SmartAuth: Multi-Factor Authentication using Smart Card and Android on Web

Muhammad Shahbaz¹, Dr. Abdul Ghafoor Abbasi² and Muhammad Sohaib Shaheen³

¹, ², ³ School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad, Pakistan

13msccsmshahbaz@seecs.edu.pk¹, abdul.ghafoor@seecs.edu.pk², 12msccsmshaheen@seecs.edu.pk³

ABSTRACT
To protect any resource in a system from an unauthorized entity, the authentication process is basic and the first line of defense. Several authentication schemes are available for user authentication or verification to solve the issue of secure authentication mostly enforcing an extra layer of security with two-factor or multi-factor authentication. In this paper, we propose SmartAuth, a secure and usable multi-factor authentication mechanism using contactless smart-card and Android device on the web. The SmartAuth provides ownership base authentication system using Android device and contactless smart-card via NFC on the web. The SmartAuth is based on PKI and a prototype of the proposed authentication mechanism is also developed and presented in this research paper.

Keywords: Smart Authentication, Multi-Factor, Smart Card, Android on Web.

1. INTRODUCTION
Over the past several years the smart phone market is exponentially increasing. There are many major players in the market, but android operating system dominates all other smart phones by holding the highest shipment volume according to a study conducted by IDC [1] [2]. The reason for the popularity of the android operating system is its open source nature and an application friendly environment developed by Google Inc [3]. Android smart phones provide many features under one umbrella, including NFC (Near Field Communication), data storage, contacts storage, camera, GPS navigation, internet with support for Wi-Fi or cellular data including Edge, 3G and 4G LTE, emails and a wide range of applications. This attracts consumers from all price ranges in the smart phone market from different manufactures and independent of carrier providers.

To protect any resource in a device or a system, the authentication process is basic and the first line of defense. Authentication process is crucial for a system which ensures a basic security goal of integrity [4] [5]. Most existing authentication processes require handling at both sides, i.e at server side and the client side. The approval of any authentication system to use mostly depends on its robustness to prevent unauthorized access against malicious attacks and also requirement of resources at the server side and the client side. Because of dynamically increasing smart phone devices over the past several years, requirement of the resource has become a prime aspect of an authentication scheme. In the literature several authentication schemes are available. The authentication schemes can be predominantly categorized as follows:

- What you know?
- What you have?
- What you are?

"What you know" is an authentication scheme based on a knowledge of a secret e.g. password or a pin. The authentication scheme with base on something which can be used for authentication, for example, smart cards, is "What you have". Which is widely used for 2-factor or multi factor authentication. Finally "What you are" is an authentication scheme based on something a person is. Authentications based on biometric (fingerprint, retinal scan, etc.) are such examples [6]. Authentication using password dominates other schemes of authentication, other forms of authentication are also available. Knowledge based authentication has some problems like remembering, e.g. an average person tends to remember only seven random characters of data approximately [7], password guessing vulnerability, brute force attack using a dictionary, key logger, social engineering and shoulder surfing [8]. Other than that a user may choose an easy to guess password [9] or store his password and become prone to theft by malicious entity or malware. The authentication schemes could get weaker by this.

However, authentication scheme based on smart card does not recognize the user unless he possesses it. The authentication can’t occur if the user doesn’t have a smart card or an authorized one [10].
One crucial problem with passwords is reusing, that is users tend to reuse their passwords for multiple services online for easy to remember. If one website is compromised, then an adversary could have access to all of his accounts if user reused her password [11]. On average, a user reuse a single password 3.9 times on different websites [12].

In android operating system applications are written with the Java language and android provides a rich application environment to build creative applications and fun games [18]. Android applications use android’s built-in Java APIs to access the different components of the system running inside the DVM. For android application development, Google provides SDK (Software Development Kit) tool with ADT plugin for Eclipse. To test android applications Eclipse ADT plugin provides an Android emulator. Android application during development phase can be tested and debugged inside the android emulator before releasing or publishing the application. Android application has four essential application components which consider as building blocks [19]. Activities: To display information on the screen, Services: run in the background, Content providers: when two applications require data to be shared between them, Broadcast receivers: another application component that is used when the message or a response require to broadcast system wide on an Android operating system [19].

In this research the required permissions are declared in the AndroidManifest.xml of the Android app. ADB is a command line tool provided by The Android SDK to Interface an emulator or communicate an Android device with client server basis.

2.1 Overview of Android

Android’s first beta operating system was released to the public in November, 2007. The Android Operating system is an open source operating system which is based on Linux Kernal also an open source operating system with built-in C/C++ libraries. All android applications are Java based applications runs inside a sandbox with reserved area for applications a Java virtual machine that is specially designed for Android DVM abbreviated Dalvik Virtual Machine optimized for Android smart phones [17].
3) Card emulation: in this mode Android device with NFC acts as an emulated NFC card that can be read by another NFC reader [20]. Android device provides support to different NFC based tag technologies for I/O operations [21]. The supported NFC technologies are listed in Table 2.

APDUs are data messages to communicate or exchange data between a smart card and reader or terminal. ISO/IEC 7816-4 standard is designed for APDUs. In Figure 2 shows command structure of APDU [25]. APDU commands are used to read, write and select etc. to communicate between smart card and reader.

2.2 Cloud Computing on Web

The emerging model of Cloud Computing technology deals with offering flexible and dynamically scalable computing resources on the Internet consequently making a lot of financial benefits. There are three types of distinct layers of Cloud Computing which are provided by the cloud providers. These three distinct layers provide different type of resources to the cloud consumers based on the client requirement.

2.2.1 GCM

Google Cloud Messaging. Google Cloud Messaging is a message service developed by Google for Android. It is free of cost. A developer can send messages or notifications from server side to its Android users via Google Cloud Messaging service.

2.3 User Authentication and Authorization

The authentication process is to determine whether an entity has access to a system. To protect any resource in a system, the authentication process is basic and the first line of defense. Authentication process is crucial for a system which ensures basic security goal of integrity [4] [5]. In this subsection we will discuss authentication attacks on the web.

2.3.1 Authentication on Web

Authentication using password dominates other schemes of authentication, other forms of authentication are also available. Knowledge based authentication has some problems like remembering, an average person tends to remember only seven random characters of data approximately [7], password guessing vulnerability, brute force attack using a dictionary, keylogger, social engineering and shoulder surfing [8]. Other than that a user may choose an easy to guess password [9] or store his password and prone to theft by malicious entity or
malware. The authentication schemes could get weaker by this.

One crucial problem with passwords is reusing, that is users tend to reuse their passwords for multiple services online for easy to remember. If one website is compromised, then an adversary could have access to all of his accounts if user reused his password [11]. On average a user reuse a single password 3.9 times on different websites [12]. In table 1 [16] listed some of the data breaches because of unauthorized access from one of the reasons mentioned above.

In paper [26] author discusses about the security of passwords and even mentions attack possibilities on two factor authentication. One is man-in-the-middle attack and other by using trojan horse. Author of [26] questions robustness of two factor authentication against phishing. Authors of paper [27], implements two factor authentication with the use of mobile phone. To generate One-Time-Password authors proposed two methods using mobile phones. One is using Java based third party application installed on the mobile phone and sync with a remote server to generate OTP on the local side. They also proposed SMS based two factor authentication scheme. In SMS based authentication, first user sends some unique information to verify its device to the server and then the server responds with the OTP in SMS. The study shows that mobile device’s unique id numbers, etc. can be generated using mobile emulators.

Security of authentication system based on those unique parameters which can be replicated or generated is not reliable and requires more secure elements.

In [28] authors proposed two factor authentication for cloud computing using PKI and out-of-band (OOB) Authentication. Out-of-band authentication using mobile phones. On login a token is generated and sent to the user’s registered mobile device via SMS from the server and as well as from the user’s local machine. The user matches and verifies both token and sends response to server with accept or reject to allow or cancel a session between a user’s workstation and cloud in a timely fashion. This approach mostly depends on GSM network and may not work with out of service area. Also to send response, the user may be charged with SMS charges by telecom operator which could make this solution costly.

2CAuth [29] presents a new scheme for two factor authentication with the use of QR-Code. QR-Code is generated with the help of smart card and mobile phone reads QR-Code using built-in camera. This proposed scheme doesn’t require synchronization between the user and Mobile Network Operator for authentication. Although smart card is used but it doesn’t store any secret information. The solution [29] for two factor authentication based on QR-Code requires a card reader and mobile phone camera to read QR-Code and generates random numbers for verification process by an application installed in mobile.

In paper [30] usability and commonly used two factor authentication schemes are discussed with surveys and its results. Most common two factor authentication is SMS based authentication. Also application based OTP is quite popular using user’s smart phone i.e Google Authenticator.

SMS OTP (One Time Password) dominates multifactor authentication on internet. Due to recent attacks and vulnerabilities found with SS7 (Signaling System 7) cellular networks [31], SMS OTP is now considered as insecure [32] and costly. As every time an SMS is sent for multi-factor authentication costs, even a small amount but on large scale it could be expensive. It may also not work in out-of-range areas of the cellular networks. Other multi-factor solutions are OTP (One Time Password) generated using a mutually shared random number. The security of this system lies in the random number and doesn’t provide mutual authentication.

2.3.2. Fips 196

ENTITY AUTHENTICATION USING PUBLIC KEY CRYPTOGRAPHY. Fips 196 is an entity authentication protocol using public key cryptography introduced and published by National Institute of Standards and Technology (NIST) [33]. This standard is used to authenticate an entity based on international standards using random number challenges and digital signatures. Fips 196 can be used to only authenticate one entity or both. When both entities authenticate each other using Fips 196 standard, it is called mutual authentication protocol [33].

3. ARCHITECTURE AND DESIGN

There are two main components of SmartAuth, one is to check and perform authentication processing with remote web server and second is to issue commands and communicate with smart card via NFC to generate digital signatures.
OpenSSL [34] an open source general purpose cryptography library to generate key pairs for public cryptography. OpenSSL can generate self signed certificate authority and user certificates based on PKI for Fips 196 protocol implementation to load on remote web server and smart card.

3.1 Authentication Module

Authentication Module is responsible for performing authentication process. It is also responsible to generate cryptographically secure random number for Fips 196, also verifies entity certificates using specified CA certificate, verifies digital signature and after successful verification authorized access to user. In this section each structural element is discussed.

To use SmartAuth as multi-factor authentication using smart card for cloud based web applications to authenticate securely by implementing Fips 196 Mutual Authentication using Public Key Cryptography 2.4.2 as shown in the Figure 4.

Following are the proposed steps for multi-factor authentication on web by implementing Fips 196 Mutual Authentication Protocol 2.4.2 with Android device and smart card:

1) To get an authorized access to a web application, user login with his traditional username and password.
2) Then user prompt to select her Android smart phone. If user doesn’t have a registered Android device then he requires to follow these steps as shown in Figure 5:
   a) To register SmartAuth application with Google Cloud Messaging, user requires to use his login id of web server.
   b) Then SmartAuth application makes an API call to Google Cloud Messaging with instance ID [35] to obtain registration token for this device.
   c) After successfully obtaining registration token from GCM, application post user’s id and token to web server with some extra information for reference.
   d) Web server receives the token and store against the user id.
3) User selects her Android device and web server sends authentication request to initialize mutual authentication via Google Cloud Messaging push notification.
4) Web server is now become initiator ‘A’ of the mutual authentication process.
5) Android and smart card are the responder ‘B’.
6) User receives authentication request on her Android device and opens SmartAuth application for mutual authentication using smart card via NFC.
7) Application detects smart card and verifies human ownership of smart card with PIN code.
8) In response to the authentication request from web server, Android gets 8 bytes random number from smart card (RB) and sends back as TokenBA1.
9) Web server then generate pseudo random number challenge (RA).
10) Web server generates SHA message digest from TokenBA1 and challenge from smart card.
1) Web server generates digital signature as TokenAB.

\[ \text{TokenAB} = \text{SignA}(\text{SHA1}(RA||RB)) \]

12) Web server sends back response with the TokenAB and public certificate of web server ‘A’ to Android application.

13) Android application gets TokenAB and public certificate CertA from web server.

14) Verifies CertA with CA certificate Then Android application verifies Digital Signature from TokenAB. On successful verification, web server ‘A’ has authenticated itself to ‘B’ on this step.

15) In response from smart card, Android Application gets digital signature

16) Android application also gets public certificate CertB from Smart Card via NFC

17) Android Application post TokenBA2 and public certificate of smart card in HEX format to server

18) Server using PHP language gets TokenBA2 and Public Certificate CertB

19) Server verifies smart card public certificate with CA certificate

20) After CA verification, server verifies digital signature from TokenBA2

\[ \text{TokenBA2} = \text{SignB}(\text{SHA1}(RB||RA)) \]

21) Successful completion of the above steps means that the responder, B (Android using Smart Card via NFC), has authenticated itself to the initiator, A (Web Server), and thus the entities have successfully mutually authenticated each other.

3.1 Smart Card Module

Smart Card Module of SmartAuth application handles communication with smart card via NFC as shown in Figure 6. Smart Card Module is responsible to send APDU commands and handle APDU responses from smart card via NFC including PIN verification, retrieve random number challenge, sending SHA1 message digest to get digital signature and also to retrieve smart cards’s public certificate.

3.2 Certificate Authority

A trusted certificate authority issues certificates. For multi-factor authentication, certificates of web server and smart card are signed by trusted certificate. Web server can verify smart card’s public certificate from certificate authority as well as on Android smart phone, application can confirm web server’s public certificate for implementing FIPS 196 protocol.

3.3 Remote Web Server

This module is on server side to communicate between web server and Android smart phone for authentication purpose.

3.4 IDMS

MySql is used as database on server to store user credentials and also to store user’s registered devices and tokens. members table stores email address and password for login on web. reg users table stores registered devices using Google Cloud Messaging. tokens table stores authenticated tokens generated using FIPS 196 protocol and session id.

3.5 Web Client Login

This module is on server side to handle user interface. User is asked to provide login credentials to get access. This module handles user input.

3.6 Authentication Server

Authentication server is responsible to allow only authorized users to access to a secure area. This module also send and receive commands for authentication between web server and smart card via Android smart phone.
4. PROOF OF CONCEPT AND RESULTS

4.1 Dependencies

The proof-of-concept SmartAuth application is designed for Google Android v4.4 with NFC enabled hardware support to use contactless smart card. SmartAuth uses spongy castle library to perform cryptography related operations e.g. verification of public certificates and uses open source Android application for lock screen functionality [36]. SmartAuth requires contactless smart card with JCOP (Java Card OpenPlatform) an open source Java Card PKI applet based on ISO-7816 and implementation of PKI standards. A web server running servers side PHP language for authentication remotely and also for multi-factor authentication using smart card.

4.2 Development of SmartAuth Android Application

To built Android application, Eclipse IDE with ADT (Android Development Tools) kit plugin is used. Android application is written predominantly using Java programming language. Like Java Object Oriented Programming, activities of applications are organized as classes and events can trigger those classes. As presented architecture of Multi-factor Authentication on Web using Smart Card App SmartAuthentication.apk of SmartAuth in section 3.1.

In this section we described development and implementation. The structure of the application is similar to the previous section. This application requires permissions for example \texttt{com.google.android.c2dm.permission.RECEIVE} to be able to receive messages from GCM 2.3.1 besides internet access with \texttt{android.permission.INTERNET} and NFC access \texttt{android.permission.NFC} defined in \texttt{SmartAuthentication's AndroidManifest.xml}.

4.3 MainActivity.java Activity

This application uses Android’s layout structure to interact and display to user defined in \texttt{res \rightarrow layout \rightarrow activity main.xml}. The main activity performs major function including but not limited to register device with Google Cloud Messaging Service and remote web server against user’s email address as shown in Figure 7. Asks user for smart card PIN verification.

\texttt{onCreate} method is called when application starts. In \texttt{onCreate} method application presents prototype layout to register device with Google Cloud Messaging Service and remote web server against user’s email. \texttt{setOnClickListener} checks if button is clicked to register GCM as shown code in Figure 7. When user clicks button to register GCM it calls \texttt{onClick} method of a new instance \texttt{OnClickListener}. In this method, it first checks for Google Play Service is installed as it is required for GCM to work properly. Then it gets registration id from GCM and posts to remote web server as shown in Figure 8.

Detection of smart card by application via NFC is only required if an authentication request is received from web server for multi-factor authentication. Detection for smart card by application is similar as previous sections, registered in \texttt{onResume} method by creating and initializing intent receiver \texttt{IntentFilter} and adding extra flags \texttt{filter.addAction()}. \texttt{onTagDiscovered} method is called when a smart card is discovered. In \texttt{onTagDiscovered} method \texttt{SmartCard} class object is initiated to communicate with smart card and remote web server to mutually authenticate for multi-factor authentication. Main activity is passed \texttt{MainActivity.this} to \texttt{SmartCard} class object to dynamically display text or information on main activity from \texttt{SmartCard} class.

When smart card is detected, application asks user for smart card PIN verification by calling \texttt{getPin()} method. When user clicks on button ‘Verify PIN’ after entering PIN to verify smart card ownership, \texttt{setPin(byte[] pin)} method is called with user’s entered PIN for smart card verification via NFC and to respond web server for mutual authentication.

4.4 GCMBroadcastReceiver.java Class

This application registers for Broadcast Receiver by GCM. When a message is received by GCM, \texttt{onReceive} method of this class is called with \texttt{intent} received by GCM Broadcast receiver. Received \texttt{intent} is passed to the \texttt{GCMIntentService} class.
4.5 GCMIntentService.java Class

Message in intent from GCM is passed to onHandleIntent method of GCMIntentService.java class. Notification message is displayed to user by using Android’s API method NotificationCompat.Builder and MainActivity.class is called when user clicks on the notification message as shown in Figure 9.

4.5.1. SmartCard.java Class

SmartCard.java class is the implementation of the smart card module architecture presented in section 3.2. SmartCard.java class handles all communication between a smart card via NFC using APDU commands and also communicate with remote web server to authenticate. SmartCard class initializes some private and string variables and also calling its constructor which sets a private variable of type activity with the activity passed when initializing SmartCard class object as shown in Figure 10.

When a smart card is discovered via NFC, it passes to setTag(final Tag tag) method. TextView is passed to the setTextView(TextView mTextView) method for displaying text on activity and setIsoDep() method is called to set tag technology to IsoDep for smart card communication and also increases smart card communication time out. Every smart card has a unique id. getTagId method returns the smart card unique id in hex string. selectPKIApplet method with exception handling calls selectPKIAppletNFC method and selects the PKI applet installed in smart card as a smart card can have multiple applications or applets installed. The APDU command for applet selection is as follows:

00A404000CA000000063504B43532D3135

The APDU command is sent using isoDep.transceive() to connected smart card via NFC. If the APDU command executes successfully on smart card then the response would be 9000. PKI applet provides different
modes including digital signature, encrypting and decrypting data. To select a specific mode, an APDU command is sent for a specific mode operation. setSignPKI() method is called to select mode to compute digital signature from smart card via NFC and the APDU command is:

002241B60784020002800102

verifyPinPKI() method checks and verifies the ownership of the smart card with PIN verification. PIN is padded with hex 3 before send and the APDU command for PIN verification is as follows:

0020000008313131313131313131

If the PIN verifications is successful by smart card then the response would be 9000. Smart card ownership is now verified with PIN verification and now can compute digital signature.

getSignaturePKI() method is called for steps to mutually authenticate with remote web server. authenticationRequest() is called to send authentication request to web based cloud server. In response server sends back TokenBA1 i.e. a random number message digest getChallengePKI() method is called to get challenge from smart card via NFC i.e 8 bytes random number from smart card. Below is the APDU command to get challenge from smart card:

0084000008

Challenge and TokenBA1 from remote web server are concatenated using Java’s method ByteBuffer.allocate() as $dataAB SHA-1 message digest is calculated from $dataAB. 20 bytes of message digest are then sent to smart card to digitally sign with smart card’s private key based on RSA algorithm. The APDU command to compute digital signature:

002A9E9A21301F300706052B0E03021A0414−< 20−Bytes−of−SHA1 > 80

In response from smart card, we have digital signature of SHA-1 message digest signed by smart card as signA. signA is posted back as response in JSON format to the remote web server along getChallengePKI, authenticationRequest and readSignatureCertificatePKI smart card public certificate as TokenAB. To read smart card public certificate, the APDU command is:

00A4000002410300

A cloud based remote web server responses with TokenBA2 after receiving and verifying digital signature signed by smart card. TokenBA2 is digital signature signed by remote web server. verifyCA method is called to verify web server’s public certificate with CA certificate and to verify digital signature signed by remote web server to authenticate. On this step successful verification of TokenBA2 means that the responder, B (Server), has authenticated itself to the initiator, A (Android using Smart Card via NFC), and thus the entities have successfully mutually authenticated. The APDU commands and responses are shown in Figure 11.

4.6 Generate Certificates using OpenSSL

OpenSSL [34] an open source general purpose cryptography library to generate key pairs for public cryptography. OpenSSL is used to generate self signed certificate authority and user certificates based on PKI related standard for Fips 196 protocol implementation to load on remote server and smart card. Three different keys are generated for three different modes of smart cards. Then certificates are generated from keys. Certificates are signed by self-signed CA with required information including user information and user’s Email id. Certificates are now converted into x509 format. Finally keys are converted to make compatible with pkcs8 format to use with smart card.

4.7 Load PKIapplet.cap in Smart Card

PKIapplet.cap is an open source applet of PKI related standard with implementation of Java Card of the ISO-7816 standard [37]. GlobalPlatformPro an open source command line tool for loading and managing applets for JavaCards. GlobalPlatformPro is used to load PKIapplet.cap in smart card as shown in Figure 12.

Java Card PKI [37] open source project also provides a Java based tool to load RSA key pairs and certificates to smart card with PKIapplet.cap.

4.8 Cloud Based Web Server

In this research DigitalOcean [38] a cloud computing provider is used to host web server for mutual authentication and multi-factor authentication for our proof of concept prototype. DigitalOcean provides infrastructure to host web applications with SSH access. The server configurations are 512 MB RAM Memory, 1 CPU with 2 core processor, 20 GB SSD Disk Space, Ubuntu 14.04.1 LTS OS, NGINX open source HTTP Server, MySQL database server and PHP server side scripting language for web applications.

4.9 Storing GCM Registration ID in Database

User posts GCM registration id against login/email id with extra information about Android device to web server. Web server stores this information in reg users database table using PHP as shown in Figure 13.
4.10 Sign In on Web with Multi-Factor Authentication

A simple sign in page with login credentials is setup as shown in Figure 14. A user enters his login credentials, email address as login id and password to enter into a secure area. PHP checks user’s credentials saved in members database table with SHA1 conversion of password. If login credentials are verified then user proceeds to next layer of security to mutually authenticate using smart card. Next user prompts to select the Android device which he registered against his id as presented in Figure 15. An authentication request is sent to his Android device to mutually authenticate when user selects his Android device for multi-factor authentication using smart card. User need to authenticate using smart card via NFC on his Android device to get authenticate on web. This authentication request is registered in tokens database table against user’s email address and valid for only 120 seconds. Using AJAX query is sent to check periodically if user has mutually authenticated with smart card. If user fails to authenticate with smart card in 120 seconds, he’ll require to resend the authentication request to his Android device so that he can mutually authenticate using smart card as shown in Figure 16. If smart card authentication is verified, user is now have successful mutual authentication/Fips196 between web server and smart card using Android shown in Figure 17. User is redirected and now have access to secure area.

4.11 Fips196.php Class

Fips196.php class is created to perform secure functions on server for public key mutual authentication with smart card. Main features of Fips196.php class are: verify tokenID() verifies that the request id is the same of the current object and sets tokenBA1 into class variable. get rand() is called to get cryptographically secure random numbers to use for Fips196 protocol using PHP. tokenAB() generates TokenAB by digitally signing SHA1 hash of RA and RB with private key and echo in JSON format along public certificate as presented in Figure 18. verify tokenBA2() verifies TokenBA2 with self signed CA certificate and updates tokens table with received TokenBA2 as this point the responder, B (Android using Smart Card via NFC), has authenticated itself to the initiator, A ( Web Server ), and thus the entities have successfully mutually authenticated.

5. CONCLUSION

In this research, we have proposed a secure authentication framework for web using smart card and Android. This research has carried out in mainly three phases, we explored and have identified the issues related to security of authentication systems in web and shortcomings on tradition user/password web based authentication system with major issues including but limited to password reuse and an easy to guess secret. Based on our findings, we have proposed an authentication framework, the proposed framework can fulfill requirements to address security related issues which are identified. The architecture of proposed authentication system is based on mutual authentication with public key cryptography. Only an authorized user with an authorized smart card can mutually authenticate with cloud based remote web server. In our final stage, we have developed a prototype application as proof-of-concept for multi-factor authentication using Android and smart card on web.
Fig. 11. APDU commands and responses

Fig. 12. Loading of PKlapplet.cap

Fig. 13. Storing GCM Registration ID

Fig. 14. Sign In
6. FUTURE WORK

As of October 19, 2015 FIPS 196 is obsolete and is being withdrawn [33]. We are currently extending the proposed architecture to use elliptical cryptography [39] which provides more security and small keys as compared to RSA [39].

REFERENCES

[7] [7]


