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-1386 May 2013

TECHNICAL REPORT SERIES

No. CS-TR-1386 May, 2013

POS Terminal Authentication Protocol to Protect EMV Contactless Payment Cards

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

Abstract

The original EMV protocol was designed to operate in a situation where the card holder removes their card from their wallet and insert the card into a Point of Sale (POS) terminal. The protocol operates predominantly in plaintext which was not a problem because the attackers needed to tamper with the POS to gain access to the information on the card. The introduction of contactless EMV cards exposes the mainly plaintext EMV protocol to a wireless interface. This allows attackers to use an off-the-shelf NFC reader to access the card without the cardholders knowledge and potentially whilst the card is still in their wallet. Research has demonstrated that contactless EMV cards are vulnerable to various attacks carried out using off-the-shelf equipment which is both cheap and easy to obtain. The proposed solution addresses these issues by having the card request that any NFC reader, attempting to initiate communication, must authenticate itself as a genuine bank issued POS. The POS does this using a Bank issued private key to sign a nonce provided by the card.

Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

EMMS, M., ARIEF. B., HANNON, J., VAN MOORSEL, A.

POS Terminal Authentication Protocol to Protect EMV Contactless Payment Cards
[By] M. Emms, B. Arief, J. Hannon, A. van Moorsel

Newcastle upon Tyne: Newcastle University: Computing Science, 2013.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1386)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1386

Abstract

The original EMV protocol was designed to operate in a situation where the card holder removes their card from their wallet and insert the card into a Point of Sale (POS) terminal. The protocol operates predominantly in plaintext which was not a problem because the attackers needed to tamper with the POS to gain access to the information on the card. The introduction of contactless EMV cards exposes the mainly plaintext EMV protocol to a wireless interface. This allows attackers to use an off-the-shelf NFC reader to access the card without the cardholders knowledge and potentially whilst the card is still in their wallet. Research has demonstrated that contactless EMV cards are vulnerable to various attacks carried out using off-the-shelf equipment which is both cheap and easy to obtain. The proposed solution addresses these issues by having the card request that any NFC reader, attempting to initiate communication, must authenticate itself as a genuine bank issued POS. The POS does this using a Bank issued private key to sign a nonce provided by the card.

About the authors

Martin Emms is studying for a research PhD at Newcastle University's Centre for Cybercrime and Computer Security (CCCS). My research into potential vulnerabilities in the EMV payments system brought about by the introduction of Near Field Communications (NFC) payment technologies (i.e. NFC payment cards, mobile phone payments applications, NFC payment tags and NFC payment / top-up wrist bands). Supervised by Professor Aad van Moorsel with the School of Computing Science at Newcastle University. Martin has also been working with a local women's support centre in the North East of England to better understand the issues faced by survivors of domestic violence. The main focus of this research has been enabling survivors to access online / electronic domestic violence support services without the fear of being caught by their abuser. His role has been to design new applications that can help survivors access support services without leaving tell-tale electronic footprints.

Budi obtained his Bachelor of Computing Science with First Class Honours from Newcastle University in 1997. He had a one year placement (industrial training) during his undergraduate study with Mari Computer System Ltd. from 1995 to 1996. He went on to study for a PhD at Newcastle University with a scholarship from the School of Computing Science and an Overseas Research Studentship (ORS) from the British Council. He completed his PhD in July 2001 with a thesis entitled "A Framework for Supporting Automatic Simulation Generation from Design". He currently works as a Research Associate at the School of Computing Science. He had previously worked as a Research associate on the TrAmS, TRACKSS, Rodin and DIRC projects, as well as a Teaching Fellow between October 2008 and September 2010.

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

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 original EMV protocol was designed to operate in a situation where the card holder removes their card from their wallet and insert the card into a Point of Sale (POS) terminal. The protocol operates predominantly in plaintext which was not a problem because the attackers needed to tamper with the POS to gain access to the information on the card.
The introduction of contactless EMV cards exposes the mainly plaintext EMV protocol to a wireless interface. This allows attackers to use an off-the-shelf NFC reader to access the card without the cardholders knowledge and potentially whilst the card is still in their wallet. Research has demonstrated that contactless EMV cards are vulnerable to various attacks carried out using *off-the-shelf* equipment which is both cheap and easy to obtain.
The proposed solution addresses these issues by having the card request that any NFC reader, attempting to initiate communication, must authenticate itself as a genuine bank issued POS. The POS does this using a Bank issued private key to sign a nonce provided by the card.

Keywords *– Contactless card payment, Elliptic Curve Cryptography, Point of Sale Authentication, EMV, Payment Protocol*

Introduction

Researchers have demonstrated that NFC enabled mobile phones and *off-the-shelf* contactless readers can be used in skimming attacks [7][8], relay attacks [6][9], bogus transactions and bogus operations [10]. The underlying issue that allows all of these attacks is that the card will communicate with any *off-the-shelf* NFC reader. These issues can be addressed if the card can distinguish between a genuine POS terminals and *off-the-shelf* NFC readers.
In the current EMV transaction protocol the card authenticates itself to the POS by signing transaction data sent by the POS with its private key. The POS validates the card's signature using the 3 tier Public Key Infrastructure (PKI) depicted in Figure 1.
Our proposed solution will follow the same logic allow the POS to authenticate itself to the card. The POS will use its private key to sign a data packet containing a nonce generated by the card.
The POS terminal will require a number of private keys to accommodate the different Certificate Authority (CA) keys used by Visa, MasterCard, American Express and JCB (see Figure 2).

Design Considerations

Potential Issue	Resolution
If attackers can compromise the private key of a single POS they will have access to all contactless EMV cards and the authentication system is seriously compromised.	The solution uses keys with a one month lifespan check against the last transaction date on the card. Attackers would have to break a new POS key each month.POS keys will be distributed on the Secure Access Module (SAM) so the keys cannot be read by brute force or intercepted during transmission. The SAM is currently used to store the POS terminal's CA public key certificates and EMV applications.

Design Considerations (cont)

Potential Issue	Resolution
EMV currently uses RSA cryptography with a maximum key length of 248 bytes. In practice the maximum key length we have observed on cards currently in circulation in the UK is 144 bytes. The maximum APDU command message length is 256 bytes. POS authentication requires a signature and 2 public keys to be passed to the card, in RSA this would be too long for a single APDU. ISO 7816 allows longer messages to split over multiple APDUs, however EMV does not support this feature.	Elliptic Curve Cryptography (ECC) uses much shorter key lengths than RSA to provide the same or a greater level of security. The ECC key length suggested by EMV is 64 bytes [2]. Information published by the US National Security Agency suggest that ECC with 64 byte keys would be much stronger than RSA with 248 byte keys (the current maximum in EMV).
Any change to EMV affects 1.6 billon cards so any change to the protocol would have to be a phased approach with old cards and new cards in circulation at the same time.	The proposed change is restricted to the card requesting the POS authentication within an existing message. This allows new cards to request the new functionality whilst the old cards continue to operate using the current protocol.

Current EMV Public Key Infrastructure (PKI) Structure

The Public Key Infrastructure (PKI) implemented by EMV is described in Figure 1 below. It employs a 3 tier PKI with (i) Certificate Authority public key (P_{CA}) at the root (ii) an issuer public key (P_I) for the Bank that issued the card (iii) a unique public (P_{IC}) / private key (S_{IC}) pair for each card.

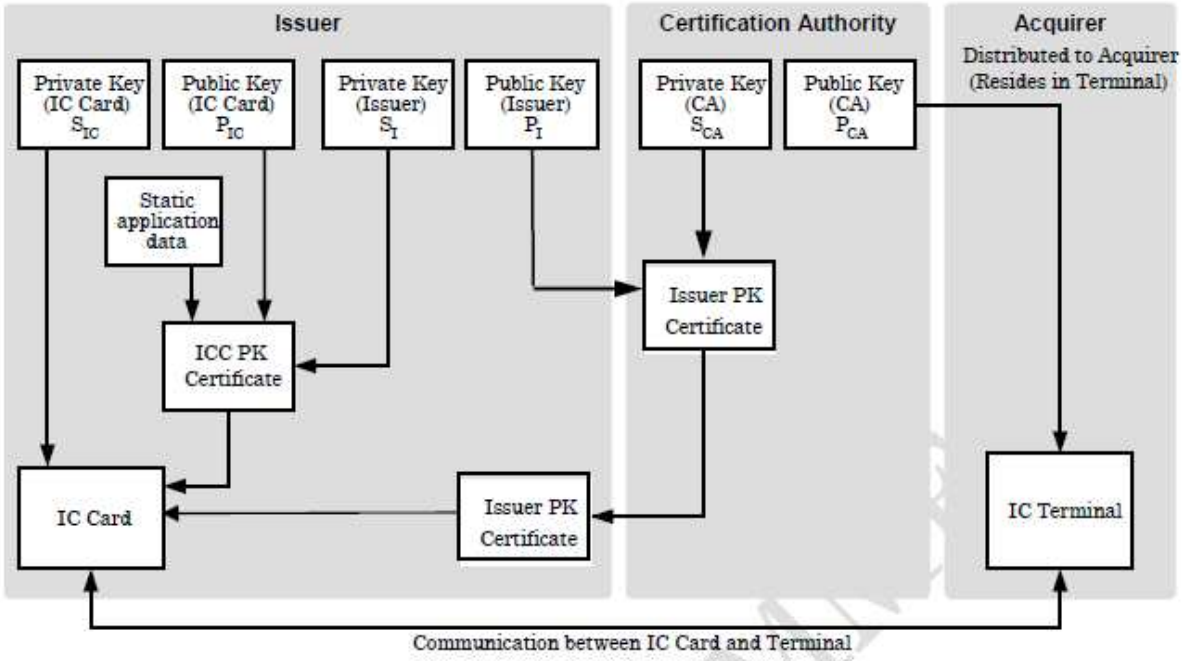

Figure 1 – EMV Public Key Infrastructure (source: [1] page 52)

The 3 tier PKI is used to allow the POS terminal to validate that the card is genuine. The card stores the Issuer public key (P_I) and card IC public (P_{IC}) / IC private key (S_{IC}) pair. The card also stores the CA public key index which indicates the Certificate Authority private key (S_{CA}) that was used to sign its Issuer public key (P_I).
The POS terminal has a copy of the CA public key (P_{CA}) which it uses to validates that the card's Issuer public key (P_I) and card IC public key (P_{IC}) were issued by the bank.
The card uses its IC private (S_{IC}) to sign the transaction data thereby proving that the card is genuine and preventing the transaction data from being altered, without detection, once it has been signed.
The POS uses the card IC public (P_{IC}) to validate that the signed transaction data was produced using a genuine bank issued IC private (S_{IC}).
This process ensures that (i) the card cannot be cloned (ii) *man-in-the-middle* attacks cannot alter the details of the transaction (iii) *replay attacks* cannot use the transaction signature to validate a second transaction of the same value on the same day as the transaction data included a nonce from the

POS which will not match in the second transaction even though the rest of the transaction data remains the same.

Proposed EMV PKI Structure

To support POS authentication the POS will need its own public / private key pair similar to that of the card described above in Figure 1. The key structure on the POS will mirror the key structure currently implemented for the card. Figure 2 shows the keys in current EMV protocol are shown in black, the new keys required for the proposed POS authentication are shown in red.
In the current implementation of EMV the POS stores multiple CA public keys (P_{CA}) for MasterCard, Visa, American Express etc. the POS selects which CA public key (P_{CA}) to use based on the CA public key index stored on the card. For each CA public key (P_{CA}) the POS will have an Issuer public key (P_I). For each Issuer public key (P_I) the POS will have 48 POS public / POS private key pairs, one key pair per month for 4 years.

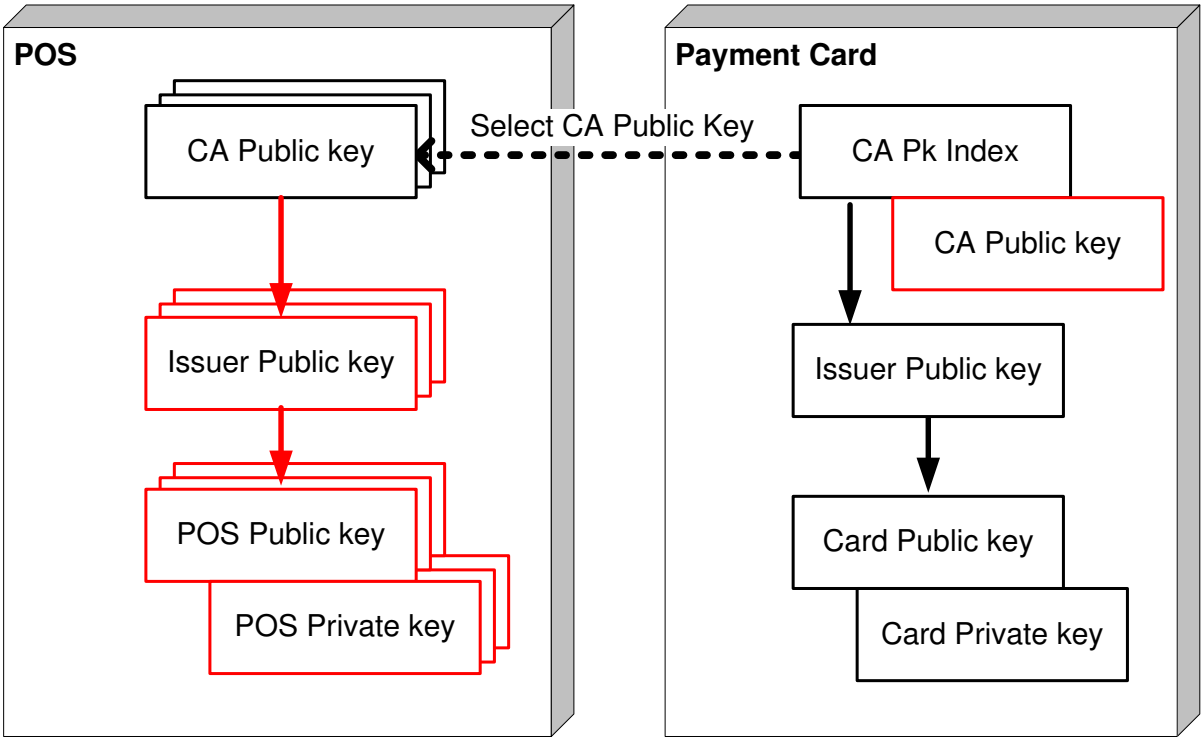

Figure 2 – Revised PKI Structure

The new keys will be stored on the POS terminal's Secure Access Module (SAM). The SAM contains tamper-resistant storage (currently used to store the CA public keys (P_{CA})), an application module and a cryptography module. The tamper resistant storage will be used to store the new keys this as will prevent the new POS private key directly read from the SAM.
The SAM is designed to perform EMV application specific functions and crypto-processing for the POS. Putting the ECC processing on the SAM would allow POS authentication to be rolled out with a reduced requirement to replace or update the POS terminals.
The card currently only stores the CA public key index, in the new key structure the card will also need to have its own copy of the CA public key that matches the CA public key index. The card will use its own copy of the CA public key (P_{CA}) to validate the POS Issuer public key (P_I) and in turn validate the POS public key (P_{IC}).
For each POS Issuer public (P_I) SAM will store 48 POS public key (P_{IC}) / POS private key (S_{IC}) pairs, each of which will have a lifespan of 1 month. The 1 month lifespan is designed to limit the impact of an attacker breaking one of the POS keys. The card will check the key's expiry date encoded within the key structure against the date of the last transaction of the card.
In the current EMV implementation the expiry dates of each of the keys in the PKI chain (P_{CA}, P_I and P_{IC}) are checked against the transaction date to ensure that they have not expired, if a key has expired the signature validation fails.
In the case of POS authentication the expiry dates of the P_{CA}, P_I and P_{IC} will be checked against the last transaction date of the card, if a key has expired POS authentication fails.

POS Authentication

The new POS Private key will be used to sign a data packet containing (i) a nonce produced by the card to eliminate playback of a recorded POS signature (ii) a nonce from the POS to ensure freshness of the data (iii) random padding[1] to make the packet up to 64 bytes.
The card nonce is returned by the card in response to a SELECT( ) command. The card will currently a generate an 8 byte nonce when it receives the GETCHALLENGE( ) command. It is envisaged that this functionality can be employed for POS authentication. The POS already generates a 4 byte nonce for the GETPROCESSINGOPTIONS( ) command.
The signed POS authentication data is returned to the card in the GETPROCESSINGOPTIONS( ) message. Also in the GETPROCESSINGOPTIONS( ) message are the keys required for the card to validate the signed data.
To validate the POS the card must perform the following steps

1. Use its own copy of the CA public key (P_{CA}) to validate /decrypt the POS Issuer public key (P_I)
2. The POS Issuer public key (P_I) is used to validate / decrypt the POS public key (P_{IC})
3. Once the POS public key (P_{IC}) is decrypted the expiry date encoded within its structure will be checked against the date of the last transaction stored on the card
4. The POS public key (P_{IC}) is used to validate the signed data packet containing the nonce
5. The card checks that the nonce matches the one it sent to the POS in the SELECT( ) command.

Authentication Failure

If any of the authentication steps 1-5 above fail the card will use the existing “Other Interface” failure mode this will cause the POS to ask for the card to be re-presented for a Chip & PIN transaction. This has the advantage that (i) it utilises the existing mechanism for contactless transaction failure (ii) transactions are not lost (iii) infrequently used cards (i.e. more than 2 months) will still be processed and will have their last transaction date reset.

[1] At the moment we have made the assumption that the data payload needs to be padded to the same length as the key (64 bytes). This is the case for the RSA cryptography currently in use in EMV. More work is required to study the ECC implementation detailed by EMV in [2] to understand the implementation details of ECC.

Messaging Structure

To minimise the changes to EMV protocol the proposed solution adds extra data fields to SELECT() and GETPROCESSINGOPTIONS() commands.

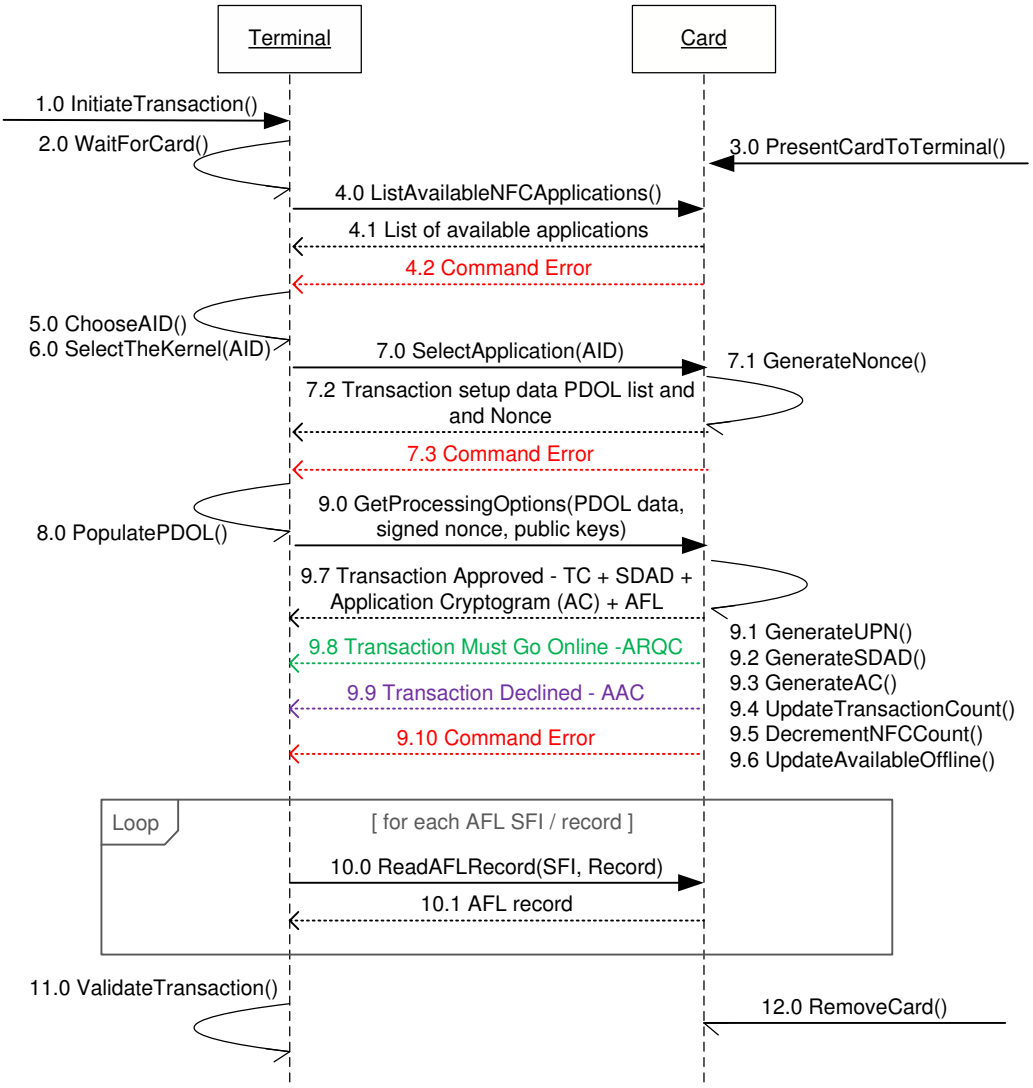

Figure 3 – Revised Transaction Sequence

In its response to the SELECT( ) command the card will return the 8 byte nonce and a new Processing options Data Object List (PDOL) containing the request for the POS authentication data.
The PDOL is used by the card to request the data fields it requires from the POS to initiate the transaction in the new protocol the card will request the authentication signature, the POS Issuer public key (P_I) and the POS public key (P_{IC}). The card will also request the fields currently requested in the PDOL which will include fields such as the transaction date, transaction amount, transaction currency, transaction type and POS capabilities.
The POS creates the authentication signature by concatenating a data packet from the card nonce and its own nonce, the POS uses the POS private key (S_{IC}) to sign the data packet.
The GETPROCESSINGOPTIONS( ) command sends the data requested by the card which will include the new data elements the authentication signature and public keys required to validate the signature.
GETPROCESSINGOPTIONS( ) has been chosen as mechanism to deliver the authentication signature as (i) the PDOL is flexible list via which each card can request the fields it requires, this allows the new cards request the new functionality whilst older cards continue to operate as per the existing spec (ii) GetProcessingOptions( ) is a key check-point in the current EMV state machine (Figure 4) allowing the new functionality to fit into the existing structure.
The new POS authentication message protocol sequence is designed allows a phased introduction of the new cards without affecting the operation of existing cards.

GETPROCESSINGOPTIONS( ) Message Structure

The message structure currently employed by Visa contactless transactions is as follows in Table 2

Field	Bytes	Comment
POS Capabilities	4	List of transactions that the POS supports
Transaction Amount	6	
POS Country Code	2	Country code of the POS SO 3166
POS Verification Results	5	Always zero at the start of a new transaction
Transaction Currency Code	2	Currency code of the ISO 4217
Transaction Date	3	YYMMDD
Transaction Type	1	Purchase '00' , Cash '01' , Refund '20'
POS nonce	4	Unpredictable number
Total Bytes	27	

Table 2: Transaction data for the Visa GETPROCESSINGOPTIONS( ) message

Field	Bytes	Comment
Card nonce	8	Nonce provided by the card in SELECT( ) response
POS nonce	4	Nonce provided by the POS
Padding[1]	56	Random padding (data packet = 64 bytes)
Total Bytes	64	

Table 3: Fields in the POS authentication signature

Field	Bytes	Comment
POS Authentication Signature	64	Table 3 data signed with the POS Private key (ECC)
POS Issuer public key header	14	Key header containing expiry date MMYY
POS Issuer public key	64	POS Issuer public key (ECC)
POS public key	14	Key header containing expiry date MMYY
POS public key	64	POS public key (ECC)
Total Bytes	220	

Table 4: New fields required for POS authentication

EMV uses APDU commands which have a maximum length of 256 bytes. The total length of the new GETPROCESSINGOPTIONS( ) is made up of the command (5 bytes) + transaction data (27 bytes) + new fields required for POS authentication (220 bytes), the total length of the new message 252 bytes.

State Machine

The EMV specification [4] describes the state machine for EMV cards. The state machine controls the order in which the EMV commands can be called in the transaction protocol. Figure 4 shows the current state machine which is detailed in the EMV specification [4] and the MasterCard specification [5]. The main control points are the SELECT( ) and the GETPROCESSINGOPTIONS( ) commands

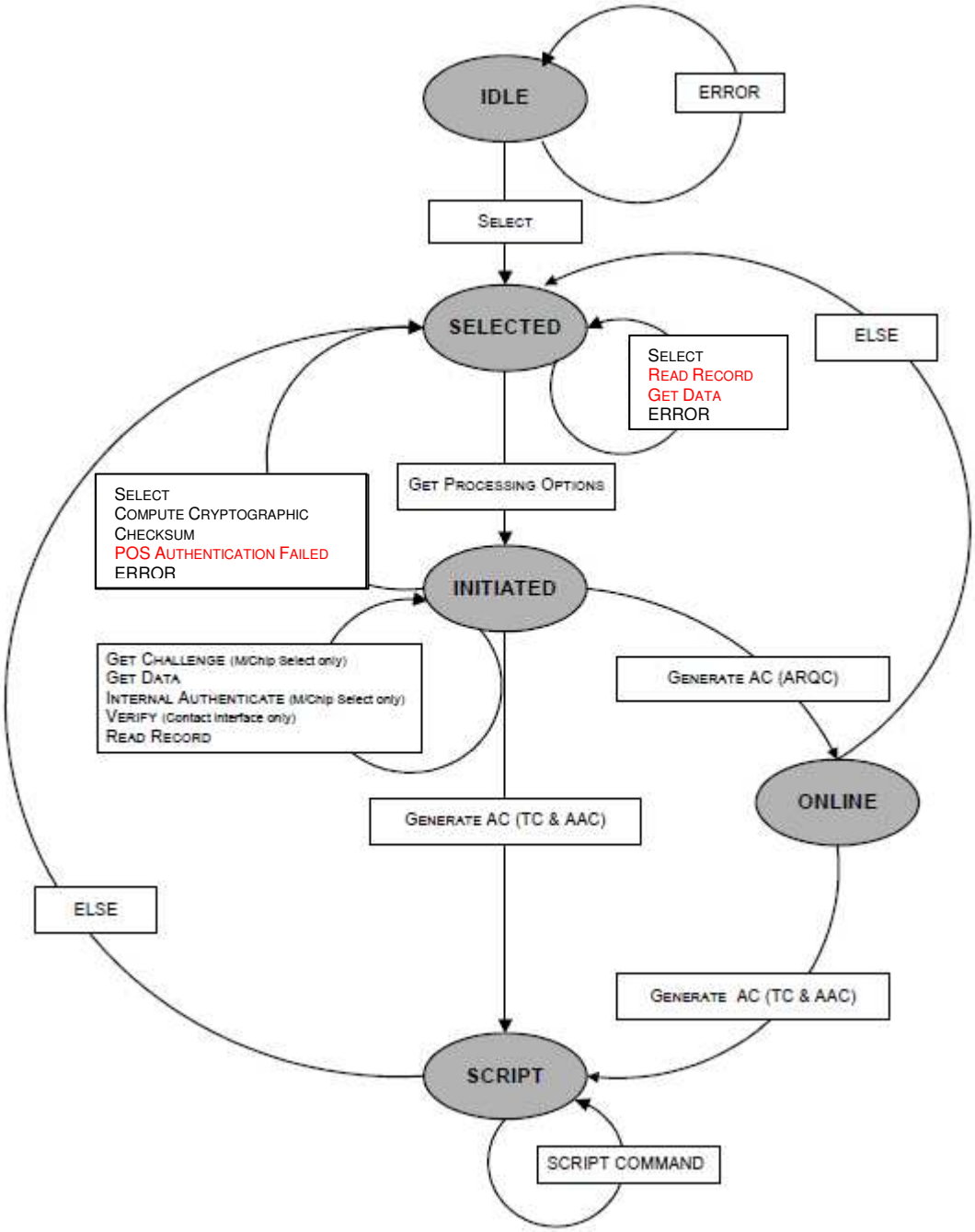

Figure 4 – State Machine (source: [5])

The proposed changes to the state machine are marked in red (i) when the card is in the SELECTED state the READRECORD( ) command and GETDATA( ) command the data fields returned by the card will be restricted to the data fields required by GETPROCESSINGOPTIONS( ) (e.g. the nonce, the PDOL and the CA public key index) (ii) if the card cannot validate the POS authentication signature the card will return a new error state "POS Authentication Failed" in response to GETPROCESSINGOPTIONS( ) (iii) if authentication was successful the card will enter the INITIATED state, once in this state READRECORD( ) and GETDATA( ) the card will return the full set of data fields.

Elliptic Curve Cryptography (ECC)

EMV cards currently in circulation in the UK use RSA keys 144 bytes in length and the maximum RSA key length in EMV is 248 bytes. It would not be possible to implement POS authentication with keys this length as the authentication process requires a signature and 2 keys to be transmitted to the card which is longer that the 256 bytes permitted by a single APDU command.
To enable POS authentication to operate in the 256 byte APDU limit the key length needs to be reduced to allow the signature and 2 keys to be transmitted in a single message, without reducing the strength of the underlying cryptography. The comparison between RSA and ECC bit strength on the US NSA website [2] shows that ECC can provide a greater level security using much shorter key lengths.
The draft EMV specification [3] contains specifications for ECC with 64 byte and 128 bytes keys, our proposed authentication scheme will use ECC with 64 byte keys. The use of ECC with 64 byte keys allows the signature and 2 keys, required by the authentication, to fit into the 256 bytes of the APDU (see tables 2, 3 and 4).

Conclusion

The solution is designed to integrate with the current implementation of EMV without having to implement a wholesale replacement of existing cards and also to minimise the upgrade requirements for the POS terminals. This is achieved by simply adding data elements to existing commands (using the EMVs flexible Data Object List structure) and by adding rules to existing check points in the state machine.
Authentication failure is handled through an existing mechanism which causes the POS terminal to request that the customer completes the transaction using the contact interface (Chip & PIN). This ensures that the sale is not lost due to an infrequently used card failing to authenticate the POS based on the date of its last trasnaction.
We propose the use of Elliptic Curve Cryptography (ECC) to reduce the key size from 144 bytes to 64 bytes whilst maintaining the security of the system [3]. The shorter key length 64 bytes allow the POS authentication keys and signature to be passed in a single APDU 256 byte message. ECC is compliant with one of the options [2] EMV has proposed for enhancing the cryptography system.

References

[1] EMV Co EMV Book 2 Security and Key Management Version 4.3 – (2011)
[2] EMV Co EMV Book 2 Security and Key Management Version 4.1z ECC With support for Elliptic Curve Cryptography – (2007)
[3] National Security Agency - The Case for Elliptic Curve Cryptography - http://www.nsa.gov/business/programs/elliptic_curve.shtml - Accessed May 2013 - (2009)
[4] EMV Co EMV Common Payment Application Specification Version 1.0 – (2005)
[5] MasterCard: PayPass - M/Chip Acquirer Implementation Requirements (2006)
[6] Lishoy, Hancke, Mayes, Markantonakis: Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones - http://eprint.iacr.org/2011/618 (2011)
[7] Emms, Van Moorsel: Practical Attack on Contactless Payment Cards - http://homepages.cs.ncl.ac.uk/p.m.dunphy/hwit/Emms.pdf (2010)
[8] Hancke: A Practical Relay Attack on ISO 14443 Proximity Cards - http://www.rfidblog.org.uk/hancke-rfidrelay.pdf (2005)
[9] Hancke: Practical Eavesdropping and Skimming Attacks on High-Frequency RFID Tokens - http://www.rfidblog.org.uk/Hancke-JoCSSpecialRFIDJune2010.pdf (2011)
[10] Emms, Arief, Little, Van Moorsel: Risks of Offline Verify PIN on Contactless Cards - http://fc13.ifca.ai/proc/9-2.pdf (2013)