A Survey of Payment Token Vulnerabilities Towards Stronger Security with Fingerprint based Encryption on Samsung Pay

Co-authored by

Student: Yiming Sun

Primary advisor: Ron Ruhl

Secondary advisor: Hamman Samuel

Project report

Submitted to the Faculty of Graduate Studies,

Concordia University of Edmonton

In Partial Fulfillment of the

Requirements for the Final

Research Project for the Degree

MASTER OF INFORMATION SYSTEMS SECURITY MANAGEMENT

Concordia University of Edmonton

FACULTY OF GRADUATE STUDIES

Edmonton, Alberta

April 2018

A Survey of Payment Token Vulnerabilities Towards Stronger Security with Fingerprint based Encryption on Samsung Pay

Yiming Sun
Information Security and Assurance
Concordia University of Edmonton
Alberta, Canada
yimingsun100@gmail.com

Ron Ruhl
Information Security and Assurance
Concordia University of Edmonton
Alberta, Canada
ron.ruhl@concordia.ab.ca

Hamman Samuel
Information Security and Assurance
Concordia University of Edmonton
Alberta, Canada
hamman.samuel @concordia.ab.ca

***Abstract*— The use of payment tokens, based on EMV® specifications and the Payment Card Industry token standard, both propels the spread of mobile payment technologies and improves the security of Mobile Payments including protection of the original payment information and primary account numbers. However, some researchers have demonstrated that attacks on payment tokens through decoding the magnetic secure transmission or near field communication signal allows an attacker to use stolen tokens to complete malicious transactions or to guess new tokens through analysis of the token format. The stolen tokens are then used to make fraudulent transactions.**

In this research we examined Samsung Pay in order to design a novel theoretical security model using a fingerprint-based master key for unlock phone authentication, and transaction authentication and encryption. Samsung Pay is an application installed in a Secure Element in a Samsung Android device. In our theoretical security model presented, this master key can be created using one biometric fingerprint pattern or two merged patterns. Sub-keys can then be generated from this master key that can be applied to transaction encryption, payment token encryption and to protect the payment token in the Secure Element in the phone where the mobile EMV® customer information is stored.

Keywords*— *Mobile Payment, Tokenization, Samsung Pay, Payment Token, NFC, MST, Key Encryption, Fingerprint feature, Biometric Key generation PKI certificate.

I. INTRODUCTION

This research examines payment tokens used in mobile payment systems, such as Apple Pay, Android Pay and Samsung Pay. Current research on mobile payments show vulnerabilities and this includes tokenized systems used in mobile payments. Recent attacks on payment tokens on Samsung Pay, for example, illustrate the ease of carrying out some of these attacks. This research will analyze these vulnerabilities using a systematic literature survey and summarize the findings using standard modeling diagrams. This research will then propose a secure design of token based transaction protection for Samsung Pay using an additional biometric scheme to strengthen the mobile payment process. It is hoped this proposed scheme can be implemented by EMV® (EMV® specifications are used by several vendors including Europay, Mastercard and Visa) participants using mobile pay platforms.

In 1992 MasterCard company launched the first international payment card and since then physical payment cards have been widely adopted by consumers. With the development of telecommunication technologies, mobile phone payment has started to replace traditional payment card methods. Mobile payment has been almost 20 years in development since 1999 when two mobile operators in the Philippines launched the first commercial mobile payments [1].

Compared to conventional physical payment cards, mobile payment is based on the mobile phone as a payment platform through the loading payment cards, into a virtual device which uses Near Field Communication (NFC) technology or Magnetic Secure Transmission (MST) technology to communicate with point-of-sale (POS) terminals without inserting a payment card into the terminal [2]. In the current mobile pay industry, there are three main competitors; Apple Pay, Samsung Pay and Google Pay. Apple Pay and Samsung Pay use a chip called Secure Element (SE) embedded into phones to complete key protection of sensitive information and data. Unlike Apple Pay and Samsung Pay, Google Pay uses a Host Card Emulator (HCM) application simulating physical payment cards, and deploys cloud servers on remote-site for the protection of information and data. More details will be presented in Section III on all three types of mobile payment systems.

Unlike traditional payment cards, mobile payment technology can allow many virtual payment cards so that customers can add, delete, or update virtual cards much more conveniently. This creates a convenient portable approach to payment card use and management.

Using mobile payments also has the potential to combine online bank applications and payment cards allowing customers to track transaction records much easier. In addition, banks in the future may develop more useful monitoring and control methods which use the mobile payment platform to better protect customer financial information and tailor the bank service to the customer. All of the benefits cannot be offered by physical payment cards and mobile payment technology can provide enhanced security and enhanced customer service.

The EMV®is global payment industry specification for payment systems. This organization published the first EMV specification in 1996. It describes the requirements for the interoperability between the chip based on the physical payment cards and the payment terminal sale machine. EMV® also provides standards and requirements for mobile payments and this includes payment tokenization to protect customer payment card information [3]. EMV® participants publishes these standards at emvco.com.

II. Background

A. Theory of Tokenization

Tokenization is the process of replacing the Primary Account Number (PAN) with non-sensitive values called tokens. Thus, tokens become the most important value used in mobile payment transactions rather than the PAN [3]. Tokens used by EMV® participants are of two types: security tokens and payment tokens, although most of payment companies use payment tokens. Payment tokens are random values that replace a cardholder's PAN when initiating a payment transaction. In the EMV® specification [4], payment tokens map to the real PANs to provide additional security. Payment tokens are mostly generated by a token service provider (TSP) or the bank which issued the payment card. They can be dynamic, static, or a combination of both. Dynamic tokens are valid either for a single transaction or for a limited number of transactions within a very short duration. Dynamic tokens currently are not used very often for mobile payments because they use additional resources to manage the related TSP token mapping. Compared to dynamic tokens, static tokens are utilized by most mobile payments following EMV® specifications. They combine one static token with a dynamic component (the uniquely generated cryptogram used to complete the transaction) for additional security. In summary, the reasons for using static tokens are: they can be easily to deployed by card issuers; they use Bank Identification Numbers (BINs) to assign to different services and domains; and, merchants can easily link payment credentials to customers and discover fraudulent transactions (without having to store the underlying PAN) [4].

The Payment Card Industry Data Security Standard (PCI DSS) is a set of specifications for securing PCI data in systems using EMV® payment cards to complete transactions. Merchants have to comply to this standard through routine audits. Since tokens do not contain the PAN, tokens may be stored in a less secure manner and are not in scope for PCI DSS compliance.

B. Format of Tokenization

The PCI DSS Tokenization Guideline [5] shows the basic format of tokenization. There are three main types of tokens used by different payment network companies: American Express, where the first number starts with 3, and uses alphabetic and numeric value combinations; Visa, which starts with 4, and only uses numeric values; and Mastercard, which starts with 5 and uses a truncated PAN where the last four digits are the same as the real PAN's last four numbers.

C. Workflow of Tokenization

Token generation includes three main approaches:

- a mathematic reversible function with a known strong cryptographic key;
- a one-way hash function; or
- an index function, sequence number or a randomly generated number.

No matter which method is used, a token cannot be calculated by a simple mathematical calculation [6]. In the Samsung Pay official website, Samsung explains the traditional workflow of tokenization (Fig.1). However, no matter which mobile payment company a customer uses, a mobile phone has one token that is mapped to one real PAN. A token request must be sent to a TSP or a bank when a real credit card number gets enrolled.

When a credit card number is verified by the card issuer, the TSP will respond with a generated token sent to the mobile device. The sequence diagram in Figure.1 demonstrates the entire tokenization flow. In addition, following the EMV® payment tokenization specifications, the TSP also generates the Payment Account Reference (PAR) including a registered BIN controller. The BIN is used as a linkage and routing mechanism for processing token based transactions to ensure the correct bank receives the transaction token for processing.

PAR has 29 characters, containing PAR field, PAR data, PAR data generation method, PAR delivery mechanism and PAR enquiry method [6,7]. The first four characters show a tokenized BIN identifier assigned by EMV®, while the remaining 25 characters are based on each PAN. The 29-random characters cannot be reversed to reveal the real PAN number. The PAR just acts as a reference list, recording all necessary transaction information for the token based transaction. Moreover, PAR cannot be used by itself for any financial transaction [7,8]. Each PAN may have many different payment tokens (one per customer's mobile device) which map to the single customer PAN and PAR. [8]. While PAR is not a mandatory requirement that must be used in payment token transaction in PCI payment token security, the PAR mechanism proposed by EMV® solves the issue of different client tokens used in different devices while all of the tokens refer to the single PAN and PAR [9].

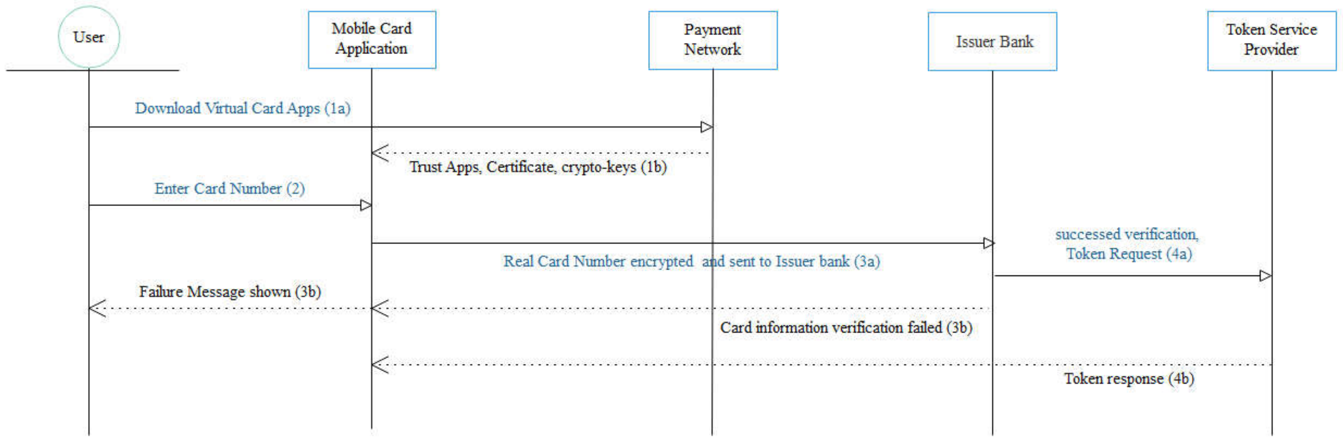

User
Mobile Card Application
Payment Network
Issuer Bank
Token Service Provider
Download Virtual Card Apps (1a)
Trust Apps, Certificate, crypto-keys (1b)
Enter Card Number (2)
Real Card Number encrypted and sent to Issuer bank (3a)
successed verification, Token Request (4a)
Failure Message shown (3b)
Card information verification failed (3b)
Token response (4b)

Fig.1. Overview of Tokenization

Figure 1 presents our synthesis of the tokenization workflow. The user downloads a virtual card application from Google Store or an online payment network which contains official EMV® authorized certificates, public keys, and private keys to be used with the trusted application in step (1a, 1b). Once the application is installed, the user enters card numbers (PAN) into the virtual card application in step (2) at which point the card application software will send them into the payment network and all card information will be verified by an issuer bank in step (3a) using the issuing banks procedures. If the card information is not correct, a failure message is sent back to the mobile phone and the user is prompted to enter the information again. Otherwise if successful, in step (4), a token request will be sent to the TSP. The TSP generates a new token matching the unique card number (PAN) and PAR token, and this is stored in the TSP vault. Upon first use a PAR token is also created. The payment token and PAR token are then sent back to the mobile phone along with a customer-specific EMV® certificates and keys used in processing transactions.

When a transaction is ready to complete and the authentication is done successfully, the PAR token accompanied with the payment token [7] is transmitted into the POS, and the POS will start to complete the rest of the steps [10] including routing the request to the appropriate bank based on the registered BIN range of the issuing bank shown in the PAR. Following the EMV® specification, a static token is used with a strong transaction cryptogram which is normally generated by the digital payment token, an application transaction counter and a secret key (known only by the issuing bank and the EMV® payment card downloaded to the). The cryptogram is a fully random and varied value to anyone without the secret key to decrypt it [11].

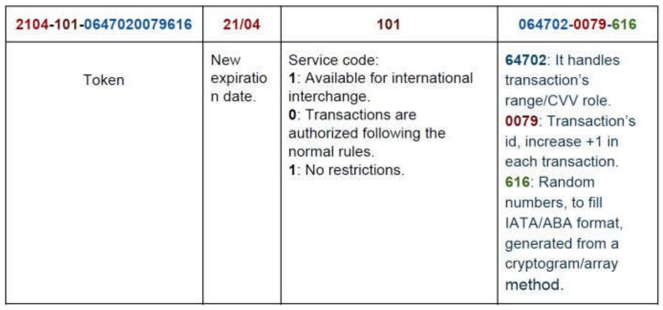
III. RELATED WORKS

Although Apple Pay, Android Pay and Samsung Pay use a similar process of tokenization, they opt for different approaches to store the tokens. Apple Pay [12,13] stores the Device Account Number (DAN) in Secure Element (SE). DAN (core payment data) is created by the payment network and it is encrypted in SE with either Elliptic Curve Cryptography (ECC) or Rivest–Shamir–Adleman (RSA) encryption. When users request a payment user authentication is checked in the Secure Enclave, which communicates with the SE. Then the payment token is generated by a SE including the transaction ID, payment network and payment token data.

Samsung Pay [10] works like Apple Pay although Samsung's Knox architecture is based on Android. One significant difference is that Samsung Pay uses an embedded chip as a SE working in a TEE, which is discussed in Section V. The Samsung Pay token and cryptogram is be put into the SE. When the payment is triggered, it creates a payment token with a cryptogram including token data, timestamps, and an application transaction counter.
Android Pay [14] uses a cloud technology to store payment tokens through isolating the physical device. Here, any payment token is sent to a Google cloud. In an Android phone, only Host Card Emulator (HCE) is used to connect to POS. So, although Android Pay and Samsung Pay use similar platforms, there is no SE in Android phones.

Salvarda [15] comprehensively analyzed Samsung token numbers that he collected from Samsung Pay and found that patterns existing in the token structure. The author explained one collected token in detail in Table 1.

Table 1. Collected Token from Samsung Pay [13]



2104-101-0647020079616	21/04	101	064702-0079-616
Token	New expiration date.	Service code: **1**: Available for international interchange. **0**: Transactions are authorized following the normal rules. **1**: No restrictions.	**64702**: It handles transaction's range/CVV role. **0079**: Transaction's id, increase +1 in each transaction. **616**: Random numbers, to fill IATA/ABA format, generated from a cryptogram/array method.

The format follows Visa token characteristics which have a 20-digit number. The first four numbers represent the token expiration date. The next three numbers represent the service mode. The remaining numbers represent other transaction related elements. Table 1 also shows that the number 064702 handles the transaction range and CVV (Card Verification Value) role, which is followed by 0079 which changes with each transaction. The last three numbers 616 are generated randomly. The CVV is the number printed on the back of a physical payment card and is normally used for card not present transactions.

After explaining the format of the collected token, the author describes how the collected tokens can be obtained with MST or NFC tools, such as "TokenGet" and "MagSpoof" which was illustrated at a Black Hat Conference [16]. Although the author explained the meaning of each number, he did not give a better solution to protect all token numbers. The main reason that the author can obtain the plaintext token is because all tokens stored in the mobile phone are not encrypted. Once any database file is leaked or decrypted or in transit, the clear text token number can be obtained and analyzed, and then used for future attacks [17].

Daeseon Choi and Younho Lee. [17] demonstrated how malicious users could use a magnetic card reader to obtain magnetic signals, decode signals and use refreshed stolen tokens to make a remote payment. In this paper, the author demonstrated exploits to other vulnerabilities existing in Samsung Pay. He showed that very strong MST signals can be stolen from a relatively long distance of about two meters. An attacker can decode stolen signals and obtain tokens transmitted in the signal. Although this paper did not talk about the stolen token in detail, an obvious issue of the transmitted token is that the token is visible in the communication stream. Due to the lack of connection between payment authentication and payment transaction, the stolen token can be used by malicious users simultaneously.

We created Table 2 to provide a summary of three mobile pay platforms and the main differences between the mobile platforms, their applications, storage payment tokens, token formats, transaction routing and the potential vulnerabilities.

Table 2. Evaluation Form of Three Mobile Pay Applications

Mobile Phone Platforms	Apple iOS	Android OS	
Mobile Payment Applications	Apple Pay	Google/Android Pay	Samsing Pay
Units sold in 2Q2016 [22]	44395(iOS)	296,912(Android)	
Market Shares % [22]	12.9 %	86.2 %	
Mobile Phone units sold in 2Q2016 [22]	44395 (12.9%) (iPhone)	76743(22.3%) (Samsung)	
Storage Location of Payment Tokens	An Embedded chip in phone (Secure Element)	Online Cloud (Google server)	An Embedded chip in phone (Secure Element).
Payment Token Format	Payment token includes: Transaction ID, Payment network, Payment token Data (signature, header, and payment data).	Payment token includes: DPAN, ExpirationMonth, ExpirationDay, Authmethod (3D secure) 3dsCryptogram etc. [18]	Payment token includes: Payment Data from DPAN, Timestamps, Account Transaction Counter.

Transaction Routing	Token BIN or BIN range (part of token, first six digitals) used to find payment network.	Token BIN or BIN range (part of token, first six digitals) used to find payment network.	Token BIN or BIN range (part of token, first six digitals) used to find payment network
Payment Token Encryption	Only payment data (amount, card name, process data etc.) using either elliptic curve cryptography (ECC) or RSA encryption in SE. When transaction starts, new token will be created	Payment token stored in Cloud. And in transaction, it is encrypted by Elliptic Curve Integrated Encryption Scheme [19]	Payment Data from DPAN encrypted. But the entire payment token is stored in SE as a plaintext, when transaction starts, cryptogram used to alter last 4-6 digitals to make a new token for each payment
Relationship between Authentication and Transaction	Step1. PIN or Fingerprint used as an unlock authentication. Step2. Before transaction starts, (in most cases), same fingerprint (recommended) or PIN authentication is requested again.	Step1. PIN, Passcode, Pattern or fingerprint used as an unlock authentication. Step2. No authentication requested before transaction.	Step1. PIN, Passcode or fingerprint used as an unlock authentication. Step2. Before transaction starts, (in most cases), same fingerprint (recommended) or PIN authentication is requested again.
Payment Token Invoke methods	Step1. Secure Enclave responsible for Touch ID authentication Step2. Apple API connects Secure Enclave and Secure Element to create plain payment token.	Step1. Trusted Auth Apps charge in unlock authentication. Step2. NFC controller directly talks to HCE (payment application) to request a payment token in cloud (Google server).	Step1. Trusted Auth Apps charge in unlock authentication. Step2. Trusted Auth Apps talks to SE via Trusted internal API to obtain an payment token.
Potential Vulnerabilities of payment applications	1. To intercept traffic to an Apple server, when payment data is added into device phone, hacker can push malware into phone to steal data. 2. Via public WiFi or 'fake' WiFi hotspot, to request users to create a profile, and hacker can steal Apple pay cryptogram, and decode cryptogram. 3. Run out of available of Token BIN 4. Hacker can use stolen card numbers to register and finish payment. etc. [20,21] 5. entire payment token is in plaintext, it can be reused or analyzed when payment traffic is jammed.	1. Due to the use of payment card emulation applications, so rogue HCE apps can be installed. 2. NFC controller directly talks to host CPU that can cause big pressure of CPU and consume lots of resources. 3. Run out of available of Token BIN 4. Malicious applications can attack Host OS and steal payment data on HCE app. 5. DoS attacks to the cloud. [21].	1. Secure storage is encrypted with static password, once it is compromised, Payment Token can be obtained and analyzed. 2. NFC and MST encode plaint payment token, once MST decoded, payment token. cryptogram and other information can be gained and be used again. 3. Run out of available of Token BIN.

IV. Description of Proposed Solution

After analysis of the vulnerabilities existing in the main three mobile payments methods listed above in Table 2, we examined Samsung Pay in order to propose the following solution to further secure payment transactions [23].

Our proposed model is based on several assumptions including:

- The payment network and all network protocols are sound, secure and efficient and that the https protocol and encryption is secure during the communication between the mobile phone and issuers.
- The mobile phone, POS machine, the bank, payment network providers, and TSP are secure and genuine. All certificates, tokens, encryption keys are secure and genuine when issued.
- The underlying software used in the mobile phone is safe without viruses and worms. All software is certified by merchants and no malicious software which might affect the SE is downloaded into the phone.

Our solution involves creating a fingerprint pattern-based value which will further secure specific transactions and is not based on the unlock key of the mobile device. How this could be embedded into the deployment of mobile payments is described below:

A. Initial Credit Card Application

A user needs to apply for a credit card in a bank or on a bank's website. During the application process, in addition to providing necessary personal identification information (PII) and security questions to the bank, they must enroll a fingerprint pattern using the bank device or their own device embedded in their mobile device. In our solution we will call this fingerprint pattern 1. (Note: this fingerprint pattern should not be the same as the fingerprint pattern enrolled for unlocking the user phone or mobile device.) This enrolled fingerprint will be applied not only to first Virtual Payment Card verification on the mobile phone, but also to fingerprint-based key generation.

B. Mobile phone unlock authentication

When a user starts to use their own mobile phone or device, the unlock authentication needs to be enrolled for the first-time use. In a standard Samsung mobile device there are two ways to unlock it:

i. The user uses PIN as an unlock phone method. When the PIN is entered into the phone via a mobile authentication application, the phone will generate an encrypted file (password.key) using SHA-1 and compare this to the file stored in the phone.

ii. The user uses fingerprint pattern named pattern as an unlock phone method. In our solution we will call this fingerprint pattern 2. When the fingerprint pattern is entered into the phone via a mobile authentication application, the phone will also generate an encrypted file in the Android phone by invoking `keygenerator()` and `keyStore()` function to obtain a key and then using `cipher()` function to input the fingerprint pattern to finish the encryption process. This pattern is stored into a file named unlock.

C. Fingerprint pattern and PAN enrollment

i. Fingerprint based binary value matrix created

When a user downloads Samsung Pay and enters or scans their payment card numbers to create the SE virtual machine for the payment card. In this research we will use Toronto Dominion (TD) Bank Visa as an example of an issuing bank. After the Visa card numbers are entered, the user needs to login to their online bank with their username, password and fingerprint pattern 1 in (1) to verify user identification. Here, Samsung Pay requests the user to confirm this enrolled fingerprint pattern 1 for the use of transaction authentication and this pattern is stored into a file named transaction. This fingerprint pattern 1 must be different from fingerprint pattern 2 used to unlock the phone. In case they are the same, the fingerprint pattern 2 in the unlock file will be deleted, and the user is asked to choose another pattern or to use PIN for unlock authentication. Samsung phone will send all information to the bank for the Virtual Visa Card registration.

This enrollment phase includes the following two cases based on the unlock mechanism in (1) (i.e. steps to unlock the phone):

i.1 If the user used a PIN for the phone unlock protection, the fingerprint Pattern 1 for the transaction will be extracted and binarized to create a binary stream and add a PIN and a hardware key to get a fingerprint binary value matrix file in the SE.

i.2 OR: If the user used the fingerprint pattern 2 for phone unlock protection, both fingerprint patterns will be merged to form a new cancellable pattern, and then minutiae will be extracted and binarized to create a binary stream. After this a hardware key (mobile device number) is added to get a fingerprint binary value matrix file in the SE specific to the customer and the mobile device hardware.

ii. PAN enrollment

The user can scan or enter the PAN into the card Virtual Machine (VM) application in the SE. The user then logs on to their secure online bank with username, password and fingerprint pattern 1 enrolled at the bank. Then the phone sends all information (PAN number, username, password and fingerprint pattern 1) and the fingerprint binary value matrix file created above to the bank in order to confirm who the user is in the web interface. If the verification is passed, the fingerprint binary value matrix file becomes the bank's master symmetric key for the cardholder. All the information will be encrypted by TD's public key and sent to the TD issuer, who can use TD's private key to decrypt all information. (Note: all fingerprint patterns will be binarized and the original pattern is never stored in the phone.)

After matching all card number information, the issuer sends a request to the TSP to generate a payment token. The TSP will send the payment token back to the bank. The issuing bank and TSP encrypt all communication using the bank/TSP previously configured interface.

D. *Fingerprint based transaction key generated*

i. Fingerprint based symmetric key created

TD bank uses the fingerprint binary value matrix file to extract a random binary stream and then the tokenized BIN value is added to create a symmetric key.

The bank then encrypts the token with a sub-key generated by a master key and puts the encrypted token object into the certificate issued to the user which contains a freshly created public key of the users' payment card (CusPuK) and corresponding private key (CusPrK). (Note that the resulting value encrypts the token which was already encrypted or hashed by the TSP as described in the workflow of tokenization above.) The TSP can map the token back to the real PAN of the customer at any time.

ii. User stores all information in SE

The user downloads all of Fingerprint pattern and PAN enrollment i.e. C (i) above to the SE:

a. Issuing bank's certificate (including PuK of bank signed by a CA).
b. PAR token which includes the registered BIN controller.
c. Certificate of cardholder (with PuK signed by the bank). Note the doubly encrypted token is in a data field of the certificate.
d. CusPk, CusPuk.
e. Tokenized BIN.
f. Encrypted Payment Token
g. PIN file or fingerprint pattern 2 matrix value file used to unlock the phone (if PIN is not used). Once this is completed the SE can then ask the user for fingerprint pattern 1 which will allow the SE to generate the binary value matrix and then sub-keys based on this. Sub-keys can be used in the SE to encrypt all files and to add Message Authentication Code on all transaction elements when is data is transmitted (i.e K_{Enc} and K_{MAC}) to the payment network. All information above will be stored in the SE which only the authorized user can access.

E. *Completion of a purchase*

When the phone is used to purchase something:

i. The user opens a card VM application.
ii. The user is required to enter a second fingerprint for the transaction beginning by pressing the fingerprint scanner on the phone with the appropriate finger previously enrolled.
iii. During this process, the POS terminal validates the VM EMV® certificate issued to the customer (checking the signatures of the certificates on the card to verify the chain of trust from the cardholder certificate up to the CA which the POS terminal must have installed on the POS).
iv. After that, the POS uses a challenge question encrypted by CA CusPuk sent to the SE, and if the SE responds to the question correctly using the CusPrK which matches the certificate public key, then EMV® card authentication is completed. The challenge used is a known challenge and completed without user interaction by the SE and the Samsung Pay application.
v. During this whole transaction process, fingerprint pattern 1 will get enrolled again and the binary value matrix file will be generated again. Then the SE can extract random the binary stream and combine the tokenized BIN to generate a symmetric key based on the known algorithm used by bank to create the same key. If the new generated key can decrypt the encrypted payment Token in the SE, then EMV® customer authentication is complete and the phone will derive the same key to encrypt the transaction. If the match is unsuccessful, the phone will show the unmatched message, and one will be added to a retry counter stored in the SE and the user is asked to try again (i.e. the same as if the customer was using a PIN and a physical EMV® card). Finally, after four unsuccessful attempts, the transaction will be cancelled. In SE, having reached the maximum retry number, the virtual payment card will be locked. Once the VM tokenized card is locked, this action taken by the VM will be communicated to the issuing bank. (Note: although the tokenized card is locked in the VM, the physical payment card would not be locked and may still be used.
vi. The transaction processing can then be completed by using K_{Enc} (symmetric key) to encrypt the transaction details and K_{MAC} (symmetric key hash value) to verify the integrity of the transaction. This step is mandatory. All keys are based on the master key known by the SE and the bank.

vii. The above is routed to the issuing bank (through the payment gateways the merchant uses to process transactions) and the issuing bank uses its similarly calculated keys to decrypt the transaction, verify its MAC, and decrypt the encrypted token to obtain the real token. The token is then sent to the TSP and a PAN is returned to the issuing bank.

viii. The bank either approves or rejects the transaction and sends the appropriate message back using the reference number for the transaction as the doubly encrypted token (i.e the one matching the token field in the certificate in the SE).

F. *Unlocking a locked VM tokenized card*

To unlock a locked VM payment card:

i. The user uses the same bank enrollment web interface to first prove identity and then request the VM tokenized card to be unlocked. Normally, the bank would automatically unlock the card using an automated response unless there was account activity which flagged that the cardholder must call the issuing bank. The bank might also provide the option to re-enroll the user's fingerprint.

ii. If re-enrolling and after answering several security questions, a new fingerprint-based key is recreated either in the bank or using the bank enrollment web interface. The user is the told to register the same card again. In registry process, the user has to enroll the second fingerprint pattern or PIN verification value (if PIN was used) again, and the matrix file and all necessary information are sent to the bank again. The bank verifies all information and sends new keys and certificates back to the user for storage in the SE. this the retry counter is set to zero.

There are two possible issues that can cause a VM card to be locked:

1. A ruined fingerprint enrollment by the user may cause information extracted to fail repeatedly.

2. The matrix file may be compromised by attackers. In either of the above cases, the user can request a new fingerprint enrollment, and revoke a previous one by either going to the bank or calling the bank or using a web interface if the issuing bank has enabled this.

False rejection rate (FRR) for biometric devices is published by each device manufacturer. FRR is the rate at which a genuine fingerprint is rejected. A common fingerprint FRR for many devices is 2% or 0.02. Using this value and a retry counter of 4 means that 0.02^4 or 0.00000016 times a user will need to re-enroll a fingerprint pattern. For many users with 10 or so transactions per day this will take several years before the fingerprint retry counter reaches four. Any successful match completes the transaction and the retry counter in the VM card is returned to zero.

False acceptance rate (FAR) is the rate at which a random user would be accepted. The rate of FAR is often the same rate as FRR. However, in our proposed model, two biometric fingerprints are used and this means that the chance of a malicious user first accessing the phone and then the SE with Fingerprint Numbel 1 is remote (0.02^2 or 0.0004). A malicious user attempting to make a payment with a stolen phone would cause the retry counter to quickly reach four.

V. New Payment Flow Model

A. *The selection of fingerprint key generation model*

To establish a direct connection between the payment authentication and the authorized payment transaction, the first concern, is to decide which feasible fingerprint key generation model can be applied to our solution. The article [24] proposed a new robust multimodal biometric system by enrolling two different biometric features; iris and fingerprint using a look-up table to store both hash values and biometric feature information.

B. Chen and V. Chandran [25] proposes a transform employed by the system as an iterative, chaotic, bispectral, one-way transform that accepts a one-dimensional vector input and is used to produce a magnitude and angle pair per iteration.

In our solution, we choose a novel fingerprint-based key generation model, proposed by Subhas Barman, Debasis Samanta and Samiran Chattopadhyay [26], using the Cartesian coordinates system transformation to extract different coordinates (x, y) to create a data matrix file extracted from fingerprint patterns. All the data will be edited via the define of (int) Array[] function, which can be developed with basic programming language (C++ or JAVA).

B. *Analysis of Samsung Phone framework and Development of New Payment Flow Model*

Apple iOS platform and the Android platform are the two popular development environments that most mobile phone companies use to develop their own smart phones. Like Samsung Phone, it employs the Samsung Knox hardware platform and keeps using the Secure Element Chip like Apple Pay, which is based on the Android development environment with ARM TrustZone Technology.

A system-on-chip (SoC) security architecture establishes two hardware-based zones; an external communication area called "Rich Execution Environment (REE), and an internal secure area named "Trust Execution Environment (TEE)". From the perspective of information technology security, all sensitive operations in the mobile phone, including storage, exchange, transmission, transaction and execution, must be implemented into TEE. For the highest level of security, a TEE deployment must meet requirements of TEE protection profile (TEE PP), which contains execution authentication, transit cryptography, application isolation, and untrusted applications. Therefore, any data or information exchange or transmission referring to a mobile payment must be processed in the TEE. [27,28].

The TEE software architecture is illustrated in Figure. 2 [29].

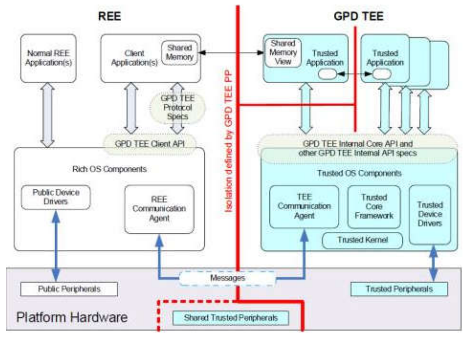


Fig.2. TEE software architecture [29]

This diagram basically elaborates an entire software deployment structure and relationship between REE and TEE. Any software a user installs must be checked with a whitelist, which is stored into the TEE service library containing all secure related drivers. All untrusted software must be installed into the REE and this includes all client payment applications, such as Google Pay, Android Pay or Samsung Pay. Otherwise, trusted applications (TAs) get deployed into the TEE directly, like Visa, Mastercard, and American Express payment applications, on a trusted operating system.

The only way to establish a connection between two zones is a TEE Application Programming Interface (TEE API). In the TEE structure, it includes six main APIs, in which TEE client API connects client apps in REE with trusted apps in TEE. The TEE internal API concentrates on the various interfaces and this enables a Trusted Application to make best use of the standard TEE capabilities. The TEE socket API provides a common modular interface for the TA to communicate to other network nodes, acting as a network client. The TEE secure element API supports communication to the SE.

A SE (one embedded chip with tamper resistance) serves as a platform existing in the TEE on the mobile phone and it stores all sensitive data, files and information, and can implement cryptogram calculation for payment tokens. After user authentication, any data or information can be extracted from the SE via the TEE SE APIs to finish all communication between TAs and SE in Figure.3 created by us.

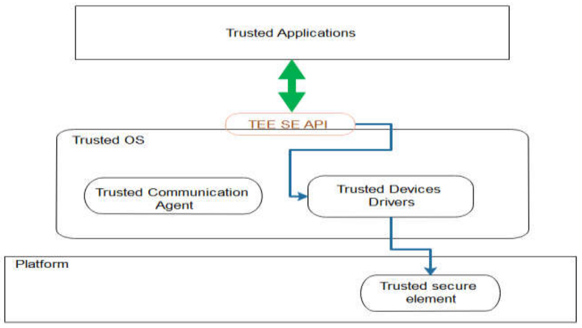
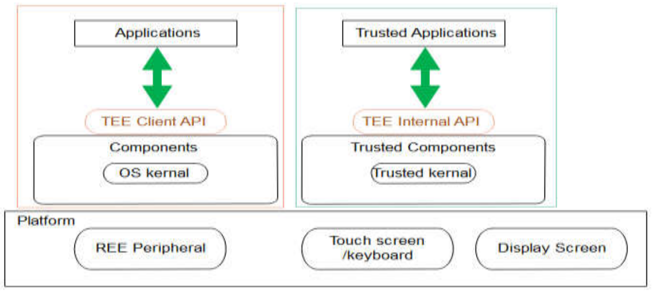


Fig.3 TEE SE API

While the TEE Internal Core API provides an interaction to execute all sensitive operations within a Trusted Application (TA) or between TAs running in the Trusted Execution Environment (TEE), certain applications need to display sensitive information to the user for validation or to obtain sensitive information from the user, such as input of payment card information, display of transaction information, etc. All operations must be implemented in TEE rather than in REE. So, TEE Trusted User Interface offers three main security objectives: security display, security input and security indicator.



F

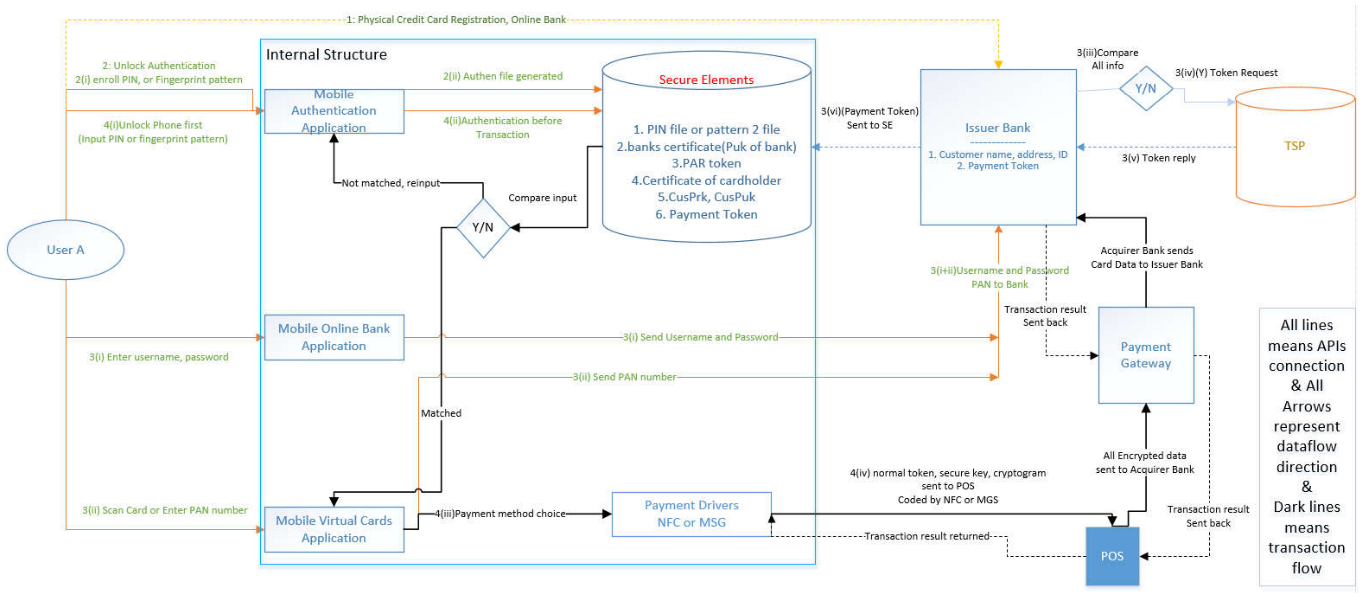
Fig.4. TEE Trusted UI

The diagram in Figure. 4 demonstrates a basic architecture of the Trusted User Interface. It consists of either a touch screen or keyboard peripheral and a display controller peripheral. When one of peripherals works, others must not be accessible to reading, writing or executing in REE. [29].

TEE APIs provide important functionality to build bridges for communication among different components not only between TEE and REE zones, but in each isolated zone. Therefore, mobile payment data and information can be obtained from REE (outside of TEE), and securely enter the TEE via TEE client API and TEE user API. The payment data and information can finish any invokes or operations between TAs or TAs and SE with TEE internal APIs and TEE internal core APIs.

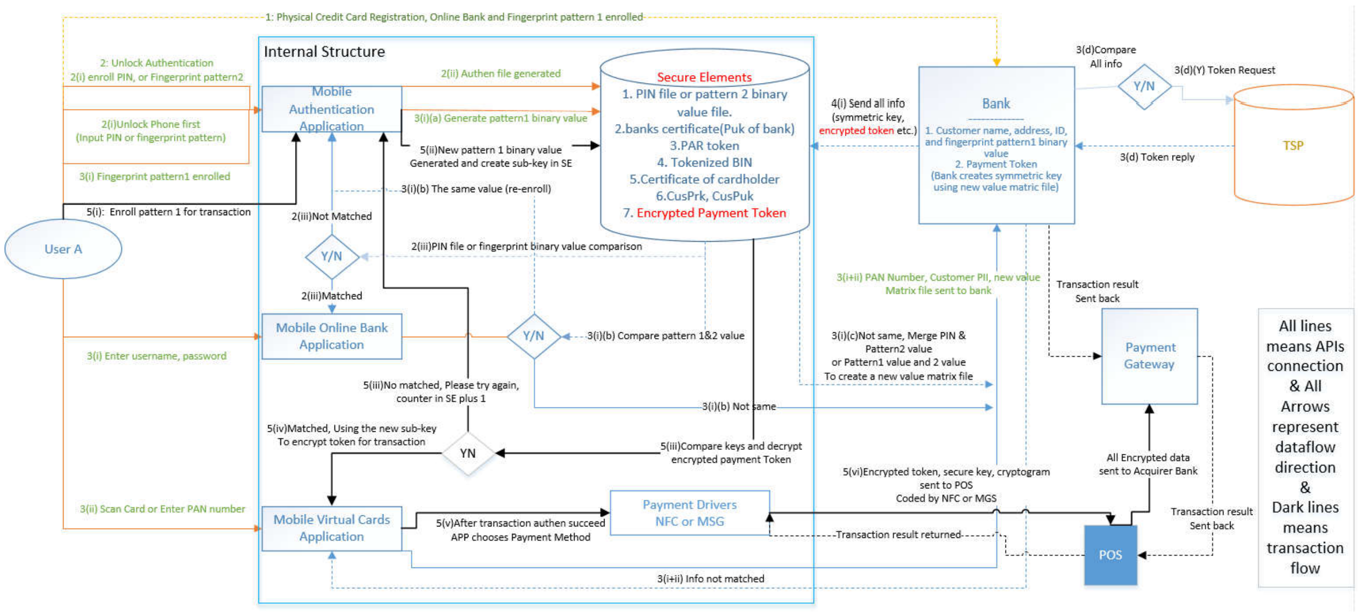
Figure 5 demonstrates the current payment flow on Samsung Phone based on our earlier discussion. The new payment flow is developed in Figure 6 and the locked/unlocked VM process in Figure 7.

In Figure 5 and 6, the full details about how the POS machine communicates through the payment gateway to the issuing bank is not illustrated. The payment gateway may include the merchant bank, acquirers and other outsourced payment services. In our solution, once the payment transaction is encrypted and sent by MST or NFC signal to the payment gateway, the user's fingerprint pattern 1 has been used to prove customer authentication, then to decrypt the payment token, and after this to create the transaction which is encrypted and routed to the issuing bank indicated by the BIN controller value. This process can prove the user is the genuine user and only the issuing bank can properly decrypt the transaction details to provide the transaction verification and response.



1: Physical Credit Card Registration, Online Bank
Internal Structure
2: Unlock Authentication
2(i) enroll PIN, or Fingerprint pattern
4(i)Unlock Phone first
(Input PIN or fingerprint pattern)
Mobile Authentication Application
2(ii) Authen file generated
4(ii)Authentication before Transaction
Secure Elements
1. PIN file or pattern 2 file
2.banks certificate(Puk of bank)
3.PAR token
4.Certificate of cardholder
5.CusPrk, CusPuk
6. Payment Token
Not matched, reinput
Compare input
Y/N
User A
Mobile Online Bank Application
3(i) Enter username, password
3(i) Send Username and Password
3(ii) Send PAN number
Matched
3(ii) Scan Card or Enter PAN number
Mobile Virtual Cards Application
4(iii)Payment method choice
Payment Drivers NFC or MSG
Transaction result returned
3(vi)(Payment Token) Sent to SE
Issuer Bank
1. Customer name, address, ID
2. Payment Token
3(iii)Compare All info
Y/N
3(iv)(Y) Token Request
TSP
3(v) Token reply
3(i+ii)Username and Password PAN to Bank
Transaction result Sent back
Acquirer Bank sends Card Data to Issuer Bank
Payment Gateway
4(iv) normal token, secure key, cryptogram sent to POS Coded by NFC or MGS
All Encrypted data sent to Acquirer Bank
Transaction result Sent back
POS
All lines means APIs connection & All Arrows represent dataflow direction & Dark lines means transaction flow

Fig.5. Current Payment Flow on Samsung Phone



1: Physical Credit Card Registration, Online Bank and Fingerprint pattern 1 enrolled
Internal Structure
2: Unlock Authentication
2(i) enroll PIN, or Fingerprint pattern2
2(i)Unlock Phone first
(Input PIN or fingerprint pattern)
3(i) Fingerprint pattern1 enrolled
Mobile Authentication Application
2(ii) Authen file generated
3(i)(a) Generate pattern1 binary value
5(ii)New pattern 1 binary value Generated and create sub-key in SE
3(i)(b) The same value (re-enroll)
5(i): Enroll pattern 1 for transaction
2(iii)Not Matched
Y/N
2(iii)PIN file or fingerprint binary value comparison
2(iii)Matched
User A
Mobile Online Bank Application
3(i) Enter username, password
Y/N
3(i)(b) Compare pattern 1&2 value
5(iii)No matched, Please try again, counter in SE plus 1
5(iv)Matched, Using the new sub-key To encrypt token for transaction
YN
5(iii)Compare keys and decrypt encrypted payment Token
3(ii) Scan Card or Enter PAN number
Mobile Virtual Cards Application
5(v)After transaction authen succeed APP chooses Payment Method
Payment Drivers NFC or MSG
Transaction result returned
3(i+ii) Info not matched
Secure Elements
1. PIN file or pattern 2 binary value file.
2.banks certificate(Puk of bank)
3.PAR token
4. Tokenized BIN
5.Certificate of cardholder
6.CusPrk, CusPuk
7. Encrypted Payment Token
4(i) Send all info (symmetric key, encrypted token etc.)
Bank
1. Customer name, address, ID, and fingerprint pattern1 binary value
2. Payment Token
(Bank creates symmetric key using new value matric file)
3(d)Compare All info
Y/N
3(d)(Y) Token Request
TSP
3(d) Token reply
3(i+ii) PAN Number, Customer PII, new value Matrix file sent to bank
3(i)(c)Not same, Merge PIN & Pattern2 value or Pattern1 value and 2 value To create a new value matrix file
3(i)(b) Not same
Transaction result Sent back
Payment Gateway
5(vi)Encrypted token, secure key, cryptogram sent to POS Coded by NFC or MGS
All Encrypted data sent to Acquirer Bank
Transaction result Sent back
POS
All lines means APIs connection & All Arrows represent dataflow direction & Dark lines means transaction flow

Fig.6. Process of PAN Enrollment, Fingerprint based key generation and Transaction on Samsung Phone

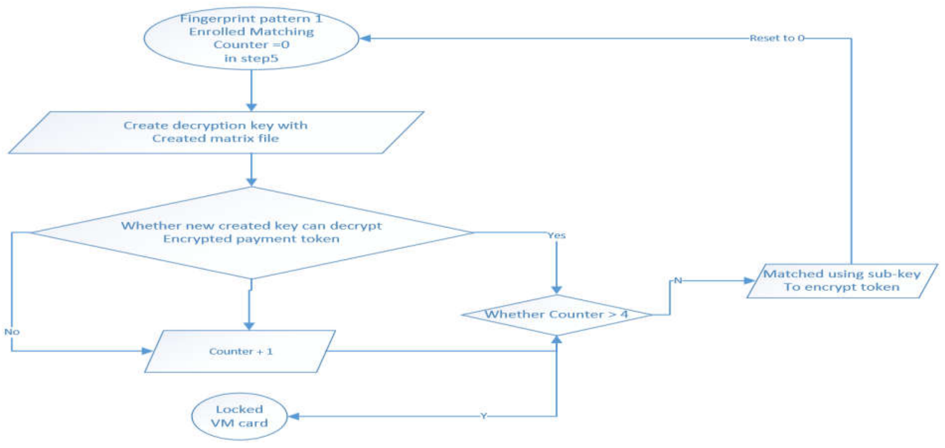


Fig. 7. Part 6: locked VM Process

VI. Benefits and Feasibility of Our Solution

A. *Benefits*

i. The payment token is encrypted using one key from the TSP and the other key from the fingerprint created by the bank. The original (encrypted, derived hashed or random value) token is never visible during the transaction nor is it present in the cardholder VM certificate.
ii. The fingerprint symmetric key can be a master key to generate sub-keys, which can be used to protect files in the SE.
iii. During each transaction, a symmetric key must be created by the transaction fingerprint authentication process described in this research. Moreover, sub-keys are created to decrypt encrypted files in the SE.
iv. During the transaction, the symmetric key must match the key from the bank to verify that: (1.) the original source to create this key is not compromised; (2.) the symmetric key is not compromised; and, (3.) the bank is the genuine bank.
v. The symmetric key must be used for the transaction otherwise the transaction rejected by the issuing bank.
vi. Non-repudiation security is an integral component of the payment transaction.

B. *Feasibility*

1) *Customer perspective*

From the customer perspective our solution will create minimal change to what they are accustomed to in mobile payments. For example, in making the payment customers unlock the phone and then use a fingerprint to approve the transaction. This should take little time and is as easy as entering a PIN. Enrolling the fingerprints with the bank in initial setup of the mobile payment app in the phone with the bank will take longer but can be minimized by allowing the customer to do this step online as part of the issuing bank card web application process.

2) *Issuing bank's perspective*

Banks will have a real cost in deploying this since they will need to generate additional code to facilitate our proposed system. If issuing banks further decide to use online bank access to load required fingerprints, then the system created would be largely a self-service system. However, once the proposed system is deployed and the addition code costs incurred, the bank will have a stronger means to combat fraud through the use of a PIN and a fingerprint or two fingerprints together with an encrypted token.

3) *EMV® perspective*

Our proposed solution will result in some implementation costs for EMV®; such as:

- Creating new specifications for handling tokens (i.e. encrypting the token obtained from a TSP through the use of fingerprint pattern 1in addition to the phone fingerprint pattern 2 or PIN).
- Implementing this proposal with industry participants and gaining their support.

In summary, after computer code costs are incurred in producing, testing and implementing our solution, much of the system can be self-serve from the customer perspective and the bank perspective. Our solution might also have unforeseen benefits as the customer might come to appreciate the increase in security around their mobile payments as they interact with the issuing bank and the self-serve tools for enrollment and re-enrollment. Customers might well be more inclined to use mobile payment systems and this could benefit both industry participants and customers alike by reduced payment card fraud.

VII. Conclusion

Through our study, research and analysis of the current mobile payment infrastructure and related payment vulnerabilities, we developed a method to enhance security in the mobile payment and transactions process in Samsung Pay by using multiple authentication and fingerprint based encryption keys. Further research can be constructed to test the industrial feasibility of the proposed model.

VIII. References

[1] Polymathconsulting.com "A Brief History of Payments" October" 2015.

[2] Vibha Raina. "Overview of Mobile Payment: Technologies and Security" in *Electronic Payment Systems for Competative Advantage in E-Commerce.* February 2014. pp 31.

[3] A Guide to EMV. Version 1.0 May 2011. Marianne Crowe and Susan Pandy.

[4] Technical Framework. "EMV®, Payment Tokenisation Specification. EMVCo, LLC. Version 2." September 2017.

[5] Scoping SIG, Tokenization Taskforce PCI Security Standards Council. *PCI Data Security Standard (PCI DSS)* "PCI DSS Tokenization Guideline." Version 2. August 2011.

[6] EMV®Co. "EMV®Co Publishes Version 2 of the EMV® Payment Tokenisation Technical Framework." September 2017.

[7] Chandra Srivastava. "Payment Account Reference Overview" presented at Smart Card Alliance Payments Summit April 2016. Available: https://www.securetechalliance.org/secure/events/20160404/CORAL-SEA-1-2_WED_1045_SRIVASTAVA_SCA-Payment-Summit-Tokenization-PAR-presentation-April-6-Chandra-Srivastava-Visa.pdf

[8] "EMV®, Tokenization, and the Changing Payment Space". Version 1.0. September 2015.

[9] David Bakker. *"TOKENIZATION & TICKETING: 5 TAKEAWAYS."* May 2016. [Online]. Available: https://blog.ul-ts.com/posts/tokenization-ticketing-5-take-aways/

[10] Mobile Tech Insights. Samsung Pay, Tokenization. Samsung Developer Website. [Online]. Available: http://developer.samsung.com/tech-insights/pay/tokenization

[11] Press_Guidance_Samsung_Pay. August 2016. [Online]. Available: http://security.samsungmobile.com/doc/Press_Guidance_Samsung_Pay.pdf

[12] *iOS Security* "iOS Security, iOS 11" in January 2018.

[13] "Payment Token Format Reference". Apple Inc. [US]. [Online]. Available: https://developer.apple.com/library/content/documentation/PassKit/Reference/PaymentTokenJSON/PaymentTokenJSON.html

[14] Ganeshji Marwaha. "Mobile Payments: What is HCE?". Sept 2014. [Online]. Available: http://www.gmarwaha.com/blog/2014/09/20/mobile-payments-what-is-hce/

[15] Salvador Mendoza. "Samsung Pay: Tokenized Numbers, Flaws and Issues". August 2016. [Online]. Available: https://www.blackhat.com/docs/us-16/materials/us-16-Mendoza-Samsung-Pay-Tokenized-Numbers-Flaws-And-Issues-wp.pdf

[16] Salvador Mendoza. "Flaw in Samsung Pay lets hackers wirelessly skim credit cards" presented at Black Hat Conference. August 2016. [Online]. Available: http://www.zdnet.com/article/flaw-in-samsung-pay-lets-hackers-wirelessly-skim-credit-cards/

[17] Daeseon Choi, Younho Lee. "Eavesdropping one-time tokens over magnetic secure transmission in Samsung Pay". June 2016. [Online]. Available: https://www.usenix.org/system/files/conference/woot16/woot16-paper-choi.pdf

[18] Android Developer. "Android API". [Online]. Available: https: developers.google.com/android-pay/integration/payment-token-cryptography

[19] John Leyden "Wallet-snatch hack: ApplePay 'vulnerable to attack', claim researchers". Jul 2017. [Online]. Available: https://www.theregister.co.uk/2017/07/28/applepay_vuln/

[20] Thomas Fox-Brewster , "Here's Proof Apple Pay Is Useful For Stealing People's Money". Mar 2016. [Online]. Available: https://www.forbes.com/sites/thomasbrewster/2016/03/01/apple-pay-fraud-test/#7b1cfaec46c6

[21] Sdog-mitre "New PAY threat: Vulnerabilities of HCE-based NFC Mobile Payment for code tampering and cryptographic key lifting attacks #52". Oct 2016. [Online]. Available: https://github.com/usnistgov/mobile-threat-catalogue/issues/52

[22] Chance Miler. "Latest Gartner data shows iOS vs Android battle shaping up much like Mac vs Windows". Aug 2016. [Online]. Available: https://9to5mac.com/2016/08/18/android-ios-smartphone-market-share/

[23] *PCI Security Standards Council.* "Tokenization Product Security Guidelines" Version 1.0.

[24] Nemanja Maček, Borislav Đorđević, Jelena Gavrilović, Komlen Lalović. "An Approach to Robust Biometric Key Generation System Design". Nov 2015. [Online]. Available: http://www.uniobuda.hu/journal/Macek_Dordevic_Gavrilovic_Lalovic_64.pdf

[25] B. Chen and V. Chandran. "Biometric Based Cryptographic Key Generation from Faces". 2007. [Online]. Available: http://eprints.qut.edu.au/10642/1/B._Chen_DICTA_2007_-_Crypto_Key.pdf

[26] Subhas Barman, Debasis Samanta and Samiran Chattopadhyay. "Fingerprint based crypto-biometric system for network security". December 2015. [Online]. Available: https://link.springer.com/article/10.1186/s13635-015-0020-1

[27] Mobile Tech Insight, Samsung Pay, Device-side Security: Samsung Pay, TrustZone, and the TEE. [Online]. Available: http://developer.samsung.com/tech-insights/pay/device-side-security#token-handling

[28] GPD_TEE_SystemArch_v1.1_Public_Release. [Online]. Available: https://www.globalplatform.org/specificationsdevice.asp

[29] Global Platform_Trusted_User_Interface_API_v1.0. [Online]. Available: https://www.globalplatform.org/specificationsdevice.asp