Secure Communication Protocol Between Two Mobile Devices Over Short Distances

Muhammad Umair Khan[1](✉), Farhana Chowdhury[2], Zarmina Jahangir[1], and Francis Ofougwuka[3]

[1] Department of Computer Science, Riphah International University, Lahore, Pakistan
umair@cs.queensu.ca, zarmina.jahangir@riphah.edu.pk
[2] Department of Information and Communication Engineering Technology, Centennial College, Toronto, ON, Canada
fchowdh3@my.centennialcollege.ca
[3] Royal Bank of Canada, Toronto, ON, Canada
francid81@gmail.com

Abstract. The security of mobile devices has become a significant issue with the increase in data and computing capacities. Such devices store security critical data such as passwords to various online services. In the event of theft of such devices, the user's credentials are at the mercy of the attacker. A secondary mobile device may be used to lock the primary mobile device if the distance between the two is larger than a specified threshold. Such proximity-based locking devices use Bluetooth and NFC technologies to communicate with the primary devices. In this paper, we propose an authentication protocol that can be used between two such mobile devices which use NFC and Bluetooth. The protocol elaborates a number of possible scenarios and how they should be implemented to maximize security of the mobile devices. The proposed protocol has been implemented and tested on Android and iOS devices.

Keywords: Software security · Security · NFC · Bluetooth
Secure communications protocol

1 Introduction

Security considerations are an integral part of today's mobile devices such as smartphones, notebooks, laptops, and tablets. The primary reason behind this consideration is that these mobile devices now have the computing and data storage capacities that were not even available on desktop computers up till recently. This computing prowess has provided users with options to conduct social and business activities on their mobile devices. However, if the data stored on the mobile devices (*e.g.*, credentials, passwords, and financial information) falls in wrong hands, the user's confidentiality and integrity may be compromised. Moreover, they may suffer emotional and financial repercussions.

Many security mechanisms have been implemented by the mobile devices' vendors to circumvent and safeguard against security threats. Such mechanisms include biometric

R. Mehmood et al. (Eds.): SCITA 2017, LNICST 224, pp. 74–85, 2018.
https://doi.org/10.1007/978-3-319-94180-6_9

authentication and remotely deleting data (*e.g.*, Apple's "Find Your Phone" [1]). A more recent security model is to use a secondary mobile device to lock/unlock the primary one based on its proximity to the secondary one. A similar idea has successfully been implemented in high-end cars where the keys stay in the driver's pocket and the vehicle detects, based on the proximity, the presence of the key and enables functions of the car. This same idea has already been implemented by Lynk from uConekt [2] for mobile devices. The secondary mobile device does not have to be a mobile phone or tablet in itself as only limited storage and processing capabilities are required for authentication and locking purposes. The secondary device is small enough to fit in a key ring as a key fob along with other keys. As most people use and keep keys (car, office, or home) close, the secondary device will be in close range of the primary device most of the time. The range of a Bluetooth device may range be from 0.1 m to 100 m depending. Most mobile devices use Class 2 Bluetooth which has a typical range of 10 m. We consider this an effective range to implement a proximity based authentication and locking mechanism. This implies that if the secondary and primary devices are more than 10 m apart, the primary device will lock.

Proximity-based mobile authentication [4] devices use Bluetooth and Near Field Communication (NFC) technologies to communicate. These technologies are not completely secure in their handling and transfer of data. Bluetooth and NFC have various security issues such as leaking of information from the system to an unwanted party resulting into confidentiality violation, unauthorized changes of information during transmission leading to integrity violations, and resources blocked by malicious attacker resulting in availability violations. Proximity-based authentication [4] devices use NFC and Bluetooth to verify the primary device because of their short and adjustable range. However, NFC and Bluetooth are prone to attacks because they do not use a central server to authenticate and most of the time the data transferred is not strongly encrypted. We, in this paper, propose an authentication and data handling protocol that offers a secure way for these devices to communicate via Bluetooth and NFC. We discuss various security related scenarios, especially when one device (primary or secondary) is taken beyond the proximity threshold either by the legitimate user or by an attacker. We propose security safeguards that should be in implemented to secure the primary device in the event of any of these scenarios.

The remainder of this paper is organized as follows. Section 2 presents the related work in this area. Section 3 describes preliminaries. Section 4 provides the details of the proposed protocol. Section 5 concludes the paper with future works and limitations.

2 Related Work

Extended usage of the mobile computing devices (*e.g.*, smart phones) in our lives is resulting in many security issues especially regarding device-to-device communication. Numerous researchers have suggested techniques involving password based authentication, proxy based security protocols and Near Field Communications (NFC) which would ensure a secure and reliable means of communication between the devices without intervention by an unauthentic device/person.

Transport Layer Security (TLS) protocol [3] works on the transport layer which provides end-to-end security. TLS is currently being used for internet communication along with its predecessor Secure Sockets Layers (SSL) protocol. This paper proposes that the advantages of TLS protocol are much more when implemented in mobile devices domain. Implementing security protocols in mobile devices also increases maintainability and extensibility. The major technique used in this paper is to hide information from intruders using bouncy castle cryptographic packages. Object oriented technique has also been used to enhance security. The implementation did not include optional TLS specifications such as client authentication, session resumption and compression.

Key agreement protocol [5] has been used to make device-to-device communication secure. This protocol activates two mobile devices to initiate a shared secret key for both the devices. This approach specifically uses Diffie-Hellman key agreement protocol [11]. The authors claim that the proposed technique is more effective and incurs less computational cost while using device-to-device communication. This paper presents an advance integration technique that allows previous methodologies to work with the proposed method. This paper also discusses their technique with regards to a cellular network using LTE dealing power control issues and Wi-Fi based Device-to-Device communications [5]. Analysis of the technique indicates that the proposed key agreement protocol enables two mobile users to securely set up a secret key with a small computation cost and low authentication overhead. A similar approach is presented in [6] which establishes a secure key between two mobile devices.

WebBee [11, 12], another communications technique, provides a complete framework that supports security sensitive applications used in mobile devices. According to this paper, there are two possibilities in terms of time when devices are compromised: the first possibility is when the user is informed about the system being compromised. The second possibility is when the security keys are initialized. This paper introduces a new technique called Challenge Response System (CRS) that supports the WebBee system to provide overall security in the worst situation. The proposed methodology also provides support to integrate existing methodologies to minimize the impact on infrastructure by using WebBee system. One of the main constraints in the CRS is that it deals with limited numbers applications.

The authors in [10] present different mobile settings that support and provide guidance on setting up a password. Results have been compiled after comparing the strength and usability of passwords that are generated on desktops, laptops and mobile devices. The proposed methodology shows that the passwords created on mobile devices are more error prone, longer, frustrating, and weaker. It is observed that password policies vary in both mobile and desktop environments and it also suggests an easy way to enter passwords in mobile devices.

Passwords can be breached with different methods like brute force and dictionary attacks. Strings of alphanumeric characters are also more difficult to remember, Graphical passwords have been used to make passwords more easy and reliable for users. In this technique user can select different images as passwords rather than entering different alphanumeric values and characters. However, such passwords are more prone to shoulder surfing attacks. An alternative method has been proposed for authentication purposes by using graphical passwords along with alphanumeric characters [9]. It is a combination of recognition and recall based system that is more reliable and more secure than the older systems. The presented technique is also more robust and reliable against the shoulder surfing attacks on graphical passwords. Moreover, the proposed methodology is more convenient and suitable for the mobile devices.

The authors in [3] discuss the security issues of wearable gadgets and software agents. They propose two separate protocols for each of these: a separate protocol for device-to-proxy communication and another one for proxy-to-proxy communication. Using two different protocols enables us to use less computation capacity of devices. Another advantage of separate protocols is the provision of reasonable authentication in communication on more strong devices. This paper gives details on lightweight wireless devices that have been used for device-to-proxy protocol and also elaborated on simple public key infrastructure and simple distributed security infrastructure for proxy-to-proxy protocols. The authors have also developed a prototype for secure and efficient access to networks and mobile devices. A qualitative analysis was performed on this prototype whose results favor proxy based security protocols in mobile devices.

3 Preliminaries

Near Field Communication (NFC) [8] is an emerging technology in wireless short range communication. It is based on different existing standards like Radio Frequency Identification (RFID) infrastructure. NFC provides support for contactless transactions that is used in different intuitive application scenarios such as over-the-air ticketing system and mobile payment system [7].

Bluetooth is a point-to-point or point-to-multipoint radio frequency technology for data exchange requiring minimal power. For a range of ten meters, the power usage is 2.5 mW. The typical range of a Bluetooth device is 10 to 100 m. Bluetooth is also being used in to internet of things (IoT) devices in a secure manner [17].

Security is an important concern in today's software systems as security failures may lead to financial losses or physical injuries [13]. Security vulnerabilities, the root of security failures, may be introduced in requirement specification, design, or implementation phases [14–16]. While communicating between two mobile devices, the security threats increase due to the use of the wireless medium.

4 Proximity-Based Authentication and Communication Protocol

We have identified a total of ten possible scenarios where the security of the primary device may be threatened. In the following paragraphs we describe these scenarios and how the server, primary device and secondary device should communicate to safeguard against threats. We represent this communication exchange using sequence diagrams. In these diagrams, $\boldsymbol{S}$ represents the server, $\boldsymbol{M_1}$ represents the primary device, $\boldsymbol{M_2}$ represents the secondary device, $\boldsymbol{S_{pr}}$ represents the server's private key, $\boldsymbol{M_{1pr}}$ represents the primary device's private key, $\boldsymbol{M_{2pr}}$ represents the secondary device's private key,

$\boldsymbol{S_{pu}}$ represents the server's public key, $\boldsymbol{M_{1pu}}$ represents for the primary device's public key, $\boldsymbol{M_{2pu}}$ represents for the secondary device's public key, $\boldsymbol{X}$ represents a random number sent as a message, and $\boldsymbol{K_1}$ and $\boldsymbol{K_2}$ are keys generated by the server during the synchronization process. The private and public keys can be generated by any appropriate asymmetric key algorithm. The remainder of this section is organized as follows. Sections 4.1 through 4.7 describe different scenarios that the devices may come across. Section 4.8 proposes an added layer of security in the form of a local password for the primary device. Section 4.9 presents the options for storing data securely. Section 4.10 concludes the protocol by presenting the details of an experiment conducted on iOS and Android mobile devices.

4.1 Synchronizing Primary and Secondary Devices

The primary and secondary devices have to be paired to each other at the beginning. This pairing is performed by exchanging and synchronizing keys. After synchronization, the two devices keep in constant contact. However, there will be many instances when the two devices are not in contact such as when the two devices are out of range, when the devices are turned off or run out of power, or when the SIM cards are changed. In such an event, the synchronization process has to be initiated to authenticate that the two devices are legitimate ones. This process is performed every seven days, even in the absence of any of the aforementioned scenarios. During this process the primary and secondary devices come to a close range of 0.0 m. Connection to the server is mandatory for this process therefore internet connection is essential for the success of the process.

The synchronization process is initiated by $\boldsymbol{M_1}$. It sends the sync command (which also reduces the power of both the devices). $\boldsymbol{M_1}$ sends the profile password to the server encrypted using the server's public key $\boldsymbol{S_{pu}}$. The server $\boldsymbol{S}$ generates two new keys $\boldsymbol{K_1}$ and $\boldsymbol{K_2}$ and sends them to $\boldsymbol{M_1}$ using $\boldsymbol{M_{1pu}}$. $\boldsymbol{M_1}$ will decrypt the keys by using $\boldsymbol{M_{1pr}}$. $\boldsymbol{M_1}$ will then re-encrypt $\boldsymbol{K_2}$ by using the same algorithm only it will use the public key $\boldsymbol{M_{2pu}}$ of $\boldsymbol{M_2}$ and then send it to $\boldsymbol{M_2}$. This is done to add another layer of security. $\boldsymbol{K_1}$ will be the new private key of $\boldsymbol{M_1}$ and $\boldsymbol{K_2}$ will be the new private key of $\boldsymbol{M_2}$. The method ***SyncS()*** sends the username and password in an encrypted format to the server. The method ***SyncM*$_2$*()*** verifies whether the distance between $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$ is zero. This scenario is depicted in Fig. 1 in the form of a sequence diagram.

4.2 SIM Change and Device Switched ON/OFF

With the assumption that $\boldsymbol{M_1}$ is a mobile device, SIM change is when the user removes the SIM to swap with another SIM or have the same SIM reinstated. The device has to be synchronised (sync) according to the steps defined in Sect. 4.1. Similarly, whenever the primary or secondary device is turned ON/OFF, the sync process defined in Sect. 4.1 has to be performed before the user can access the data in the primary device

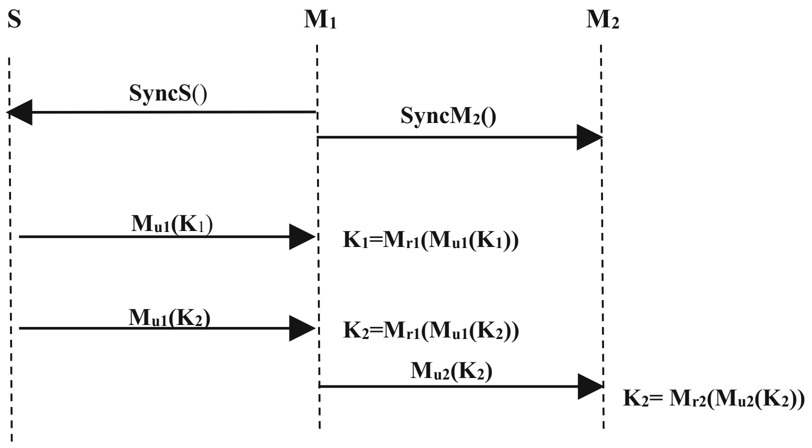

Fig. 1. Synchronizing primary and secondary.

4.3 M_1 and M_2 are Within the Defined Proximity of Each Other

To ensure a device within range is the correct device, we need a means of authenticating the two devices as part of the same pair. After every 30 s, M_1 generates a random value X and sends it to M_2. We use the M_{2pu} to encrypt the X resulting in M_{1x} before it is sent. M_2 decrypts the M_{1x} using the M_{2pr} and overwrites the X in M_2 with the new X. M_2 responds to M_1 message by sending the X to M_1 using the M_{1pu} to encrypt X resulting in M_{2x}. M_1 validates the message by decrypting the M_{2x} and checks if it matches with the initial X that was sent. If M_1 detects a discrepancy between the sent and received X, it locks M_1 deducing that the M_2 is fake or has been tampered with. When both the devices have X, then they can recognize each other as legitimate as only M_1 will have its private key. This scenario is represented in Fig. 2. If a fake or tampered device is detected, a sync process is required which involves communicating with the server S.

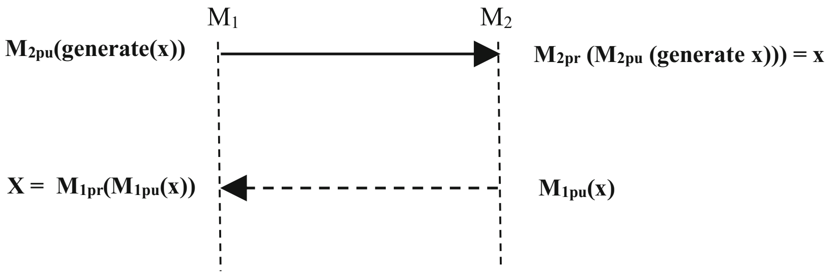

Fig. 2. Verifying proximity between M_1 and M_2.

4.4 M_1 and M_2 are Outside of the Defined Proximity of Each Other

This scenario occurs when M_1 and M_2 are not close to each other (within the pre-defined proximity). When this situation occurs, M_1 sends an encrypted X to M_2 and does not receive a valid response (X) from M_2. M_1 waits for 30 s (or any other time interval set by the user) before resending a new X to M_2. After 3 attempts without any response from M_2, M_1 locks itself. We do not distinguish whether M_1 or M_2 were stolen by an attacker or M_1 or M_2 were misplaced by the legitimate user. This scenario is described in Fig. 3.

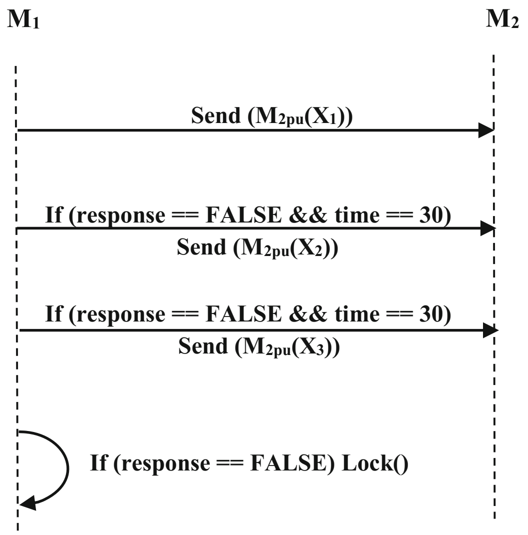
M1
M2
Send (M2pu(X1))
If (response == FALSE && time == 30)
Send (M2pu(X2))
If (response == FALSE && time == 30)
Send (M2pu(X3))
If (response == FALSE) Lock()

Fig. 3. Lock $\mathbf{M_1}$ if proximity between $\mathbf{M_1}$ and $\mathbf{M_2}$ is not verified.

4.5 Unlocking M_1

Once the device M_1 is locked because of M_2 being out of range, the unlock process has to be carried out. This process, to be successful, requires M_1 and M_2 to be present in close proximity with each other (zero meters to use NFC) and with access to the internet to allow connection to **S**. This scenario is represented in Fig. 4. Using the provided software application, the user logs in to the server by using the *ServerID* and *password* set up during the initial registration. The communication in this process has to be encrypted using the private and public key of S and M_1 (not shown in the figure).

Upon successful login, the server instructs M_1 to request from M_2 the last message X which is stored in its. This is required to ensure that the M_2 has not been swapped with another device since each distinct M_2 will have different X store in its memory. M_2 responds by sending the X value to M_1. All communication is carried out in encrypted form. M_1 validates the X sent by M_2 and when it does match with the stored X in M_1, then the sync process is initiated (Scenario I described in Sect. 4.1) with successful unlocking of M_1. If the X sent by M_2 does not match with the X stored in M_1, then the

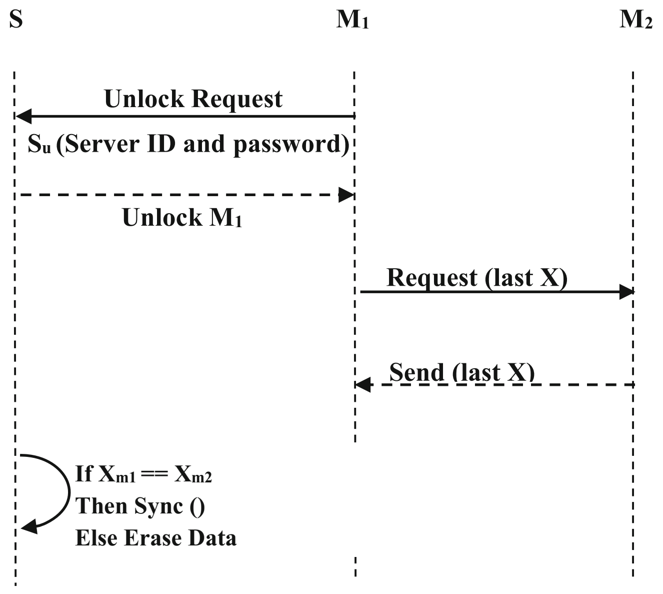

Fig. 4. Unlocking **M$_1$**.

data stored in ***M$_1$*** is erased because this situation indicates that ***M$_2$*** has been tampered with and someone is trying to deceive ***M$_1$***.

4.6 Internet Connection not Available

There could be a scenario that the user has no internet connection and the server cannot be reached. We propose to use a session token which has been proven effective in many other applications and protocols. We propose that a token should remain valid for only one hour after the internet connection is lost. After the passage of one hour, ***M$_1$*** should be locked. To unlock ***M$_1$*** we need to perform the sync process defined in Sect. 4.1.

There are multiple rationales behind this lock and sync. First, consider that the legitimate user loses both ***M$_1$*** and ***M$_2$***. He/she may need to send an erase data command to safeguard his data. However, if ***M$_1$*** not connected to the internet, it should lock itself as the erase command will not reach it. This does create a usability issue as the legitimate user will not be able to use his device in the absence of internet. However, the user can setup the time after the device locks itself.

4.7 M$_1$ and M$_2$ have been Stolen

In the unlikely case where both the devices are lost together, there are the following two possibilities:

- User is aware: When a user knows the devices are gone, he/she can login to the application server and send an erase command to ***M$_1$***. If the device ***M$_1$*** is still connected to the internet, the command will be successful and all data on the device will be erased. This scenario is presented in Fig. 5.

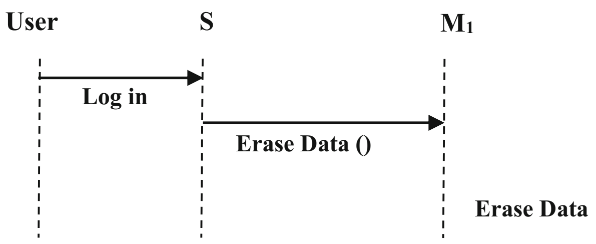

Fig. 5. Erase data.

- User is not aware: When a user has no knowledge of the devices been stolen, and both the devices are in close proximity, $\boldsymbol{M_1}$ will not lock itself. However, to resolve this, an added layer of protection can be provided in which the software application installed on the device $\boldsymbol{M_1}$ asks for a password after a specified interval of time and locks itself if that password is not entered. This scenario is described in detail in Sect. 4.8.

4.8 Local Password

No matter if $\boldsymbol{M_1}$ is connected to Server or $\boldsymbol{M_2}$ or not, in all cases $\boldsymbol{M_1}$ will ask for a local password which will be different from the server password. The password will be at least 10 characters long. This password will work as a screen lock and will be enabled after 30 s of inactivity. The user will have only three chances to enter the right password. If the local password is not entered correctly, $\boldsymbol{M_1}$ will let the user enter it two more times and after three incorrect attempts, $\boldsymbol{M_1}$ will be locked and user has to go through the unlock procedure via the server. This scenario is represented in the sequence diagram in Fig. 6.

4.9 Data Storage

There are two places where data can be stored: on the cloud/server and on the device $\boldsymbol{M_1}$. Each of them requires a process to make sure data is not compromised

- On the Cloud/Server: In this process, we synchronize $\boldsymbol{M_1}$ with the cloud/server. $\boldsymbol{M_1}$ only has the last values of X on it while the remaining data is on the cloud/server. $\boldsymbol{M_1}$ encrypts the data using $\boldsymbol{S_{pu}}$ and sends it to the server $\boldsymbol{S}$. Server decrypts the data using $\boldsymbol{S_{pr}}$. The server then stores the data and sends "delete data" command to $\boldsymbol{M_1}$. $\boldsymbol{M_1}$ deletes all data and sends confirmation to the server. If $\boldsymbol{M_1}$ requests any data, the server encrypts data using $\boldsymbol{M_{1pu}}$ and sends it. $\boldsymbol{M_1}$ then decrypts the data using $\boldsymbol{M_{1pr}}$.
- On $\mathbf{M_1}$: For this process, all the data is stored in $\boldsymbol{M_1}$. When $\boldsymbol{M_1}$ needs to save any data, it requests an ***EKey*** from the server. The server encrypts the ***EKey*** with $\boldsymbol{M_{1pu}}$ and sends it to $\boldsymbol{M_1}$. $\boldsymbol{M_1}$ decrypts the key using $\boldsymbol{M_{1pr}}$ and encrypts the data using that

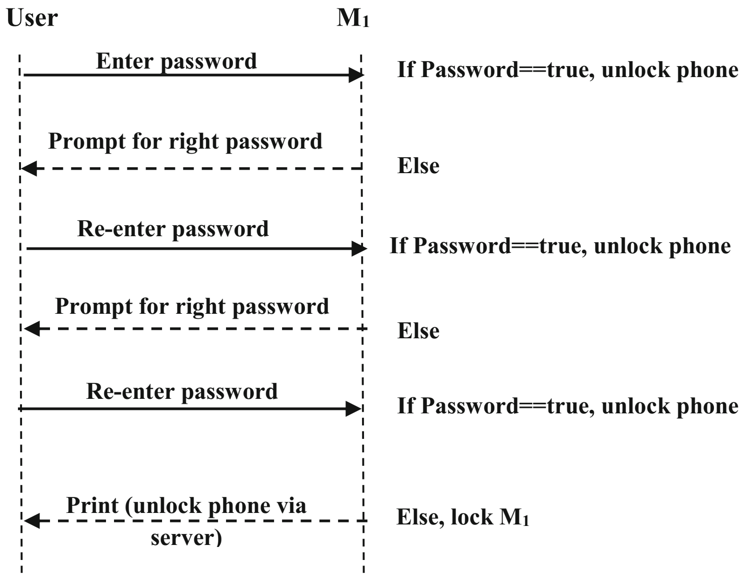
User
M1
Enter password
If Password==true, unlock phone
Prompt for right password
Else
Re-enter password
If Password==true, unlock phone
Prompt for right password
Else
Re-enter password
If Password==true, unlock phone
Print (unlock phone via server)
Else, lock M1

Fig. 6. Local password.

key. $\boldsymbol{M_1}$ then deletes the ***EKey*** from its memory. When $\boldsymbol{M_1}$ needs to access the data on it, it requests the ***DKey*** from the server. Server encrypts the ***DKey*** using $\boldsymbol{M_{1pu}}$ and sends that to $\boldsymbol{M_1}$. $\boldsymbol{M_1}$ decrypts the ***DKey*** using $\boldsymbol{M_{1pr}}$ and uses this key to decrypt the data. This decrypted data is only viewable and is not saved at any time in $\mathbf{M_1}$. The keys are always stored in the server. This will help to mitigate the risk of the key or data been discovered when the device is lost.

4.10 Experimental Implementation

The above-mentioned communication protocol was implemented in two separate Android and iOS based primary devices. The secondary device was an embedded system in the form of a key fob. For the primary devices, JAVA was used to develop an App. This app was responsible for authentication of the user using a password, communicating with the server to verify keys and accessing user information, and locking the primary device when the secondary device was out of range.

The code for the embedded device was written in C due to space and computational restrictions. The secondary device was first registered with the server. This registered secondary device was then paired with a primary device and the pairing information was added to the user database in the server. Once the pairing was complete, the secondary device was locked to the primary one and it could not be paired with any other primary device with making changes to the secondary device database in the server.

The server was installed in the cloud and was used to host client credentials and generate keys. All of the described scenarios were tested multiple times. The results indicate that based on the discussed scenarios, the primary device was locked whenever it was out of range of the secondary device.

5 Conclusions and Future Work

With the extensive use of mobile devices in our everyday lives, security of data stored on these devices is becoming of paramount importance. Lost mobile devices may not only represent the cost of the device itself. The data stored in such devices, such as banking information, social websites' credentials, and mailboxes, maybe used to exploit the user financially, politically, socially, and emotionally.

Usually the security features available in mobile devices are not sufficient if the device is not connected to the internet (*e.g.*, erase data remotely). We propose a communication protocol in this paper which uses a secondary device as an authenticator. A similar idea has been successfully implemented in other domains such as the automobile industry. We consider multiple scenarios which the mobile device may encounter. The proposed protocol was implemented on Android and iOS devices and tested.

The only limitation observed in the protocol is the time interval between losing both the devices and the user defined time to lock the device. As a future work, this communication protocol can also be used for internet of things (IoT) devices however it may require some modification because of their distributed nature. Moreover, this protocol has to be modified and tested for devices which cannot reasonably store data on the cloud and completely depend on storing the data within itself. This will be the case when the amount of data to be protected is too large.

References

1. iCloud - Find my iPhone. https://www.apple.com/ca/icloud/find-my-iphone/. Accessed July 2017
2. uConekt. http://uconekt.com/lynk/. Accessed July 2017
3. Kayayurt, B., Tuglular, T.: End-to-end security implementation for mobile devices using TLS protocol. J. Comput. Virol. **2**(1), 87–97 (2006)
4. Burnside, M., Clarke, D., Mills, T., Devadas, S., Rivest, R.: Proxy-based security protocols in networked mobile devices. In: Symposium on Applied Computing (SAC 2002), pp. 265–272 (2002)
5. Shen, W., et al.: Secure key establishment for device-to-device communications. In: IEEE Global Communications Conference (GLOBECOM), pp. 336–340 (2014)
6. Aldosari, W., El Taeib, T.: Secure key establishment for device to device communications among mobile devices. Int. J. Eng. Res. Rev. **3**(2), 43–47 (2015)
7. Siira, E., Tuikka, T., Tormanen, V.: Location-based mobile wiki using NFC tag infrastructure. In: 1st International Workshop on Near Field Communication (NFC 2009), pp. 56–60 (2009)

8. Burkard, S.: Near field communication in smartphones, Department of Computer Engineering, Berlin Institute of Technology, Germany, Technical report. https://www.snet.tu-berlin.de/fileadmin/fg220/courses/WS1112/snet-project/nfc-in-smartphones_burkard.pdf. Accessed July 2017
9. Routi, S., Andersen, J., Seamons, K.: Strengthening password-based authentication. In: Symposium on Usable Privacy and Security (SOUPS) (2016)
10. Khan, W.Z., Aalsalem, M.Y., Xiang, Y.: A graphical password bases authentication based system for mobile devices. Int. J. Comput. Sci. Issues (IJCS) **8**(5)-2, 145–154 (2011)
11. Yang, B.S., Dreijer, S., Jamin, S., Mukherjee, S., Wang, L.: Secure communication framework for mobile devices. Technical report, University of Michigan at Ann Arbor, MI, USA, CSE-TR-543-08 (2008). https://www.cse.umich.edu/techreports/cse/2008/CSE-TR-543-08.pdf. Accessed July 2017
12. Upatkoon, K., Wang, W., Jamin, S.: WebBee: an architecture for web accessibility for mobile devices. In: 10th IFIP International Conference on Personal Wireless Communications, Colmar, France (2005)
13. Khan, M.U.A., Zulkernine, M.: Developing components with embedded security monitors. In: 2nd International ACM SIGSOFT Symposium on Architecting Critical Systems (ISARCS 2011), pp. 133–142 (2011)
14. Khan, M.U.A., Zulkernine, M.: Quantifying security in secure software development phases. In: 2nd IEEE International Workshop on Secure Software Engineering (IWSSE 2008), pp. 955–960 (2008)
15. Khan, M.U.A., Zulkernine, M.: On selecting appropriate development processes and requirement engineering methods for secure software. In: 4th IEEE International Workshop on Security, Trust, and Privacy for Software Applications (STPSA 2009), pp. 353–358 (2009)
16. Khan, M.U.A., Zulkernine, M.: Activity and artifact views of a secure software development process. In: International Workshop on Software Security Process (SSP 2009), pp. 399–404 (2009)
17. Hussain, S.R., Mehnaz, S., Nirjon, S., Bertino, E.: Secure seamless bluetooth low energy | connection migration for unmodified IoT devices. IEEE Trans. Mob. Comput. **PP**(99), 17 (2017)