08 0a 00 01 77 99 06 05   ...e.... ....w...
0040  54 92 6d 6c 73 36 4c 6f  61 77 67 6b 4f 36 63 33   T.mls6Lo awgkO6c3
0050  66 6f 53 7a 32 50 47 58  30 78 73 74 36 77 6a 2b   foSz2PGX 0xst6wj+
0060  36 79 48 5a 51 2b 6e 59  68 79 5a 6b 6d 39 48 65   6yHZQ+nY hyZkm9He
0070  43 79 7a 35 4a 49 57 48  38 77 77 6d 41 31 69 46   Cyz5JIWH 8wwmA1iF
0080  34 73 0a                                           4s.
```

Figure 36: Sending Encrypted Credentials.

6.2.2 Testing over the Bluetooth medium.

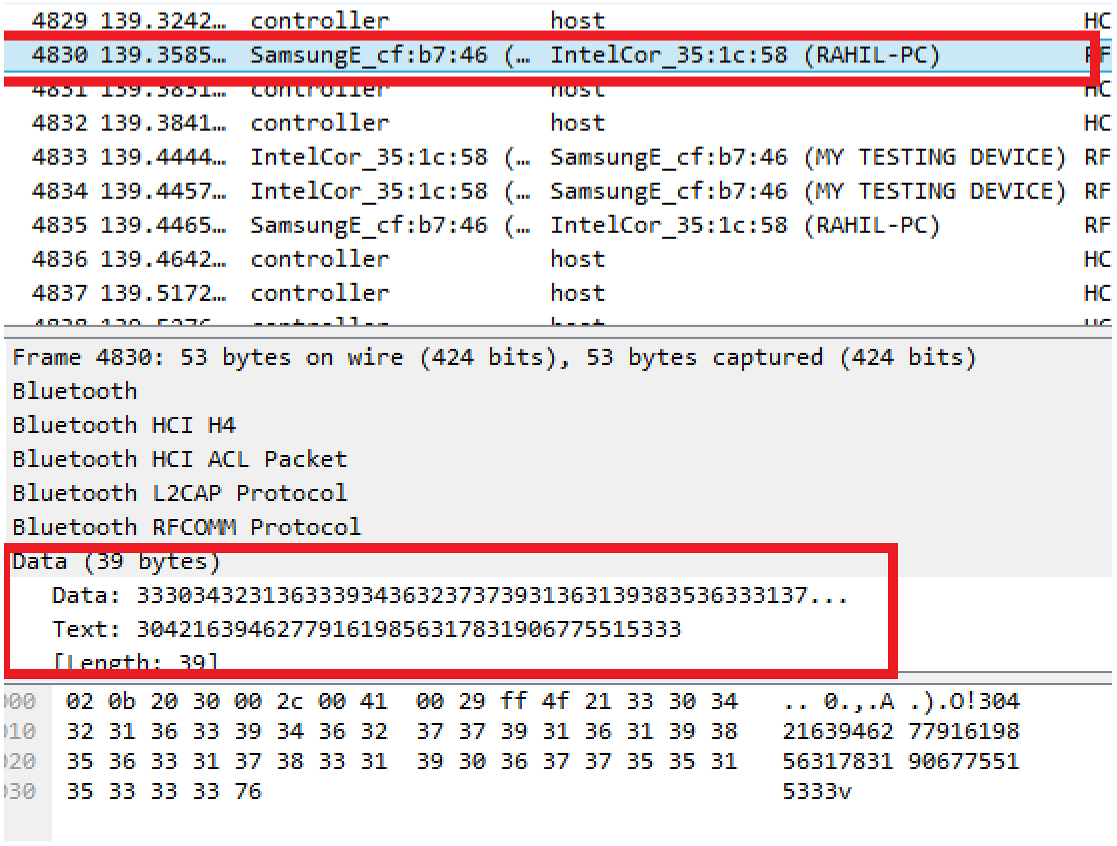
To capture the packets on a Bluetooth medium we have enabled the ‘Bluetooth HCI snoop logging’ on device. The device captures all the packets that are sent and received over the Bluetooth. The log file is maintained under the location “\Phone\Android\data\btsnoop_hci.log”. We have import the log file to the Wireshark to analyse the captured packets. The following figure shows the packet captured during Bluetooth transfer.



```
4520 131.5442…  controller          host                              HCI_EVT     7 Rcvd Command Status (Read Clock offset)
4521 131.5452…  controller          host                              HCI_EVT     8 Rcvd Read Clock Offset Complete
4522 131.5471…  controller          host                              HCI_EVT     6 Rcvd Max Slots Change
4523 131.5485…  IntelCor_35:1c:58 () SamsungE_cf:b7:46 (MY TESTING DEVICE) L2CAP  17 Rcvd Connection Request (SDP, SCID: 0x0040)
4524 131.5585…  SamsungE_cf:b7:46 (… IntelCor_35:1c:58 ()              L2CAP      15 Sent Information Request (Extended Features
4525 131.5594…  host                controller                        HCI_CMD     6 Sent Read Remote Version Information
4526 131.5621…  controller          host                              HCI_EVT     7 Rcvd Command Status (Read Remote Version Inf
4527 131.5654…  controller          host                              HCI_EVT    11 Rcvd Read Remote Version Information Complet
4528 131.5676…  SamsungE_cf:b7:46 (… IntelCor_35:1c:58 ()              L2CAP      21 Sent Connection Response - Pending (SCID: 0x

▷ Frame 4523: 17 bytes on wire (136 bits), 17 bytes captured (136 bits)
▷ Bluetooth
▷ Bluetooth HCI H4
▷ Bluetooth HCI ACL Packet
▷ Bluetooth L2CAP Protocol

0000  02 0b 20 0c 00 08 00 01  00 02 6c 04 00 01 00 40   .. ..... ..l....@
0010  00                                                 .
```

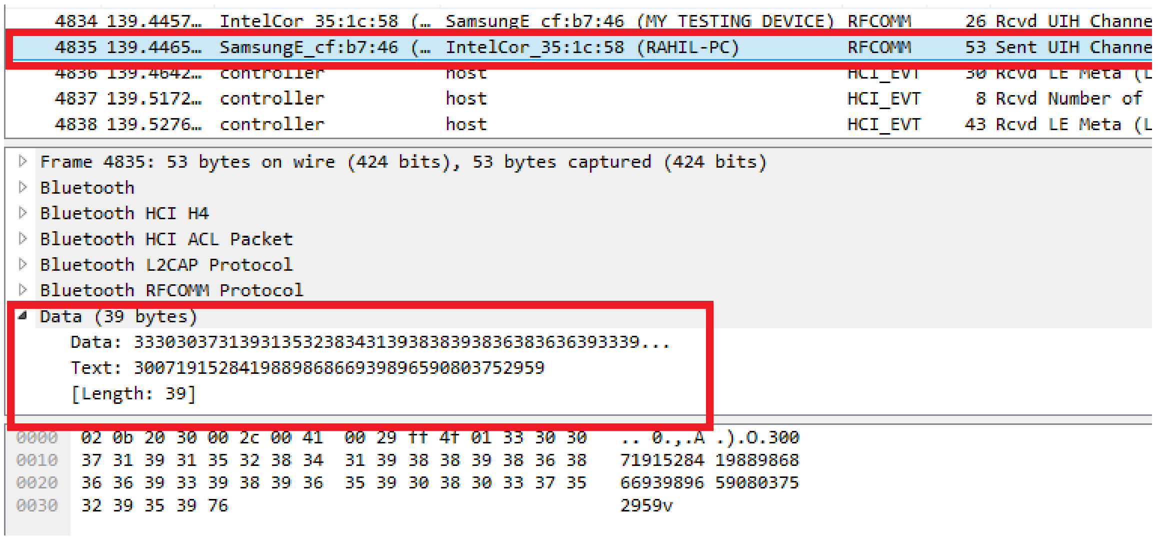
Figure 37: Packets captured during Bluetooth transfer.

As shown in the above figure the desktop application initiates the connection and once the mobile application accepts the connection both the devices transfers the Diffie-Hellman parameters. All the data is transferred in the plain text except for the credentials which is sent in an encrypted form as shown in the following figures.



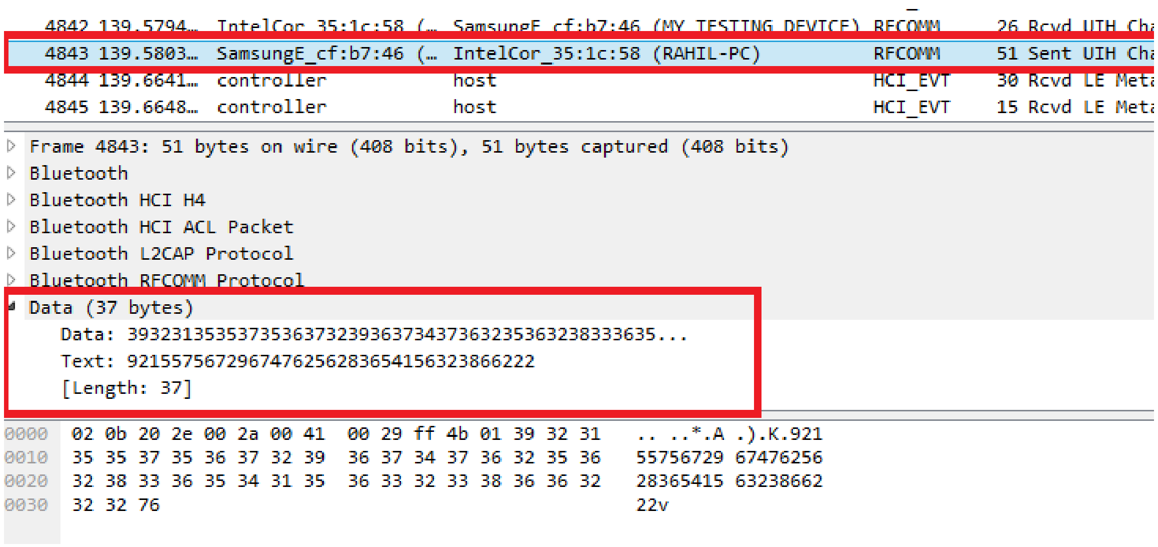
```
4829 139.3242…  controller            host                              HC
4830 139.3585…  SamsungE_cf:b7:46 (…  IntelCor_35:1c:58 (RAHIL-PC)      RF
4831 139.3831…  controller            host                              HC
4832 139.3841…  controller            host                              HC
4833 139.4444…  IntelCor_35:1c:58 (…  SamsungE_cf:b7:46 (MY TESTING DEVICE) RF
4834 139.4457…  IntelCor_35:1c:58 (…  SamsungE_cf:b7:46 (MY TESTING DEVICE) RF
4835 139.4465…  SamsungE_cf:b7:46 (…  IntelCor_35:1c:58 (RAHIL-PC)      RF
4836 139.4642…  controller            host                              HC
4837 139.5172…  controller            host                              HC

Frame 4830: 53 bytes on wire (424 bits), 53 bytes captured (424 bits)
Bluetooth
Bluetooth HCI H4
Bluetooth HCI ACL Packet
Bluetooth L2CAP Protocol
Bluetooth RFCOMM Protocol
Data (39 bytes)
   Data: 333034323136333934363237373931363139383536333137...
   Text: 304216394627791619856317831906775515333
   [Length: 39]

000  02 0b 20 30 00 2c 00 41  00 29 ff 4f 21 33 30 34   .. 0.,.A .).O!304
010  32 31 36 33 39 34 36 32  37 37 39 31 36 31 39 38   21639462 77916198
020  35 36 33 31 37 38 33 31  39 30 36 37 37 35 35 31   56317831 90677551
030  35 33 33 33 76                                     5333v
```

Figure 38: Sending Diffie-Hellman parameter P via Bluetooth



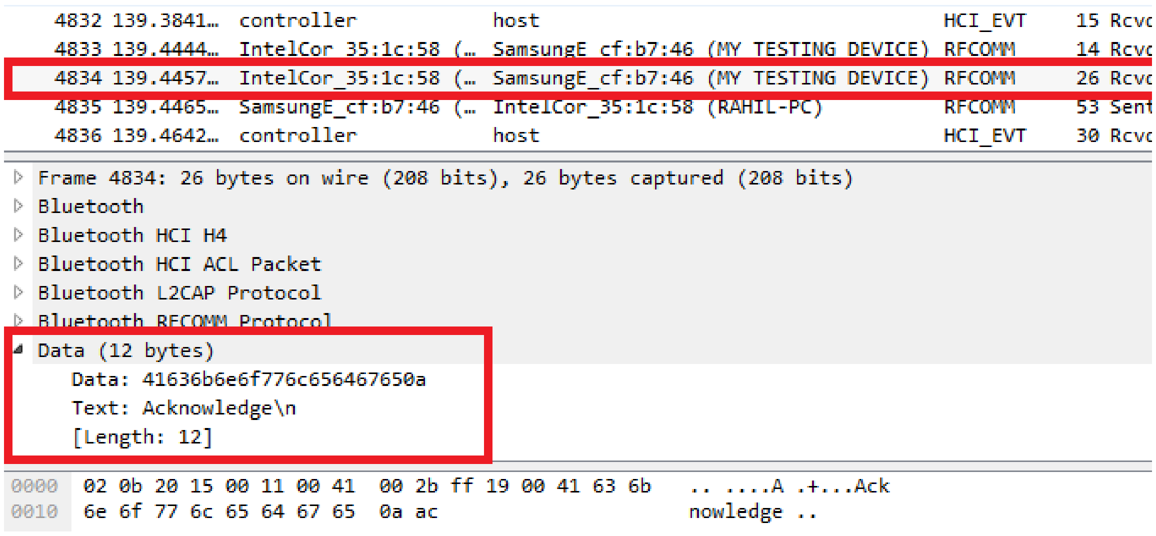
```
4834 139.4457…  IntelCor_35:1c:58 (…  SamsungE_cf:b7:46 (MY TESTING DEVICE) RFCOMM   26 Rcvd UIH Channe
4835 139.4465…  SamsungE_cf:b7:46 (…  IntelCor_35:1c:58 (RAHIL-PC)          RFCOMM   53 Sent UIH Channe
4836 139.4642…  controller            host                                  HCI_EVT  30 Rcvd LE Meta (L
4837 139.5172…  controller            host                                  HCI_EVT   8 Rcvd Number of
4838 139.5276…  controller            host                                  HCI_EVT  43 Rcvd LE Meta (L

▷ Frame 4835: 53 bytes on wire (424 bits), 53 bytes captured (424 bits)
▷ Bluetooth
▷ Bluetooth HCI H4
▷ Bluetooth HCI ACL Packet
▷ Bluetooth L2CAP Protocol
▷ Bluetooth RFCOMM Protocol
◢ Data (39 bytes)
     Data: 333030373139313532383431393838393836383636393339...
     Text: 300719152841988986866939896590803752959
     [Length: 39]

0000  02 0b 20 30 00 2c 00 41  00 29 ff 4f 01 33 30 30   .. 0.,.A .).O.300
0010  37 31 39 31 35 32 38 34  31 39 38 38 39 38 36 38   71915284 19889868
0020  36 36 39 33 39 38 39 36  35 39 30 38 30 33 37 35   66939896 59080375
0030  32 39 35 39 76                                     2959v
```

Figure 39: Sending Diffie-Hellman parameter G via Bluetooth.



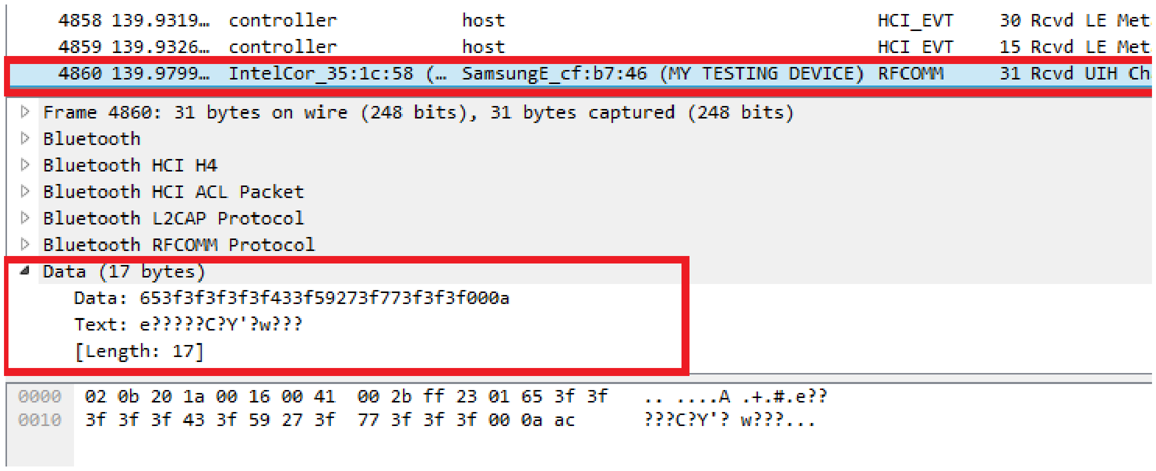
```
4842 139.5794… IntelCor_35:1c:58 (… SamsungE_cf:b7:46 (MY TESTING DEVICE) RFCOMM    26 Rcvd UIH Cha
4843 139.5803… SamsungE_cf:b7:46 (… IntelCor_35:1c:58 (RAHIL-PC)           RFCOMM    51 Sent UIH Cha
4844 139.6641… controller          host                                   HCI_EVT   30 Rcvd LE Meta
4845 139.6648… controller          host                                   HCI_EVT   15 Rcvd LE Meta

Frame 4843: 51 bytes on wire (408 bits), 51 bytes captured (408 bits)
Bluetooth
Bluetooth HCI H4
Bluetooth HCI ACL Packet
Bluetooth L2CAP Protocol
Bluetooth RFCOMM Protocol
Data (37 bytes)
   Data: 393231353537353637323936373437363235363238333635...
   Text: 9215557567296747625628365415632386622
   [Length: 37]

0000  02 0b 20 2e 00 2a 00 41  00 29 ff 4b 01 39 32 31   .. ..*.A .).K.921
0010  35 35 37 35 36 37 32 39  36 37 34 37 36 32 35 36   55756729 67476256
0020  32 38 33 36 35 34 31 35  36 33 32 33 38 36 36 32   28365415 63238662
0030  32 32 76                                           22v
```

Figure 40: Sending Diffie-Hellman parameter TA via Bluetooth



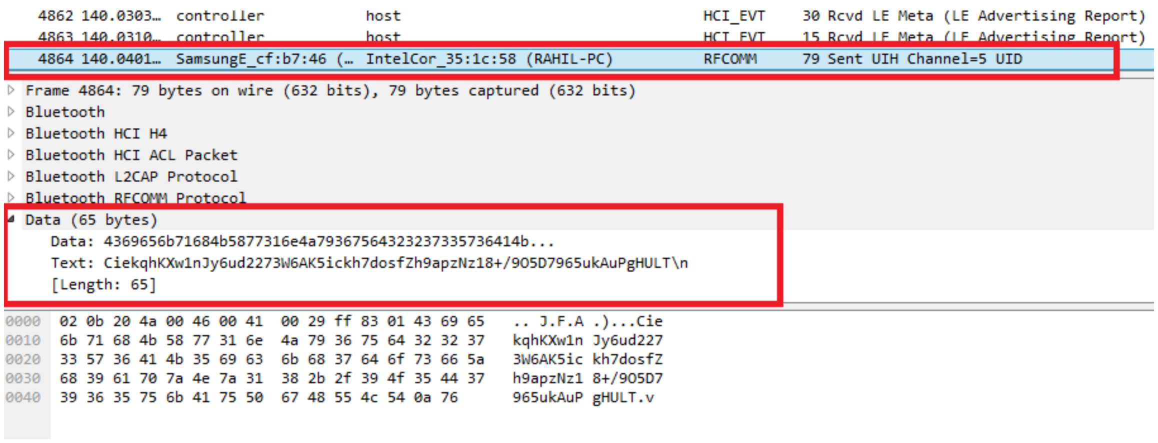
```
4832 139.3841… controller          host                                   HCI_EVT   15 Rcvd
4833 139.4444… IntelCor_35:1c:58 (… SamsungE_cf:b7:46 (MY TESTING DEVICE) RFCOMM    14 Rcvd
4834 139.4457… IntelCor_35:1c:58 (… SamsungE_cf:b7:46 (MY TESTING DEVICE) RFCOMM    26 Rcvd
4835 139.4465… SamsungE_cf:b7:46 (… IntelCor_35:1c:58 (RAHIL-PC)           RFCOMM    53 Sent
4836 139.4642… controller          host                                   HCI_EVT   30 Rcvd

Frame 4834: 26 bytes on wire (208 bits), 26 bytes captured (208 bits)
Bluetooth
Bluetooth HCI H4
Bluetooth HCI ACL Packet
Bluetooth L2CAP Protocol
Bluetooth RFCOMM Protocol
Data (12 bytes)
   Data: 41636b6e6f776c656467650a
   Text: Acknowledge\n
   [Length: 12]

0000  02 0b 20 15 00 11 00 41  00 2b ff 19 00 41 63 6b   .. ....A .+...Ack
0010  6e 6f 77 6c 65 64 67 65  0a ac                     nowledge ..
```

Figure 41: Bluetooth Acknowledgment packet.



```
4858 139.9319…  controller          host                               HCI_EVT    30 Rcvd LE Met
4859 139.9326…  controller          host                               HCI_EVT    15 Rcvd LE Met
4860 139.9799…  IntelCor_35:1c:58 (… SamsungE_cf:b7:46 (MY TESTING DEVICE) RFCOMM     31 Rcvd UIH Ch

▷ Frame 4860: 31 bytes on wire (248 bits), 31 bytes captured (248 bits)
▷ Bluetooth
▷ Bluetooth HCI H4
▷ Bluetooth HCI ACL Packet
▷ Bluetooth L2CAP Protocol
▷ Bluetooth RFCOMM Protocol
◢ Data (17 bytes)
     Data: 653f3f3f3f3f433f59273f773f3f3f000a
     Text: e?????C?Y'?w???
     [Length: 17]

0000  02 0b 20 1a 00 16 00 41  00 2b ff 23 01 65 3f 3f   .. ....A .+.#.e??
0010  3f 3f 3f 43 3f 59 27 3f  77 3f 3f 3f 00 0a ac      ???C?Y'? w???...
```

Figure 42: Sending message digest via Bluetooth.



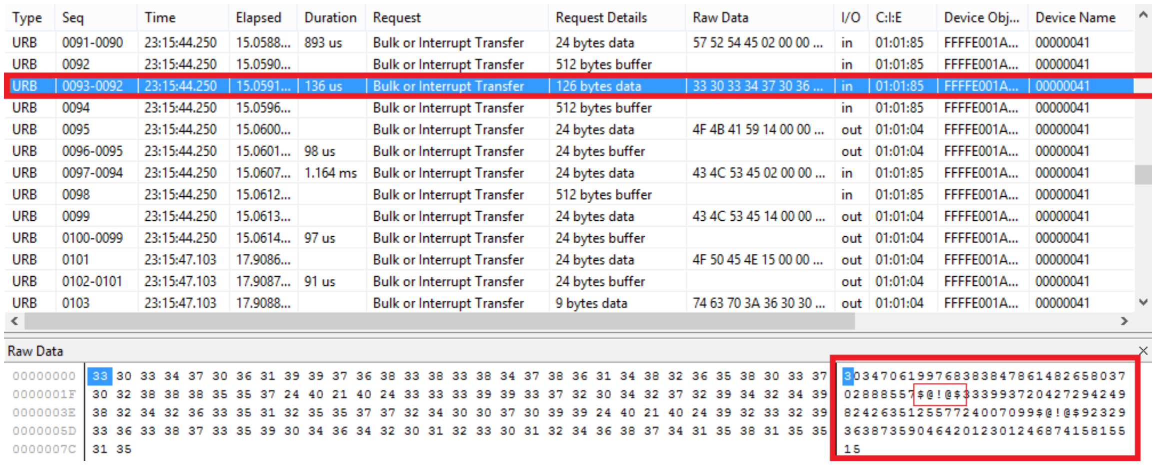
```
4862 140.0303…  controller          host                       HCI_EVT    30 Rcvd LE Meta (LE Advertising Report)
4863 140.0310…  controller          host                       HCI_EVT    15 Rcvd LE Meta (LE Advertising Report)
4864 140.0401…  SamsungE_cf:b7:46 (… IntelCor_35:1c:58 (RAHIL-PC) RFCOMM     79 Sent UIH Channel=5 UID

▷ Frame 4864: 79 bytes on wire (632 bits), 79 bytes captured (632 bits)
▷ Bluetooth
▷ Bluetooth HCI H4
▷ Bluetooth HCI ACL Packet
▷ Bluetooth L2CAP Protocol
▷ Bluetooth RFCOMM Protocol
◢ Data (65 bytes)
    Data: 4369656b71684b5877316e4a7936756432323737335736414b...
    Text: CiekqhKXw1nJy6ud2273W6AK5ickh7dosfZh9apzNz18+/905D7965ukAuPgHULT\n
    [Length: 65]

0000  02 0b 20 4a 00 46 00 41  00 29 ff 83 01 43 69 65   .. J.F.A .)...Cie
0010  6b 71 68 4b 58 77 31 6e  4a 79 36 75 64 32 32 37   kqhKXw1n Jy6ud227
0020  33 57 36 41 4b 35 69 63  6b 68 37 64 6f 73 66 5a   3W6AK5ic kh7dosfZ
0030  68 39 61 70 7a 4e 7a 31  38 2b 2f 39 4f 35 44 37   h9apzNz1 8+/9O5D7
0040  39 36 35 75 6b 41 75 50  67 48 55 4c 54 0a 76      965ukAuP gHULT.v
```

Figure 43: Sending encrypted credentials via Bluetooth.

6.2.3 Testing over the NFC medium

The NFC packets are transferred from the mobile application to the desktop application over the USB port. To test the NFC data we have attached another NFC enabled device (android tablet) and we have written an application that takes the NFC data and send it to the USB port. The data that should be sent over the USB port from the mobile application are Diffie-Hellman parameters and the credentials. The Diffie-Hellman parameters are sent as a plain text while the credentials are sent in encrypted form. We have used

USBlyzer version 2.2 to capture the traffic on the USB port as shown in the following figures.



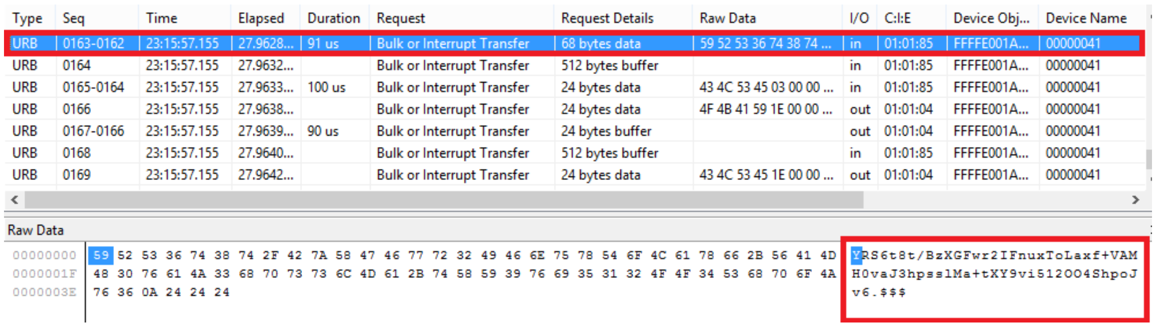
Type	Seq	Time	Elapsed	Duration	Request	Request Details	Raw Data	I/O	C:I:E	Device Obj...	Device Name
URB	0091-0090	23:15:44.250	15.0588...	893 us	Bulk or Interrupt Transfer	24 bytes data	57 52 54 45 02 00 00 ...	in	01:01:85	FFFFE001A...	00000041
URB	0092	23:15:44.250	15.0590...		Bulk or Interrupt Transfer	512 bytes buffer		in	01:01:85	FFFFE001A...	00000041
URB	0093-0092	23:15:44.250	15.0591...	136 us	Bulk or Interrupt Transfer	126 bytes data	33 30 33 34 37 30 36 ...	in	01:01:85	FFFFE001A...	00000041
URB	0094	23:15:44.250	15.0596...		Bulk or Interrupt Transfer	512 bytes buffer		in	01:01:85	FFFFE001A...	00000041
URB	0095	23:15:44.250	15.0600...		Bulk or Interrupt Transfer	24 bytes data	4F 4B 41 59 14 00 00 ...	out	01:01:04	FFFFE001A...	00000041
URB	0096-0095	23:15:44.250	15.0601...	98 us	Bulk or Interrupt Transfer	24 bytes buffer		out	01:01:04	FFFFE001A...	00000041
URB	0097-0094	23:15:44.250	15.0607...	1.164 ms	Bulk or Interrupt Transfer	24 bytes data	43 4C 53 45 02 00 00 ...	in	01:01:85	FFFFE001A...	00000041
URB	0098	23:15:44.250	15.0612...		Bulk or Interrupt Transfer	512 bytes buffer		in	01:01:85	FFFFE001A...	00000041
URB	0099	23:15:44.250	15.0613...		Bulk or Interrupt Transfer	24 bytes data	43 4C 53 45 14 00 00 ...	out	01:01:04	FFFFE001A...	00000041
URB	0100-0099	23:15:44.250	15.0614...	97 us	Bulk or Interrupt Transfer	24 bytes buffer		out	01:01:04	FFFFE001A...	00000041
URB	0101	23:15:47.103	17.9086...		Bulk or Interrupt Transfer	24 bytes data	4F 50 45 4E 15 00 00 ...	out	01:01:04	FFFFE001A...	00000041
URB	0102-0101	23:15:47.103	17.9087...	91 us	Bulk or Interrupt Transfer	24 bytes buffer		out	01:01:04	FFFFE001A...	00000041
URB	0103	23:15:47.103	17.9088...		Bulk or Interrupt Transfer	9 bytes data	74 63 70 3A 36 30 30 ...	out	01:01:04	FFFFE001A...	00000041

Raw Data

```
00000000  33 30 33 34 37 30 36 31 39 39 37 36 38 33 38 33 38 34 37 38 36 31 34 38 32 36 35 38 30 33 37  3034706199768383847861482658037
0000001F  30 32 38 38 38 35 35 37 24 40 21 40 24 33 33 33 39 39 33 37 32 30 34 32 37 32 39 34 32 34 39  02888557$@!@$3339937204272942 49
0000003E  38 32 34 32 36 33 35 31 32 35 35 37 37 32 34 30 30 37 30 39 39 24 40 21 40 24 39 32 33 32 39  824263512557724007099$@!@$92329
0000005D  33 36 33 38 37 33 35 39 30 34 36 34 32 30 31 32 33 30 31 32 34 36 38 37 34 31 35 38 31 35 35  3638735904642012301246874158155
0000007C  31 35                                                                                        15
```

Figure 44: Sending Diffie-Hellman parameters via NFC

The value of ‘P’, ‘G’ and ‘TA’ and concatenated using a predefined string “$@!@$” and is sent to the desktop application. Once the secret key is generated and verified the mobile application send the credentials to the desktop application in encrypted form. The following figure shows the credential packet sent over the NFC.



Type	Seq	Time	Elapsed	Duration	Request	Request Details	Raw Data	I/O	C:I:E	Device Obj...	Device Name
URB	0163-0162	23:15:57.155	27.9628...	91 us	Bulk or Interrupt Transfer	68 bytes data	59 52 53 36 74 38 74 ...	in	01:01:85	FFFFE001A...	00000041
URB	0164	23:15:57.155	27.9632...		Bulk or Interrupt Transfer	512 bytes buffer		in	01:01:85	FFFFE001A...	00000041
URB	0165-0164	23:15:57.155	27.9633...	100 us	Bulk or Interrupt Transfer	24 bytes data	43 4C 53 45 03 00 00 ...	in	01:01:85	FFFFE001A...	00000041
URB	0166	23:15:57.155	27.9638...		Bulk or Interrupt Transfer	24 bytes data	4F 4B 41 59 1E 00 00 ...	out	01:01:04	FFFFE001A...	00000041
URB	0167-0166	23:15:57.155	27.9639...	90 us	Bulk or Interrupt Transfer	24 bytes buffer		out	01:01:04	FFFFE001A...	00000041
URB	0168	23:15:57.155	27.9640...		Bulk or Interrupt Transfer	512 bytes buffer		in	01:01:85	FFFFE001A...	00000041
URB	0169	23:15:57.155	27.9642...		Bulk or Interrupt Transfer	24 bytes data	43 4C 53 45 1E 00 00 ...	out	01:01:04	FFFFE001A...	00000041

Raw Data

```
00000000  59 52 53 36 74 38 74 2F 42 7A 58 47 46 77 72 32 49 46 6E 75 78 54 6F 4C 61 78 66 2B 56 41 4D  YRS6t8t/BzXGFwr2IFnuxToLaxf+VAM
0000001F  48 30 76 61 4A 33 68 70 73 73 6C 4D 61 2B 74 58 59 39 76 69 35 31 32 4F 4F 34 53 68 70 6F 4A  H0vaJ3hpsslMa+tXY9vi512OO4ShpoJ
0000003E  76 36 0A 24 24 24                                                                            v6.$$$
```

Figure 45: Sending credentials via NFC

6.3 SUMMARY OF THE EXPERIMENTAL RESULTS

The experimental results of the network traffic from the Wireshark and the USBlyzer shows that, all the Diffie-Hellman parameters ‘P’, ‘G’ and ‘TA’ are sent in plain text. The secret Integer and the secret key is never transmitted to one another. Once the key is computed on both ends, the secret key is verified using one-way hash function. If the key is compromised using the MITMA the verification fails and the credentials are not transferred to the desktop application. If the key is verified, the credentials are sent in the encrypted form which is then decrypted on the other end.

Chapter 7 CONCLUSION AND FUTURE WORK

Smart phones have become part of our daily living and almost every person carries a smart phone with them all the time. Our proposed approach is an attempt to overcome the shoulder surfing and keylogging attacks using a smart phone. The motivation behind this proposed approach is to use the smart phone to authenticate users to different website that can prevent them from various attacks such as shoulder surfing and keylogging.

The proposed approach has three major phases: the connection phase, where the user can select any medium (Wi-Fi, Bluetooth, NFC) to transfer credentials; the key exchange phase, where both the devices compute the secret key using Diffie-Hellman key exchange; and the encryption phase, where the credentials are encrypted and transfer to the desktop application.

The proposed approach also uses QR code to verify the secret key. If any Trojan is install on the system that captures the screenshot of the QR code and sent it to the attacker, it will be of no use because the QR code is changed each time when the user selects a medium.

The Wireshark traffic analysis shows that all the Diffie-Hellman parameters, ‘P, ‘G’ and ‘TA’, are sent in plain text. The secret Integer and the secret key are never transferred to another device. The Diffie-Hellman key exchange algorithm is susceptible to MITMA; the proposed approach uses one-way hash function to verify that the secret key is not altered and is same on both ends. The credentials are transferred and sent in encrypted form using the verified secret key.

Based on the available resources and security situations user can select different medium to transfer credentials. NFC is the most secure way to transfer the credentials as the devices needs to be as close as 10mm to transfer the NFC data. In order to steal the data via NFC, the attacker needs to be physically present near the user. If the system does not have support for NFC, the user can select Bluetooth as the second option to transfer credentials. The user should always keep the Wi-Fi as a last option if they are using a public Wi-Fi or the Wi-Fi is not personal.

7.1 LIMITATIONS

We have captured and analyzed the traffic between the mobile and the desktop application to ensure that the credentials are sent in an encrypted form. Although the proposed approach uses standard AES-128 encryption to encrypt and decrypt data, the packets need to be tested against various attacks as the packets contains credentials that are private and sensitive. One of the other limitations is users are bound to use our desktop application to automatically login to the websites, which might not be feasible for all users. This limitation is due the web browsers, as most of the web browsers do not allow extensions to access system resources, such as Bluetooth or USB port. Our proposed approach require access to these resources to transfer credentials between the mobile and the desktop application.

7.2. Future Work

The proposed approach only focuses on preventing shoulder surfing and keylogging attacks and can be extended as a tool for password management. Nowadays, the growth of web accounts are increasing day by day and it is difficult to remember different passwords for different web accounts. It is recommended to use strong passwords with a combination of alpha numeric characters and special symbols which makes it difficult for attackers to crack the passwords. The proposed approach can be extended to add the functionality of a password manager. With the proposed approach, the user can manually enter their password for websites and can be managed on the device. These passwords cannot be accessed from any other device or website as they are stored locally on the device which makes it difficult to manage. If the device is lost, all the passwords are lost as well. In order to provide the functionality of the password manager, the database can be host to any hosting server and a simple user interface can be developed to manage the passwords.

Our proposed approach requires the users to manually enter their credentials in the mobile application. The proposed approach can be extended to add functionality to automatically capture user credentials from the desktop application. If the user creates a new account or changes the password of any existing account, the desktop application can suggest strong passwords. Additionally, it can automatically capture the new password set by the user and can transfer it to the mobile application. As a result, the user does not have to manually manage the passwords each time when they creates a new account or changes the password of an existing account.

User study can be done to understand different aspect of the proposed approach such as user-friendliness and efficiency in terms of time. Furthermore, this study might also help us to understand whether the user uses web accounts on their personal or public computers. All the results from the study will help us to understand the need and importance of the proposed approach in our daily living.

REFERENCES

[1] "Malware," Wikipedia, 12-Nov-2017. [Online]. Available: http://en.wikipedia.org/wiki/Malware. [Accessed: 13-Nov-2017].

[2] Raza, M., Iqbal, M., Sharif, M., & Haider, W. (2012). A survey of password attacks and comparative analysis on methods for secure authentication. World Applied Sciences Journal, 19(4), 439-444.

[3] Haque, M. A., & Imam, B. (2014). A New Graphical Password: Combination of Recall & Recognition Based Approach. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, 8(2), 320-324.

[4] Stallings, W., & Brown, L. (2012). Computer security. Principles and practice (2 nd ed). Edinburgh Gate: Pearson education limited.

[5] Suo, X., Zhu, Y., & Owen, G. S. (2005, December). Graphical passwords: A survey. In Computer security applications conference, 21st annual (pp. 10-pp). IEEE.

[6] Md. Asraful Haque, Babbar Imam, Nesar Ahmad, "2-Round Hybrid Password Scheme", International Journal of Computer Engineering and Technology (IJCET), Vol. 3, Issue 2, July-September (2012), page. 579-587.

[7] The Activity Lifecycle," Android Developers. [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle.html. [Accessed: 3-Sep-2017].

[8] Kessler, G. C. (2002). Passwords-Strengths and Weaknesses. Jan-1996. URL: http://www. garykessler.net/library/password.html.

[9] Pathak, N., Pawar, A., & Patil, B. (2015). A Survey on Keylogger: A Malicious Attack. International Jourcal of Advanced Research in Computer Engineering and Technology.

[10] Fujita, K., & Hirakawa, Y. (2008, September). A study of password authentication method against observing attacks. In Intelligent Systems and

Informatics, 2008. SISY 2008. 6th International Symposium on (pp. 1-6). IEEE.

[11] Canbek, G. (2005). Analysis, design and implementation of keyloggers and anti-keyloggers. Gazi University, Institute Of Science And Technology, M. Sc. thesis 103.

[12] Lynch, P. (2004). The Naked Employee: How Technology Is Compromising Workplace Privacy. Journal of Applied Management and Entrepreneurship, 9(2), 116.

[13] Feng, H., & Choong Wah, C. (2002). Private key generation from on-line handwritten signatures. Information Management & Computer Security, 10(4), 159-164.

[14] Kieseberg, P., Leithner, M., Mulazzani, M., Munroe, L., Schrittwieser, S., Sinha, M., & Weippl, E. (2010, November). QR code security. In Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia (pp. 430-435). ACM.

[15] "Diffie-Hellman Protocol," from Wolfram MathWorld. [Online]. Available: http://mathworld.wolfram.com/Diffie-HellmanProtocol.html. [Accessed: 14-Nov-2017].

[16] "Hash function," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Hash_function. [Accessed: 14-Nov-2017].

[17] "Movable Type Scripts," SHA-256 Cryptographic Hash Algorithm implemented in JavaScript | Movable Type Scripts. [Online]. Available: http://www.movable-type.co.uk/scripts/sha256.html. [Accessed: 14-Nov-2017].

[18] Burr, W. E. (2003). Selecting the advanced encryption standard. IEEE Security & Privacy, 99(2), 43-52.

[19] V. Lynch, "Difference Between SHA-1, SHA-2 and SHA-256?," [Online]. Available: https://www.thesslstore.com/blog/difference-sha-1-sha-2-sha-256-hash-algorithms/. [Accessed: 14-Nov-2017].

[20] Ebrahim, M., Khan, S., & Khalid, U. B. (2014). Symmetric algorithm survey: a comparative analysis. arXiv preprint arXiv:1405.0398.

[21] Massey, J. L. (1988). An introduction to contemporary cryptology. Proceedings of the IEEE, 76(5), 533-549.

[22] Advanced Encryption Standard - tutorialspoint.com." [Online]. Available: https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm. [Accessed: 14-Nov-2017].

[23] "Advanced Encryption Standard," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard. [Accessed: 14-Nov-2017].

[24] Diffie-Hellman Key Exchange," cryptography - Information Security Stack Exchange. [Online]. Available: https://security.stackexchange.com/questions/45963/diffie-hellman-key-exchange-in-plain-english. [Accessed: 4-Nov-2017].

[25] "Diffie–Hellman key exchange - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Diffie–Hellman_key_exchange. [Accessed: 14-Nov-2017].

[26] "How does the man in the middle attack work in Diffie–Hellman?," public key encryption - [Online]. Available: https://stackoverflow.com/questions/10471009/how-does-the-man-in-the-middle-attack-work-in-diffie-hellman. [Accessed: 14-Nov-2017].

[27] Bresson, E., Chevassut, O., & Pointcheval, D. (2002). Group Diffie-Hellman key exchange secure against dictionary attacks. Advances in Cryptology—ASIACRYPT 2002, 603-610.

[28] "Dictionary attack," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Dictionary_attack. [Accessed: 14-Nov-2017].

[29] Steiner, M., Tsudik, G., & Waidner, M. (1996, January). Diffie-Hellman key distribution extended to group communication. In Proceedings of the 3rd ACM conference on Computer and communications security (pp. 31-37). ACM.

[30] Günther, C. G. (1989, April). An identity-based key-exchange protocol. In Workshop on the Theory and Application of of Cryptographic Techniques (pp. 29-37). Springer, Berlin, Heidelberg.

[31] Boyko, V., MacKenzie, P., & Patel, S. (2000). Provably secure password-authenticated key exchange using Diffie-Hellman. In Advances in Cryptology—Eurocrypt 2000 (pp. 156-171). Springer Berlin/Heidelberg.

[32] Li, N. (2010, April). Research on diffie-hellman key exchange protocol. In Computer Engineering and Technology (ICCET), 2010 2nd International Conference on (Vol. 4, pp. V4-634). IEEE.

[33] Adrian, D., Bhargavan, K., Durumeric, Z., Gaudry, P., Green, M., Halderman, J. A & VanderSloot, B. (2015, October). Imperfect forward secrecy: How Diffie-Hellman fails in practice. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 5-17). ACM.

[34] Bao, F., Deng, R. H., & Zhu, H. (2003, October). Variations of diffie-hellman problem. In International Conference on Information and Communications Security (pp. 301-312). Springer, Berlin, Heidelberg.

[35] Parekh, A., Pawar, A., Munot, P., & Mantri, P. (2011). Secure authentication using anti-screenshot virtual keyboard. International Journal of Computer Science Issues (IJCSI), 8(5).

[36] Olzak, T. (2008). Keystroke logging (keylogging). Adventures in Security, April.

[37] "What is Shoulder surfing," Solutions 24h. [Online]. Available: http://solutions24h.com/what-is-shoulder-surfing/. [Accessed: 14-Nov-2017].

[38] Haque, M. A., & Imam, B. (2014). A New Graphical Password: Combination of Recall & Recognition Based Approach. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, 8(2), 320-324.

[39] Martin, J., & Leben, J. (1994). TCP/IP networking: architecture, administration, and programming. Prentice-Hall, Inc..

[40] Xue, M., & Zhu, C. (2009, May). The socket programming and software design for communication based on client/server. In Circuits, Communications and Systems, 2009. PACCS'09. Pacific-Asia Conference on (pp. 775-777). IEEE.

[41] Law, K. E., & Leung, R. (2003). A design and implementation of active network socket programming. Microprocessors and Microsystems, 27(5), 277-284.

[42] Meier, R. (2012). Professional Android 4 application development. John Wiley & Sons.

[43] Rogers, R., Lombardo, J., Mednieks, Z., & Meike, B. (2009). Android application development: Programming with the Google SDK. O'Reilly Media, Inc..

[44] “Package java.security,” java.security (Java Platform SE 7) [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html. [Accessed: 11-Nov-2017].

[45] “Package Javax.crypto.” Javax.crypto (Java Platform SE 7) [Online]. Available: docs.oracle.com/javase/7/docs/api/javax/crypto/package-summary.html. [Accessed: 21-Oct-2017].

[46] “Package java.net,” java.net (Java Platform SE 7) [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html. [Accessed: 11-Nov-2017].

[47] Socket (Java Platform SE 7), 09-Oct-2017. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html. [Accessed: 11-Nov-2017].

[48] “Java (programming language),” Wikipedia, 08-Nov-2017. [Online]. Available: https://en.wikipedia.org/wiki/Java_(programming_language). [Accessed: 11-Nov-2017].

[49] Zxing, “zxing/zxing,” GitHub. [Online]. Available: https://github.com/zxing/zxing. [Accessed: 11-Nov-2017].

[50] “C Sharp (programming language),” Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/C_Sharp_(programming_language). [Accessed: 11-Nov-2017].

[51] BillWagner, “Introduction to the C# Language and the .NET Framework,” Microsoft Docs. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-

framework. [Accessed: 11-Nov-2017].

[52] Inthehand, “inthehand/32feet,” GitHub [Online]. Available: https://github.com/inthehand/32feet. [Accessed: 11-Nov-2017].

[53] System.Security.Cryptography Namespace [Online]. Available: https://msdn.microsoft.com/en-us/library/system.security.cryptography(v=vs.110).aspx. [Accessed: 11-Nov-2017].

[54] “System.Threading Namespaces” [Online]. Available: https://msdn.microsoft.com/en-us/library/mt481587(v=vs.110).aspx. [Accessed: 11-Nov-2017].

[55] Codebude, “QRCoder,” GitHub. [Online]. Available: https://github.com/codebude/QRCoder/wiki. [Accessed: 11-Nov-2017].

[56] “Android (operating system),” Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Android_(operating_system). [Accessed: 11-Nov-2017].

[57] “Android 6.0 APIs,” Android Developers. [Online]. Available: https://developer.android.com/about/versions/marshmallow/android-6.0.html. [Accessed: 11-Nov-2017].

[58] Wireshark [Online]. Available: https://www.wireshark.org/. [Accessed: 11-Nov-2017].

[59] “An Overview of the Android Architecture,” [Online]. Available: http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture. [Accessed: 11-Nov-2017].

[60] “USBlyzer - USB Protocol Analyzer and USB Traffic Sniffer,” [Online]. Available: http://www.usblyzer.com/. [Accessed: 11-Nov-2017].

[61] Soon, T. J. (2008). QR code. Synthesis Journal, 2008, 59-78.

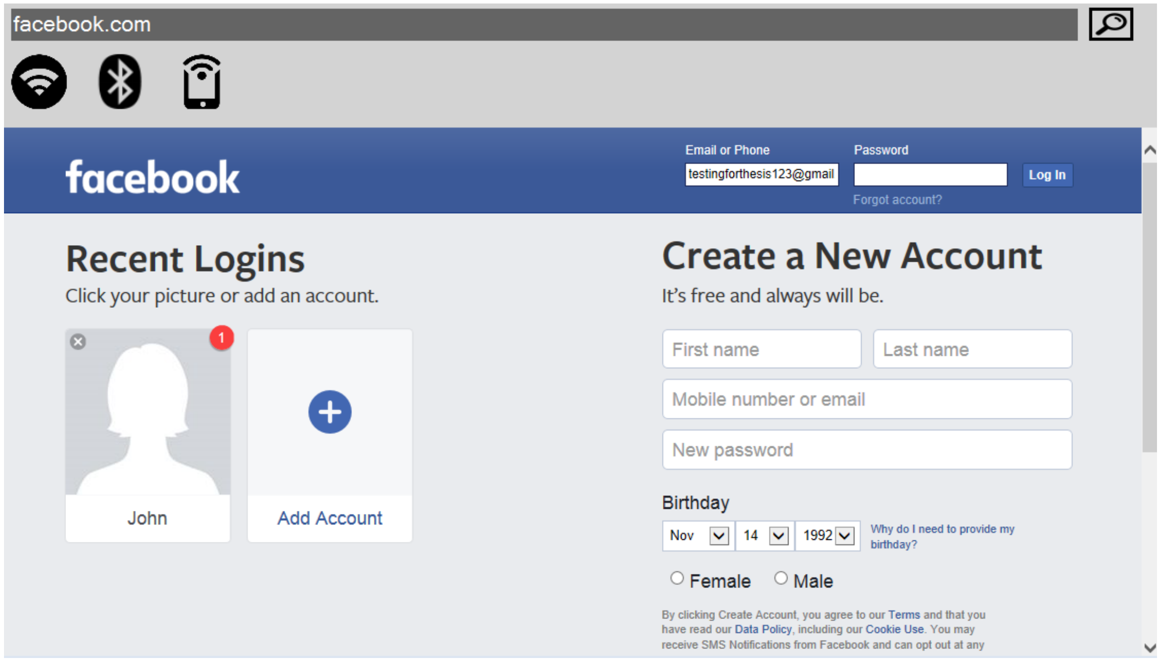
[62] Liu, Y., Yang, J., & Liu, M. (2008, July). Recognition of QR Code with mobile phones. In Control and Decision Conference, 2008. CCDC 2008. Chinese (pp. 203-206). IEEE.

[63] Nand, P., Singh, P. K., Aneja, J., & Dhingra, Y. (2015, March). Prevention of shoulder surfing attack using randomized square matrix virtual keyboard. In Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in (pp. 916-920). IEEE.

[64] Vinothini, Saranya, and Vasumathi, "A Study On Diffie-Hellman Algorithm in Network Security," International Journal Of Engineering And Computer Science ISSN:2319-7242, vol. 3, no. 7 July 2014, pp. 7346–7349.

[65] Michel Abdalla, Mihir Bellare, and Phillip Rogaway, "DHIES: An encryption scheme based on the Diffie-Hellman Problem", In Proc.of ACM CCS '01, ACM Press September18,2001.

APPENDIX A

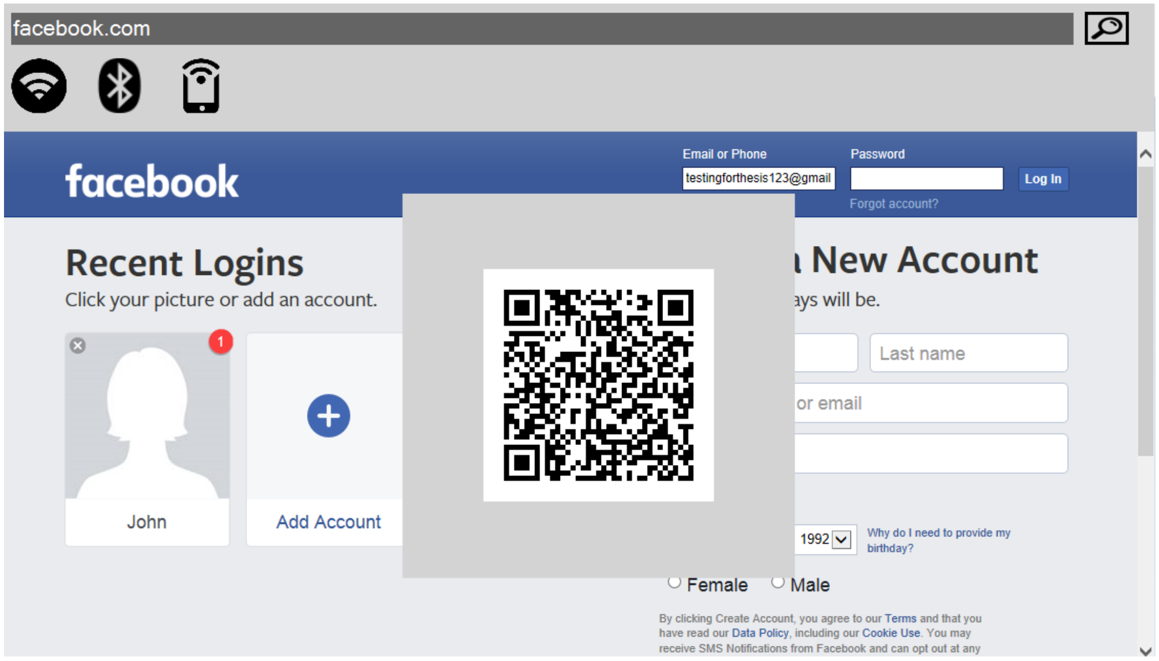


Email or Phone: testingforthesis123@gmail

Birthday: Nov 14 1992

By clicking Create Account, you agree to our Terms and that you have read our Data Policy, including our Cookie Use. You may receive SMS Notifications from Facebook and can opt out at any

Figure 46: Screenshot of the desktop application home page.

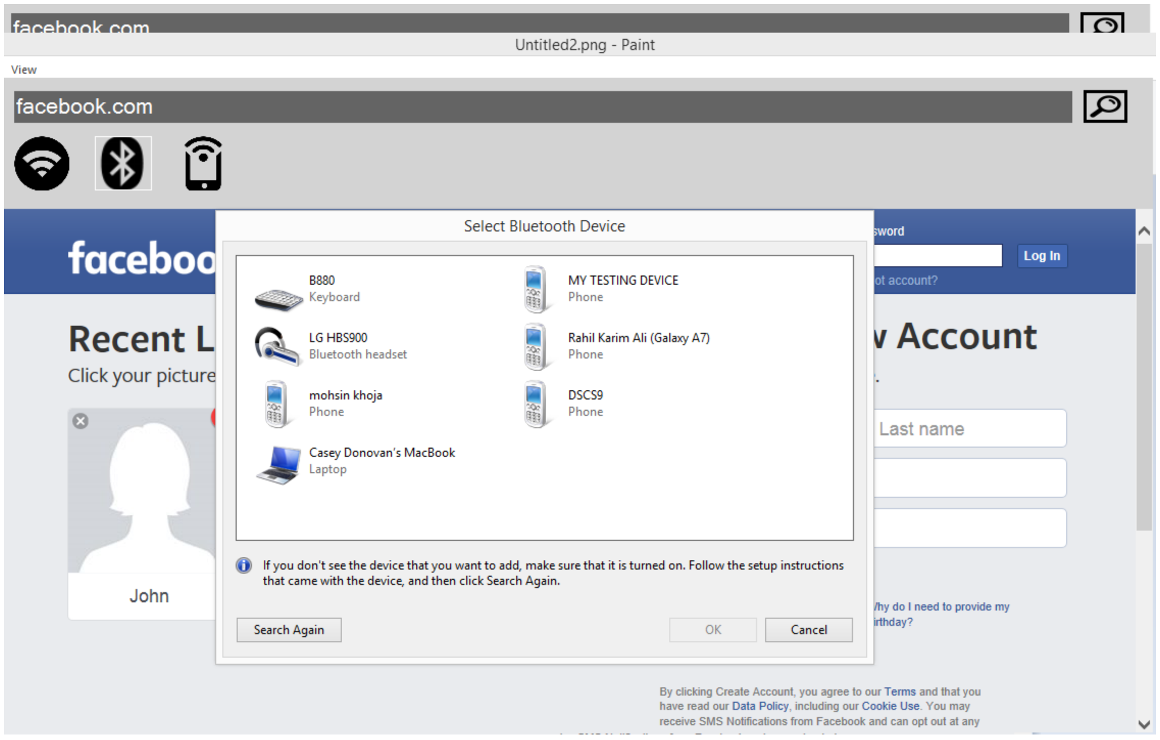


Email or Phone: testingforthesis123@gmail

1992

By clicking Create Account, you agree to our Terms and that you have read our Data Policy, including our Cookie Use. You may receive SMS Notifications from Facebook and can opt out at any

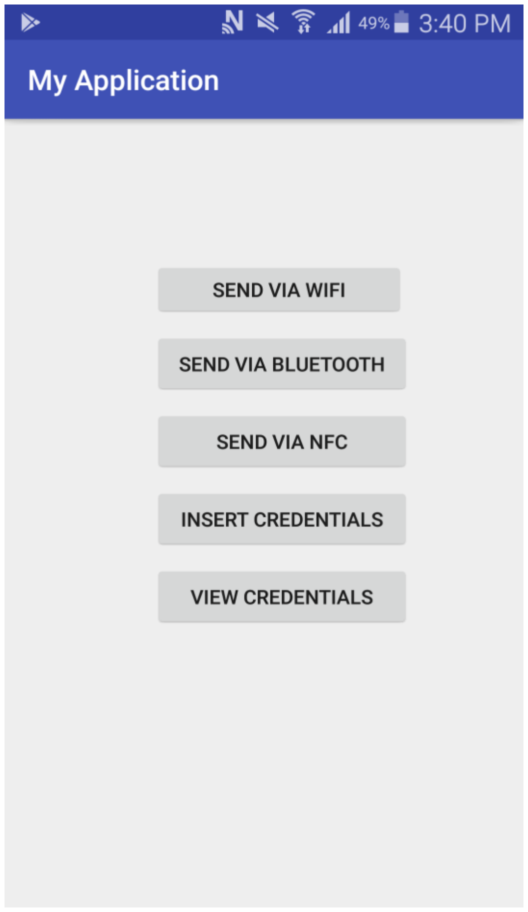
Figure 47: Desktop application displaying QR code.



If you don't see the device that you want to add, make sure that it is turned on. Follow the setup instructions that came with the device, and then click Search Again.

Figure 48: Desktop application searching Bluetooth devices.

APPENDIX B



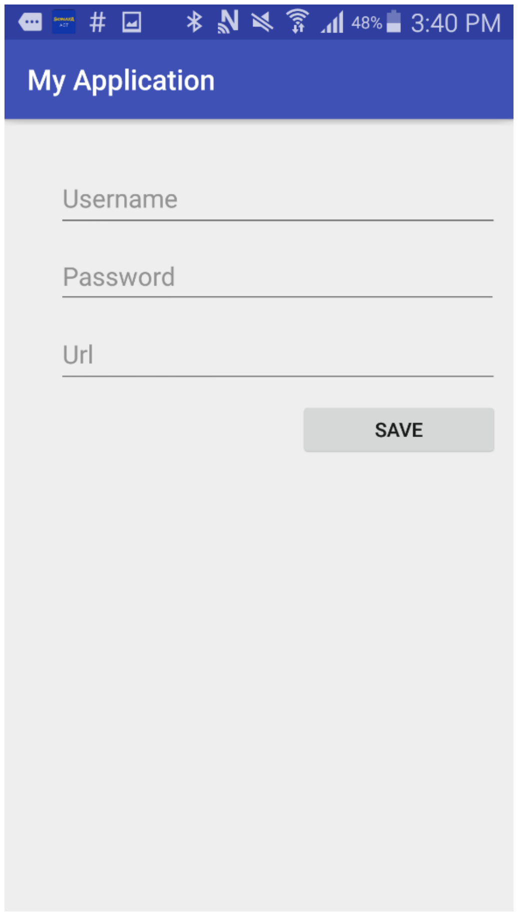
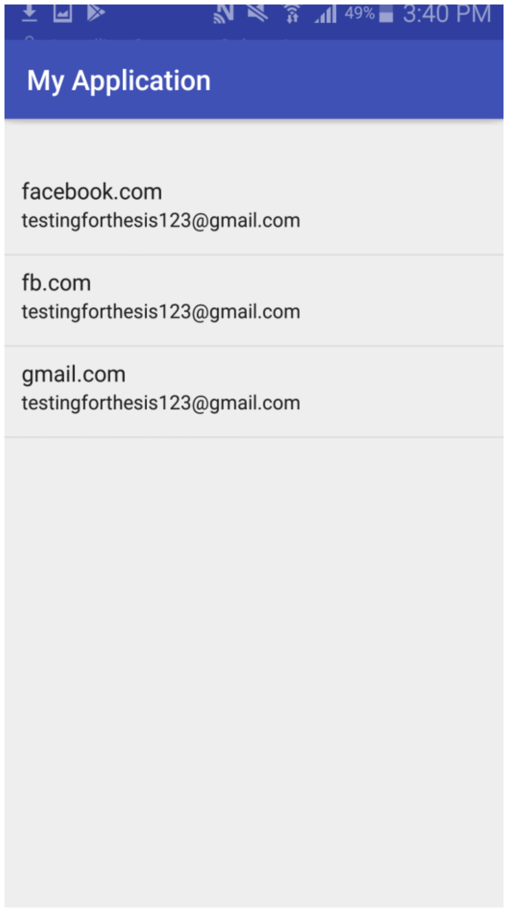


Figure 49: Mobile application home page. Figure 50: Mobile application insert credential page



facebook.com
testingforthesis123@gmail.com

fb.com
testingforthesis123@gmail.com

gmail.com
testingforthesis123@gmail.com

Figure 51: View Credential page

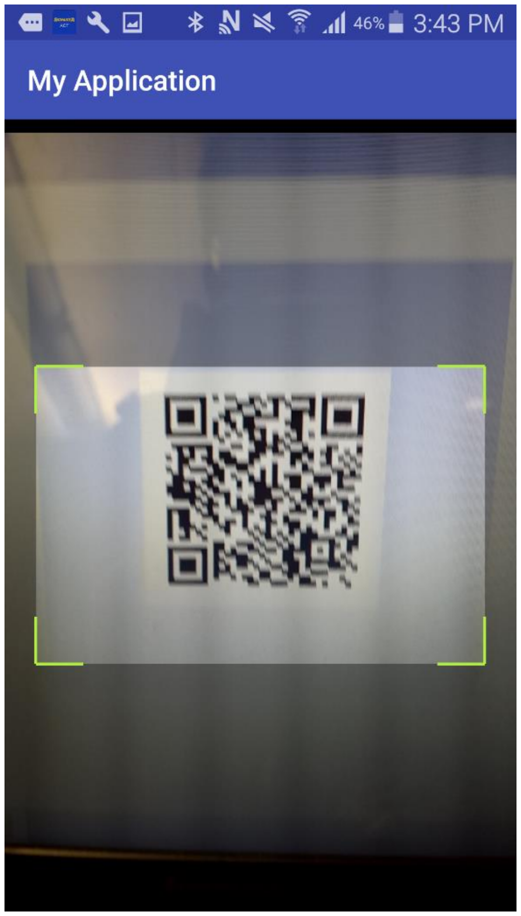


Figure 52: Scan QR code

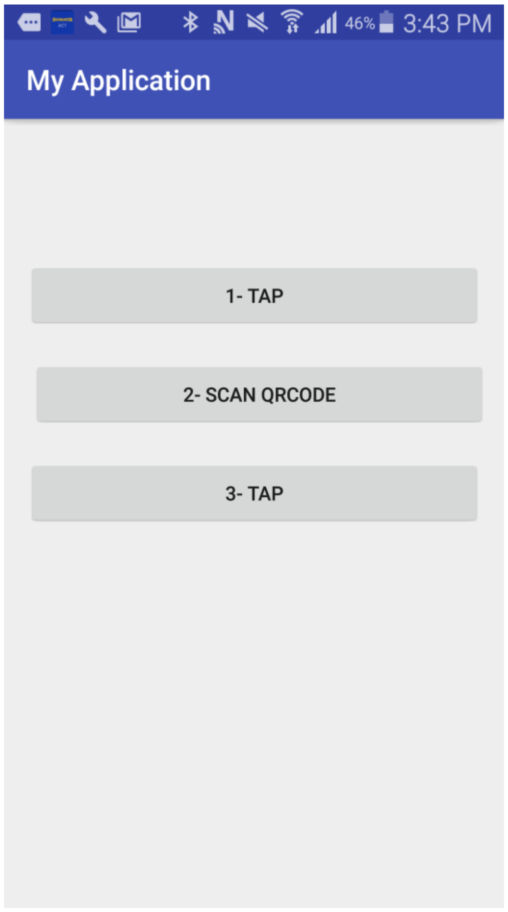


Figure 53: NFC page