# Chapter 1 – Related Work

1) EVM Contactless Credit Cards:

Keywords: EMV (Europay, Mastercard, Visa), POS (Point of Sale)

The EVM Contactless credit cards have six functional characteristics:

1. Visa credit cards approve unlimited amount to be transacted in foreign currency.
2. The contactless interface allows the transaction to be carried out even when still being in the cardholder’s wallet.
3. The cardholder’s PIN is not required to be entered to carry out the contactless transaction.
4. Visa cards approve the transaction in offline mode, which allows attack to be carried out without connecting to the card payment system, thereby avoiding any additional security checks.
5. The merchant details are not included in the data cryptographically protected by the card; this allows the details to be added later.
6. The EVM protocol requires the card to authenticate itself to the Point of Sale (POS) terminal but doesn’t make it mandatory for the POS to authenticate itself.

But when all these 6 characteristics line up (like the Swiss cheese model), fraudulent transactions can be carried out without the cardholder’s knowledge. For example, the Visa credit card will approve foreign currency transaction for any amount upto 999,999.999 Euro without the cardholder’s PIN. The fraudulent transaction details are then transmitted over the Internet to a rogue merchant, who uses the transaction data to take money from the victim’s account. Usually, the criminals would choose any amount between 100-200 Euro, which is low enough to be in the victim’s account and not raise suspicion.

Reference: Harvesting High Value Foreign Currency Transactions from EMV Contactless Credit Cards without the PIN.

2) IBM’s Method for Secure Credit Card Transaction:

This approach aims at storing user’s credit card information in his/her mobile phone, which is bluetooth or network-enabled. It is similar to mobile wallet, wherein the mobile phone acts as a purchasing system. The mobile phone doesn’t store the credit card information, but instead a credential that identifies the user and the credit card company (Eg. Chase, Citibank, Discover, etc.). So if the cardholder wants to pay for the restaurant bill, his/her credit card information is never exposed to the server or to anyone else. A hand-held device is used by the server to connect with the user’s mobile phone to share information about the transaction via an encrypted session.

The user’s device contacts the credit card company to request a temporary credit card number for the transaction. The temporary card number would be associated with the specific retailer, for a specified amount (including tip) and for a specific date and time. This temporary number is passed back to the retailer, who then processes it like a normal transaction.

Another application of this method include Point of Sale(POS) terminals where a customer checking out of a retail store like Walmart can use his/her mobile device to interact with the POS instead of exposing credit card information by swiping it through the magnetic strip reader.

Problem of this method - If the mobile is lost/stolen, the information stored in it can be misused.

3) Full EVM Smartcard:

Reference: The Implementation of a Full EMV Smartcard for a Point-of-Sale Transaction and its Impact on the PCI DSS

4) Safepay:

Existing magnetic credit cards adopt plain text to store confidential information, making it vulnerable to untrusted magnetic card reader. To deal with such credit card information leakages, the use of IC cards (also known as EVM) that has strong security authentication is recommended. However, IC cards face backward compatibility issues and therefore all existing EVM cards have magnetic strip to maintain backward compatibility.

Safe-Pay protects user’s credit card information while still being backward compatible with the existing magnetic stripe readers. First, the user needs to download the mobile banking application. Then during transaction, the mobile app acquires disposable credit card numbers from the bank server, generates a wave file, plays the file to generate electrical current, and then drives the magnetic card chip via audio jack or Bluetooth. Disposable credit card information is a virtual credit card number that expires after a limited time or number of usage (eg. One time). So, the misuse of the same credit card number will not take place even if an attack is carried out.

Reference: SafePay: Protecting against Credit Card Forgery with Existing Magnetic Card Readers

5) Mutual authentication between cardholder and merchant:

Credit cards are widely used in on-site transactions and also for online shopping. With it’s rising popularity, credit cards are becoming more vulnerable to various attacks like

6) Credit Card Verification System by Biometric Method

Ways and means to steal the credit card identity and numbers:

1)   Shoulder surfing

2)   DB Stealing

3)   Packet Intercepting

4)   Dumpster Diving

The first use of this system, requires the user to register his or her fingerprints with the credit card company, or a bank or a store along with the personal credit or debit card information. At the very first instance, payment has to be done with fingerprint scan and residential password.

Registration process involves the following steps:

1. Registration of credit card information which involves dissemination of credit card information using card reader after swiping the card
2. Fingerprint registration can be done using continuous inputs of three times on fingerprint sensor. It also takes into consideration verification of fingerprint to be registered and then transferring to the central server for registration.
3. Just in case, a request can be made to the server for modification and deletion of customer’s fingerprint and information.

When actually we use this system at POS, following procedures are followed:

1. When a fingerprint input is detected a confirmation check is done with the customer with his or her personal key. If the confirmation check fails, the user is asked to re-enter the fingerprint input else, card data request is made to the credit card company’s database.
2. The company’s database verifies this with the specific key that the user entered, if the company is able to confirm the credentials of the user the credit information of the customer is transferred to the POS from the server else the request is immediately terminated.
3. Once the credit information of the customer is received, an authorization check is performed by the POS terminal. If authorized, the payment is processed, else a message is sent to the customer and the transaction is denied.

7) Virtual credit card for mobile for m-Commerce payment.

This paper involves the development of the MCAT (Mobile Credit-card with Analytical Transaction) system. THis system has two main components namely the virtual credit card (VCC) application and the Wireless Credit Card Reader (WCCR), which would be attached to the POS machine for the cashiers. VCC application manages a range of credit cards the customer possibly has.

A wireless access point is generated so that the communication can take place between the two specified components for the purpose of transaction. The process to carry out payment though involves the VCC to connect to a specific WCCR, in case a store has many WCCR.

1. Customer enters the IP address of the WCCR notified by the cashier to connect to it.
2. VCC shall display the name of the WCCR if the connection is successful.
3. The customer can then select any one of the credit card to make the payment. During the entire process the data that is being disseminated to the WCCR is card number, cardholder name, card network and the security code.
4. WCCR verifies the authorization of the credentials from the card issuing company.
5. If accepted by the company, the WCCR will receive an acceptance notification from the card issuer.
6. WCCR sends the purchase details to the VCC to store in its’ database and at the same time transaction is processed.

8) Combining Biometric ID Cards and Online Credit Card Transaction

This project aims at implementing the credit card security by linking the online credit card transaction information with biometric ID card like the Turkish e-ID pilot system.  Biometric ID card provides multi - factor authentication (MFA). Turkish e-ID system offers various methods for verification of a person’s identity. For example, a photo that is imprinted on card for a basic visual check, PIN and biometric information.

The transaction is done using 3 - way authentication of the customer:

1. Select the product you want to buy online
2. The merchant creates an online ODP (orders detail package)
3. User is supposed to enter the PIN of the credit card along with the information such as the credit card number.
4. Turkish e-ID card is used for authentication
5. The system asks for the fingerprint input.
6. The user inputs the fingerprint using sensor on the Card Access Device
7. Perform the biometric authentication
8. Once authenticated the transaction is accepted.

Note: If any step results in the output to be false, the transaction is denied with an immediate effect.

9) Mobile Phone based RFID architecture for secure electronic Payment using RFID credit cards

The author aims at redesigning the credit cards such that they have an embedded RFID chip within them. The RFID credit cards would have the encrypted information that could be decrypted with the use of private keys that would be available with the user.  This private key is obtained by public key cryptography method that is stored at the RFID middleware collocated with middleware server.

The system intends to use a mobile phone with the capability of reading RFID chips. The mobile phone is responsible for registering the RFID credit card with the RFID middleware on the server. It would also download and stores encryption key provided by the RFID middleware after successful registration.

When performing a payment, the phone encrypts the RFID tag using the key that was obtained previously and transfers it with the use of user’s internet service. In case of credit card theft or loss or any kind of security compromises, the phone would have the capability to reset the entire registration phase.

The entire system works on the three phases:

1. Registration phase
2. Transaction phase
3. Reset phase

Though all these are done to ensure authorized person to use the credit card.

# Chapter 2 – Implementation

ZXing Library:

ZXing library is an open source project in order to scan and generate the barcodes and the QR (quick response) codes. This library helps the user to perform 2D or 3D scans on such codes. An abbreviation for the term Zebra Crossing absolutely, fits its name considering the kind of zebra striped code.

Working:

During the implementation of the project, the ZXing library was imported as is, this increased the application size to 18.7 MBs. Such a huge application size was undesirable and hence, the ZXing library was minimalized to be only Android specific, since the original library contained files which would help ZXing to be incorporated in all platforms like Windows, Android, iOS, Linux, etc.

# Table of Contents

1. UI Design
   * 1. Customer scan activity
     2. Vendor scan activity
     3. Amount confirmation
2. Database Creation (MS Access)
3. Application Coding
   1. Barcode scanner (ZXing Library)
   2. IMEI identification
   3. Async Task
4. PHP Creation for database CRUD functions
   1. SQL Queries
   2. MySQLi Procedural
5. PHP to Database Connectivity
   1. $conn HTTP protocol
   2. Uploading PHP file to GoDaddy server
6. Android to PHP Connectivity
   1. HTTP Connection protocol
   2. Accessing the server

Sub - Topic 1) UI Design:

UI Design was one of the most important aspect of this system. Considering the complexity, it was imperative to keep the UI as simple as possible because, any further difficulty in using the system would only drive the users away from it. Therefore, for the purpose of explanation, we have further classified the UID in three sub-categories which we will discuss in detail.

1. Customer activity:

This activity’s main focus is on the credit/debit card holder. A rectangular box with higher contrast as compared to the rest of the screen appears along with a red blinking line. The box and the red line together demarks the area in which the card holder has to scan the QR Code. This would enable user to not get confused about where to hold the phone’s camera to scan the QR Code.

A title is placed on the top of the activity stating “Scan the QR Code of the card”, which helps user to get a hint of what he or she has to do.

1. Vendor scan activity:

This activity has almost the same UID as that of the customer activity. The difference is just the activity title stating, “Scan the vendor’s QR Code”. The rest of the UI is reused.

1. Amount Confirmation:

This activity asks the vendor to enter the due amount of the entire transaction, which the customer will confirm. A plain text, which can accept numeric values only is shown with a submit button which will further go across to confirm the amount.

Sub - Topic 2) Database Creation (MS Access)

An access database was created. This database was populated with the data which would be required to show the transactions and indeed the application of security. The extension for this database is .mdb. The database has four fields viz. card\_no, name, IMEI and Balance.

Card No is the primary key in this database and also a 16 digits field size is pre-defined for it. Name is column with datatype as a text field. This text field has a character limit of 255 characters. IMEI is a unique key with 16 digits field size and balance is the amount that is present in the bank account of these users.

Other fields in this database could be added depending upon the future use and extension. If by chance a database is changed from MS Access to open source database it wouldn’t be a problem too because, the PHP code is irrespective of the code of the database used. Only the extension of the file must be changed should the database is chosen as open source.

Sub - Topic 3) Application coding

The programming language used to code the android application is java. For every activity in the android application, a java code is written. This java code accepts the values from the UI of the activity, which is in XML format and runs the code for further backend computations.

JAVA coding here had three extremely pivotal things to do viz. scanning the QR Codes, Getting the IMEI# of the cell phone, Asynchronous tasks to connect to the database from behind the scenes, hiding from the user the actual background processes.

1. Scanning the QR Code.

File name: CScan.java

Class name: CScan, AppCompatActivity

Library used: ZXing

Method containing the module: onCreate(), handleResult()

Corresponding UI: Customer Activity (activity\_cscan.xml)

Packages used: me.dm7.barcodescanner.zxing.ZXingScannerView, Alert Dialog, Bundle, zxing.Result, Toast

Description:

CScan.java involves the use of camera. In order to access the camera through an external application like Fortified Cards, permissions have to be gathered so that the user is made aware of the needs of the application. Hence, in the “Manifest” file gets the permission of “android.permission.CAMERA”

The library and the UI were overwriting the display of the camera with the demarked area. The scanner was initialized using the ZXingScannerView method of the ZXing Library. This initialization was used to invoke the camera of the phone.

The scanning process gave the scan results in the form of rawResult, which was further typecast as text. Since, in the database the QR Code is stored as String variable, we had to typecast the text form of rawResult to String. The string is saved in the variable named “scanresult”.

For the sake of finding out if the result is indeed correct a reference has been made on Alert Dialog Box through the object “alert1” which displays the value of text by typecasting rawResult.

1. IMEI identification.

File name: CScan.java

Class Name: CScan, AppCompatActivity

Method containing the module: handleResult()

Corresponding UI: Customer Activity (activity\_cscan.xml)

Packages used: Toast, TelephonyManager.

Description:

IMEI is a unique phone identification number containing 16 digits which can be obtained by dialing \*#06#. Since, it is a unique phone identification number and one of the sensitive data about a device, we have to make user aware of it and thus, ask the permission to do so. In the manifest file, a permission exists for the same as “android.permission.READ\_PHONE\_STATE”.

An object of the package “Telephony Manager” is used to get the IMEI number of the device. This package provides a method called as .getDeviceId(), which directly gives the IMEI number in a String format as the output. For the purpose of simplicity to understand the code further, a simple String variable imei is created and is made public.

The choice of keeping the string private or public is tricky. Since, the variable has to be passed on to the database in the near future to check the authenticity of the device scanning the card. As a result, the variable imei is kept public.

1. Asynchronous Task:

File name: CScan.java

Class Name: CScan, AsyncTask <>

Method containing the module: AsyncLogin

Packages used: AsyncTask, HttpURLConnection, MalformedURLException, URL

Description:

The user shouldn’t be made aware of the task that the application is performing in the background viz. connection to the database, the server HTTP protocol, etc. This not only saves the time delay but also, improvises the security of the application against the potential hackers.

Asynchronous task has in itself two methods dynamically overridden onPreExecute(), doInBackground().

onPreExecute() – This method shows a progress dialog “Loading”, this happens to let user know to hang on for a while.

doInBackground() – This method is actually the one which connects to the database in the background. The asynchronous task gets linked to the url of the GoDaddy server where the PHP file is placed. A connection timeout (setConnectTimeout) is set to 10000ms. If the server doesn’t respond within this time limit an exception is caught and a message “Connectivity Problem! Please check your internet connection” is displayed. A read timeout is set to 15000ms. This happens to check the response of the PHP file. If the CRUD functions on the PHP file don’t work within the read timeout, a message “Couldn’t read the database! Please try again later.”

This method also handles various exceptions like, “Malformed URL”. If the URL to be connected with the application is not appended properly to the connection string, an error is generated and the corresponding system error message is displayed.

Sub - Topic 4) PHP Creation for database with CRUD functions

Description:

A PHP code is used to connect the cross platforms of Android and the database server. It was imperative to choose the database to be online as compared to being on the device, since the database on the device is only going to create the application which has application size of more than 25 MBs which is against the requirement specification that has been developed by the developers before. Not only will the database presence increase the size of the application, but also the cross functionality of the application is lost if the database is just present within the application. Not to forget, if the database resides within the system itself, it becomes more vulnerable to attacks.

In this PHP code, we get the IMEI number and the scanned value from the Android java code. It is then received using the two variables $imei and $scan. CRUD functionalities are implemented using SQL Queries and since, the support for HTTPGet is deprecated by Android Studio we had to use the MySQLi Procedurals to process the queries.

The SQL query is: Select imei From CCDB Where “CCNo=$scan”

From the database the PHP code from the scan result checks what is the IMEI number of the Credit Card No. It stores this value in $imei and sends it back to the Android code which further checks if the $imei value is equal to String imei derived from the .getDeviceId() method earlier specified in the IMEI identification module.

If this value matches, the result is to proceed further to scan the vendor’s QR Code else an error message pops up “Incorrect Device! Use the registered phone to scan the card.”

Sub - Topic 5) PHP to database connectivity

Using the $conn = new com ( adodb.connection ) the PHP is initiated to connect to the database on the server. Before that the access file is uploaded to the GoDaddy server. Through the web hosting services the server is managed using a Linux platform cPanel. In the file manager, the .mdb file is uploaded. This file is given the read privileges.

$conn has to be followed by a COM class object since, PHP 5.1 and above direct connection has been deprecated.

Sub - Topic 6) Android to PHP connectivity.

A try – catch block is created to instantiate the connection between the two platforms. This try catch block seeks for the HTTP protocol with opening the ports for connection using HttpURLConnection)url.openConnection()

When the connection is established, a link is passed to the index.php, where the code for further connectivity is written. This method sends strings of parameters along with it. These parameters essentially are a dummy variable vars, imei number and the scanresult,

The dummy variable is sent out to just adjust the connectivity criteria since, the doInBackground process accepts a range of parameters and two was just not enough to pass the two strings. When these three strings are received by the PHP code, the first dummy variable is just dumped or rather not used in the further computations of the Android Programming.

# Chapter 3 – Architecture

REST (Representational State Transfer) – The reason to use the RESTful style is because, we wanted the server to be off the extra load. Using the REST style, it’s the client who remembers everything it needs (for eg. The imei number, derived from the cellphone using the .getDeviceId() method and the scan result)