Enhancing EMV Tokenisation with Dynamic Transaction Tokens

Danushka Jayasinghe, Konstantinos Markantonakis, Raja Naeem Akram and Keith Mayes

Smart Card Centre, Information Security Group, Royal Holloway, University of London, Egham, Surrey, UK, TW20 0EX.
`Danushka.Jayasinghe.2012@live.rhul.ac.uk,(K.Markantonakis, R.N.Akram, Keith.Mayes)@rhul.ac.uk`

Abstract. Europay MasterCard Visa (EMV) Tokenisation specification details how the risk involved in Personal Account Number (PAN) compromise can be prevented by using tokenisation. In this paper, we identify two main potential problem areas that raise concerns about the security of tokenised EMV contactless mobile payments, especially when the same token also called a static token is used to pay for all transactions. We then discuss five associated attack scenarios that would let an adversary compromise payment transactions. It is paramount to address these security concerns to secure tokenised payments, which is the main focus of the paper. We propose a solution that would enhance the security of this process when a smart phone is used to make a tokenised contactless payment. In our design, instead of using a static token in every transaction, a new dynamic token and a token cryptogram is used. The solution is then analysed against security and protocol objectives. Finally the proposed protocol was subjected to mechanical formal analysis using Scyther which did not find any feasible attacks within the bounded state space.

Keywords: Tokenisation, Security, Dynamic Transaction Token, EMV Contactless Mobile Payments, Cryptography, Scyther, Formal Analysis.

1 Introduction

EMV is a globally accepted standard, initially introduced for "Chip & PIN" payment transactions [2, 3] and contactless transactions [6]. However, compromising the Personal Account Number (PAN) sent during EMV transactions to be used in card-holder not present or magnetic-stripe transactions was a problem. EMV Tokenisation was adopted as a countermeasure to PAN compromise [4]. Tokenisation replaces the PAN by a substitutive value called the Token which is a 13-19 digit numeric value that does not reveal the PAN and passes validation checks set by the payment scheme [4]. Since its introduction, EMV tokenisation has seen early adoption in contactless mobile payment applications [5, 7, 16].

Near Field Communication (NFC) modules in smart phones and portable devices enable users to carry out close proximity communication which also include contactless payments. Here the mobile emulates a contactless smart card. In this paper, the payment device that emulates a contactless smart card is referred to as a mobile. Mobiles let users store a number of payment applications in one place and have hardware or software secure element technologies. Secure elements provide a secure execution environment to carry out sensitive executions. Compared to a contactless smart card, one of the additional capabilities of a mobile is the readily available communication channels via the network operator or Wi-Fi.

In this study, we identify two problem areas that raise concerns about the security of tokenised contactless mobile payments. The first problem area is that the payment terminal is assumed to be a trusted device and during a transaction, the terminal authenticates the card/mobile but the card/mobile does not authenticate the terminal. Because of this lack of mutual authentication, an adversary at a rogue terminal, can carry out a number of attacks during a tokenised payment transaction. The second problem area is that similar indelible trust assumptions are placed on the intermediary entities between the terminal and the Scheme Operator (SO) / Card Issuing Bank (CIB). When this trust assumption is disregarded, an adversary compromising one of the intermediaries is able to compromise payment transaction details. The acronyms used in this paper are listed in Table 1.

Table 1. Acronyms used in the paper

ARC	: Authorisation Response Code	PAN	: Primary Account Number
CDA	: Combined Data Authentication	SDA	: Static Data Authentication
CIB	: Card Issuing Bank	SO	: Scheme Operator
DDA	: Dynamic Data Authentication	SPDL	: Security Protocol Description Language
DTD	: Dynamic Token Data		
DTT	: Dynamic Transaction Token	TAR	: Token Authorisation Request
EMV	: Europay MasterCard Visa	TSP	: Token Service Provider
NFC	: Near Field Communication	TVR	: Terminal Verification Result

The main contributions of this paper are threefold. They are: 1) The proposed solution provides mutual-authentication, so that both the mobile and the terminal is able to authenticate each other. 2)The protocol uses a new Dynamic Transaction Token (DTT) that is unique to a particular transaction instead of using a static token for improved security. 3) The protocol also provides end-to-end encryption between the terminal and the Token Service Provider (TSP) as well as the terminal and the mobile, eliminating the need of placing indelible trust assumptions on intermediary nodes between the terminal and the TSP.

The paper is structured as follows. In Section 2, the use of tokenisation in the current contactless mobile payment architecture is introduced. In Section 3 the two potential problem areas and the corresponding attack scenarios are discussed. The proposed protocol is introduced in Section 4 and evaluated in Section 5 against protocol objectives. Finally the protocol is subject to mechanical formal analysis in Section 6.

2 EMV Tokenisation

In this section, we expand our discussion to tokenisation on contactless mobile payments. One potential security issue in contactless payments is PAN compromise, where PAN related data is compromised by adversaries during EMV point-of-sale transactions or from merchants' databases. The compromised payment card details are then used to carry out cross channel fraud[1]. PAN compromise can be prevented by mapping the PAN with a substitute value. The process that manages the conversion from a PAN to a token and vice-versa is called tokenisation and the substitute value is called a token. The EMV Tokenisation Specification details requirements to support payment tokenisation in EMV transactions [4]. From the merchants' perspective, storing and managing tokens as opposed to PANs in databases, simplifies compliance audits such as the Payment Card Industry-Data Security Standard (PCI-DSS) [8, 23]. The tokenisation discussed in this study refers to the EMV Tokenisation Specification [4]. The payment architecture and the transaction message flow of a generic EMV contactless mobile transaction based on tokenisation is illustrated in Figure 1.

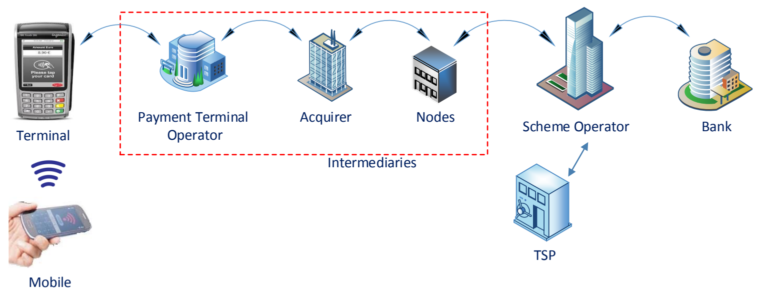

Fig. 1. Generic EMV Tokenised Payment Architecture

At the start of the transaction, the mobile passes the token and token related data to the terminal. The terminal then sends the additional token related data in the transaction authorisation message to the SO/CIB via a number of intermediaries that engage in key-translation for approval. This means that two entities connected with arrows on both ends in Figure 1 share a symmetric key to communicate with each other. When the message is forwarded to the next entity, it is deciphered and enciphered using the shared symmetric key with the next entity. Once the authorisation request is received at the SO, the TSP is contacted to de-tokenise the token in order to retrieve the corresponding PAN of the token and to validate the token cryptogram [4, 16]. The retrieved PAN and validation results are forwarded to the SO. The SO then forwards the authorisation request with the mapped PAN to the bank. Following this, the bank generates an Authorisation Response Code (ARC) and sends it in an authorisation response message to the SO [4, 19]. The SO now forwards the authorisation response to the terminal. The terminal then approves or declines the transaction.

[1] Cross-Channel Fraud: capturing card details in a point-of-sale transaction and using the details fraudulently in other payment channels such as e-commerce payments.

2.1 Operating Environment

In this section, we discuss the operating environment of tokenised EMV contactless mobile payments in its current architecture as illustrated in Figure 1. The SO has a direct communication channel to the TSP and the bank. The payment terminal operator supplies terminals to a number of merchants. It also engages in collecting transactions originating from the merchant's terminals and forwards them towards the SO/CIB. A payment terminal operator can either be a third party, or an acquirer's subcontractor. However, this does not change our attack scenarios discussed later. The same communication path between the terminal and the SO that was taken to send the transaction authorisation request is also taken in reverse to send the transaction authorisation response back to the terminal.

In this study, we consider the bank, SO, mobile and TSP as secure and trusted. In contrast to this, we consider that the terminal has the potential to be compromised. This is evident from reports and research shown in [11, 13, 18, 24]. We also consider that the intermediaries have the potential to be compromised. This assumption is reasonable, based on reports and research shown in [9, 19, 21, 22, 25]. Taking this operating environment into consideration, we expand the discussion to outline potential attack scenarios in the next section.

3 Potential Attacks

In this section, we outline potential attack scenarios associated with two main problem areas in tokenised contactless mobile payments.

3.1 Adversary Compromises A Terminal

In this problem area, there are three different potential attacks. The attack scenarios are outlined and discussed in Attacks 1, 2 & 3 given below. The terminal is considered to be a trusted device but, there is no mutual-authentication between the terminal and the card/mobile. When the trust assumption is taken out, a rogue terminal controlled by an adversary could be in EMV contactless payment process. For these attacks, we assume an adversary with the following capabilities. An adversary:

- can gain full control of the terminal including what is displayed on screen for the payer.
- can change transaction related details such as the amount.
- cannot break standardised encryption algorithms.
- might collude with another adversary that compromises and controls an intermediary between the terminal and the SO/CIB.

Attack 1: *Over Charging*
In this attack scenario, the adversary fraudulently enters a large payment amount (within the contactless limit) for a transaction but displays the correct purchasing product price on the terminal screen for the consumer. It is not possible for

the mobile to detect whether the terminal is genuine or rogue as the transaction amount is not displayed on the mobile at the time of payment. Therefore the user does not have any alternative option other than to believe the amount displayed on the merchants terminal is true. So the user continues the payment unaware of the fraudulently over charged amount.

Attack 2: ***Capturing Static Token & Related Data***
In this attack scenario, an accomplice controlling the rogue terminal transacts with a genuine mobile making a tokenised contactless mobile payment. The genuine mobile sends the static token and token cryptogram to the rogue terminal. The accomplice captures the static token, its associated cryptogram and other transaction related data. The rogue terminal may display an authentication failed message on the terminal and refuse purchase for the consumer. The captured details are used by the adversary in Attack 4.

Attack 3: ***Capturing The Unpredictable Number***[2]
The EMV tokenisation specification does not specify whether offline data authentication needs to be carried out by the terminal [4]. Because tokenised payments operate in an online setting, at first, it is not apparent as to why offline data authentication is actually needed. However, we highlight why failing to carrying out offline data authentication aggregates the identified security concern. In this attack scenario, an adversary attempts a payment at a genuine terminal to obtain the unpredictable number generated by the genuine terminal. At the absence of offline data authentication, the terminal is unable to verify whether the payment application related data presented by the mobile is genuine. Therefore, the terminal nonce is sent to the mobile as a challenge to be signed by the mobile in order to carry out dynamic data authentication. The nonce forms part of the dynamic application data which is later signed by the mobile to generate the digital signature expected by the terminal. Soon as the nonce is received, the adversary captures the nonce and halts any further communication with the terminal.

In some instances, even if Static Data Authentication (SDA) is carried out by the terminal, it may still be possible to compromise the terminal nonce if SDA is not carried out before the nonce is sent. For example, as explained in [12], Visa's payWave qVSDC protocol sends the terminal generated nonce before SDA. This would enable an adversary to obtain the terminal unpredictable number. Potential attacks and other security concerns related to compromising terminal unpredictable numbers are shown in [10, 11].

3.2 Adversary Compromises An Intermediary

In the current EMV architecture, indelible trust assumptions are placed on the intermediaries between the terminal and the SO/CIB. When this trust assumption is disregarded, an adversary compromising one of the intermediaries has

[2] The EMV Specification defines the Unpredictable Number as a "Value to provide variability and uniqueness to the generation of a cryptogram [3]". In this paper we refer to this as the terminal nonce.

a potential attack scenario to infiltrate transaction details and make fraudulent transactions. The adversary at the compromised intermediary observes all transaction data passing through it, which also include transaction authorisation requests, tokens and token related data. For these attacks, we assume the following adversary's capabilities. An adversary:

- can compromise any of the intermediaries.
- can gain access to transaction data at the compromised intermediary.
- can break standardised and strong encryption algorithms.
- cannot compromise smart cards, the SO or the CIB.
- might collude with the adversary that compromises a terminal.

Attack 4: ***Adversary Replays An Authorisation Response For Cloned Token Data***
The attack scenario is realised when the transacting terminal fails to carry out adequate offline data authentication method such as Dynamic Data Authentication (DDA) or Combined Data Authentication (CDA) [2], but sends the transaction data for online transaction authorisation. The adversary at the compromised intermediary gets to observe all transaction data passing through it and these include transaction authorisation requests intended for the SO/CIB. The attack steps are described below.

1. The adversary works together with the accomplice, who captured the static token and the corresponding token data in the precursor Attack 2.
2. The accomplice chooses a terminal that has an established communication path to the SO/CIB via the compromised intermediary and makes a contactless payment with the captured static token data.
3. The terminal carries out SDA on the presented static token data. As the data were captured from a genuine mobile, the SDA verification at the terminal completes successfully. However, without DDA or CDA where a dynamic signature is generated by the mobile and verified by the terminal, the terminal is not able to detect the cloned data.
4. The terminal sends the transaction data online for transaction authorisation.
5. The adversary, instead of passing the transaction authorisation request to the authorising entity, stops the request from reaching the authorising entity by identifying static token included in the message.
6. Instead, the adversary replays an ARC pretending to have come from the authorising entity. Once the authorisation response is received, the terminal approves the transaction.
7. Unlike in a contact-based EMV transaction, the transaction authorisation response cryptogram is not sent to the contactless card/mobile [2]. One of the reasons for this is that in contactless EMV, there is no assurance that the card is kept in the reader's field by a cardholder, so the transaction authorisation response is not enciphered by the bank with a key shared between the card/mobile and the bank.

Attack 5: ***Replaying An Authorisation Response For DDA/CDA***
In Attack 4, the attack was realised when the terminal did not carry out DDA/CDA as offline data authentication. However, in this attack scenario, we assume

that the terminal is carrying out DDA/CDA and identify a similar compromise. The attack steps are described below.

1. The adversary works together with an accomplice, who is in possession of a number of lost & stolen contactless mobiles. The attack is carried out during the time-slot between the cards/mobiles are lost/stolen and the relevant issuing banks are notified by the owners.
2. The accomplice chooses a terminal that has an established communication path to the SO/CIB via the compromised intermediary and makes a contactless payment.
3. The terminal carries out the dynamic offline data verification. As the dynamic signature is generated by a genuine mobile, the terminal verification finishes successfully. The terminal then sends the transaction data online for authorisation.
4. The adversary, instead of passing the transaction authorisation request to the authorising entity, captures it.
5. The adversary replays a previously communicated ARC generated by the authorising entity. Once the authorisation response is received, the terminal approves the transaction.

4 Proposed Solution

In this section, a solution that addresses the security concerns discussed in Section 3 is proposed. The main objectives of the protocol are listed below.

1. Should prevent Attacks 1, 2, 3, 4 & 5.
2. Mutual authentication should be carried out between the terminal and the mobile.
3. End-to-end encryption should be provided between the secure element and the terminal, as well as between the terminal and the TSP.

4.1 Protocol Assumptions

The following have been assumed in our proposed solution:

– The communication between the mobile and the TSP is carried out using a secure channel.
– The TSP is a trusted entity that provides transaction token issuing, detokenisation, token updates and management on behalf of the CIB.
– The SO acts as the TSP in the payment architecture.

The notation used in the proposed solution is given in Table 2. The tokenised contactless mobile payment architecture of the proposed protocol is illustrated in Figure 2. The protocol proposed in this study has a setup stage and a payment stage. In the *Setup Stage*, the payment app and related data are securely provisioned to the mobile. The *Payment Stage* is used when making a contactless mobile payment. The transaction scenario focused in this paper is when both the terminal and the mobile are online capable to reach the TSP. Providing offline tokenised payments is not the focus of this paper and related work can be found in [20].

Table 2. Notation used in the proposed protocols

T/SE/x	: Terminal/Secure Element/Identity of X.
TATC	: Token Application Transaction Counter, count of token transactions since personalisation. It is shared between $mobile, bank$ & TSP and used during key derivations.
K	: SE generated Symmetric Session Key.
K_{s1}	: Symmetric Encryption Session Key shared between TSP and SE.
K_{s2}	: TSP generated Symmetric Encryption Session Key used by the terminal to communicate with the TSP.
$K_{To'}$	: Token Cryptogram Generation Symmetric Session Key derived by a key derivation function used by TSP.
$E_K\{Z\}$	: Symmetric Encryption of data string Z using key K.
S_X	: Private Signature Key of entity X.
$sS_X\,[Z]$	: Digital signature outcome (without message recovery) from applying the private signature transformation on data string Z using S_X of X.
P_X, P_X^{-1}	: Public Encryption/Decryption Key Pair of entity X.
$eP_X\{Z\}$	: Encryption of data string Z using a public algorithm with P_X.
$Cert_Y(X)$	: Public Key Certificate of X issued and certified by Y.
$h(Z)$	: Hash of data string Z.
n_X / n_{2X}	: First / second nonce issued by entity X.
$A\|\|B$	: Concatenation of A and B in that order.

4.2 Setup Stage

During the setup phase of the protocol, the personalisation of the payment application and provisioning of security sensitive data elements of the payment application and credentials are carried out using a secure channel. Following the application personalisation, the security sensitive data elements of the payment application reside in the SE and the user interface part of the payment application reside in the mobile platform. The data elements in the SE consist of all cryptographic keys needed by the mobile eg: K_{SE}, K_{s1}, S_{SE} & P_{SE}/P_{SE}^{-1}. The SE also stores: $Cert_{bank}(TSP)$, $Cert_{bank}(SE)$, Token Application Transaction Counter ($TATC$). Following personalisation of the payment app, the user is required to enter a strong pass-code on first access which is used for future authentication of the user to the payment app. The subsequent transaction protocols are constructed based upon the above mentioned data elements.

Terminals and secure elements participating in the payment scheme can verify certificates issued by the SO or entities that have been certified to be trusted in the certificate hierarchy. The TSP also takes part in the payment scheme.

4.3 Payment Phase

The protocol messages of the proposed solution are illustrated in Table 3 and explained as follows. To make a payment, the user opens the payment application by entering the pass-code and taps the device on the terminal.

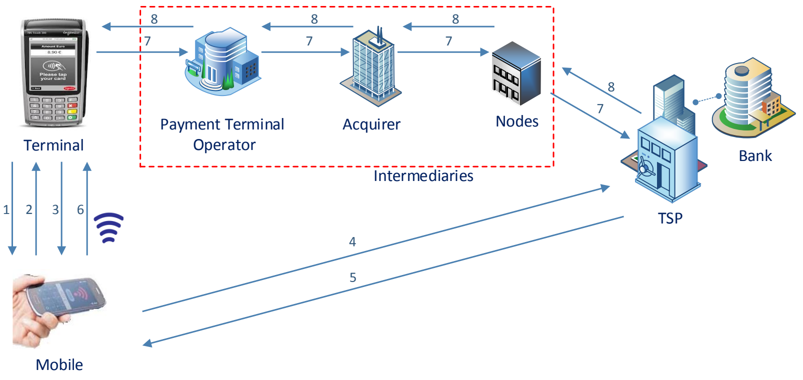

Fig. 2. Tokenised Contactless Mobile Payment Architecture of the Proposed Protocol

Message 1: The T provides its identity, n_t and $Cert_{SO}(T)$ to the SE.
Message 2: The SE obtains P_T after verifying $Cert_{SO}(T)$. A message is constructed that includes: both identities, n_{se}, n_t, the Processing Options Data Object List (PDOL) that instructs the T what information to send back to the T [6], a session key generated by the SE to be used in further communication between the T and the identity of the TSP. The SE enciphers the message using P_T. A digital signature of the message is generated by the SE and both the enciphered part and the digital signature is sent to the T. The SE's public key certificate is also sent in the same message.
Message 3: The T authenticates the device by verifying the signature using the certificate hierarchy, then deciphers message 2. The T encrypts and signs a message which includes: the identities, n_{se}, n_{2t}, the amount of the transaction and an encipherment carried out using P_{TSP} on the T's identity, amount and n_{3t}. The full message and the signature is then sent to the SE.
Message 4: Using the certificate hierarchy, the SE verifies the signature, authenticates the T and deciphers message 3. To request a token from the TSP, a message is constructed which includes: the identities, n_{2se}, Token Requester ID, amount, the Token Application Transaction Counter (TATC), $Cert_{SO}(T)$ and $eP_{TSP}\{t||amount||n_{3t}\}$. The message is then encrypted using the symmetric key K_{s1} shared between the TSP and the SE before sending.
Message 5: The TSP, first verifies the $Cert_{SO}(T)$ and obtains the T's public key. The TSP deciphers $eP_{TSP}\{t||amount||n_{3t}\}$ and checks whether the amount recovered from this matches the amount requested by the SE. If satisfied, the TSP queries the CIB and verifies the user credibility for a new token. Then the TSP generates a DTT and a session key K_{s2}. The TSP then creates Dynamic Token Data (DTD) which includes: the identities, n_{2tsp}, DTT and K_{s2}. The TSP signs the hash of DTD. The DTD is then enciphered using P_T. The TSP then creates a message that includes: the identities, n_{2se}, n_{tsp}, $eP_T\{DTD\}$ and $sS_{TSP}[h(DTD)]$. The message is then enciphered using K_{s1} before sending. DTT is constructed as follows;

Table 3. Dynamic Transaction Token Protocol Messages.

1. $T \rightarrow SE$	:	$t\|\|n_t\|\|Cert_{SO}(T)$
2. $SE \rightarrow T$	:	$V\|\|sS_{SE}\left[h(V)\right]\|\|Cert_{Bank}(SE)$ $V = eP_T\{se\|\|t\|\|n_{se}\|\|n_t\|\|PDOL\|\|K\|\|tsp\}$
3. $T \rightarrow SE$	:	$W\|\|sS_T\left[h(W)\right]$ $W = E_K\{t\|\|se\|\|n_{se}\|\|n_{2t}\|\|amount\|\| \; eP_{TSP}\{t\|\|amount\|\|n_{3t}\}\}$
4. $SE \rightarrow TSP$	:	$E_{K_{s1}}\{se\|\|tsp\|\|n_{2se}\|\|TokenR\text{-}ID\|\|amount\|\|$ $TATC\|\|Cert_{SO}(T)\|\|eP_{TSP}\{t\|\|amount\|\|n_{3t}\}\}$
5. $TSP \rightarrow SE$	:	$E_{K_{s1}}\{tsp\|\|se\|\|n_{2se}\|\|n_{tsp}\|\|eP_T\{DTD\}\|\|sS_{TSP}\left[h(DTD)\right]\}$ $DTD = tsp\|\|se\|\|t\|\|n_{2tsp}\|\|n_{3t}\|\|DTT\|\|K_{s2}$
6. $SE \rightarrow T$	:	$E_K\{se\|\|t\|\|n_{3se}\|\|n_{2t}\|\|eP_T\{DTD\}\|\|sS_{TSP}\left[h(DTD)\right]\}$
7. $T \rightarrow TSP$	:	$Token\|\|E_{K_{s2}}\{t\|\|tsp\|\|Token\|\|n_{2tsp}\|\|n_{4t}\|\|DTT\|\|POSem\|\|TVR\}$
8. $TSP \rightarrow T$	:	$Token\|\|E_{K_{s2}}\{tsp\|\|t\|\|n_{3tsp}\|\|n_{4t}\|\|Token\|\|TokenAssuranceLevel\|\|$ $PANlast4digits\|\|ARC\}$

$$DTT = TokenData||TokenCryptogram$$
$$TokenData = TokenID||TokenExpiry||TokenR\text{-}ID$$
$$TokenCryptogram = E_{K_{To'}}\{TokenData||amount||n_{3t}\}$$

Message 6: The SE, prepares a message to send the $eP_T\{DTD\}$ and $sS_{TSP}\left[h(DTD)\right]$ to the T. The message includes: the identities, n_{3se}, n_{2t}, $eP_T\{DTD\}$ and $sS_{TSP}\left[h(DTD)\right]$. The message is then enciphered using K before sending. If the SE is not in the NFC field, the user taps the SE on the T again to transmit the message. The SE may leave the NFC field once the message is successfully sent to the T.

Message 7: After deciphering the message received from the SE, the T first examines the nonces to detect any replay attempts. Then deciphers the $eP_T\{DTD\}$ to obtain DTD and verifies $sS_{TSP}\left[h(DTD)\right]$ to have been generated by the TSP. Once satisfied the T carries out *dynamic token data authentication* to verify the authenticity of the presented data. For this the T generates the hash of the DTD received in the previous message and compares this with the hash recovered in the $sS_{TSP}\left[h(DTD)\right]$. If the two hashes match, *dynamic token data authentication* is verified successfully, otherwise the transaction is declined due to the potential of a replay attack.

Depending on the outcome of the *dynamic token data authentication*, the T constructs a Token Authorisation Request (TAR) and forwards it to the TSP for payment authorisation. To construct the payment authorisation message, the T first constructs a message which includes: identities, Token, n_{2tsp}, n_{4t},

DTT, Point-Of-Sale Entry Mode (POSem)[3] and the Terminal Verification Result (TVR) indicating the outcome of the offline dynamic token data verification. This message is then enciphered using K_{s2}. The T before forwarding the message to the TSP also appends the $Token$ to the encipherment. The T uses the key translation mechanism to forward the TAR to the TSP via the Intermediaries, for financial transaction authorisation.

The only data sent in the clear is the $Token$, which on its own cannot be used by the $Intermediary$ to obtain any useful information corresponding to the PAN. In the operating environment of the proposed solution, the SO acts as the TSP, therefore the message is received at the TSP.

Message 8: After receiving the TAR, the TSP carries out the following checks to validate the token:

- queries its database records in-relation to the issued tokens and checks details such as: expiry, requester ID, amount and the token cryptogram.
- if the token related data is validated properly, the TSP conducts payment token de-tokenisation to map the token details into PAN details.

Following these verifications the TSP retrieves the PAN details and contacts the CIB to obtain an ARC. The TSP provides information such as: the PAN, PAN expiry date, amount, POSem, token, token expiry, token requester ID and the Token Authorisation Request Result (TAR_{result}) in order to obtain the ARC. The TAR_{result} contains three main components. They are: the outcome of TSP's token verification has passed or failed, *TokenAssuranceLevel* which indicates the level of assurance that the TSP has assigned to the token depending on the confidence of the TSP and *TokenAssuranceData* which indicates the data used by the TSP to assign a token assurance level. The CIB before issuing the ARC carries out the following account level validations:

- retrieve account details corresponding to the PAN.
- check whether there are sufficient funds available and no account restrictions.
- verify POSem and the token has not been presented for authorisation before.
- check the outcome of the TAR_{result} validation carried out by the TSP.

Following all the validation steps, the ARC is issued and the TSP constructs a message that includes: the identities, n_{3tsp}, n_{4t} the Token, Token Assurance Level, the last 4 digits of the PAN and the ARC generated by the CIB. The message is then enciphered using K_{s2}. The TSP also appends the $Token$ to the encipherment before the message is sent to the T via the Intermediaries. The Intermediary cannot deduce any information corresponding to the PAN or the authorisation response other then the $Token$.

Once the message is received, the T deciphers the message using the session key and examines the results in order to approve/decline the transaction. The outcome is displayed on the T. The merchant may produce a receipt that includes transaction details such as the amount, last 4 digits of the PAN, date, time and ARC to be given to the user upon request.

[3] The POSem acts as a Token Domain Restriction Control [4] to prevent other cross channel fraud by restricting the tokens to a specific payment channel (contactless mobile payments in this scenario).

5 Analysis

In this section, the proposed protocol is analysed for its security and protocol objectives. The operating environments outlined in Sections 2.1, adversary capabilities outlined in Sections 3.1 & 3.2 and protocol assumptions outlined in Section 4.1 have been taken into consideration in this analysis.

At the beginning of the protocol, both the secure element and the terminal are authenticated to each other. This establishes a mutual-authentication before security sensitive transaction data are communicated. Due to the unforgeability of the digital signature algorithm used, only a genuine secure element and the terminal is able to generate their own signatures. The signatures can be verified using the certificate hierarchy.

The proposed protocol provides end-to-end encryption between the terminal and the secure element. This eliminates the need for placing indelible trust assumptions on the intermediaries. The protocol also provides end-to-end encryption for the communication between the terminal and the TSP which provides confidentiality to token transaction related data by preventing adversaries from eavesdropping. Below we describe how the identified attacks that compromise token transaction data in Section 3 are prevented in the proposed protocol. Table 4 categorises different countermeasures used for each attack.

Attack 1 ***(Over Charging):*** In the proposed protocol, message 3 sent by the terminal to the secure element has transaction related data including the amount which is displayed on the users mobile. The mobile contacts the TSP to request the dynamic token, only after the user authorises the amount displayed on the user's mobile screen. If the merchant is trying to overcharge the user, this will be detected and the transaction can be cancelled. Furthermore, the corresponding token is requested by the mobile using the data received in message 3, hence a rogue merchant is not in a position to change the amount to a different value. The DTD also includes the amount and any changes to the transaction amount can be detected TSP.

Attack 2 ***(Capturing Static Token & Related Data):*** In the proposed protocol, before the token and token related data is given to the terminal, a mutual authentication process is carried out. The mobile sends n_{se} as a challenge in message 2 for the terminal to sign with other related data. The signature is verified by the mobile after receiving message 3 where the terminal is authenticated. If the terminal is not authenticated at this stage the mobile aborts the protocol. Furthermore, the token used in the proposed protocol is a dynamic transaction token, meaning the token issued by the TSP is unique and can only be used in the particular transaction. A replay of DTT can be detected by a genuine terminal due to a replayed message 6 not having the terminal-generated n_{3t} in the $sS_{TSP}\,[h(DTD)]$. These countermeasures prevent rogue terminals from carrying out Attack 2. In case the token was compromised and attempted on another fraudulent transaction, the TSP would not authorise the transaction for the second time. As the mobile requests a DTT for every transaction, the TSP is aware of a transaction even before a payment authorisation request is made by a terminal, this introduces an additional layer of security to prevent unauthorised transactions, as well as facilitating accurate approvals & risk assurance levels for the tokenised payment transaction.

Attack 3 ***(Capturing The Unpredictable Number):*** Due to the unforgeability of the digital signature scheme used, the terminal can verify the signature to have been generated by a genuine secure element. Furthermore, n_t included in the signature prevents a signature from being replayed and indicates to the terminal that the transaction is fresh. If the verification fails, then the terminal declines the overall transaction which prevents the terminal from generating the third nonce n_{3t} which is used in the DTT. Furthermore, the protocol provides end-to-end encryption between the terminal and the SO which also prevents any malicious entity from compromising n_{3t}.

Attack 4 ***(Adversary Replays An Authorisation Response For Cloned Token Data):*** The proposed protocol uses the following countermeasures. Firstly, mutual authentication is established at the beginning of the protocol. This way the terminal only proceeds to the transaction by sending transaction related details in message 3, only if the secure element is authenticated in message 2. Secondly, the solution uses a DTT rather than a static token. This means that the token is specific for a transaction and it includes nonces from both the transacting terminal and the TSP. A replay of DTT can be detected by the terminal due to a replayed message 6 not having the terminal-generated n_{3t} in the $sS_{TSP}\,[h(DTD)]$. Furthermore, the protocol provides end-to-end encryption for the communication between the terminal and the TSP. This prevents the adversary at the compromised intermediary from replaying an authorisation response back to the terminal and any such attempts are detected by the terminal.

Attack 5 ***(Replaying An Authorisation Response For DDA/CDA):*** Unlike Attack 4 where only SDA is carried out, this attack scenario is even possible with DDA/CDA. To prevent this attack, the solution provides end-to-end encryption which prevents the adversary at the compromised intermediary from learning the data communicated between the terminal and the TSP. Also, nonces generated from both the terminal and the TSP are included in messages communicated between each other as well as in the DTT. This prevents any replay attempts detectable for the terminal in the event of any authorisation response replay attempts. Furthermore, the payment application needs the user to enter a passcode before use. It must be also noted that as the mobile requests a DTT online before each transaction, in the event of the mobile being lost or stolen, the user can inform the CIB in order to deny access to token requests.

Table 4. Attacks and countermeasures used in the proposed protocol

Attack	**Mutual Authentication**	**End-To-End Encryption**	**DTT**	**Other**
1:Over Charging	✓		✓	Amount displayed on mobile
2:Capturing Static Token & Related Data	✓		✓	
3:Capturing The Unpredictable Number	✓	✓	✓	
4:Adversary Replays An Authorisation Response For Cloned Token Data	✓	✓	✓	
5:Replaying An Authorisation Response For DDA/CDA	✓	✓	✓	Passcode for payment app

6 Mechanical Formal Analysis

In this section, the proposed protocol is subject to mechanical formal analysis using Scyther [15]. The proposed protocol was modelled and provided as input to Scyther using the Security Protocol Description Language (spdl) defined in [14]. The spdl provides three main protocol modelling features: *roles, events and claims.* The roles define the entities in a protocol, which characterise events. The send and receive operations are classed as `send` and `recv` events respectively; each corresponding `send` and `recv` event has the same sequence number. The security goals and objectives of a protocol that require verification are specified using `claim` events. We used the Dolev-Yao model as the adversarial model used in this analysis [17]. The following security claims are verified in the analysis: *Aliveness (`Alive`), Weak agreement (`Weakagree`), Non-injective agreement (`Niagree`) Non-injective synchronisation (`Nisynch`) and Secrecy of data (`Secret`) for: DTT, ARC, K, K_{s2}* [14, 15].

The script was run on an Intel CORE-i7 2GHz machine with 8GB of RAM. When the security claim events were run together during protocol analysis, Scyther tool was crashing. We identified that the reason for this was the RAM getting full after a few hours of protocol analysis. To overcome this issue, the security claims were analysed one by one. Following successful execution of the script, the security of data in the claim events were verified and Scyther did not find any feasible attacks within the bounded state space. The Scyther script can be downloaded from [1].

7 Conclusion & Future Work

The work carried out in this paper first looked into the current architecture of EMV contactless mobile payments based on tokenisation. Then five potential attack scenarios in two problem areas that would compromise tokenised contactless mobile payments were discussed. To meet the objectives of the paper and to address the raised security concerns a protocol was proposed in the paper. The proposed protocol was analysed for its security and objectives. Finally the protocol was subjected to mechanical formal analysis which did not find any feasible attacks within bounds. In our further research directions, we are in the process of implementing the protocol in order to carry out measurements. We also aim to extend to include additional transaction modes and expand our threat model to include the mobile being compromised by an adversary.

References

1. Scyther script for the protocol `https://www.dropbox.com/s/euqnwf0ds17zd6v/DTT%20Protocol%20Scyther%20Script.spdl?dl=0`
2. EMV integrated circuit card specifications for payment systems, Book 2: security and key management, Version 4.3, EMVCo, LLC (November 2011)
3. EMV integrated circuit card specifications for payment systems, Book 3: application specification, Version 4.3, EMVCo, LLC (November 2011)

4. EMV payment tokenisation specification: technical framework, Version 1.0, EMVCo, LLC (March 2014)
5. Apple pay (July 2015), `http://www.apple.com/uk/apple-pay/`
6. EMV contactless specifications for payment systems, EMVCo, LLC (March 2015)
7. Android pay (June 2016), `https://developers.google.com/android-pay/`
8. Askoxylakis, I., Pramateftakis, M., Kastanis, D., Traganitis, A.: Integration of a secure mobile payment system in a GSM/UMTS SIM smart card. System 12, 13
9. BBC News: US and UK accused of hacking SIM card firm to steal codes (20th February 2015), `http://www.bbc.co.uk/news/technology-31545050`
10. Bond, M., Choudary, O., Murdoch, S.J., Skorobogatov, S., Anderson, R.: Chip and Skim: cloning EMV cards with the pre-play attack. In: 2014 IEEE Symposium on Security and Privacy. pp. 49–64. IEEE (2014)
11. Bond, M., Choudary, O., Murdoch, S.J., Skorobogatov, S., Anderson, R.: Be prepared: the EMV pre-play attack. IEEE Security & Privacy (2015)
12. Chothia, T., Garcia, F., de Ruiter, J., van den Breekel, J., Thompson, M.: Relay cost bounding for contactless EMV payments. In: Financial Cryptography and Data Security, vol. 8975, pp. 189–206. LNCS, Springer Berlin Heidelberg (2015)
13. Computerworld.com: Vulnerabilities found in three popular payment terminal models can result in credit card data theft (26th July 2012), `http://www.computerworld.com/article/2504956/security0/payment-terminal-flaws-shown-at-black-hat.html`
14. Cremers, C., Mauw, S.: Operational semantics of security protocols. In: Scenarios: models, transformations and tools. LNCS, Springer Berlin Heidelberg (2005)
15. Cremers, C.: The Scyther tool: verification, falsification, and analysis of security protocols. In: Computer Aided Verification. Springer Berlin Heidelberg (2008)
16. Crowe, M., Pandy, S., Lott, D., Mott, S.: Is payment tokenization ready for primetime? perspectives from industry stakeholders on the tokenization landscape. Tech. rep., Federal Reserve Bank of Boston & Federal Reserve Bank of Atlanta (2015)
17. Dolev, D., Yao, A.C.: On the security of public key protocols. Information Theory, IEEE Transactions on 29(2), 198–208 (1983)
18. Drimer, S., Murdoch, S.J., et al.: Keep your enemies close: distance bounding against smartcard relay attacks. In: USENIX Security. vol. 2007 (2007)
19. Jayasinghe, D., Akram, R.N., Markantonakis, K., Rantos, K., Mayes, K.: Enhancing EMV Online PIN Verification. In: Trustcom/BigDataSE/ISPA, 2015 IEEE. vol. 1, pp. 808–817 (Aug 2015)
20. Jayasinghe, D., Markantonakis, K., Gurulian, I., Akram, R., Mayes, K.: Extending EMV Tokenised Payments To Offline-Environments. The 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-16), IEEE Computer Society (6 2016)
21. Kaspersky Lab: Equation group: the crown creator of cyber-espionage (16th February 2015), `http://www.kaspersky.com/about/news/virus/2015/equation-group-the-crown-creator-of-cyber-espionage`
22. Kaspersky Lab: The great bank robbery (16th February 2015), `http://www.kaspersky.com/about/news/virus/2015/Carbanak-cybergang-steals-1-bn-USD-from-100-financial-institutions-worldwide`
23. PCI Security Standards Council: Information Supplement: PCI DSS Tokenization Guidelines, Version 2.0, PCI Data Security Standard (PCI DSS) (Auguest 2011)
24. Symantec: A special report on: attacks on point-of-sales systems (November 2014)
25. TheHackerNews: 324,000 financial records with CVV numbers stolen from a payment gateway (13th September 2016), `http://thehackernews.com/2016/09/bluesnap-payment-gateway-hack.html?m=1`