Identity-Enabled Transactions Based on the EMVCo Payment Tokenization Specification

Authors:

Yue Zhu

Asmaa Aljohani

Gyan Singh Namdhari

Mentors:

Dr. Seth Nielson Maria Vachino

December 12, 2016

Table of Contents

Abstract

Instances of attacks that target payment systems have raised the awareness to reshape the traditional payment infrastructure. As a response, Europay, MasterCard, and Visa network payment systems established the EMVCo framework to ensure secure and robust payment infrastructures. The framework aims at facilitating "consistent, secure and globally interoperable digital payments when using a mobile handset, tablet, personal computer or other smart devices."[1] To ensure secure identity transactions, we designed and implemented an identity-enabled transaction system based on the EMVCo payment tokenization technologies. In particular, the proposed system uses the tokenization concept that is adopted by major mobile payment technologies to assure both the security and privacy of identity-enabled transactions. In addition, a secure protocol to prevent threats such as replay attacks is implemented.

[1] EMVCo. (n.d.). Retrieved October 19, 2016, from https://www.emvco.com/faq.aspx?id=270

Introduction

In 2013, approximately 40 million credit cards were stolen from 200 Target stores by accessing data stored in Point of Sale systems.[2] Such an incident has raised the awareness to replace the "outdated payment system" that relies on magnetic stripe cards that offer no protection against threats with a new payment system that relies on technologies that provide more protection against such incidents. Payment tokenization systems hold substantial promise to provide such protection.[3] Mobile payment tokenization technologies such as Apple Pay, Android Pay, and Samsung Pay, which are based on the EMVCo Tokenization Specifications, are increasingly available for Point of Sale Payments. This research project investigates various aspects of using this technology for conducting identity enabled transactions.

EMVCo Tokenization Specification provides general guidelines to entities involved in a payment process. However, this specification is not an ideal industry standard. After reading through the document and investigating mobile payment technologies that use the framework, we found many ambiguities in the specification, specifically when it comes to the actual implementation. As an example, it has been mentioned in the specification that a token must be stored in "an approved secure element." A closer look at some major mobile payment service providers, one can see that each service provider implements the specification based on its own favor and capability. The methods used by Android Pay and Apple Pay to store tokens, for instance, are different. In other words, Android Pay uses the Host Card Emulation (HCE) technology to store the token, while Apple Pay uses the Secure Element (SE) instead. Each one of these adopted technologies has its own pros and cons. In general, the way that is used to implement mobile payment systems using the EMVCo specification is highly independent.

As a result, designing a secure identity-enabled system based on the EMVCo framework in a way that brings up clearness regarding the EMVCo specification becomes the new goal of this project.

As a system that focuses on providing an identity service, besides general security concerns, privacy is another focus that worth to be discussed. Adversaries who target an identity-based system are also interested in stealing sensitive identity information, such as driver's license number and social security number. For example, when a customer makes a purchase at an alcohol store, it is important to think about if the alcohol store needs to have knowledge regarding customers' identity information, or it only needs to know if the customer is above 21 or not rather than knowing the exact date of birth.

In general, this research project ends up with designing a system that provides a secure identity service by using the tokenization technology. Before delving into the details of the proposed system, this report will provide an overview of the EMVCo framework components, requirements, recommendations, and how to use the framework to design our proposed system. Then, it will present major impediments that were encountered and justifications of how these impediments can bring challenges for future implementations. Next, this report will propose a design of a secure system based on the EMVCo framework, and potential challenges along with recommended countermeasures will be provided.

[2] Teri Radichel, 2014, Case Study: Critical Controls that Could Have Prevented Target Breach
[3] EMV Payment Tokenisation Specification – Technical Framework, March 2014, Version 1.0.

About Mobile Payment Systems

Mobile payment systems, such as Apple Pay, Android Pay and Samsung Pay, are reshaping our payment conventions, which are either making a purchase with cash or sliding debit/credit card. To adapt to new payment technologies, the way the Point of Sale (POS) terminal works also is changed. Details about this section can be found in Appendix I.

When it comes to the communication technologies that are used in order for the mobile payment applications to communicate with the POS terminals, there are two common technologies that are used. Almost all mobile payment systems use Near Field Communication (NFC) technologies. Samsung Pay, however, provides the user with an additional option, known as the Magnetic Secure Transmission (MST) technology. In brief, MST is a type of technology that mimics a magnetic stripe card in order to be accepted at all POS terminals.

Among Mobile Payment Systems

One of the most common features among most mobile payment systems is the use of NFC technologies and special NFC enabled POS terminals that are required in order to send packets to or receive packets from mobile devices.

The other most common feature, which is also the focus of this project, is the use of tokenization technologies. Different service providers use different technologies to store and secure tokens. As mentioned before, Apple uses Secure Element (SE), and Android Pay uses the Host Card Emulation (HCE.) Details about SE and HCE can be found in Appendix II.

Based on research and meetings with practitioners who have knowledge of these major payment systems, there was a lack of well-documented resources related to how these mobile payment systems implement the EMVCo tokenization framework. However, we found that it is possible to integrate the identity management into mobile payment systems. For example, Android Pay allows merchants to sign up and integrate their loyalty cards as part of Android Pay. When customers pay with Android Pay, they can use their mobile device as a loyalty card first. For future integrations, Google can enable the identity management within Android Pay.

EMVCo Framework Briefs and Highlights

Tokenization

In brief, the tokenization is a process of turning a meaningful piece of data into a random one. One efficient way to find out the original value of a given token is to use a lookup table that stores original value-token mappings. This process differs from the encryption in that the encryption obfuscates data and makes it inaccessible unless a user has the proper key for the decryption.[4]

Tokenization is used for many different types of purposes, and it makes the interception of data useless since a random value is meaningless if intercepted. It is one of the technologies adopted by EMVCo and is used as a vital requirement in the implementation of secure mobile payment systems. One of the biggest challenges we faced regarding the implementation of the EMVCo framework is the lack of sufficient details about the tokenization. For example, the EMVCo framework does not specify what types of algorithms or methods should be used to tokenize a primary account number. We found that, however, one of the most common ways to do the tokenization is using hash functions. The purpose of the hash function is to transform data into a unique value, and ideally there should be no collisions among each unique value. Nonetheless, security considerations must be taken into account to avoid threats such as dictionary attacks. In other words, if a hash function is applied directly without adding a salt, adversaries may conduct brute force attacks to find out the primary account number of the targeted token; the reason is that the length of the primary account number is fixed (e.g., 11-13 digits). Additionally, hash functions are available to the public.

Ecosystem

According to the EMVCo Payment Tokenization Framework, the EMVCo Ecosystem consists of six different entities:

- User
- Merchant
- Acquirer (Usually Merchant's bank)
- Payment Network
- Token Service Provider (TSP)
- Issuer (Usually User's card issuer)

Figure 1 below shows an example of how a payment token flows through six different entities when a user uses the token to make payments:

1. The token is passed from Card Holder to Merchant, and from Merchant to Acquirer.
2. Acquirer passes the token to Payment Network, and Payment Network processes the request.
3. Payment Network interacts with Token Service Provider to exchange the token with its associated Personal Account Number (PAN).
4. Payment network then replaces tokens and token-related data fields with PAN and PAN-related data, and passes it to Issuer.

[4] S. (n.d.). Tokenization vs Encryption - Benefits & Uses Cases Explained. Retrieved November 03, 2016, from https://www.skyhighnetworks.com/cloud-security-university/tokenization-vs-encryption/

5. Issuer receives PAN information and either approves the transaction or declines it and passes an authorized packet back to Payment Network for processing.
6. Payment network processes it according to Issuer's authorized information, and passes final result back to Acquirer, then Merchant, then finally Card Holders.

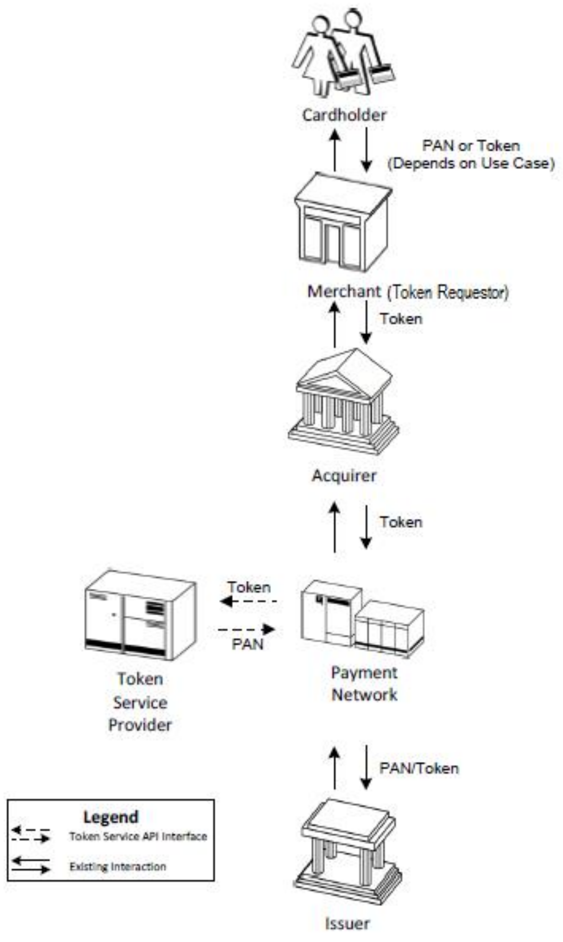


Figure 1: Payment Token Ecosystem. [5]

Besides Payment Token Ecosystem, the EMVCo also introduces processes such as Token Issuance. Token Requestor Registration, and Token Assurance. Among all different processes, situations may vary from case to case, and every entity is assigned responsibilities based on a specific use case.

For Token Payment, Token Issuance and Token Requestor Registration, one of the most important entities that is involved in the processes is the Token Requestor. In general, the Token Requestor is a legitimate entity who is allowed to send detokenization queries. The next section is going to explain the EMVCo's specifications of the Token Requestor.

[5] Payment Tokenization Technical Framework issued by ... - EMVCo. (n.d.). Retrieved October 10, 2016, from https://www.emvco.com/faq.aspx?id=270

EMVCo Token Requestor Specifications

According to the EMVCo Mobile Payment Tokenization Framework, Token Requestors can be different entities within the EMVCo Ecosystem depending on specific use cases.

Case 1: Card Holder as Token Requestor

In a situation where a user is the Token Requestor, the user is the entity that sends out a payment request with a token added to the request packet. The payment token can be stored in an NFC-enabled mobile device, and when the user uses an NFC-enabled mobile device to make payments at a merchant's POS terminal, a token is passed on behalf of user.[6]

Case 2: Merchant as Token Requestor

In a situation where a merchant is the Token Requestor, the merchant is the entity that sends out a payment request with a token added to the request packet. When a Cardholder initiates a payment request at an e-commerce Merchant that supports a specific digital wallet, the wallet will pass the Payment Token in lieu of the PAN along with additional Payment Token-related fields through the wallet API to the Merchant.[7]

Case 3: Acquirer as Token Requestor

In a situation where an acquirer is the Token Requestor, the acquirer usually is responsible for capturing and clearing flow progress. The information for Capture File processing is created by the Acquirer based on the information provided by the consumer during a transaction initiation.[8]

Case 4: Card Issuer as Token Requestor

Card Issuer is the Token Requestor in the case when a Payment Token is processed for a chargeback request.[9]

Identity Tokenization System

In the design of our identity-enabled system, we focused on one of the previous use cases of the Token Requestor. In other words, we assumed that the Card Holder is the only token requestor that initiates the tokenization and detokenization queries. . This design is based on few elements after considerations:

1. ID can only be used by the person for whom the ID was issued.
2. Identity Authentication Tokenization System is only designed to be used by a user to be identified officially, and there should be no other cases for other entities to use people's IDs.
3. ID Token has the same effect as the user ID. Although a token is useless without token-ID mapping, it still has to be treated as same as normal ID.

The next section will discuss the EMVCo Token Assurance Specifications, ID&V Methods Specifications and Token Service Provider Requirements.

[6] EMV Payment Tokenisation Specification – Technical Framework, March 2014, Version 1.0.
[7] Ibid
[8] Ibid
[9] Ibid

EMVCo Token Assurance Specifications

Token Domain Restriction Controls

There are three ways to perform Token Domain Restriction Controls: Token Requestor ID, POS Entry Modes, and Merchant Information. These Controls are intended to ensure that any exposure of Payment Tokens does not result in significant levels of subsequent fraud. The permitted Token Domain Restriction Controls for a given Token Requestor Registration are approved by the Token Service Provider, and these controls SHALL be stored in the Token Vault, which is managed and maintained by the Token Service Provide.

1. **Token Requestor ID**

   Each Token Requestor ID assigned by a TSP SHALL be unique and not conflict with other assigned Token Requestor IDs from that same TSP or another TSP. Token Requestor ID is an 11-digit numeric value assigned by the TSP with the following convention:

   - Positions 1-3: TSP Code
   - Positions 4-11: Assigned by TSP for each requesting entity and Token Domain

2. **POS Entry Modes**

   The use of POS Entry Mode values that are carried in the POS Entry Mode Code field to limit the use of Tokens to only those POS Entry Modes agreed to during Token Requestor Registration.

3. **Merchant Information**

   If the Merchant is the Token Requestor, Merchant-related data elements, such as the Card Acceptor ID in a combination with Acquirer-identifying data elements SHOULD be used to limit the use of a Payment Token by comparing these fields in the transaction processing messages with controls maintained in the Token Vault.[10]

[10] EMV Payment Tokenisation Specification – Technical Framework, March 2014, Version 1.0.

EMVCo Token Assurance ID&V Methods Specifications

According to the EMVCo Framework Documentation, there are four different concepts and ID&V methods for Assurance: the account verification, the TSP risk score, the TSP risk score with Token Requestor data, and the Card Issuer authentication of the Cardholder.[11] There are different cases related to performing ID&V methods. These cases are as follows:

Case 1: No ID&V Performed
In this case, no ID&V method is performed. When the Payment Token has been issued without any ID&V step performed at the time of Token Issuance.[12]

Case 2: Account Verification
In this case, the ID&V method is performed to see if the PAN is valid.[13]

Case 3: Token Service Provider Assurance
In this case, the TSP provides ID&V method to perform a risk-based assessment of the likelihood that the request to Tokenize a PAN is assured with sufficient level of confidence.[14]

Case 4: Token Service Provider Assurance with Requestor Data
In this case, TSP provides ID&V method that involves the use of data elements provided by the Token Requestor that can be used as indicative of Fraud. For example:

- Account age and history
- Bill to/ship to addresses and contact information
- IP address
- Device ID and device Information
- Geo Location
- Transaction velocity

The TSP defines methods to communicate the assurance level and the ID&V steps performed to the applicable parties, including the Card Issuer.[15]

Case 5: Card Issuer Verification of the Cardholder
In this case, Card Issuer provides ID&V methods that include, but not limited to:

- Use of a 3-D Secure ACS
- Mobile banking verification of the Cardholder with an authentication code
- Federated login systems
- API functionality capable of generating, delivering, and validating data from the Token Requestor
- One-time password (OTP), activation code, or other shared secret between the Card Issuer and the Cardholder
- Two-way email confirmation

[11] Ibid
[12] Ibid
[13] Ibid
[14] Ibid
[15] Ibid

In the case of using the OTP or activation code, the shared secret SHOULD be delivered to the consumer through an out-of-band channel, and static authentication data and Enrollment in an authentication service at the time of Cardholder authentication should not be used.

For OTP, Card issuer should follow the following standards:

- OTPs SHOULD have a length of at least 6 and no more than 8 characters
- OTPs SHOULD be generated in a manner such that they are not predictable
- Preferred method for delivery is a secure channel from the Card Issuer to the consumer device.[16]

Token Service Provider Requirement

Token Service Provider is the most important entity within this system. It is the only entity that performs tokenization, and moreover, it is responsible for other important tasks such as: detokenization, validate provided ID with issuer, and other potentially assurance methods.

This section provides descriptions of the services Token Service Provider should offer, the exact data flow of the processes, and reasons why this data flow and format must be followed.

About Tokenization and Detokenization
Tokenization is the process of substituting one value with another, and detokenization is the process of retrieving back the original value of a token by using a mapping table.

There are many different methods to perform tokenization, and one of the most straightforward way is to use hash functions. Hash function theoretically is a one-way function that outputs a unique and equal-length values for different inputs. However, using hash function only to tokenize an ID is not an ideal and secure solution. The reason is that the length of the Primary Account Number or ID is usually not very long; for example, the length of a driver's license ID is only 9 digits long. If a Tokenization Service Provider tokenizes a 9 digits long ID by using a hash function only, an adversary can easily find out the original value of a token by a brute force attack. An adversary only needs to know what hash function is used and try at most 9^{10} times to find out all tokens and relevant original values. In order to make the brute-force attack extremely difficult, a random salt value should be added as part of the input to the hash function. In this way, the input is not only the ID number. Instead, the input to the hash function is an ID plus a randomly generated salt.

For a first time user who requests tokenization service, a Token Requestor ID should be assigned if the validation with the issuer succeeds. Token Requestor Number is an identifier for Token Requestor. Token Requestor (the User) is an entity that initiates the communication by sending out the token. A Token Requestor ID can be linked to many different Tokens, but one token is only linked to one Token Requestor ID because each User can have different types of ID.

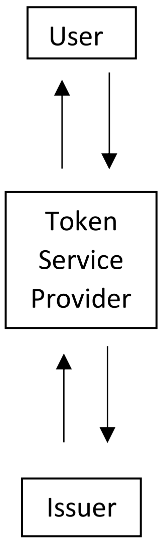
It is worth mentioning that before the Tokenization and Detokenization begin the Token Service Provide must validate the status of the Primary Account Number or ID with Issuer. Only valid and active Accounts and IDs can be tokenized or detokenized. Additionally, Detokenization can only be conducted after verifying Token Requestor ID and Merchant ID. Furthermore, Token Service

[16] EMV Payment Tokenisation Specification – Technical Framework, March 2014, Version 1.0.

Provider is responsible for ensuring and maintaining the security of the Token Vault, a storage of token mapping tables, by using effective defense mechanisms.

Identity Tokenization Ecosystem

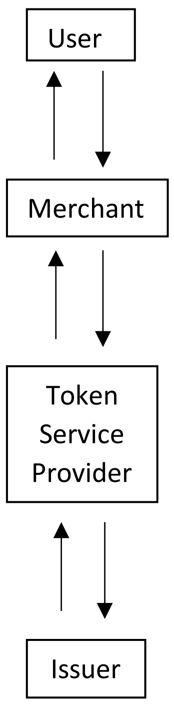
Tokenization



The flow diagram above describes the Tokenization process. The Tokenization process takes place when a user adds an ID for the first time use. In the case, User will be asked to input ID information in order to be verified. After user's input, User's packet is sent to Token Service Provider first. Token Service Provider will validate the provided ID information with Issuer; after receiving a validation from Issuer, Token Service Provider generates Token Requestor ID, Token, and Token Expiry Date for User. In this way, the first time User does not only acquire a Token from the Token Service Provider, but also he/she acquires a Token Expiry Date and Token Requestor ID. Each Token Requestor ID can be linked to several different Tokens. In the case when the User wants to remove or add more IDs, the Token Service Provider can link or unlink tokens to/from relevant Token Requestor ID.

The EMVCo Framework does not specify when the Token Requestor ID is assigned. However, according to our meeting with practitioners of mobile payment systems, Token Requestor ID is assigned during the first time use. It is important to mention that the Token Requestor ID represents a unique identifier to the Token Service Provider.

Detokenization



The flow diagram above describes the detokenization process. The detokenization process takes place when a user uses his/her digital ID. First, the user initiates the communication by using digital ID at Merchant’s side. Merchant passes the packet to Token Service Provider in order to detokenize the Token. Token Service Provider will first validate Token Requestor ID, Merchant ID and Token Expiry Date. The Token Requestor ID is essential during the detokenization process.

The following sections will talk about what each entity should know and should not. It is extremely important to clarify this because identity information is sensitive and must be kept confidential.

Entities

This section provides details about the functionalities of each entity within the Identity Tokenization System.

User

In the real world, the User is the owner of his/her ID, and within this system, the User is also the owner of his/her ID's relevant Token. User also initiates communications for both Tokenization and Detokenization process. In addition, the User's mobile device is the medium to make transmissions.

Merchant

In this system, besides the responsibilities of traditional merchants, Merchant also must provide a terminal to accept Token. Moreover, Merchant is responsible for passing User's packet back and forth and add additional information as required.

Token Service Provider

Token Service Provider is responsible for several essential functionalities in the whole system:

- Tokenization
- De-Tokenization
- Token Requestor ID Issuance
- Merchant ID Issuance
- Token Assurance

Token Service Provider can be a separate entity from Issuer, but it can also be an entity that cooperates with Issuer, or even it can be the Issuer. The idea here is, Token Service Provider actually accesses sensitive ID data. Practically, it might be easier if the Issuer is the Token Service Provider.

Issuer

Issuer, in our system, is almost the same as traditional ones. Additionally, it provides services to validate ID. Issuer can also provide an assurance method in order to improve the overall security level. This topic will be discussed later. In the next section, flow diagrams will be introduced to illustrate the design.

Knowledge of Each Entity

This section aims at providing details about each entity within the whole ecosystem. In brief, the title of this section can be informally paraphrased as "who knows what and why". This section is also essential for discussing potential security and privacy concerns.

User

User only knows the following data elements:

- ID
    - User is the owner of the ID
- ID Expiry Date
    - User knows the expiry date of ID
- Token Requestor ID
    - User knows the Token Requestor ID as User will be the Token Requestor
- Token
    - User knows the Token after the Token is issued
- Token Expiry Date
    - User knows the expiry date of the Token

Merchant

Merchant only knows the following data elements:

- Merchant ID
    - Each Merchant must obtain a valid Merchant ID before providing service
- *Query
    - This data element is optional
    - If queries are assigned to registered Merchant, Merchant may include Query as an additional element in order to identify a user for some special scenarios
        - E.g. An alcohol merchant may send query to ask if user's age is 21 or above or not

Token Service Provider (TSP)

Token Service Provider only knows the following data elements:

- Token Requestor ID
    - Token Requestor ID is issued by TSP
- Merchant ID
    - Merchant ID is issued by TSP
- Token
    - Token is issued by TSP
- Token Expiry Date
    - Token Expiry Date is defined by TSP
- *ID
    - ID is stored after it is verified by Issuer
- *ID Expiry Date
    - ID Expiry Date is stored after it is verified by Issuer

Issuer

Issuer only knows the following data elements:

- ID
    - Issuer issued ID
- ID Expiry Date
    - Issuer defined ID Expiry Dates

Justifications of Entities' Knowledge
When it comes to the Identity Tokenization System, one of the biggest concerns is privacy. Traditionally, when a merchant requires a customer to show his/her ID to prove their age, the merchant does not record it. In this way, merchant verifies customer's ID information manually and does not have any hardcopy of it.

Each entity within this system only knows things that are related to their own role. When a User's packet is passed to Merchant, Merchant should not be able to read anything from it, and in order to achieve it, mechanisms such as encryptions can be applied.

By using such a design, only authorities such as the Token Service Provider and Issuer can read User and Merchant's Information. Merchant, as a third party entity, does not have any knowledge of the User's packet. Moreover, in some special cases, if Merchant wants to query about User's ID information, such queries have to be controlled in order to prevent inference attacks by sending several different queries.

Identity Tokenization Data Format

This section provides information regarding the basic data format used in the system. It does not, however, provide data formats that might require recommended assurance methods. The EMVCo's requirements on the data format can be found in Appendix III.

Tokenization Request

User → Token Service Provider

Packets that are sent from User to Token Service Provider must include the following data elements at minimum:

- ID Number
- ID Expiry Date

Token Service Provider → Issuer

Packets that are sent from Token Service Provider to Issuer must include the following data elements at minimum:

- ID Number
- ID Expiry Date

Issuer → Token Service Provider

Packet that are sent from Issuer back to Token Service Provider must include the following data elements at minimum:

- Status of the Request
- *Reasons (If failed)

Token Service Provider → User

Packets that are sent from Token Service Provider back to User must include the following data elements at minimum:

- Token Requestor ID
- Token
- Token Expiry Date

Detokenization Request
User → Merchant

Packets that are sent from User to Merchant must include the following data elements at minimum:

- Token Requestor ID
- Token
- Token Expiry Date

Merchant → TSP

Packets that are sent from Merchant to TSP must include the following data elements at minimum:

- Merchant ID
- User Packet (The packet that is sent from User to Merchant)

TSP → Issuer

The packet that is sent from TSP to Issuer must include the following data elements at minimum:

- Token Requestor ID
- ID
- ID Expiry Date

Identity Tokenization System Token Assurance Specifications

Based on the EMVCo Token Assurance Specifications, Token Requestor ID and POS Entry Modes can be implemented in the Identity Tokenization System. Below is the descriptions of how this can be done.

1. Token Requestor ID:

Each user who uses the EMVCo Digital Identity Service should be assigned a unique Token Requestor ID. This can follow the EMVCo's specifications which defines what each Token Requestor ID consists of. Each Token Requestor ID needs to be uniquely identified and associated with one specific Token Service Provider.

2. Entry Mode:

Entry Mode Code should be defined by ID-TSP to ensure that the entry mode of a digital ID is agreed-upon between both user and ID-TSP. For example, Entry Mode can be set to a numerical value on which both entities agree.

3. Acceptor ID:

Acceptor information should be included as a data field to ensure the acceptor is registered with ID-TSP. Only registered and approved acceptor may provide Digital Identification Service.

In the next section, Identity Tokenization System Token Assurance ID&V Methods Specifications will be reviewed.

Identity Tokenization System Token Assurance ID&V Methods Specifications

Based on the EMVCo's specifications on ID&V methods, there are many useful information can be used to design an Identity Authentication System. The following ID&V methods are considered as important and useful:

1. **Account Verification:**

   ID should be verified by ID Issuer to make sure it is still active. This is done when the Token Service Provider performs the detokenization process (I.e., verifying the ID with the issuer is part of this process).

2. **ID-TSP Assurance:**

   ID-TSP should provide some level of risk assessments to predict if a token request is fraud. Components that can be used to perform risk assessments include, but not limited to:

   - ID History
   - Geo Location
   - IP Address

3. **ID Issuer Assurance:**

   There should be an assurance method between ID Issuer and User. One of the most effective way is to use One Time Passwords. One Time Passwords can prevent replay attacks of using token or security parameters.

   One Time Password should be securely transmitted between ID Issuer and ID Holder, and it should be transmitted with the token request and verified by Card Issuer at the end of the data flow.

4. **Onsite Registration:**

   This assurance is one of the options that can be performed. Requiring users to register an account onsite can guarantee that the person who owns the ID is using a mobile payment service's account that is created by him/herself and approved by ID Service Provider on site. This assurance method ensures a person who adds IDs digitally are real owners. Thus, it prevents fraud in cases such as a lost ID. Overall, these methods are very important for the whole system because fraud resulted from the use of a digital ID service may raise dangers to personal privacies. Some major challenges and threats to the identity system along with some mitigation techniques will be provided in the next section.

Challenges and Threats

Challenges and threats to this system can be summarized as below:

- Lost ID
- Lost Phone
- Replay Attack
- Privacy Leakage
- Spoofed Merchant

1. Lost ID

When someone loses his/her ID, an adversary could add this ID to his/her own phone. In this way, the adversary could register the lost ID and use it.

Solution

Issuer can require ID owners who are willing to use ID Tokenization Service to register onsite and obtain a security code in order to add the ID. This method can prevent adversaries from adding other people's ID since adversaries in such case will not be able to obtain a security code for a stolen ID by an onsite registration.

2. Lost Phone

If User loses his/her phone, adversaries may use the phone and the service.

Solution

Before using service, User needs to provide a password or biometric identity to use the service. In this way, adversaries who use other Users' mobile devices cannot use the service without entering Users' passwords and or using biometrics.

3. Replay Attack

Token is a static value that does not change over time. This could be vulnerable to a replay attack. For example, an adversary can potentially figure out what the token is if the packet is the same all the time, and an adversary may be able to use the token anywhere else.

Solution

In order to prevent such a replay attack, the packet that contains a token needs to be dynamic each time. In other words, before a packet is sent out to the Merchant, the User gets a nonce from the Token Service Provider, and the User also provides a client nonce in order to prevent an active replay attack where an adversary sends its own nonce to User. In this way, a packet that is sent out by User is not a static value anymore. Instead, it becomes dynamic. Details of this method will be introduced in the next section.

4. Privacy Leakage

Privacy leakage is one of the biggest concerns for many systems, especially for systems that provide identity services. The concern is that Merchant is not an authoritative entity to access identity information which is of interest to merchant to perform data analysis.

Solution

In the previous section, it is already stated that Merchant should not have access to contents of User’s packet by using mechanisms such as encryption.

5. Spoofed Merchant

If the only requirement is to have a terminal that accepts a correct data format, an adversary can spoof merchant to collect identity information.

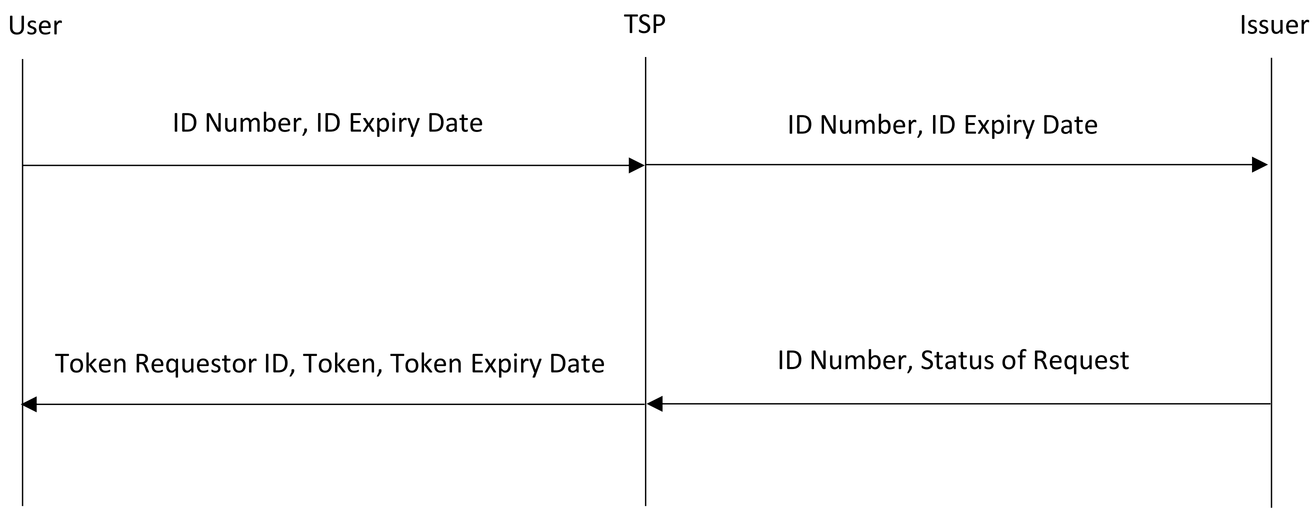
Solution

Merchant should be required to register with Token Service Provider before providing any service.

The next section shows data flows of Tokenization and Detokenization process with the Dynamic Security Code included. In addition, a simulation will be provided to illustrate the design.

Identity Tokenization System: Final Design (with DSC included) and Simulation

Tokenization


User
TSP
Issuer
ID Number, ID Expiry Date
ID Number, ID Expiry Date
Token Requestor ID, Token, Token Expiry Date
ID Number, Status of Request

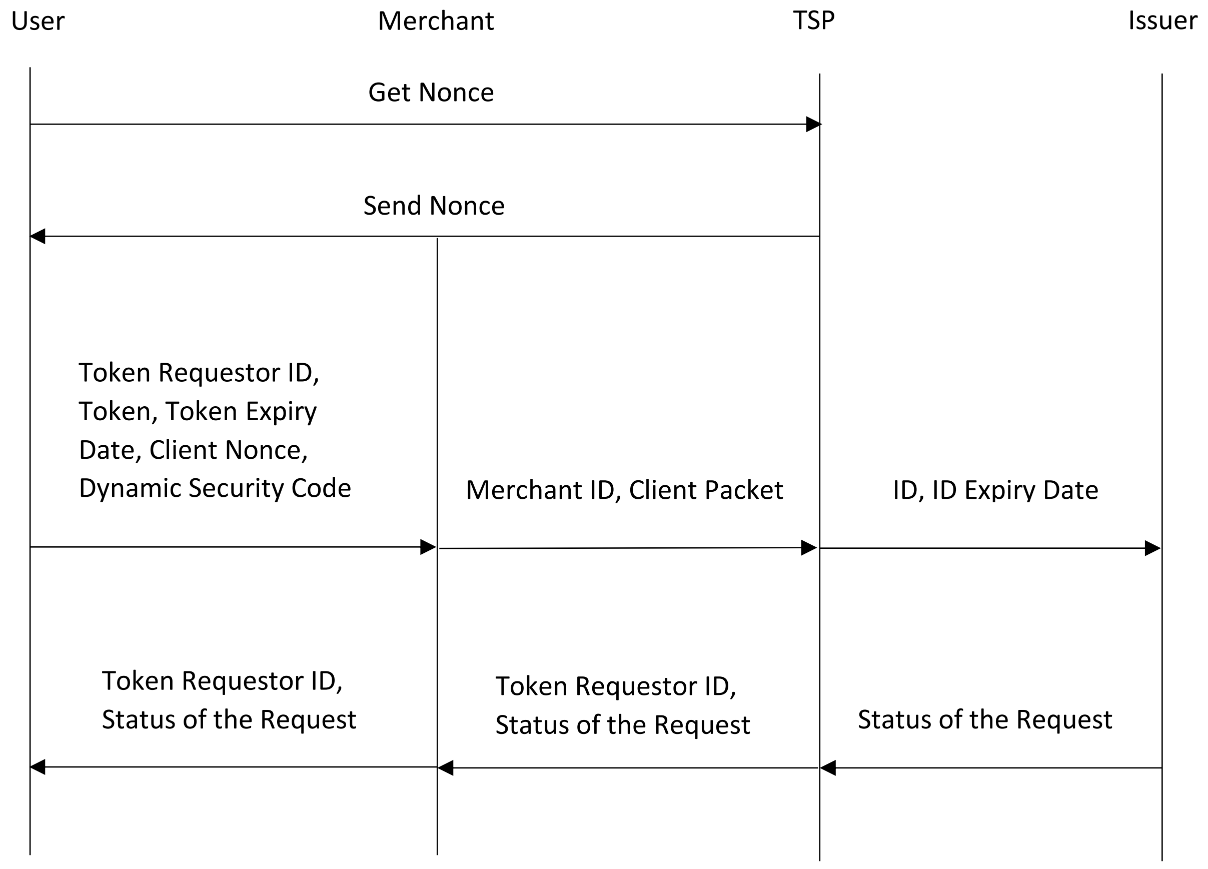
Token Issuance (First Time Add ID)

1. Client → TSP: {ID Number, ID Expiry Date}
2. TSP → Issuer: {ID Number, ID Expiry Date}
3. Issuer → TSP: {ID Number, Status of Request}
4. TSP → Client: {Token Requestor ID, Token, Token Expiry Date}

Explanations

1. Client sends ID Number and ID Expiry Date to TSP in order to add ID for the first time use.
2. TSP forwards ID information to ID Issuer for verifications.
3. ID Issuer sends back a packet to TSP that includes Status of Request, and if it fails, a reason for failure is also included.
4. TSP generates Token Requestor ID and Token for the client if validation by ID Issuer succeeds; stores them in Token Vault, and sends back to Client.

Detokenization:


User
Merchant
TSP
Issuer
Get Nonce
Send Nonce
Token Requestor ID, Token, Token Expiry Date, Client Nonce, Dynamic Security Code
Merchant ID, Client Packet
ID, ID Expiry Date
Token Requestor ID, Status of the Request
Token Requestor ID, Status of the Request
Status of the Request

Simulation Explained

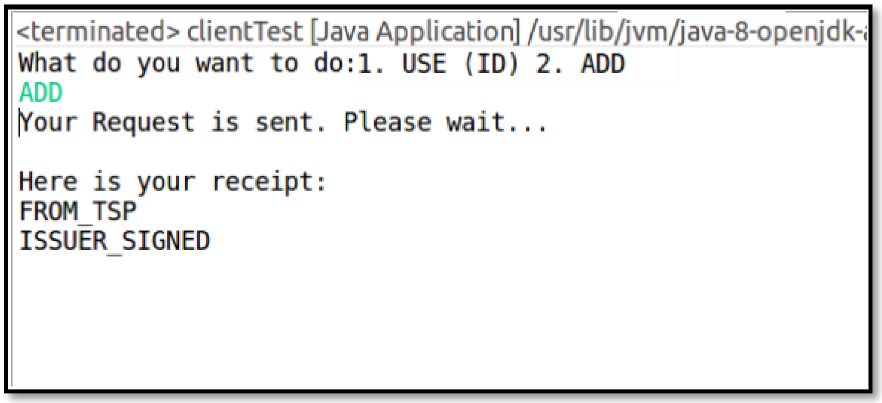
This simulation program has the following features other than basic functions:

- Dynamic Security Code
- Token generation by adding random salt
- Merchant ID is a must-check field

On initially executing the code with the client being executed at the end, we are presented with 2 options:

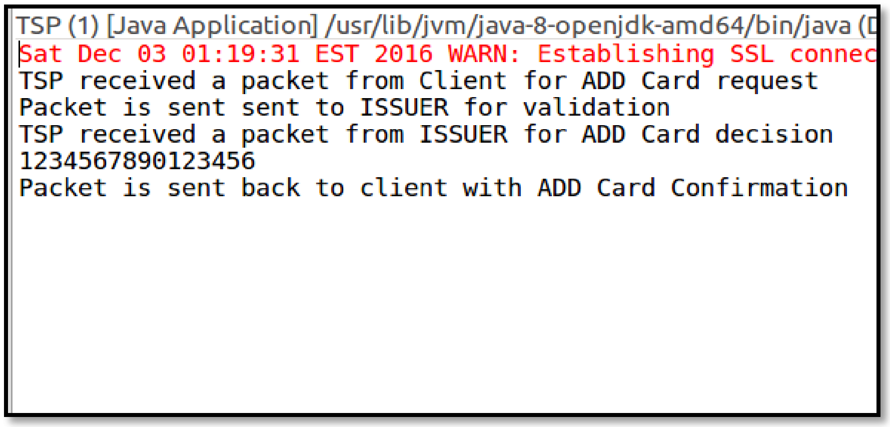
1. USE (ID)
2. ADD

We select the ADD option to add a User's ID to the system:



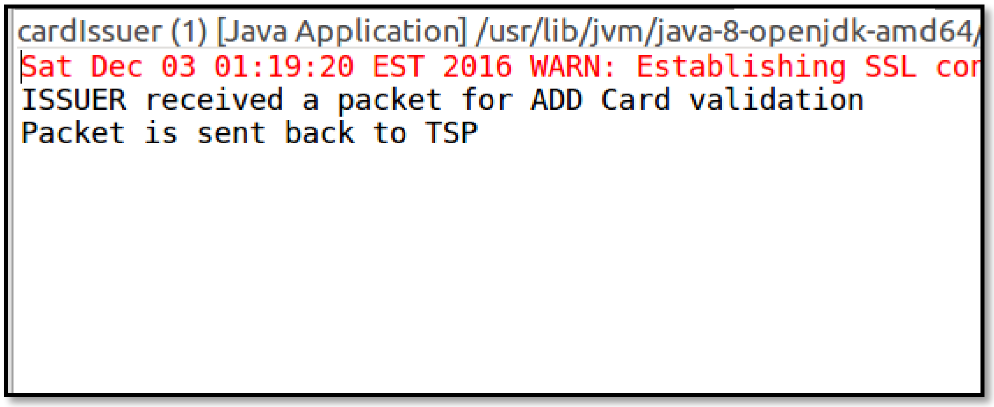
```
<terminated> clientTest [Java Application] /usr/lib/jvm/java-8-openjdk-
What do you want to do:1. USE (ID) 2. ADD
ADD
Your Request is sent. Please wait...

Here is your receipt:
FROM_TSP
ISSUER_SIGNED
```

That ID is sent to the TSP which forwards it to the ISSUER for Adding to the database:

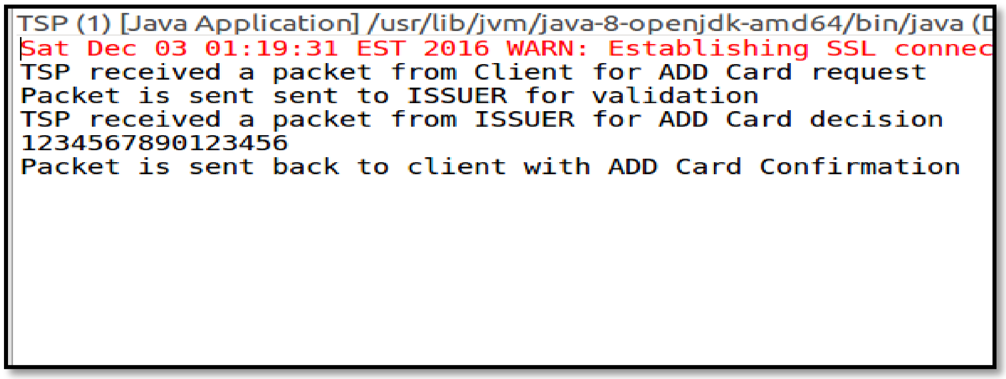


```
TSP (1) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (
Sat Dec 03 01:19:31 EST 2016 WARN: Establishing SSL connec
TSP received a packet from Client for ADD Card request
Packet is sent sent to ISSUER for validation
TSP received a packet from ISSUER for ADD Card decision
1234567890123456
Packet is sent back to client with ADD Card Confirmation
```

The ISSUER sends back the confirmation to the TSP. Finally, the client receives a token for use with merchant:

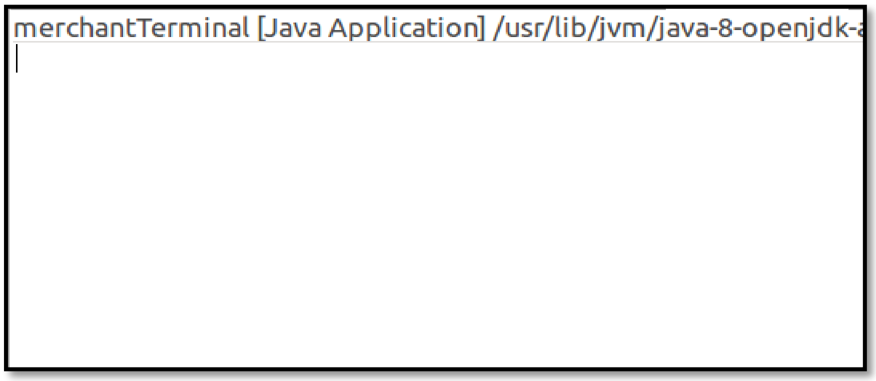


```
cardIssuer (1) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/
Sat Dec 03 01:19:20 EST 2016 WARN: Establishing SSL cor
ISSUER received a packet for ADD Card validation
Packet is sent back to TSP
```



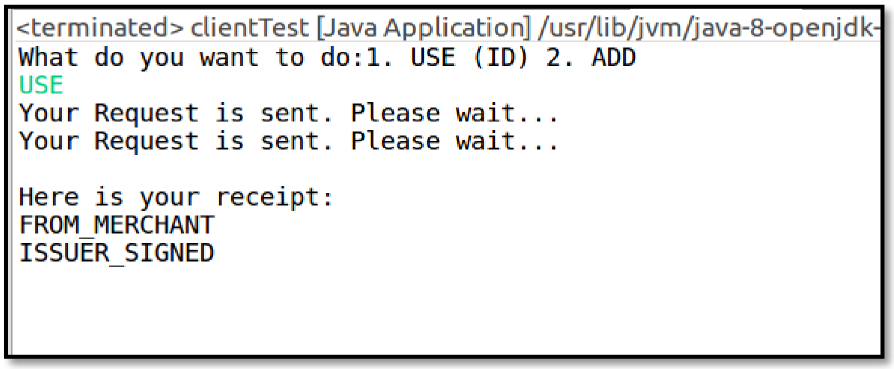
```
TSP (1) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (D
Sat Dec 03 01:19:31 EST 2016 WARN: Establishing SSL connec
TSP received a packet from Client for ADD Card request
Packet is sent sent to ISSUER for validation
TSP received a packet from ISSUER for ADD Card decision
1234567890123456
Packet is sent back to client with ADD Card Confirmation
```

Merchant has no role in the tokenization process and when the ID is added to the system:



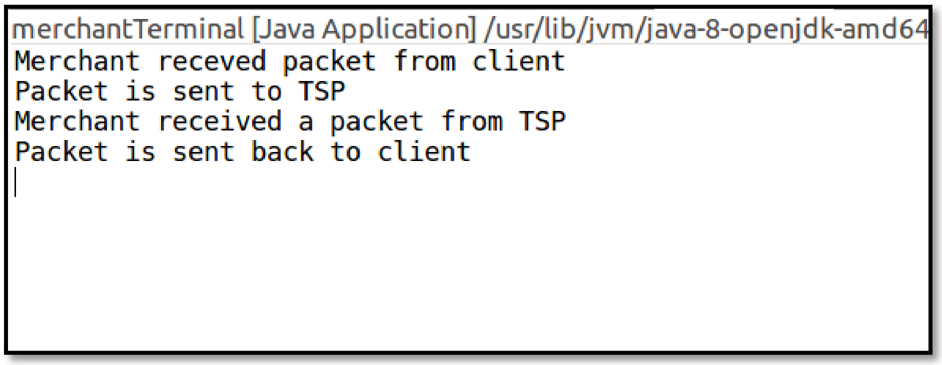
```
merchantTerminal [Java Application] /usr/lib/jvm/java-8-openjdk-
```

For the second case, the client selects the USE option. Here, initially, a nonce for the transaction is requested from the TSP before it is passed on to the Merchant for use. The client then uses the token received while adding the ID to the payment system and transfers it to the Merchant.

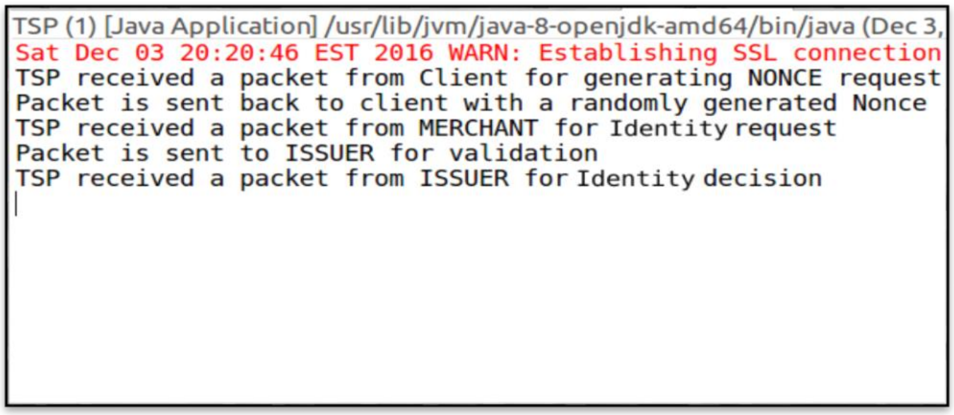


```
<terminated> clientTest [Java Application] /usr/lib/jvm/java-8-openjdk-
What do you want to do:1. USE (ID) 2. ADD
USE
Your Request is sent. Please wait...
Your Request is sent. Please wait...

Here is your receipt:
FROM_MERCHANT
ISSUER_SIGNED
```

The data gets transferred to the Merchant, which then passes it to the TSP. The TSP also verifies the nonce provided with the token and then transfers the packet containing the token to the ISSUER for verification.

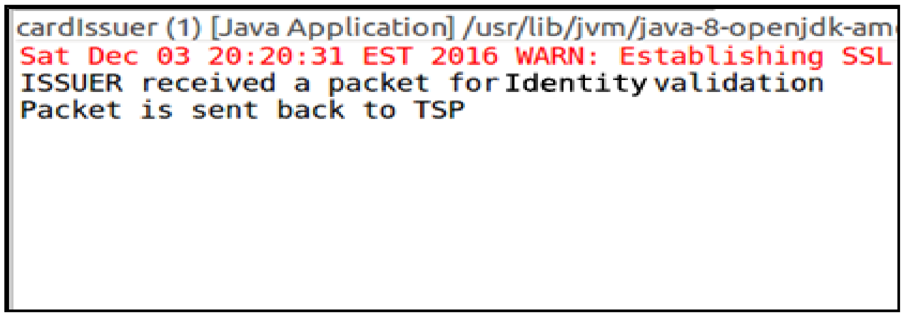


```
merchantTerminal [Java Application] /usr/lib/jvm/java-8-openjdk-amd64
Merchant receved packet from client
Packet is sent to TSP
Merchant received a packet from TSP
Packet is sent back to client
```



```
TSP (1) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Dec 3,
Sat Dec 03 20:20:46 EST 2016 WARN: Establishing SSL connection
TSP received a packet from Client for generating NONCE request
Packet is sent back to client with a randomly generated Nonce
TSP received a packet from MERCHANT for Identity request
Packet is sent to ISSUER for validation
TSP received a packet from ISSUER for Identity decision
```

After the Issuer receives the packet from the TSP, it validates the entry in its database. If the entry is there, it accepts the token and sends the confirmation back to the TSP. This confirmation is then sent to the Merchant and finally to the Client. If for any reason it fails, an appropriate error reason is returned.



```
cardIssuer (1) [Java Application] /usr/lib/jvm/java-8-openjdk-am
Sat Dec 03 20:20:31 EST 2016 WARN: Establishing SSL
ISSUER received a packet for Identity validation
Packet is sent back to TSP
```

Conclusion:

To ensure secure identity transactions, this research project implements an identity-based transaction system based on the EMVCo payment tokenization technologies. In particular, this project employs the tokenization concept that is adopted by major mobile payment technologies to assure both the security and privacy of identity transactions. Moreover, a secure protocol to prevent threats such as relay attacks is implemented. Overall, the project helps in understating how major entities in the EMVCo ecosystem interact with each other, the type and format of data that is transmitted between these entities, and the constraints and requirements each entity must adhere to.

Appendix I

ISO 8583 Financial Transaction Message Format

ISO 8583 message format is one of the most widely used format for financial messages.[17] ISO 8583 is a complete specification which not only allows card originated transactions including purchase, withdrawal, deposit, refund, reversal, balance inquiry, payments and inter-account transfers but also defines system-to-system messages for secure key exchanges, reconciliation of totals, network sign-on/sign-off and other administrative messages.

An ISO 8583 message is structured as follows:

- Message Type Identifier
- One or more bitmaps indicating which data elements are present in the message
- Data elements, or fields

Message Type Identifier	Primary Bitmap	Secondary Bitmap	Data Elements

Message Type Identifier [18]
A message type indicator includes the ISO 8583 version, the Message Class, the Message Function and the Message Origin.

Bitmaps

It indicates which elements are present in the body of the message.

There are majorly three parts of bitmaps:

- Primary bitmap
- Secondary bitmap
- Ternary bitmap

Primary bitmap

- It represents the first 64 data elements of the message

Secondary bitmap

- It is present if any of the fields 65-128 are present in the message

Ternary bitmap

- It is only present if any of the 128-192 fields are present in the message

The first bit of each bitmap signifies the presence of the next bitmap.

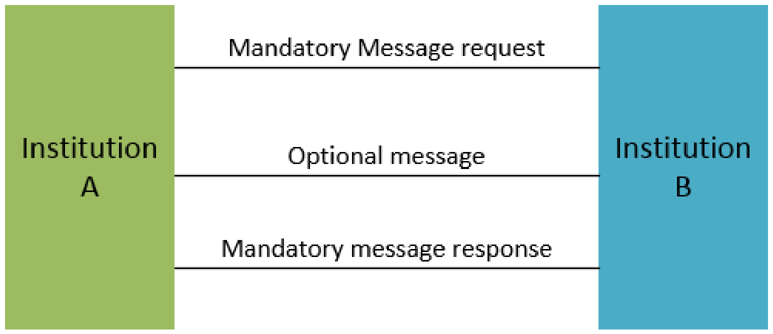
[17] http://www.admfactory.com/iso8583-financial-transaction-message-format/
[18] http://www.admfactory.com/iso8583-flows-data-elements-meaning-and-values/

Data Elements

There are up to 192 data elements in the ISO 8583 standard and the message can have a maximum of three bitmaps. In the next section, it is going to talk about flows, data elements meaning and values.

Flows, data elements meaning and values

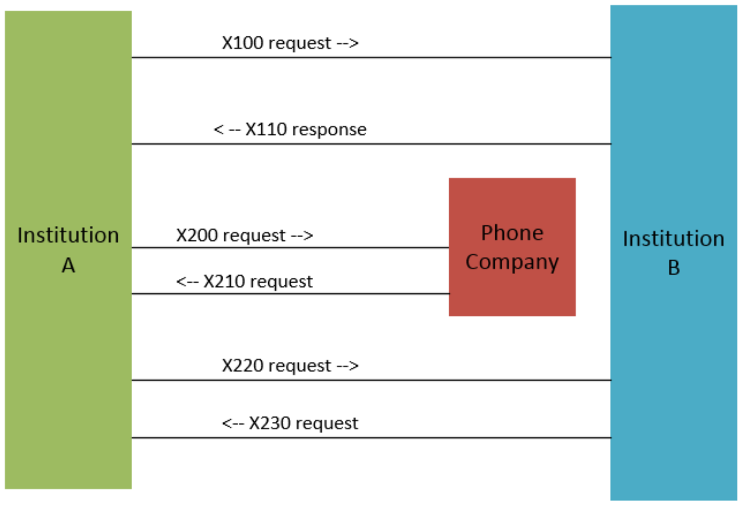
According to ADMFactory.com, a generic message flow between 2 institutions regardless the scope can be illustrated as below:



Message Types

There are message type representations defined by ISO 8583, and below is some examples according to ADMFactory.com.

- X100/X101 – Authorization message
- X200/X201 – Financial presentment message
- X220/X221 – Financial accumulation presentment message
- X300/X301 – File action message
- X400/X401 – Reversal message
- X420/X421 – Reversal advice message
- X500/X501 – Reconciliation message
- X600/X601 – Administrative message
- X800/X801 – Network management message



Explanations:
The above figure illustrates a system that can accept credit/debit cards to increase a phone account balance.

1. X100 request is sent to bank to authorize the transaction and verify the amount availability.
2. X110 request is sent back as an authorization response.
3. X200 request is then sent to Phone Company which asks for increasing the phone account balance.
4. X210 request is sent back as an authorization response.
5. X220 request is sent to bank to complete transaction.
6. X230 request is sent back as a response.

Data Elements
According to ADMFactory.com's descriptions of ISO 8583, depends on the message scope, some of the data elements are used more than the others. Here is a list with some of the most used data elements and the possible values.

- Field 2 – Primary account number
- Field 3 – Processing code
- Field 4 – Amount transaction
- Field 7 – Transmission date & time
- Field 11 – System trace audit number (STAN)
- Field 12 – Time, local transaction (hhmmss)
- Field 13 – Date, local transaction (MMDD)
- Field 22 – Point of service entry mode
- Field 37 – Retrieval reference number
- Field 39 – Response code
- Field 41 – Card acceptor terminal identification
- Field 42 – Card acceptor identification code
- Field 43 – Card acceptor name/location
- Field 49 – Currency code, Transaction

ISO 8583 formats can be used for identity transmission for sure. This format uses numerical coding to represent data element, and it is widely used by Point of Sale terminals. However, ISO 8583 format is different from data formats that are defined by EMVCo Framework. ISO 8583 defines data format that is being transmitted for transactions, and for same case, EMVCo defines something different:

Input Data Elements
Required data elements

- Token Requestor ID
- PAN
- PAN Expiry Date
- Version Number
- Length of PAN

- Account Verification Reference Length
- Length of Cardholder Data
- Device Information Length

Conditional or optional data elements

- Protocol (It is required unless inherent in the Token Requestor API.
- Account Verification Results (It is required unless Account Verification Reference Length is not 0).
- Account Verification Reference (It is required unless Account verification Reference Length is not 0).
- Token Requestor Risk Score (It is required if a fraud risk score is provided by the Token Requestor).
- Address Mismatch Indicator (It is required if shipping and billing addresses are different).
- Cardholder Data (It is required if additional data is needed to support the Requested Assurance).
- Device Information (It is required if Device Information Length is not 0).

There is an overlap between ISO 8583 and EMVCo. However, there are many different parts. ISO 8583 is more flexible and practical. For example, ISO 8583 data formats use bitmap to indicate what information it conveys. Moreover, ISO 8583 data formats use numerical value to represent data elements. In this way, it is more flexible to represent different types of content via Point of Sale terminal. On the other hand, EMVCo defines data format by using categorical representation. As the result, mobile payment systems, such as Android Pay and Apple Pay, use different types of terminals to transmit data based on types of data they selected for their own systems. Below is an example packet format according to Android Pay’s official website:

```
{
“dpan”: “4444444444444444”,
“expirationMonth”: 10,
“expirationYear”: 2015,
“authMethod”: “3DS”,
“3dsCryptogram”: “AAAAAA...”,
“3dsEciIndicator”: “eci indicator”
}
```

It is in JSON format. In order to be compatible with EMVCo Framework’s data format, it is necessary to reprogram POS terminal in order to accept a packet like the above one.

Appendix II

Secure Element (SE)

Description: "An industry-standard and certified chip that runs the Java Card Platform, which is compliant with financial industry requirements for electronic payments."[19]

Security Implications:

- SE's Tamper resistance prevents any modifications.

User Experience Implications:

- Consumer needs to have an NFC-enabled SIM. If new SIM is required, this will add cost and friction to the provisioned process.
- Works anywhere including at online and offline POS terminals, and should work regardless of whether the mobile is powered up or not.

Host-Card Emulation (HCE)

Description: "A software architecture that provides exact virtual representation of various electronic identity cards using only software."[20]

Security implications:[21]

- HCE removes the main security element (SE).
- Primary Account Number (PAN) data is at risk in HCE payments. (Tokenization solves this issue.)
- Storing credentials in the cloud (SE-in-the-cloud) is problematic when it comes to user authentication.
- Introduce a DOS threat in case the routing table of existing NFC services can be changed by a malware application.
- Features like sandbox which prevents apps from accessing data from any other app are lost when the device is rooted.

User Experience Implications:

- Tap and Pay settings may lead to confusion for the user. However, an Android API function can be used to avoid such situation.
- Limited use of payment credentials such as single-use keys.
- Phone needs to be turned on and relevant app should be running in the mobile OS.

[19] Pannifer,, S., Clark, D., & Birch, D. (n.d.). HCE and SIM Secure Element: It's not Black and White. Retrieved October 19, 2016,from http://www.chyp.com/assets/uploads/Documents/2014/06/HCE_and_SIM_Secure_Element.pdf
[20] HCE security implications - UL New Science. (n.d.). Retrieved October 19, 2016, from http://newscience.ul.com/wp-content/uploads/2014/07/hce_security_implications.pdf
[21] Ibid

Appendix III[22]

EMVCo Payment Tokenization Data Input & Output Specification

Token Request and Issuance

Input Data Elements

Required data elements

- Token Requestor ID
- PAN
- PAN Expiry Date
- Version Number
- Length of PAN
- Account Verification Reference Length
- Length of Cardholder Data
- Device Information Length

Conditional or optional data elements

- Protocol (It is required unless inherent in the Token Requestor API).
- Account Verification Results (It is required unless Account Verification Reference Length is not 0).
- Account Verification Reference (It is required unless Account verification Reference Length is not 0).
- Token Requestor Risk Score (It is required if a fraud risk score is provided by the Token Requestor).
- Address Mismatch Indicator (It is required if shipping and billing addresses are different).
- Cardholder Data (It is required if additional data is needed to support the Requested Assurance).
- Device Information (It is required if Device Information Length is not 0).

Output Data Elements

The output data should contain the following basic data elements:

- Status of the Request – Successful or Failure
- Reason code – Code indicating the type of failure (if failed)
- Payment Token (If succeeded)
- Payment Token Expiry Date (If succeeded)

[22] EMV Payment Tokenisation Specification – Technical Framework, March 2014, Version 1.0.

The following are optional or conditional output data:

- Token Reference ID (It represents a reference identifier for the Payment Token)

Token Assurance Level Method Update

This method is used for situations where after the issuance of a Payment Token, the Token Requestor wishes to have an updated assurance level assigned to the Payment Token.

Input Data Elements
Required Input Data Elements

- Version Number
- Token Length
- Payment Token / Token Reference ID
- Token Requestor ID
- Account Verification Reference Length
- Length of Cardholder data
- Device Information Length

Optional or conditional data elements:

- Requested Token Assurance Level (It is required if Token Requestor is requesting a specific Token Assurance level)
- Account Verification Results (It is required if Account Verification is performed by Token Requestor)
- Account Verification Reference (It is required if Account Verification is performed by Token Requestor)
- Token Requestor Risk Score (Fraud risk score)
- Address Mismatch Indicator (It is presented of shipping and billing addresses are different)
- Cardholder Data (It is required if additional data is necessary to support the Requested Assurance Level. E.g., billing address, shipping address, postal code)
- Device Information (If device information is needed to perform certain level of assurance)

Output Data Elements
Required data elements:

- Version Number
- Request Status
- Reason Code Length
- Token Length
- Payment Token
- Assigned Token Assurance Level

Conditional or optional data elements:

- Reason Code (It is presents if Request Status is not successful)

Detokenization Query

Input Data Elements

Required data elements:

- Version Number
- Token Length
- Payment Token
- Token Expiry Date

Conditional or optional data elements

- Token Requestor ID (It is presents if it is available)

Output Data Elements

Required data elements:

- Version Number
- Request Status
- Reason Code Length

Conditional or optional data elements:

- Reason Code (It is presented if Request Status is not successful)
- PAN Length (It is presented if Request Status is successful)
- PAN (It is presented if Request Status is successful)
- PAN Expiry Date (It is presented if Request Status is successful)

Detokenization with Verification Request

Input Data Elements

Required Data Elements

- Version Number
- Token Length
- Payment Token
- Token Expiry Date
- Length of Transaction data elements

Conditional or optional data elements

- Token requestor id (If it is available)
- Transaction data elements (If other transaction data elements are necessary for TSP to execute the request)

Output Data Elements

Required Data Elements

- Version Number
- Request status
- Reason code Length
- Length of Transaction Data Elements

Conditional or optional data elements

- Reason Code (It is presented if Request Status is not successful)
- PAN Length (It is presented if Request Status is successful)
- PAN (It is presented if Request Status is successful)
- PAN Expiry Date (It is presented if Request Status is successful)
- Transaction Data Elements (It is presented if such data is necessary for TSP to process)