COMPUTING SCIENCE

POS Terminal Authentication Protocol to Protect EMV Contactless Payment Cards

Martin Emms, Budi Arief, Joseph Hannon and Aad van Moorsel

TECHNICAL REPORT SERIES

No. CS-TR-1401 November 2013
POS Terminal Authentication Protocol to Protect EMV Contactless Payment Cards

M. Emms, B. Arief, J. Hannon and A. van Moorsel

Abstract

The introduction of contactless payment cards into the global EMV payment system introduces a potential vulnerability, in that non-authorised devices can interact with the card, even when still in the cardholder’s wallet. This paper presents a solution addressing this issue, in which the card prevents malicious access to its sensitive data and functionality. The card issues an authentication request which the Point of Sale (POS) terminal must sign with its bank issued private key, before the card will divulge any sensitive information. The proposed solution uses Elliptic Curve Cryptography, a combination of ECQV implicit certificates and ECDSA signatures, which provide both cryptographic strength and efficient use of the limited message size in EMV. There are 23.8 million EMV POS terminals installed globally; the design therefore focuses on integrating POS authentication without altering the EMV protocol. Finally, the paper presents an innovative solution for revocation of the POS terminal keys.
Abstract

The introduction of contactless payment cards into the global EMV payment system introduces a potential vulnerability, in that non-authorised devices can interact with the card, even when still in the cardholder’s wallet. This paper presents a solution addressing this issue, in which the card prevents malicious access to its sensitive data and functionality. The card issues an authentication request which the Point of Sale (POS) terminal must sign with its bank issued private key, before the card will divulge any sensitive information. The proposed solution uses Elliptic Curve Cryptography, a combination of ECQV implicit certificates and ECDSA signatures, which provide both cryptographic strength and efficient use of the limited message size in EMV. There are 23.8 million EMV POS terminals installed globally; the design therefore focuses on integrating POS authentication without altering the EMV protocol. Finally, the paper presents an innovative solution for revocation of the POS terminal keys.

About the authors

Martin Emms is a member of the SiDE team researching technologies that can help survivors of domestic violence. Survivors of domestic violence can be excluded by the technology designed to assist them, the internet browser, SMS and the mobile phone all leave electronic traces behind which their abuser can follow to detect that the survivor has been seeking help which may cause more abuse. Our research aims to create technologies that help survivor access domestic violence support services. The Hyper-DoVe (Hyper-privacy: Case of Domestic Violence) project, develops privacy enhancing technologies for victims of domestic violence. Martin is currently studying for a research PhD at Newcastle University’s Centre for Cybercrime and Computer Security (CCCS), the research focuses on the potential vulnerabilities in Near Field Communications (NFC) based payment technologies and solutions that can reduce or eliminate these vulnerabilities. Prior this Martin was a Solutions Architect specialising in back office, transactional and e-commerce systems for major financial institutions. Working in the UK, US and Australia Martin was responsible for designing, costing and implementing multi-million dollar systems for clients such as Bank of America, Chase (Australia), Commonwealth Bank of Australia, ANZ, Westpac, Computershare, HBoS, Alliance & Leicester and Royal Bank of Scotland.

Dr. Budi Arief is the leading Research Associate (RA) at the Centre for Cybercrime and Computer Security at Newcastle University, one of the recognised UK Academic Centres of Excellence in Cyber Security Research. His research interests include human aspects of computer security, the dependability of computer-based systems, and mobile/ubiquitous computing. In particular, he is applying his extensive experience in interdisciplinary research to address issues related to cybercrime and cyber security. These include investigation into how technology can be used to help everyday users and vulnerable groups, such as children and domestic violence survivors. Budi also manages the UK EPSRC-funded Cybercrime Network, and is a Researcher Co Investigator of the Hyper-DoVe (Hyper-privacy: Case of Domestic Violence) project aiming to help domestic violence survivors. In the past, Budi had worked as an RA on various projects, including the EPSRC-funded Dependability Interdisciplinary Research Collaboration (DIRC), two EU-funded projects (FP6-IST RODIN and FP6-IST TRACKSS), and the EPSRC-funded Trustworthy Ambient Systems (TrAmS) platform grant. He has published over 35 papers, has acted as reviewers for journals/conferences, and he is currently serving as an expert reviewer of an EU FP7 project in the area of Internet of Things.

Joseph Hannon is a final year Computer Science student (at Newcastle University) on route to a 1st in his MComp. His research interests include credit card security, malware and mobile development.
Aad van Moorsel is a Professor in Distributed Systems and Head of School at the School of Computing Science in Newcastle University. His group conducts research in security, privacy and trust. Almost all of the group's research contains elements of quantification, be it through system measurement, predictive modelling or on-line adaptation. Aad worked in industry from 1996 until 2003, first as a researcher at Bell Labs/Lucent Technologies in Murray Hill and then as a research manager at Hewlett-Packard Labs in Palo Alto, both in the United States. He got his PhD in computer science from Universiteit Twente in The Netherlands (1993) and has a Masters in mathematics from Universiteit Leiden, also in The Netherlands. After finishing his PhD he was a postdoc at the University of Illinois at Urbana-Champaign, Illinois, USA, for two years. Aad became the Head of the School of Computing Science in 2012.

**Suggested keywords**

- CONTACTLESS CARD PAYMENT
- ELLIPTIC CURVE CRYPTOGRAPHY
- POINT OF SALE AUTHENTICATION
- EMV
- PAYMENT PROTOCOL
- ECQV
- ECDSA
POS Terminal Authentication Protocol to Protect EMV Contactless Payment Cards

Martin Emms  Budi Arief  Joseph Hannon  Aad van Moorsel
Newcastle University
Centre for Cybercrime & Computer Security
Claremont Tower
Newcastle NE1 7RU
{martin.emms, budi.arief, joseph.hannon, aad.vanmoorsel}@newcastle.ac.uk

Abstract. The introduction of contactless payment cards into the global EMV payment system introduces a potential vulnerability, in that non-authorised devices can interact with the card, even when still in the cardholder’s wallet. This paper presents a solution addressing this issue, in which the card prevents malicious access to its sensitive data and functionality. The card issues an authentication request which the Point of Sale (POS) terminal must sign with its bank issued private key, before the card will divulge any sensitive information. The proposed solution uses Elliptic Curve Cryptography, a combination of ECQV implicit certificates and ECDSA signatures, which provide both cryptographic strength and efficient use of the limited message size in EMV. There are 23.8 million EMV POS terminals installed globally; the design therefore focuses on integrating POS authentication without altering the EMV protocol. Finally, the paper presents an innovative solution for revocation of the POS terminal keys.

Keywords: Contactless card payment, Elliptic Curve Cryptography, Point of Sale Authentication, EMV, Payment Protocol, ECQV, ECDSA

1 Introduction

Researchers have demonstrated that NFC\(^1\)-enabled mobile phones and off-the-shelf NFC readers can be used for skimming attacks [5], eavesdropping attacks [7], access the card’s secure functionality [6] and to make fraudulent transactions [7]. These attacks are possible because the cards will communicate with any device equipped with an NFC reader that comes within range, even without the cardholder’s knowledge. To address this vulnerability, this paper presents a POS terminal authentication solution for EMV contactless payment cards. Terminal authentication allows contact-

\(^1\) Contactless POS terminals use Near Field Communication (NFC) to communicate with contactless payments cards
less card to distinguish genuine POS terminals from other NFC readers and thereby restrict access to its sensitive data and secure application functionality.

Our solution combines several existing technologies in an innovative way to solve the complex issues related to adding functionality to a global payment system such as EMV. The contribution of the solution is outlined below.

**POS Authentication process.** Any NFC reader attempting to access the card will be challenged by the card to authenticate itself as a genuine bank-issued POS terminal. The card sends an unpredictable number (nonce) to the POS terminal, and the terminal must authenticate itself by producing an Elliptic Curve Digital Signature Algorithm (ECDSA) signature based on this nonce challenge. The card is able to validate the ECDSA signature (Fig. 3) using a three-tier Public Key Infrastructure (PKI) based on the Certificate Authority public key that is stored on the card (Fig. 4).

**Restricting access to the card’s sensitive data / secure functionality.** The card will restrict access to sensitive data and secure functionality until POS authentication has taken place. This is achieved using a state machine similar to that currently implemented by MasterCard [15]. The state machine will control the sequence in which EMV commands can be called, and thereby the data that can be accessed (described in Section 2.4).

**Integration with the EMV protocol sequence.** There are four EMV contactless protocol command sequences, one each for Visa, MasterCard, American Express and JCB. The POS authentication functionality has been designed so that it can be incorporated into all four protocol sequences without changing the command sequence. To achieve this, the solution adds new data fields (Table 2) to the two commands that occur at the start of all four protocol sequences: Select() and GetProcessingOptions(). Fig. 1 shows how the POS authentication functionality is incorporated into the Visa protocol sequence. Further details can be found in Section 5.2.

**Use of Elliptic Curve Cryptography.** The EMV protocol has a restricted message size of 256 bytes. This does not cause problems in the existing RSA authentication of the card by the POS terminal, as the card can pass several 256-byte messages in response to a single POS terminal message. However this does cause a problem in the proposed POS authentication solution as the POS terminal must pass all of authentication information in a single 256-byte message. The solution for this problem is to use Elliptic Curve Cryptography (ECC), which provides increased cryptographic strength over the existing RSA scheme [4][16] whilst allowing the authentication information to be passed in a single message. The ECC curve and algorithms used in the solution are as per EMV’s 2007 proposal document regarding the use of ECC [12].

The rest of the paper is organised as follows. Section 2 outlines our proposed solution in detail; it illustrates how POS authentication and ECC (which are the key building blocks of our solution) can be integrated into the EMV protocol. Section 3 provides an overview a fully implemented prototype to demonstrate the feasibility of the solution. Test results from the prototype are given in Section 4. Section 5 discusses the challenges involved in implementing the POS terminal authentication protocol. Each
challenge is followed by an explanation on how our proposed solution addresses the issues. Finally, Section 6 concludes our paper, and Section 7 outlines potential improvement and other issues that still need to be addressed in our future work.

2 Outline of Proposed Solution

The Chip & PIN technology on EMV payment cards have been proven relatively secure against card fraud since the introduction of Dynamic Data Authentication (DDA) cards in 2009. Contactless payment technology adds a wireless interface to EMV cards, which allows it to be accessed whilst it is still the cardholder’s pocket, even without their knowledge.

In the current EMV payment system the cards must prove to POS terminals that they are genuine, the reverse is not true. NFC enabled devices such as smart phones are becoming more common, these can mimic POS terminal’s behaviour, the lack of terminal authentication could lead to cases where rogue terminals might carry out skimming and eavesdropping attacks, even leading to fraudulent transactions.

Our solution aims to address this imbalance by making it necessary for any NFC enabled device attempting to access the card to authenticate itself. In other words, our solution will prevent the card from revealing sensitive information, unless the NFC reader can prove it is a genuine POS terminal issued by a bank.

2.1 Transaction Protocol

There are four variations of the EMV contactless transaction protocol sequence, one each for Visa, MasterCard, American Express and JCB cards. The prototype implements Visa’s Fast Dynamic Data Authentication (fDDA) contactless transaction protocol sequence (Fig. 1). Visa fDDA was selected as it contains the least number of commands of any of the four protocol sequences, and thereby it provides the most stringent constraints regarding which commands can be modified to incorporate the POS authentication functionality.

In Fig. 1, the new elements of the protocol sequence required for POS authentication are marked in blue. POS authentication uses the Select() command (Fig. 1 point 6.0) and the GetProcessingOptions() command (Fig. 1 point 8.0), since these command are common to all four contactless protocols. Select() and GetProcessingOptions() are also the first two commands in each of the protocol sequences which allow the protocol sequence to be halted before any sensitive data is divulged by the card.

The request for POS authentication has been added to the PDOL returned by the Select() command (Fig. 1 point 6.3). Details of the new structure of the PDOL are given in Section 2.2. The PDOL was chosen as the trigger for POS authentication as it is currently the way that the card requests information from the POS terminal.
The nonce is also contained in the response to the Select() command (Fig. 1 point 6.3). The nonce is an 8-byte unpredictable number which the POS terminal must sign with its private key to produce the ECDSA authentication signature (Fig. 3 - ECDSA Signed Nonce). A nonce is used to ensure that the ECDSA signature cannot be recorded by an attacker and replayed to gain access to the card.
The POS terminal generates the ECDSA authentication signature (Fig. 1 point 7.0), which is then returned in the message data contained in the GetProcessingOptions() command (Fig. 1 point 8.0). The message also contains the Elliptic Curve Qu-Vanstone (ECQV) implicit certificates and other data required by the card to validate the ECDSA signature.

If POS authentication fails, the card returns “Try Another Interface” (Fig. 1 point 8.2) which will cause the POS terminal to request a Chip & PIN contact transaction. This has the advantage that it ensures that the transaction is not lost if the POS terminal is not compatible with POS authentication. POS terminals which have not been updated to include POS authentication will follow the existing EMV protocol and continue the transaction in Chip & PIN mode [10].

### 2.2 Processing options Data Object List (PDOL)

The PDOL is a list of data fields that the card requests from the POS terminal. For POS authentication, the PDOL must contain all of the standard transaction fields (Table 1). In addition, it must also contain the new fields required for POS authentication (Table 2).

<table>
<thead>
<tr>
<th>TAG</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9F06</td>
<td>4 bytes</td>
<td>Terminal Transaction Qualifiers (Visa specific; this tag will need be changed for each of the different issuer card types)</td>
</tr>
<tr>
<td>9F02</td>
<td>6 bytes</td>
<td>Transaction amount</td>
</tr>
<tr>
<td>9F03</td>
<td>6 bytes</td>
<td>Amount other (used for cashback always zero)</td>
</tr>
<tr>
<td>9F1A</td>
<td>2 bytes</td>
<td>Terminal country code</td>
</tr>
<tr>
<td>95</td>
<td>5 bytes</td>
<td>Terminal verification results (always zero at this stage)</td>
</tr>
<tr>
<td>5F2A</td>
<td>2 bytes</td>
<td>Transaction currency code</td>
</tr>
<tr>
<td>9A</td>
<td>3 bytes</td>
<td>Transaction date</td>
</tr>
<tr>
<td>9C</td>
<td>1 bytes</td>
<td>Transaction type (always 00 for purchase)</td>
</tr>
<tr>
<td>9F37</td>
<td>4 bytes</td>
<td>POS terminal nonce</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAG</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9F81</td>
<td>64 bytes</td>
<td>ECDSA signed nonce (see Fig. 3)</td>
</tr>
<tr>
<td>9F82</td>
<td>64 bytes</td>
<td>Acquirer implicit certificate (see Fig. 3 - Aic)</td>
</tr>
<tr>
<td>9F83</td>
<td>9 bytes</td>
<td>Acquirer ID data (see Table 3)</td>
</tr>
<tr>
<td>9F84</td>
<td>64 bytes</td>
<td>Terminal implicit certificate (see Fig. 3 - Tic)</td>
</tr>
<tr>
<td>9F85</td>
<td>9 bytes</td>
<td>Terminal ID data (Table 3)</td>
</tr>
</tbody>
</table>
2.3 Acquirer ID (Aid) and Terminal ID (Tid) information

The Aid and Tid contain the plaintext data which is used to calculate the ECQV Acquirer implicit certificate (Aic) and Terminal implicit certificate (Tic). Details of the data contained in Aid and Tid are given in Table 3.

**Table 3. Acquirer and Terminal ID information**

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID number</td>
<td>4 bytes</td>
<td>Identity of the acquirer (Fig. 3 – Aid)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Identity of the terminal (Fig. 3 – Tid)</td>
</tr>
<tr>
<td>Expiry Date</td>
<td>2 bytes</td>
<td>MMYY</td>
</tr>
<tr>
<td>Serial Number</td>
<td>3 bytes</td>
<td>Version number of implicit certificate</td>
</tr>
<tr>
<td>Total</td>
<td>9 bytes</td>
<td></td>
</tr>
</tbody>
</table>

The format of the Aid and Tid data has been designed to follow the issuer information contained in the current EMV RSA public key certificates [11] (i.e. 4-byte issuer ID, 2-byte expiry date, and 3-byte serial number).

2.4 Controlling access to the card’s data and functionality

The cards will require a state machine similar to that currently implemented by MasterCard (Fig. 2).

The MasterCard state machine is relevant to the proposed solution in that the two major control points used there are the Select() and GetProcessingOptions() commands.

When the card is “IDLE”, Select() is the only command that can be called. Successful completion of Select() places the card into the “SELECTED” state.

Being in the “SELECTED” state allows access to GetProcessingOptions() which puts the card into the “INITIATED” state. In turn, this state gives full access to the card.

In order to make our proposed solution work, there is a need to alter the existing state machine,
whereby the `ReadRecord()` command is moved to after the “INITIATED” state, which prevents the card’s sensitive information from being read prior to POS authentication being completed.

### 2.5 Elliptic Curve Cryptography (ECC)

A 512-bit ECC scheme has been selected for the proposed solution as it provides a higher cryptographic bit-strength than EMV’s current 1984-bit RSA scheme. The shorter ECC scheme also allows the extra data required for POS authentication to fit into the `GetProcessingOptions()` PDOL message.

The solution utilises two elliptic curve schemes, ECQV and ECDSA, which perform different tasks in the implementation. The CA public key stored on the card and the two ECQV implicit certificates supplied by the POS terminal form a three-tier Public Key Infrastructure (PKI), which is used to verify the ECDSA digital signature generated by the POS terminal (Fig. 3). Brown et al., 2011 [3] describes a scheme for ECQV-certified ECDSA such as the one implemented in the prototype.

With ECQV implicit certificates, the CA public key is used to generate a public key from the implicit certificate, which in turn is used to generate the public key from the implicit certificate on the subsequent tier of the PKI [2].

### 2.6 Elliptic Curve POS Authentication Process

![Diagram](image_url)

**Fig. 3.** POS Authentication by Card
The POS terminal authenticates itself by creating an ECDSA signature of the nonce produced by the card (see Fig. 3) using its private key (Tsk). The card validates the ECDSA signature using the CA public key (CApk) and the two ECQV implicit certificates: the Acquirer (Aic) and Terminal (Tic). The card generates the Acquirer public key (Apk) and Terminal public key (Tpk) from the ECQV implicit certificates (Aic) and (Tic). The card authenticates ECDSA signature using the Acquirer public key (Apk) and Terminal public key (Tpk) and the card’s copy of the nonce produced by the card.

The solution depends on the POS terminal signing a nonce generated by the card; the nonce ensures that the ECDSA signature is fresh each time the card requests authentication. The three-tier Public Key Infrastructure (PKI) links the ECDSA signature produced by the POS to the CA public key (CApk), which proves that the Terminal’s private key was issued by an authorised Acquirer bank.

The POS terminals keys will be stored on the Secure Access Module (SAM), the SAM protects the keys from being read by brute force. The SAM also allows additional cryptographic functionality to be added to the POS without requiring an upgrade. Distributing the keys on the SAM avoids transmitting the keys to the POS terminal and thereby risking interception.

2.7 Elliptic Curve Generation of POS Terminal Keys

To perform the POS authentication process detailed in Fig. 1, the POS terminal must have two ECQV implicit certificates: the Acquirer implicit certificate (Aic) and the Terminal implicit certificate (Tic), as well as a Terminal private key (Tsk). These are generated in the process detailed in Fig. 4.

Fig. 4 shows that the Acquirer implicit certificate (Aic) and Acquirer private key (Ask) are generated by the Certificate Authority based on the data supplied by the
Acquiring bank. The Acquiring bank generates the Terminal implicit certificate (Tic) and Terminal private key (Tsk) using the Acquirer keys (Aic) and (Ask).

The Certificate Authority, Acquirer and Terminal keys form a three-tier Public Key Infrastructure (PKI) that enables the card to validate the ECDSA digital signature produced by the POS terminal based on the Certificate Authority public key (CApk) stored on the card.

In the current EMV PKI, there is a CApk for each of the four card issuers (Visa, MasterCard, American Express and JCB). The POS terminal will have to store a set of keys (CApk, Aic, Tic and Tsk) for each card issuer that the POS terminal wants to accept.

2.8 Elliptic Curve Card authentication

Based on the current EMV specification, the card authenticates itself by producing an RSA signature of the transaction data, referred to as the Signed Dynamic Application Data (SDAD) (Fig. 1 – points 8.4 and 8.9). The POS terminal validates the SDAD RSA signature and thereby the card (Fig. 1 – point 10). In the proposed solution, the SDAD has been altered to be an ECDSA signature rather than RSA. This alteration means that all of the cryptography used in the proposed solution is based on elliptic curve.

3 Prototype implementation

The prototype implementation consists of a prototype POS terminal and a prototype payment device. The payment device is a mobile phone emulation of a Visa contactless card which incorporates the POS authentication functionality.

3.1 Prototype payment device

The payment device has been implemented as a card emulation on a Nexus S Android mobile phone. Implementation on Java Card would have been preferred, however this was not practical since Java Card does not support ECQV natively using the card’s cryptographic co-processor. The current version of Java Card 2.2.2 does support ECDSA and Elliptic Curve Diffie-Hellman (ECDH) natively [17], it is therefore assumed that EMV payment cards could support ECQV in the future if the demand from the banks was great enough.

The Nexus S Android mobile phone was selected for prototype development as it provides both contactless (NFC) communication and the functionality required to generate the ECQV public key certificates and verify the ECDSA signature.

Elliptic Curve Cryptography. The ECQV and ECDSA cryptography software for the Android mobile phone platform has been implemented from scratch as the Android SDK does not natively support ECQV implicit certificates. This required the implementation of the methods for elliptic curve point addition and multiplication.
To ensure compatibility with future implementations of EMV, the prototype uses the NIST elliptic curve P-256 defined by EMV in their ECC proposal document [12]. For consistency the payment device application also uses ECDSA to generate the SDAD signature, which the POS terminal uses to authenticate the card.

### 3.2 Prototype POS terminal

The prototype POS terminal is implemented on a PC with an ACR-122U contactless reader. The POS terminal implements the Visa fDDA protocol sequence outlined in Fig. 1 with some additional data fields (Table 2). The Visa fDDA protocol was chosen because it is the shortest (i.e. it has the most stringent requirements regarding size) of the contactless protocol sequences and thereby demonstrates that the prototype can be implemented in any of the other contactless protocol sequences (i.e. MasterCard, American Express or JCB).

The prototype POS terminal implements the Visa fDDA transaction protocol sequence as defined in the EMV Contactless Specification Book C-3 [16]. Should POS authentication fail, the prototype payment device (Android phone) will return a “Try Another Interface” as the transaction outcome (see Book C-3 [16] Section 5.2.2.2). This will cause the POS terminal to initiate Chip & PIN completion of the transaction.

**Elliptic Curve Cryptography.** The prototype POS terminal implements ECQV and ECDSA cryptographic functionality required to generate the ECDSA POS authentication signature and verify the SDAD ECDSA signature produced by the prototype payments device.

### 4 Prototype test results

We have carried out initial experiments of running the prototype solution using a Nexus S Android mobile phone as a card emulation. For now, we have to use this emulation method because the current Java Card 2.2.2 does not support ECQV. However, it is envisaged that the ECDSA signature verification process will be faster when ECQV support is available on payment cards’ cryptographic co-processor. The protocol sequence timing is given here to illustrate the feasibility of the solution; future work on improving the performance will be carried out, as outlined in Section 7.

#### 4.1 Prototype transaction timings

Table 4 shows the average time to complete each stage of the transaction protocol when the prototype implementation is run on a Nexus S Android mobile phone. Significantly, we see an average of 3.6 seconds for the phone to generate the ECQV public keys and validate the ECDSA signature.
### Table 4. Prototype Transaction Timings

<table>
<thead>
<tr>
<th>Protocol Sequence Activity</th>
<th>Avg (ms)</th>
<th>Stddev(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction initialisation</td>
<td>(Fig. 1 - 3.0 to 6.3)</td>
<td>198.8</td>
</tr>
<tr>
<td>ECDSA signature generation</td>
<td>(Fig. 1 – 7.0)</td>
<td>89.0</td>
</tr>
<tr>
<td>PDOL Generation</td>
<td>(Fig. 1 – 7.1)</td>
<td>29.7</td>
</tr>
<tr>
<td>Signature Verification + SDAD</td>
<td>(Fig. 1 – 8.1 to 8.9)</td>
<td>3622.7</td>
</tr>
<tr>
<td>Signature Verification</td>
<td>(Fig. 1 – 8.1 + 8.2)</td>
<td>2172.0</td>
</tr>
<tr>
<td>SDAD generation</td>
<td>(Fig. 1 – 8.3 to 8.9)</td>
<td>1450.7 (calculated)</td>
</tr>
<tr>
<td>Read Card Data</td>
<td>(Fig. 1 – 9.0 to 9.1)</td>
<td>237.5</td>
</tr>
<tr>
<td>ECC SDAD verification</td>
<td>(Fig. 1 – 10.0)</td>
<td>95.0</td>
</tr>
<tr>
<td><strong>Average Transaction Time</strong></td>
<td></td>
<td><strong>4265.9</strong></td>
</tr>
</tbody>
</table>

The prototype takes 2.172 seconds to verify the POS authentication ECDSA signature (Fig. 1, points 8.1 and 8.2) and 1.4507 seconds to generate the SDAD ECDSA signature (Fig. 1, points 8.3 to 8.9). Cryptography operations represent 85% of the total transaction time.

The total transaction time of 4.2659 seconds is slow when compared to a typical Visa contactless transaction which takes approximately 1 second. However, the prototype performs the cryptography in the Java application rather than on a dedicated cryptographic co-processor such as the one available on EMV cards. Future work will compare the time for the prototype to perform ECDSA generation and verification with the time taken by a Java Card with cryptographic co-processor. We will also examine the possibility of speeding up the ECDSA signature verification process, for example using the technique outlined by Antipa et al., 2005 [1], which claims to speed up the process by 40%.

#### 4.2 Prototype payment device log file

Verifying the POS authentication ECDSA signature requires a number of calculations to (i) derive the issuer public key from the EQCV implicit certificate; (ii) derive the terminal public key from the EQCV implicit certificate; (iii) calculate the ECDSA POS authentication signature; (iv) compare the calculated ECDSA signature with the POS authentication sent by the POS terminal.

An extract of the Android payment device log file has been included in Appendix 1 showing the step-by-step verification of the ECDSA signature. This information is included for the benefit of readers who are familiar with ECC and would like to see the detailed steps involved.
5 Challenges

The solution is designed to prevent attacks on contactless cards using devices with NFC readers, such as mobile phones. During the process of designing the solution, the following challenges and possible modes of attack on the solution were identified and solved.

5.1 Integration with the existing EMV infrastructure

Challenge. Globally there are 23.8 million EMV POS terminals and 1.6 billion EMV credit / debit cards [13]. Therefore any change to the EMV protocol has a major financial impact across many organisations. It is therefore essential that any change to EMV can be implemented as a gradual replacement program rather than an enforced step change. This requires that cards and POS terminals which implement the new functionality must be capable of running in parallel with existing cards and POS terminals for many years.

Solution. The POS authentication solution was designed to deal with the two scenarios without the loss of the transaction or the customer having to use a different card:
(i) an existing card making a payment at a POS terminal with the new functionality;
(ii) a card with the new functionality making a payment at an existing POS terminal.
Note: the other two potential scenarios (existing card – existing POS terminal and new card – new POS terminal) do not have any issue here.

In the first scenario, the new POS terminal will perform the existing EMV contactless protocol which is appropriate for the type of the card (Visa, MasterCard, American Express or JCB).

In the second scenario the card will request POS authentication. It does this by requesting specific data fields in the PDOL as described in Section 2.2. An existing POS terminal will not be able to supply the fields required (Table 2) and will therefore exit the transaction with transaction outcome of “Try Another Interface” [10]. In all of the contactless protocols, the outcome of “Try Another Interface” causes the POS terminal to request that the transaction is completed in Chip & PIN mode. This allows the transaction to continue with minimum interruption.

5.2 Integration with existing EMV contactless protocol sequence

Challenge. As stated in Section 5.1 the cost of any change to EMV is very large. The POS authentication process must therefore be incorporated into the existing contactless protocols without changing the command sequence.

Solution. The solution is integrated into the Select() and GetProcessingOptions() commands which are the first two commands in all four contactless protocol sequences. In the Select() command, the card requests the fields required for POS authentication (Table 2). In GetProcessingOptions(), the POS terminal returns the ECDSA signature that authenticates it as genuine.
The solution takes advantage of EMV’s existing Data Object List functionality, which allows the card to request a flexible list of data fields from the POS terminal. In other words, we are using mechanisms that are already there, without the necessity of changing the protocol (Table 3).

To perform POS authentication, it is necessary to return a digital signature and two key certificates in a single 256-byte GetProcessingOptions() message. This is not possible under the current EMV scheme (1984-bit RSA). The solution implements a 512-bit ECC scheme which is cryptographically stronger than the current RSA scheme. It allows two ECQV implicit certificates and an ECDSA signature to form a POS authentication to be sent in a single GetProcessingOptions() message.

5.3 Revocation of POS terminal keys

Challenge. The ability to revoke POS terminals keys is an essential part of the design. This is because the cards have no way of communicating with the outside world apart from through a connection with a terminal. This creates a situation where a single compromised POS terminal could result in every EMV card being potentially compromised.

Consider the following two scenarios: (i) a POS terminal’s private key was compromised and used to generate ECDSA signatures; (ii) a genuine POS terminal is stolen or misused to generate ECDSA signatures. In both cases, the ECDSA signatures produced will appear to be genuine to every EMV-compliant contactless card, since the card has no means of directly receiving and storing information about revoked POS terminal keys.

Informing every EMV card of the revoked keys is impractical, as the bank’s backend servers can only communicate with the card when it is connected to a POS terminal or ATM. In addition, there is insufficient storage on the cards to store the revoked keys.

The keys must therefore be revoked at the POS terminal. However it is too late for the acquiring bank to send a message to tell the POS to revoke its key once the POS key is compromised or the POS has been stolen.

Solution. The proposed solution forces the POS terminal to regularly request a “stay alive” authorisation message from the acquiring bank’s backend servers. The POS terminal will only be allowed to issue a limited number of ECDSA signatures (e.g. 50) before it must request another “stay alive” authorisation. Limiting the number of ECDSA signature produced by a stolen / compromised POS does not entirely prevent contactless attacks but it does prevent large scale attacks, as these would produce unusually high numbers of POS authorisation requests that could be detected by the bank and shut down accordingly.

The “stay alive” requests would be made in between contactless transactions when the POS terminal was inactive thereby not impacting on the speed of a contactless transaction.
To prevent the private keys being read directly from the POS terminal’s storage, the proposed solution recommends the storage of the private keys on the SAM which is the current method of providing safe key storage for EMV POS terminals (see Section 5.4 below).

5.4 Safe Storage of POS terminal keys

Challenge. Given that there is monetary impact associated with the loss of POS terminal keys, it is important to protect them. There are 23.8 million EMV POS terminals in circulation worldwide [13]; you can even buy POS terminals on eBay. This gives criminals easy access to POS terminals from which they could attempt to extract the POS terminal keys.

Solution. The POS keys must therefore be stored in secure storage. Modern POS terminals already have secure storage for their cryptographic keys, in the form of the Secure Access Module (SAM). The proposed solution will make use of SAM to store the keys required for POS authentication.

6 Conclusion

Traditional Chip & PIN transactions have required the cardholder to explicitly authorise access to their card by taking it out of their wallet and putting it into the Chip & PIN POS terminal. Contactless payment changes this. The card can now be accessed without the cardholder’s knowledge or consent.

The proposed POS authentication solution presented in this paper will protect contactless cards against unauthorised access. This in turn prevents a range of attacks including skimming [5], eavesdropping [7], fraudulent transactions [7], and malicious access to secure functionality [6] by off-the-shelf NFC readers and NFC mobile phones. The authors acknowledge that the solution does not address man-in-the-middle relay attacks [9]; there is no real answer to this at the moment, but research into distance bounding may provide a solution [8].

The design of the proposed solution particularly emphasises issues associated with introducing new functionality into the globally deployed EMV payment system. New cards and new POS terminals are interoperable with existing cards and existing POS terminals, thus allowing new and existing cards and terminals to run in parallel. This is shown by our prototype implementation, which was implemented without altering the EMV transaction protocol sequence.

The prototype gives a working example of how we can combine ECQV implicit certificates with ECDSA signatures to create the PKI that supports POS authentication. Furthermore, the 512-bit ECQV and ECDSA provide improved cryptographic strength over the current 1984-bit RSA used by EMV. The shorter keys of ECQV and ECDSA enable the new POS authentication data fields to be sent within a single 256-byte EMV message.

The “stay alive” key revocation technique described in this paper prevents stolen POS terminals or maliciously misused POS terminals from being used on large-scale
attacks on EMV cards. The proposed solution does not entirely prevent a rogue POS terminal from accessing a small number of cards. However the revocation technique will make such attack economically unattractive to the criminals.

7 Future work

Future work will focus on improving the performance of the ECDSA signature verification process. This will include carrying out comparative timing tests between the Android prototype and Java Card with a cryptographic co-processor. We will also investigate the technique outlined in [1], which claims to speed up the process by 40%, in order to improve performance.

Detailed analysis will be required regarding the number of elliptic curve point additions and multiplications needed in each ECDSA signature generation and ECDSA signature validation. This will give a greater understanding of any time difference observed between the Java Card and the Android prototype.

8 Acknowledgements

Dylan Clarke for reviewing the technical content of the paper.

References

10. EMV Co EMV Books C-1, C-2, C-3, C-4 and C-5 Contactless Specification for Payment Systems Version 2.3 (2013)

Appendix 1: POS authentication protocol sequence log

The following is an extract from the Android prototype payment device log file, showing the ECDSA verification process (Fig. 1 – 8.1).

** Select() Command **
Select(A0000000071010)
Response to Select()
Application ID = A0000000071010
Application Name = Visa
PDOL = 9F6604 9F0206 9F1A02 9505 5F2A02 9A03 9C01 9F3704 9F8140 9F8240 9F8309 9F8440 9F8509
Nonce = 0102030405060708 (Constant for testing)
** GetProcessingOptions() **
GetProcessingOptions(ED3A2080 00000000000400 0826 000000000
0826 131021 00 1AFCD98A
1CCCB7D1D46PA5418059527118E296E53FEC0BA0926ECD8753F3758BA81C0FDE
2C52FD23CF55D1FDFFE105A84B365A3A5CED08C84BF1EC8AA9D08A56E49D
FCA36ADCD29343171EA7A8C1F9393843213AF2704B8B522285C8DFE23A687B466
87E461BEE83A2629E0E4D5A95AF0B7E16E594D9CE46D084713DDBC641EAAE90
0102030412176666)

776803357640AC4FC4ADEP22379947C6D380E4313D2D2417D75DA26778E5920B
3C0F5438EACA7BF450OB27A3684DB71B714FC289FA34C526975ACA4D5630D6
0506D70712156666)
** Getting public key from certificate **
Issuer derived Public Key X: 97986087729825723673195818672536728
594270276851404712149676231740139011657373
Issuer derived Public Key Y: 95158591496548167422614644321862058
148177431389807297302562970080780223776567
** Getting public key from certificate **
Terminal derived Public Key X: 826531636244651841116166456501549
6247171791143557293805165847465994170340624
Terminal derived Public Key Y: 40090609784714070551056753046102138635066213841587075932051885461658493126430
** Verifying ECDSA **
Verifying sig: 1CCCB7D1D4EFA5418059527118E296E53FEC0BA0926EC8D753E3755BA81C0FDE2C52FD23CF55D1FDFFE105A84B365A34A5C6D08C84BF1EC8C
AA9D080A560649D
r at verify: 13026465238181355672314974214986372181274063321153999611778992039625001537502
s at verify: 20048393904234368758029244348630494325279790267974405052619129105903173788829
w (hex): 7CC758B14EB3986BF90E243B8C1D5CF63DF6E6C229E47F364ED229217188C0
(w is the inverse of s mod the order n)
Hash (hex): A344BF51D01AB187E205293800374119531F6BEA3ACEAEF9CF1FEF91A53230D1
u1 (hex): 6656DE7E5D9AE5DF9A0293F0B1533D14CBE5F7CD3292D985C0225F18C39445C2
u2 (hex): 0E099626D8D667E7F3E8D8E3BEA3C4730560CB25555ECA5D75BB1B8E80C21791456724EA46D895458546692604B8A2A47C60D73E19902CB81D3099C77D680
v: 13026465238181355672314974214986372181274063321153999611778992039625001537502
r: 13026465238181355672314974214986372181274063321153999611778992039625001537502
(v and r need to match)
ECDSA signature Verified: true

Appendix 2: Glossary

Acquirer - Refers to the bank that holds the destination bank account for the transaction, which is typically the bank that issued the POS terminal to the merchant. Also referred to as the “acquiring bank”.

AAC - Application Authentication Cryptogram - Response code from the card that indicates that the transaction has been declined.

AFL - Application File Locator – List of records on the card that contain the cryptographic keys required by the POS terminal to validate the transaction.

Aic - Acquirer implicit certificate – ECQV implicit certificate used to generate the acquirer public key during the POS authentication process.

Aid - Acquirer identification data - Data to uniquely identify the acquirer, which is encoded into the acquirer’s ECQV implicit certificate (Tic), consisting of the acquirer ID, expiry date and a sequence number.

Apk - Acquirer public key – Generated by the card from the Aic.

ARQC - Authorization Request Cryptogram - Response code from the card that indicates that the card wants to complete the transaction online.

Ask - Acquirer private key – Generated by the Certificate Authority and stored by the acquirer.

CA - Certificate Authority
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAind</td>
<td>Certificate Authority public key index – An EMV POS terminal supports multiple CA public keys. The card indicates the CA public key that it supports using the CA public key index.</td>
</tr>
<tr>
<td>CApk</td>
<td>Certificate Authority public key.</td>
</tr>
<tr>
<td>CASk</td>
<td>Certificate Authority private key.</td>
</tr>
<tr>
<td>ECC</td>
<td>Elliptic Curve Cryptography</td>
</tr>
<tr>
<td>ECDSA</td>
<td>Elliptic Curve Digital Signature Algorithm.</td>
</tr>
<tr>
<td>ECQV</td>
<td>Elliptic Curve Qu-Vanstone</td>
</tr>
<tr>
<td>Issuer</td>
<td>The bank that issued the card used in the transaction and holds the source bank account for the transaction.</td>
</tr>
<tr>
<td>Nonce</td>
<td>An unpredictable number which is used as the challenge sent by the card to the POS terminal. The POS terminal digitally signs the unpredictable number thereby preventing replay attacks.</td>
</tr>
<tr>
<td>PDOL</td>
<td>Processing options Data Object List – A flexible list of data fields requested by the card. The POS returns the requested data fields in the GetProcessingOptions() message.</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>POS</td>
<td>Point of Sale terminal</td>
</tr>
<tr>
<td>SAM</td>
<td>Secure Access Module – Tamper proof storage module used by POS terminals for cryptographic key storage.</td>
</tr>
<tr>
<td>SDAD</td>
<td>Signed Dynamic Authentication Data – In the current EMV system the card authorised the transaction data by signing the transaction data with its private key to produce the SDAD.</td>
</tr>
<tr>
<td>SFI</td>
<td>Short File Indicator – The storage on EMV cards is in blocks of 16 records, each block is referenced by a unique SFI.</td>
</tr>
<tr>
<td>TC</td>
<td>Transaction Certificate – Response code from the card that indicates that the transaction has been successful.</td>
</tr>
<tr>
<td>Tic</td>
<td>Terminal implicit certificate - ECQV implicit certificate used to generate the terminal public key during the POS authentication process.</td>
</tr>
<tr>
<td>Tid</td>
<td>Terminal identification data – Data to uniquely identify the terminal, which is encoded into the terminal’s ECQV implicit certificate (Tic), consisting of the terminal ID, expiry date and a sequence number.</td>
</tr>
<tr>
<td>Tpk</td>
<td>Terminal public key - Generated by the acquirer and stored on the POS terminal’s Secure Access Module.</td>
</tr>
<tr>
<td>Tsk</td>
<td>Terminal private key - Generated by the acquirer and stored on the POS terminal’s Secure Access Module.</td>
</tr>
<tr>
<td>UPN</td>
<td>Unpredictable Number – This is a nonce generated by the card and included in the SDAD signed transaction data. This is a separate from the nonce used in POS authentication.</td>
</tr>
</tbody>
</table>