Extending EMV Tokenised Payments To Offline-Environments

Danushka Jayasinghe, Konstantinos Markantonakis, Iakovos Gurulian, Raja Naeem Akram and Keith Mayes
Smart Card Centre, Information Security Group,
Royal Holloway, University of London, Egham, Surrey, UK, TW20 0EX
Email:{Danushka.Jayasinghe.2012, K.Markantonakis, Iakovos.Gurulian.2014, R.N.Akram, Keith.Mayes}@rhul.ac.uk

***Abstract*— Tokenisation has been adopted by the payment industry as a method to prevent Personal Account Number (PAN) compromise in EMV (Europay MasterCard Visa) transactions. The current architecture specified in EMV tokenisation requires online connectivity during transactions. However, it is not always possible to have online connectivity. We identify three main scenarios where fully offline transaction capability is considered to be beneficial for both merchants and consumers. Scenarios include making purchases in locations without online connectivity; when a reliable connection is not guaranteed; and when it is cheaper to carry out offline transactions due to higher communication/payment processing costs involved in online approvals. In this study, an offline contactless mobile payment protocol based on EMV tokenisation is proposed. The aim of the protocol is to address the challenge of providing secure offline transaction capability when there is no online connectivity on either the mobile or the terminal. The solution also provides end-to-end encryption to provide additional security for transaction data other than the token. The protocol is analysed against protocol objectives and we discuss how the protocol can be extended to prevent token relay attacks. The proposed solution is subjected to mechanical formal analysis using Scyther. Finally, we implement the protocol and obtain performance measurements.**

***Keywords*—EMV Contactless, Mobile Payments, Tokenisation, Ambient Sensor Data, Security, Cryptography, Offline Transaction Tokens.**

I. Introduction

Unlike a contactless smart card, a mobile[1] has additional capabilities including increased computing ability, a greater variety of accessible Application Programming Interfaces (API), and readily available communication channels via a Mobile Network Operator (MNO) or Wi-Fi. Furthermore, modern mobile devices typically feature Near Field Communication (NFC) and hardware or software Secure Element (SE) technologies that provide secure execution environments in which to execute sensitive applications [1]–[3]. Hardware SEs provide a secure storage environment for credentials and offer tamper resistance against physical attacks [4]. Mobile payment applications provide additional features including having a number of virtual contactless payment cards issued by different financial institutions in one place; passcode unlocking mechanisms to access virtual payment cards; and the ability to block such cards if a mobile is lost or stolen.

The properties and features discussed above provide an ideal platform for running mobile-based payment solutions. In addition, at the time of writing, tokenisation is increasingly being adopted by the payments industry. Until recently, tokenisation was used by merchants and payment processing service providers to store consumer card details in a tokenised format to mitigate the risk of card data being hacked from databases. Following the standardisation of EMV tokenisation specification [5], there has been a dramatic move towards early adoption of this technology in contactless mobile payment applications. One example is the release of Apple Pay [6, 7].

Personal Account Number (PAN) compromise involves obtaining PAN-related data for financial fraud. PAN data can be compromised by adversaries during EMV transactions or by hacking merchants' databases that store consumer payment card details. Tokenisation, discussed in this paper, is a method that replaces the sensitive PAN used during an EMV transaction with a substitute value called the token[2]. An adversary who captures the token cannot deduce the actual PAN. Tokenised payments thus prevent PAN compromise in EMV transactions.

The EMV Tokenisation Specification details the requirements for supporting payment tokenisation in EMV transactions [5]. There are many challenges yet to be addressed in the tokenisation landscape [7]. The lack of support for making or accepting tokenised payments in an offline transaction environment is the main focus of this paper. The current tokenisation architecture requires online connectivity on the terminal[3] in order to reach the payment authorisation entity during a tokenised transaction. However, it is not always possible to have online connectivity in certain transaction scenarios. In this paper, we identify three scenarios where a fully offline transaction capability is considered beneficial.

1) Connectivity is not possible due to the geographical location of a transaction, such as purchases made on aeroplanes and underground subway systems.
2) Steady/continuous connectivity is not guaranteed. If a business is operating in a non-stationary environment, it is most unlikely that the merchant's portable payment acceptance terminal has continuous connectivity to the payment network; for example, a merchant who sells snacks on a fast-moving train using a portable payment terminal. If the merchant is able to accept payments from customers wanting to make tokenised contactless mobile

[1] In this study, the payer's contactless mobile payment device that emulates a contactless smart card is referred to as the mobile.

[2] A 'token' is a 13-19 digit numeric value that passes validation checks set by the payment scheme. A token is generated such that it does not reveal or conflict with the real PAN [5].

[3] In this study, the terminal is a stationary or a portable payment acceptance device which accepts EMV contactless payments.

payments, this may significantly improve the merchant's business on the day. Also, the consumers would be protected by the security of being able to make tokenised payments conveniently.

3) It is significantly cheaper to carry out offline transactions due to the communication and processing costs involved with establishing each online transaction individually. An example is carrying out a number of transactions offline and then forwarding all the transactions simultaneously for batch payment processing at a later time.

From the above discussion and example scenarios, the inability to make/accept tokenised payments in an offline environment may act as a deterrent for both consumers and merchants. This could hinder the wider adoption of contactless mobile payment solutions based on tokenisation.

In this paper, we propose a contactless mobile payment protocol based on EMV tokenisation that allows offline token payments when no online connectivity is present on either the terminal or the mobile. The proposed solution also provides end-to-end encryption between the secure element of the mobile and the terminal. This provides security for transaction data other than the token. The protocol is analysed against protocol objectives and subjected to mechanical formal analysis using Scyther. In our analysis, we show that while tokenised payments prevent PAN compromise during transactions, they are still susceptible to token relay attacks (discussed in Section V-A). We then discuss how the proposed protocol can be extended to detect and prevent potential token relay attacks using ambient sensing. Finally we implement the protocol and provide performance measurements. The main contributions of this paper are the following:

1) The protocol introduces the *Offline Transaction Token (OTT)*, providing the ability to make fully offline tokenised payments
2) End-to-end encryption between the secure element of the mobile and terminal provides additional security for transaction data other than the Token

The remainder of the paper is structured as follows. In Section II, EMV tokenisation and the current operating environment of tokenised EMV payments are presented. In Section III, the prospective offline operating environment and the adversary's capability are discussed. The proposed protocol is presented in Section IV. A security analysis of the protocol is carried out in Section V and the protocol is subjected to mechanical formal analysis in Section VI. The practical implementation of the protocol and performance measurements are given in Section VII. Finally, in Section VIII, the discussion is concluded and further research directions are identified.

II. EMV Tokenisation and Current Operating Environment

In this section, an introduction to tokenisation and its current online operating environment is presented. A generic payment architecture and the transaction message flow for a tokenised contactless mobile payment are shown in Figure 1 and explained according to [5, 7].

One potential security issue in contactless payments is PAN compromise. The harvested PAN-related data is then used to carry out cross-channel fraud[4]. Tokenisation has been adopted by the payment industry as a method to prevent PAN compromise by mapping the PAN with a substitute value. EMV tokenisation is also in the interest of merchants, as storing and managing tokens instead of PANs in merchants' servers and databases can simplify compliance audits such as the Payment Card Industry-Data Security Standard (PCI-DSS) [8, 9]. The tokenisation discussed in this study refers to replacing the PAN used during EMV transactions with a token defined in EMV Tokenisation Specification [5]. This paper does not refer to independent tokenisation methods used by various merchants and their payment processors in order to manage and store consumer payment card details in token format.

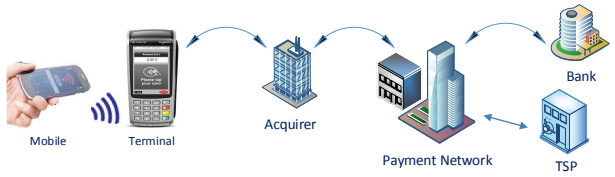

Fig. 1. Generic EMV Tokenised Payment Architecture

At the start of an online tokenised EMV contactless mobile payment transaction, the secure element in the mobile passes the payment token and token-related data to the terminal. The terminal forwards the authorisation request to the acquirer, who then forwards it to the payment network. The entities involved in this process engage in a key translation process, which means that two entities share a symmetric key to communicate, as shown by connecting arrows on both ends in Figure 1. When a message is forwarded to a particular entity, it is deciphered and enciphered using the shared symmetric key with the next entity. The payment network then communicates with the Token Service Provider (TSP) to de-tokenise the token in order to retrieve the PAN and validate the cryptogram [5, 7]. Following this, the payment network forwards the authorisation request with the mapped PAN details to the bank for authorisation. The bank, after carrying out necessary validations, sends an Authorisation Response Code (ARC) to the payment network in an authorisation response [5, 10]. The payment network may or may not send a TSP-generated response cryptogram in the authorisation response message to the terminal via the acquirer [5]. As explained, the current architecture requires online connectivity during transactions.

III. Offline Operating Environment and Adversary's Capability

In this section, first the prospective offline operating environment in order to provide offline payments based on EMV tokenisation is provided. Then the capabilities of the adversary in this environment and potential risk scenarios are discussed.

In the offline transaction environment considered in this paper, online connectivity to reach the authorising entity is not available on either the secure element or the terminal. The secure element and the terminal are the only two parties involved during the transaction. Therefore, in such a scenario the terminal needs to decide whether to accept or decline a

[4]Cross-Channel Fraud, is capturing card details in a Point of Sale (POS) transaction and using the details in other payment channels such as e-commerce payments.

tokenised transaction. As the payment transaction is carried out offline, it is paramount to secure the payment and the communication between the secure element and the terminal. The payment settlement phase is carried out when the terminal has online connectivity at a later time.

Taking the potential offline-based operating environment as discussed above into consideration, the capabilities of the potential adversary who may compromise the tokenisation-based payment system are listed below:

- Cannot break the standardised (strong) encryption algorithms.
- Has the capability to eavesdrop unencrypted messages passed between the secure element and the terminal.
- Cannot compromise the terminal's public key certificate introduced in IV-A. In Section VI, we discuss why the terminal's public key certificate is not in the adversary knowledge in the adversary model.
- Cannot compromise the secure element, terminal, scheme operator, token service provider or the bank.

Based on the adversary's capability and the current operating environment, an adversary who compromises token transaction data during an offline transaction scenario may carry out fraudulent transactions. The risk scenarios involved may include the adversary changing the transaction amount to a new value, capturing the OTT, or replaying the same OTT to carry out multiple offline transactions with different terminals. Therefore, as well as providing offline token transaction capability, it is vital to secure the sensitive token transaction data communicated between the secure element and the terminal.

In order to store the OTT securely on the mobile phone for offline use and to prevent the OTT being compromised, tamper resistant storage and secure execution are needed. In our threat model for the complete offline payment environment, the OTT needs to be protected from the adversary. A mobile phone with a software trusted execution environment such as Host Card Emulation (HCE) does not offer tamper resistance. Therefore, we consider a mobile phone with an embedded secure element which offers both security guarantees in our proposal. In the next section, we take these concerns into consideration and propose our protocol.

IV. Proposed Solution

In this section, an offline contactless mobile payment protocol based on EMV tokenisation is proposed. The payment protocol is used to make offline payments when there is no online connectivity on either the terminal or the mobile during a tokenised transaction. The objectives of the proposed protocol are listed below.

1) The protocol should be able to make secure offline payment transactions.
2) End-to-end encryption should be provided between the SE and terminal. This offers additional security to protect transaction data other than the Token.

A. Protocol Assumptions

The assumptions made in the proposed solution are listed below:

- The Token Service Provider (TSP) is a trusted entity that securely generates, issues and de-tokenise transaction tokens on behalf of the bank.
- The TSP in the proposed protocol takes part in the same payment scheme and the TSP generated signatures can be verified by the terminal following the certificate hierarchy shown in Figure 3.
- The terminal has been issued with a public key certificate signed by the scheme operator.
- The Offline Transaction Token (OTT) and security-sensitive data including the cryptographic keys are securely stored in the secure element. It is not possible for an adversary to steal/compromise the OTT or data residing in the secure element.
- The nonces generated by the terminal are random and unpredictable. An adversary cannot deduce the second nonce by examining the first nonce.

The notation used in the proposed solution is included in Table I. The tokenised contactless mobile payment architecture of the proposed protocol is illustrated in Figure 2. The protocol proposed in this study comprises a setup phase, a payment phase and a settlement phase. In the *Setup Phase*, the payment app and the Offline Transaction Token (OTT) are securely provisioned to the secure element of the mobile. The payment phase can be used to make an offline tokenised payment when there is no online connectivity on either the terminal or the mobile.

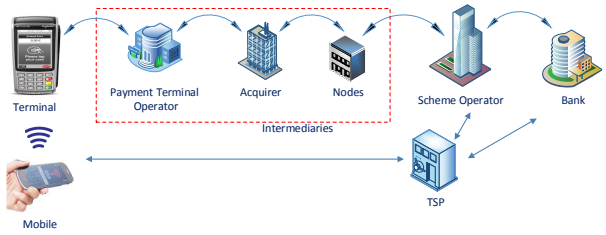

Fig. 2. Tokenised Contactless Mobile Payment Architecture of the Proposed Protocol

TABLE I. Notation used in the proposed protocol

T/SE/SO/x	:	Terminal/Secure Element/Scheme Operator/Identity of X.
TSP/OTT	:	Token Service Provider/Offline Transaction Token.
$TATC$	:	Token Application Transaction Counter, count of token transactions since personalisation. It is shared between $mobile, bank$ & TSP and used during key derivations.
$MaxValue$	:	Maximum value of the total offline token transactions allowed per OTT. Predefined value set by the bank.
T_{limit}	:	Transaction Limit is a record kept in the secure element. It is the total value of previous offline token transactions.
$K_{To'}$	:	Token Cryptogram Generation Symmetric Session Key derived by a key derivation function used by TSP.
K	:	SE generated Symmetric Session Key.
$E_K\{Z\}$	:	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.
a_X	:	Ambient sensor details issued by entity X.
$h(Z)$	:	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.

B. Setup Phase

During the setup phase, the personalisation of the payment application and provisioning of OTT related data

is accomplished. The provision of payment application and security-sensitive data to the secure element is carried out using a secure Over-The-Air (OTA) channel [11]. Following application personalisation, the payment application's sensitive data elements reside in the secure element and its user interface is located in the mobile platform. The elements in the secure element consist of all cryptographic keys needed by the mobile, i.e. S_{mobile} and P_{mobile}. The secure element also stores: $Cert_{bank}(TSP)$, $Cert_{bank}(SE)$, Token Application Transaction Counter $(TATC)$, OTT and the token service provider's digital signature on the hash of Static Token Data (STD) which has the notation $sS_{TSP}\,[h(STD)]$. The OTT is generated by the TSP and has a strong cryptographic binding with the $bank$, TSP, $Token$ and n_{tsp}. The OTT consists of a TokenData part and an encrypted part (Token Cryptogram). The construction of the OTT is shown below.

$$OTT = TokenData||TokenCryptogram$$

$$TokenData = Token||TokenExpiry||TokenR\text{-}ID^{5}$$

$$TokenCryptogram = E_{K_{To'}}\{TokenData||MaxValue||CurrencyCode||n_{tsp}\}$$

The OTT is securely provisioned to the secure element using an OTA channel. Once the OTA is provisioned the secure element keeps a record of the $MaxValue$ and the T_{limit}. The $MaxValue$ is the total offline token transactions value allowed for a particular OTT. This is a predefined value set by the issuing bank, taking the liability involved in offline transactions into consideration. This is similar to how a daily liability cap on EMV contactless card transactions is set for offline use. The T_{limit} is the total value of previous offline token transactions carried out using a particular OTT. In a given transaction scenario, the secure element adds the prospective transaction value to the T_{limit} to check whether the combined value exceeds the $MaxValue$. The secure element only presents the OTT to a terminal if the prospective transaction does not exceed the $MaxValue$.

However, whenever there is online capability and the $MaxValue$ of a particular OTT has been reached, a new OTT is provisioned automatically to the secure element using an OTA channel. The provisioning of the new token simultaneously resets the T_{limit}.

Following personalisation of the payment application, the user is required to enter a passcode on first access. A strong hash function such as an SHA256 hashing algorithm is then applied to the passcode, and this hash $\mathcal{H}$ is placed in the secure element for future authentication of the user with the payment application.

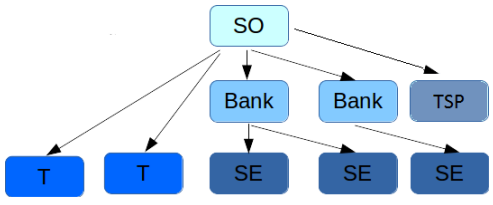

Fig. 3. Certificate Hierarchy used in the Proposed Architecture

The certificate hierarchy used to verify the public encryption keys and signature verification keys of the entities is shown in Figure 3. The scheme operator is at the top level of the certificate hierarchy used in the proposal. Terminals and secure elements participating in the payment scheme can verify certificates issued by the scheme operator or entities that have been certified to be trusted in the certificate hierarchy. The TSP also takes part in the payment scheme. As illustrated in Figure 3, the secure element's public key is certified by the bank and the terminal's public key is certified directly by the scheme operator. We have constructed the certificate hierarchy in this manner because the number of secure elements that need certifying is greater than the number of terminals. In the current payment architecture, a terminal is deployed by the merchant's acquiring bank or by a subcontractor of the acquiring bank. In our proposal, the scheme operator certifies the terminal's public key. However, on a practical note, in the current payment architecture, terminal manufacturers need to have their terminals certified by a scheme operator. Therefore, it is within the capability of the scheme operator to certify each terminal's public key directly, probably at the same time.

C. Payment Phase

The protocol messages of the proposal are illustrated in Table II and explained as follows. To make a contactless mobile payment, the user opens the payment application by entering the passcode and taps the device on the terminal's Near Field Communication (NFC) field.

TABLE II. OFFLINE TRANSACTION TOKEN PROTOCOL.

1.	$T \rightarrow SE$	:	$t\|\|n_t\|\|Cert_{SO}(T)$
2.	$SE \rightarrow T$	:	$eP_T\{se\|\|t\|\|n_{se}\|\|n_t\|\|PDOL\|\|K\}$
3.	$T \rightarrow SE$	:	$E_K\{t\|\|se\|\|n_{se}\|\|n_{2t}\|\|amount\|\|CurrencyCode\}$
4.	$SE \rightarrow T$	:	$E_K\{se\|\|t\|\|n_{2se}\|\|n_{2t}\|\|OTT\}\|\|sS_{TSP}\,[h(STD)]\,\|\|\,sS_{SE}\,[h(DAD)]\,\|\|Cert_{bank}(SE)$ $STD = se\|\|OTT$ $DAD = n_{2t}\|\|n_{2se}\|\|OTT$
a.	T	:	SE read complete
b.	T	:	offline token & dynamic data authentication
c.	T	:	approved/declined - post-transaction clearing request

Message 1: In the first transaction message, the terminal provides its identity, terminal-generated first nonce and the terminal's public key certificate to the secure element.

Message 2: The secure element then verifies $Cert_{SO}(T)$ and recovers P_T. Only a genuine terminal can provide a public key certificate that verifies correctly. The second message includes; the identities, n_{SE}, received nonce, the Processing Options Data Object List (PDOL) [12] and the session key. The message is enciphered using P_T before sending to the terminal. The PDOL instructs the terminal what information to transmit back to the secure element.

Message 3: The terminal first deciphers the received message and prepares to send the data requested in the PDOL. The prepared message includes the identities, the secure element's nonce received previously, a fresh nonce, and the amount and the currency code of the transaction. The message is then enciphered using K before sending it to the secure element.

Message 4: The secure element deciphers the previous message and then carries out the following verification steps before constructing message 4:

[5]Token Requester ID: 11-digit unique numeric value, positions 1-3 indicating TSP and positions 4-11 indicating the requester and token domain [5].

- The secure element checks whether it has received the expected n_{SE} in order to detect any replay attempts.
- The secure element examines the prospective transaction amount to check it is within the maximum value of a single offline token transaction set by the issuer. If the transaction amount exceeds the maximum value the secure element declines the transaction, otherwise it proceeds to the next verification.
- The secure element then verifies whether the amount of the transaction is with the required limits. This is done by examining the Transaction Limit T_{limit} kept in the secure element's record. The T_{limit} includes the total value of previous offline token transactions. The secure element adds the prospective transaction amount to the value in the T_{limit} and checks to see whether the final amount exceeds the *MaxValue*, which is the total offline token transaction value allowed for a particular OTT. If the *MaxValue* has been reached, the secure element declines the transaction; otherwise the secure element proceeds to construct message 4. The secure element updates the T_{limit} record when message 4 is successfully issued.

The secure element uses the Offline Transaction Token (OTT) for the payment transaction and includes the following data in the constructed message: the identities; n_{2se}; n_{2t}; and the OTT. The message is enciphered using K. The n_{2t} forms part of the Dynamic Application Data (DAD) used by the terminal to detect any replay attempts.

Message 4 also includes the $sS_{TSP}\,[h(STD)]$, $sS_{SE}\,[h(DAD)]$ and the public key certificate. The $TSP's$ signature on static token data can be used by the terminal to carry out *offline token data authentication* to verify the authenticity of the OTT related data offline. The $SE's$ signature on the DAD can be used by the terminal to carry out *offline dynamic data authentication* to verify that it is communicating with a genuine secure element. Finally the public key certificate can be used to verify the signature through the certificate hierarchy.

Once message 4 is successfully sent, the secure element may leave the NFC field of communication. The terminal then carries out the following four verification steps before the OTT transaction is approved or declined:

- The terminal verifies $sS_{TSP}\,[h(STD)]$ by generating $h(SE||OTT)$ and comparing it with the hash recovered from $sS_{TSP}\,[h(STD)]$. If the two hashes match, this verifies that the TSP has signed the STD presented to the terminal by the secure element. This provides assurance to the terminal regarding the authenticity of the OTT. If *offline token data authentication* fails, the terminal declines the transaction.
- The terminal then verifies the $sS_{SE}\,[h(DAD)]$ produced by the secure element. The terminal generates the hash of the DAD received in message 4 and then compares this with the hash recovered in the $sS_{SE}\,[h(DAD)]$. If the two hashes match, *offline dynamic data authentication* is verified successfully, otherwise the transaction is declined due to the potential of a replay attack. A replay of OTT can be detected by the terminal due to a replayed message 4 not having the terminal-generated n_{2t} in the $sS_{SE}\,[h(DAD)]$.

If the verification steps are completed successfully, then the terminal approves the offline token transaction. If thay fail, then the transaction is declined. In both cases, the outcome is displayed to the user on the terminal and a printed transaction receipt may be produced. The terminal issues a *token payment clearing request* when the terminal is online capable at a later time, following a successful transaction of an offline token payment. In the event of an unsuccessful offline token payment, the terminal declines the transaction, displays a decline message on the terminal and a *token payment clearing request* is not sent. The token payment clearing request starts the settlement phase. The settlement process of the offline token payment may follow the same transaction processing channel as specified in the EMV tokenisation specification [5] and discussed in section II. The terminal may forward the token payment clearing requests which include: OTTs and transaction details from a number of transactions, in bulk to the acquirer. The acquiring bank then forwards the request to the scheme operator who then communicates with the TSP. The TSP is able to detokenise and validate the OTTs. The retrieved PAN and transaction details are forwarded to the issuing bank for payment clearing. The acquiring bank is settled via the scheme operator.

V. Analysis

In this section, the proposed solution is evaluated to see whether it achieves the protocol objectives. The analysis discusses how the protocol can be extended to prevent token relay attacks.

The analysis takes into consideration the operating environments outlined in Section II, protocol assumptions outlined in Section IV-A and those described during the setup stage in Section IV-B.

1) **Secure offline payment transactions:** Achieving offline transaction capability during a tokenised payment was the main focus of the paper. The proposed contactless mobile payment protocol based on EMV tokenisation provides capability of making tokenised payments in a fully offline environment. The proposed protocol, in order to achieve the objective of making offline token payments, uses the OTT which includes STD as payment data.

   The terminal carries out four verification steps to verify whether the offline transaction is genuine. The terminal carries out *offline token data authentication*, by verifying $sS_{TSP}\,[h(STD)]$. This gives an assurance regarding the authenticity of the OTT. The terminal does *offline dynamic data authentication* by verifying the $sS_{SE}\,[h(DAD)]$. By doing this, a replay of OTT is detected by the terminal as the message would not include n_{2t} if it is not genuine. If these verifications fail, the terminal declines the transaction.

2) **End-to-end encryption to protect transaction data:** The protocol provides end-to-end encryption between the terminal and the secure element. This provides confidentiality by preventing adversaries from eavesdropping on sensitive token transaction-related data during transactions. The end-to-end encryption provides security for

transaction-related data other than the token. We further establish this in Section VI where we analyse the protocol in Scyther. Scyther did not find any feasible attacks including attacks on the secrecy of transaction data.

A. Token Relay Attack

Relay attacks during EMV contactless payment transactions have been examined in [13]–[15]. Even though tokenisation prevents PAN compromise during an EMV transaction, the current EMV tokenisation architecture specified in [5] does not address concerns about relaying EMV tokenised payments. Consider the example of a token relay attack where a genuine consumer makes a token-based contactless mobile payment at a compromised terminal. The transaction is then relayed to a rogue secure element elsewhere, which makes a payment at a genuine terminal simultaneously.

Our proposed protocol can be extended to detect and prevent token relay attacks by adopting ambient sensing, as discussed and illustrated in Figure 4. An introduction to ambient sensing and the explanation of how this can be used is given in Section A. A mobile device and a supported terminal capture their own ambient environment-related data on each device using on-board sensors. In the protocol, the mobile sends its ambient sensor data a_{mobile} in message 4 as shown below.

TABLE III. EXTENDED PROTOCOL MESSAGES.

4.	$SE \rightarrow T$	:	$E_K\{SE\|\|n_{2se}\|\|n_{2t}\|\|OTT\|\|\mathbf{a_{mobile}}\}\|\|$ $sS_{TSP}\left[h(STD)\right]\|\|sS_{SE}\left[h(DAD)\right]$
	DAD	=	$n_{2t}\|\|n_{2se}\|\|OTT\|\|\mathbf{a_{mobile}}$
b.	T	:	offline token relay detection

In the protocol stage, any attempted token relay attacks are detected offline by the terminal. This is due to the transaction being offline and a trusted third party, such as the TSP, being unavailable to act as a comparing party. To this end, the terminal generates its own ambient sensor data $a_{terminal}$ and compares it with the a_{mobile} received in message 4. This verification can be completed in step b of the protocol. As a_{mobile} forms part of the DAD, it provides data origin authentication of a_{mobile} and other dynamic application data to the terminal by verifying $sS_{SE}\left[h(DAD)\right]$. If the two components match or meet the expected threshold, then the terminal proceeds to the next verification stage; otherwise the terminal declines the offline token transaction because of the potential of a token relay attack.

VI. MECHANICAL FORMAL ANALYSIS

In this section, the proposed protocol is subjected to mechanical formal analysis using Scyther [16].

The description of a protocol was modelled and provided as input to Scyther using the Security Protocol Description Language (spdl) defined in [17]. The spdl provides three main protocol modelling features: *roles, events and claims*. The entities in a protocol are described using a set of roles, 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.

The adversarial model used in this analysis is the Dolev-Yao model in [18]; Scyther is capable of supporting additional adversarial models. The following security claims are verified in the analysis: *Secrecy of data* `(Secret)`, *Aliveness* `(Alive)`, *Weak agreement* `(Weakagree)`, *Non-injective agreement* `(Niagree)` *and Non-injective synchronisation* `(Nisynch)` [16, 17]. In addition to the claim types defined above, the *verify automatic claims* feature on Scyther was used to verify other claims [16]. In the adversary model, we have excluded the terminal's public key certificate from the adversary's knowledge. This is because only a genuine terminal is able to provide its public key certificate to the secure element and the adversary is unable compromise a genuine public key certificate that follows a certificate hierarchy.

Following successful execution of the script, the security of data in the claim events were verified and Scyther did not find any feasible attacks other than the secrecy of the terminal's first generated nonce (nt) sent in Message 1. The Scyther script is available in Appendix B and can be downloaded from [19]. This is because, in our protocol construction, (nt) is sent in clear text in Message 1. However, according to the protocol assumptions in Section IV-A, nonces are random and unpredictable. Therefore, an adversary cannot deduce the second generated nonce $(nt2)$ by examining (nt). As a result, knowledge of (nt) is of no value to the adversary because for the construction of the DAD, only the second nonce $(nt2)$ is used.

VII. PRACTICAL IMPLEMENTATION

In this section, we provide details of the protocol implementation, our experience, and performance measurements for the protocol.

The protocol was implemented to obtain performance measurements and to provide a comparison with other protocols. In our implementation, a Java application was developed to run as the terminal on a Microsoft Windows 7 PC with a 3.2GHz processor and an 8GB RAM. Then a separate Java card applet was developed to run the payment application on the mobile. The applet was provisioned to the 16-bit hardware secure element of a Nokia 6131 mobile phone. For our implementation, obtaining a mobile phone with an embedded secure element that gave read/write permissions to the secure element was a challenging task. The only mobile phone with an embedded secure element with read/write access to provision our payment applet that was available at the time was the Nokia 6131 mobile phone. In our applet development phase, we found that Java card frameworks v2.2.2 or above were not supported by the secure element. In order to provide compatibility with the secure element, the Java card applet was compiled using Java card framework v2.2.1.

All four messages of our proposed protocol detailed in Table II were implemented. The communication between the terminal and the secure element was carried out by command and response Application Protocol Data Units (APDU) [20, 21]. In our implementation for asymmetric encryption, we used plain RSA [22] with 1024-bit key and recommended padding. We used MD5 [23] as the hashing algorithm and the RSA Digital Signature Algorithm for signatures [24].

During our implementation, we found that the secure element and the Java card framework v2.2.1 did not support the Advanced Encryption Standard (AES). Because of this limitation on the secure element, we used double-length key Triple DES [25] in Electronic Codebook mode for symmetric encryption. The algorithm was also supported by default in the Java card framework v2.2.1. It is important to note that, double-length key Triple DES is an approved cryptographic algorithm and electronic codebook is an approved mode of operation in the EMV specification [26, see: p135-145].

The protocol is scalable to use other advanced algorithms, modes of operations and hashing algorithms. If the implementation had been carried out on a modern mobile phone with an embedded secure element, we would have used AES for symmetric encryption and SHA256 as the hashing algorithm.

TABLE IV. PERFORMANCE MEASUREMENTS COMPARISON IN MILLISECONDS

Measures	**Proposed Protocol OTT**	**SSL [27]**	**TLS [28]**	**P-STCP [29]**	**STCP [30]**
Specification	16bit	32bit	32bit	16bit	16bit
Time to complete	3971ms	4200ms	4300ms	4344ms	3875ms

A performance measurements comparison between the proposed Offline Transaction Token (OTT) protocol and some other protocols implemented on smart cards is shown in Table IV. At the time of writing, we could not find any offline token protocols or their timing measurements in published literature to enable us to carry out a more accurate comparison. In order for us to understand the performance of the protocol, we choose performance measurements from four different protocols available in the literature. The chosen protocols were also implemented on smart cards for measurements in their corresponding papers.

Three different timing measurements were obtained from our implemented protocol. We recorded timing measurements for Message 2, Message 4 and the total time for the protocol to complete. For Messages 2 and 4, timing was measured from the time the command APDU [20] was sent from the terminal to the time it received the response APDU [20] from the secure element. The overall protocol completion time was measured from the time the applet selection command APDU was sent from the terminal to the time it completed all the verifications after receiving Message 4. The measurements for Message 2 and Message 4 were 751 milliseconds and 3086 milliseconds respectively. The overall protocol completed in 3971 milliseconds. The overall time it took to complete the protocol fell in the same performance range as the compared protocols in Table IV. Another point to note is that, the 16bit hardware secure element on the Nokia 6131 mobile phone used in our implementation was released in 2006. A more recent 32bit secure element on a modern smart phone would most probably give improved performance measurements.

VIII. CONCLUSION & FUTURE WORK

In this paper, an Offline Transaction Token (OTT) protocol based on EMV tokenisation was proposed. The proposed solution achieves the two main objectives of the protocol: to be able to make secure offline token transactions and to provide end-to-end security between the terminal and the secure element. The proposal was analysed and we further identified how the protocol could be extended to prevent potential token relay attacks. Finally, we subjected the protocol to mechanical formal analysis using Scyther and provided performance measurements from a practical implementation. At the time of writing, apart from [31, 32], there is no publicly available academic research based on EMV tokenisation. To the authors' knowledge, the work carried out in this study is the first to propose an offline transaction token protocol with mechanical formal analysis, practical implementation and performance measurements.

In further research, we aim to explore other transaction scenarios such as making an offline token payment when the terminal is online-capable. We also aim to implement our protocol on a 32bit secure element on a modern smart phone to compare performance variations.

REFERENCES

[1] *NFC Technical Specifications, NFC Forum, (Accessed Online, July 2015)*, Std. [Online]. Available: http://members.nfc-forum.org/members/specifications/technical_specs/

[2] *EMV Contactless Mobile Payment: Application Activation User Interface, Version 1.0, EMVCo, LLC*, Std., December 2014.

[3] *EMVCo Mobile Contactless: EMV Profiles of GlobalPlatform UICC Configuration, Version 1.0, EMVCo, LLC*, Std., December 2014.

[4] *EMV Mobile Contactless Payment: Technical Issues and Position Paper, Version 1.0, EMVCo, LLC*, Std., October 2007.

[5] *EMV Payment Tokenisation Specification: Technical Framework, Version 1.0, EMVCo, LLC*, Std., March 2014.

[6] Apple, "Apple pay," July 2015. [Online]. Available: http://www.apple.com/uk/apple-pay/

[7] M. Crowe, S. Pandy, D. Lott, and S. Mott, "Is payment tokenization ready for primetime? perspectives from industry stakeholders on the tokenization landscape," Federal Reserve Bank of Boston and Federal Reserve Bank of Atlanta, Tech. Rep., June 2015.

[8] PCI Security Standards Council, *Information Supplement: PCI DSS Tokenization Guidelines, Version 2.0, PCI Data Security Standard (PCI DSS)*, Std., Auguest 2011.

[9] P. S. S. Council, *Tokenization Product Security Guidelines: Irreversible and Reversible Tokens, Version 1.0, PCI Data Security Standard (PCI DSS)*, Std., April 2015.

[10] D. Jayasinghe, R. N. Akram, K. Markantonakis, K. Rantos, and K. Mayes, "Enhancing EMV Online PIN Verification," in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, vol. 1, Aug 2015, pp. 808–817.

[11] *GlobalPlatform System: Messaging Specification for Management of Mobile-NFC Services, V 1.1.2, GlobalPlatform.*, Std., December 2013.

[12] *EMV Contactless Specifications for Payment Systems, Version 2.5, EMVCo, LLC*, Std., March 2015.

[13] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis, "Practical nfc peer-to-peer relay attack using mobile phones," in *Proceedings of the 6th International Conference on Radio Frequency Identification: Security and Privacy Issues*, ser. RFIDSec'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 35–49.

[14] M. Roland, J. Langer, and J. Scharinger, "Relay attacks on secure element-enabled mobile devices," in *Information Security and Privacy Research*. Springer Berlin Heidelberg, 2012, vol. 376, pp. 1–12. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30436-1_1

[15] T. Chothia, F. Garcia, J. de Ruiter, J. van den Breekel, and M. Thompson, "Relay Cost Bounding for Contactless EMV Payments," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2015, vol. 8975, pp. 189–206.

[16] C. Cremers, "The scyther tool: Verification, falsification, and analysis of security protocols," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, vol. 5123, pp. 414–418. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-70545-1_38

[17] C. Cremers and S. Mauw, "Operational semantics of security protocols," in *Scenarios: Models, Transformations and Tools*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, vol. 3466, pp. 66–89. [Online]. Available: http://dx.doi.org/10.1007/11495628_4

[18] D. Dolev and A. C. Yao, "On the security of public key protocols," *Information Theory, IEEE Transactions on*, vol. 29, no. 2, pp. 198–208, 1983.

[19] "Scyther script for the protocol." [Online]. Available: https://www.dropbox.com/s/9pppyx091si3uxk/OTT%20Scyther%20Script%20Final.spdl?dl=0

[20] W. Rankl and W. Effing, *Smart card handbook*. John Wiley and Sons, 2010.

[21] A. Umar, K. Mayes, and K. Markantonakis, "Performance variation in host-based card emulation compared to a hardware security element," in *Mobile and Secure Services (MOBISECSERV), 2015 First Conference on*. IEEE, 2015, pp. 1–6.

[22] B. Schneier, *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. New York, NY, USA: John Wiley & Sons, Inc., 1995.

[23] R. Rivest, "The md5 message-digest algorithm," 1992.

[24] M. Bellare and P. Rogaway, "The exact security of digital signatures-how to sign with rsa and rabin," in *Advances in CryptologyEurocrypt'96*. Springer, 1996, pp. 399–416.

[25] J. Kelsey, B. Schneier, and D. Wagner, *Advances in Cryptology — CRYPTO '96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings*. Springer Berlin Heidelberg, 1996, ch. Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES, pp. 237–251.

[26] *EMV Integrated Circuit Card Specifications for Payment Systems, Book 2: Security and Key Management, Version 4.3, EMVCo, LLC*, Std., November 2011.

[27] P. Urien, "Collaboration of ssl smart cards within the web2 landscape," 2009.

[28] P. Urien and S. Elrharbi, "Tandem smart cards: enforcing trust for tls-based network services," in *Applications and Services in Wireless Networks, 2008. ASWN'08. Eighth International Workshop on*. IEEE, 2008, pp. 96–104.

[29] R. N. Akram, K. Markantonakis, and K. Mayes, "A privacy preserving application acquisition protocol," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. IEEE, 2012, pp. 383–392.

[30] Akram, Raja Naeem and Markantonakis, Kostantinos and Mayes, Keith, "A secure and trusted channel protocol for the user centric smart card ownership model," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*. IEEE, 2013, pp. 336–345.

[31] F. Corella and K. Lewison, "Interpreting the EMV Tokenisation Specification," 2014.

[32] D. Ortiz-Yepes, "A critical review of the EMV payment tokenisation specification," *Computer Fraud & Security*, no. 10, pp. 5 – 12, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1361372314705391

[33] T. Halevi, D. Ma, N. Saxena, and T. Xiang, "Secure Proximity Detection for NFC Devices Based on Ambient Sensor Data," in *Computer Security, ESORICS 2012*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7459, pp. 379–396.

[34] B. Shrestha, N. Saxena, H. T. T. Truong, and N. Asokan, "Drone to the rescue: Relay-resilient authentication using ambient multi-sensing," in *Financial Cryptography and Data Security*. Springer, 2014, pp. 349–364.

APPENDIX A
AMBIENT SENSING

Figure 4 illustrates how ambient sensing can be used in the proposed EMV tokenisation-based offline contactless mobile payment protocol to detect and prevent token relay attacks. Previous work related to relay attacks detection between an NFC mobile phone and a reader using ambient sensing can be found in [33, 34]. Considering two different ambient environments, AE1 and AE2, a token relay from a genuine mobile in AE1 to a genuine terminal in AE2 via a rogue terminal and mobile can be detected by the genuine terminal or a trusted third party acting as a comparing entity.

Fig. 4. Ambient Sensing as a Token Relay Attack Prevention Countermeasure

As illustrated in Figure 4, this detection is possible due to ambient sensor data $a_{terminal}$ generated by the genuine terminal in AE2 being significantly different to the relayed a_{mobile} produced by the genuine mobile in AE1. The difference in ambient sensor data is detected when the comparison is made. The attributes collected as ambient sensor data may include atmospheric pressure, ambient noise, ambient light, Global Positioning System (GPS) data, and others [34].

APPENDIX B
SCYTHER SCRIPT - OFFLINE TOKEN PROTOCOL

```
usertype Data;
hashfunction h;
usertype SessionKey;
const Cert: Function;
secret Cert1: Function;

protocol ot2(SE,T)
{
role SE {
fresh nse: Nonce;  fresh PDOL: Data;
var nt: Nonce;
fresh OTT: Data;  fresh TSPsig: Data;
macro DAD = nt2, OTT;
fresh K: SessionKey; fresh nse2: Nonce;
var X: Ticket;  var nt2: Nonce;

recv_1(T,SE, T,nt,Cert1(T));
send_2(SE,T, {SE,T,nse,nt,PDOL,K}pk(T));
recv_3(T,SE, {T,SE,nse,nt2,X}K);
send_4(SE,T, {SE,T,nse2,nt2,DAD,TSPsig}K,
{h(DAD)}sk(SE),Cert(SE) );

claim(SE, Alive);
claim(SE, Secret, K);
claim(SE, Niagree);
claim(SE, Nisynch);
claim(SE, Secret, OTT);
claim(SE, Secret, X);
}

role T {
var nse: Nonce; var PDOL: Data;
fresh nt: Nonce;  fresh TTQ: Data;
fresh amount: Data; fresh nt2: Nonce;
fresh CurrencyCode: Data;
var K: SessionKey; var Y: Ticket;
macro m1 = amount,CurrencyCode;
var TSPsig: Data;  var nse2: Nonce;

send_1(T,SE, T,nt,Cert1(T));
recv_2(SE,T, {SE,T,nse,nt,PDOL,K}pk(T));
send_3(T,SE, {T,SE,nse,nt2,m1}K);
recv_4(SE,T, {SE,T,nse2,nt2,Y,TSPsig}K,
{h(Y)}sk(SE),Cert(SE) );

claim(T, Alive);
claim(T, Secret, K);
claim(T, Niagree);
claim(T, Nisynch);
claim(T, Secret, Y);
}
}
```