Harvesting High Value Foreign Currency Transactions from EMV Contactless Credit Cards without the PIN

Martin Emms, Budi Arief, Leo Freitas, Joseph Hannon, Aad van Moorsel
School of Computing Science, Newcastle University
Newcastle upon Tyne NE1 7RU, United Kingdom
{martin.emms, budi.arief, leo.freitas, joseph.hannon, aad.vanmoorsel}@ncl.ac.uk

ABSTRACT
In this paper we present an attack, which allows fraudulent transactions to be collected from EMV contactless credit and debit cards without the knowledge of the cardholder. The attack exploits a previously unreported vulnerability in EMV protocol, which allows EMV contactless cards to approve unlimited value transactions without the cardholder's PIN when the transaction is carried out in a foreign currency. For example, we have found that Visa credit cards will approve foreign currency transactions for any amount up to €999,999.99 without the cardholder's PIN, this side-steps the £20 contactless transaction limit in the UK. This paper outlines our analysis methodology that identified the flaw in the EMV protocol, and presents a scenario in which fraudulent transaction details are transmitted over the Internet to a "rogue merchant" who then uses the transaction data to take money from the victim's account. In reality, the criminals would choose a value between €100 and €200, which is low enough to be within the victim's balance and not to raise suspicion, but high enough to make each attack worthwhile. The attack is novel in that it could be operated on a large scale with multiple attackers collecting fraudulent transactions for a central rogue merchant which can be located anywhere in the world where EMV payments are accepted.

Categories and Subject Descriptors
K.4.4 [**Computers and Society**]: Electronic Commerce – *Cybercash, digital cash, Payment Schemes, Security;* C.3 [**Special-Purpose and Application-Based Systems**]: – *Smartcards*

General Terms
Security

Keywords
Contactless cards, EMV, fraudulent transaction, foreign currency transaction limits, rogue merchant.

1. INTRODUCTION
Our research has identified a practical attack on EMV[1] contactless credit and debit cards, which allows large-scale "harvesting" of fraudulent payments from unsuspecting cardholders. The attack exploits six functional characteristics of EMV contactless credit and debit cards:

- Many Visa[2] credit cards will approve *unlimited value transactions in a foreign currency*; this allows the attack to maximise the money extracted from each credit / debit card.
- The *contactless interface* allows transactions to be extracted whilst the card is still in the cardholder's wallet.
- The cardholder's *PIN is not required for contactless transactions*; this allows the fraudulent transaction to be extracted from the card without any further interaction from the cardholder.
- Visa contactless cards will approve *transactions in offline mode*; this allows the attack to be performed without connecting to the card payment system, thereby avoiding any additional security checks by the bank.
- The *merchant details* are *not included* in the data cryptographically protected by the card; this allows the merchant details to be added later, making the attack more flexible and scalable.
- While the EMV protocol requires payment cards to authenticate themselves to the Point of Sale (POS) terminals, currently *there is no requirement for POS terminals to authenticate themselves.*

The main contribution of this paper is the identification of a newly discovered vulnerability of the EMV protocol centred on the card's handling of foreign currencies. This is made possible by a combination of the six functional characteristics described above. The introduction of EMV contactless cards has created a situation comparable to that described by Reason in his "Swiss cheese" model [10] where layers of protection can be compromised if holes on each layer line up to create an exploitable attack. In this case, the six characteristics line up in a way that defeats the safeguards put in place by EMV. Through this paper we also contribute two potential solutions which will block this vulnerability.

The ability to capture fraudulent transactions and store them for later transmission to a rogue merchant makes this attack different from previously described relay attacks [3][6] on EMV contactless cards. The relay attack depends upon very close synchronisation between two attackers; the first attacker has to be in contact with the victim's card whilst the second attacker makes a purchase at a POS terminal. This makes relay attacks difficult to operate on a large scale.

[1] EMV (Europay, MasterCard, and Visa) is a global standard to support interoperable card payment system between Visa, MasterCard, American Express and JCB.

[2] The attack presented in this paper has only been observed on contactless Visa cards. However our testing has showed that the underlying flaw also exists in MasterCard, but additional security measures implemented by MasterCard have prevented the manifestation of this attack.

Similar to the "Chip & PIN is broken" attack [9], our attack can potentially be operated on a large scale. "Chip & PIN is broken" allows attackers to buy goods from retailers, whereas the attack described in this paper is different in that it targets the money in the victim's bank account.

The very recent "Chip and Skim" attack [1] is similar to our attack in that it could be operated on a large scale and it extracts money from the victim's account. It would be interesting to explore the possibility of using our mobile phone contactless-transaction-collecting app as the "skimming" platform for the Chip and Skim attack.

The rest of the paper is organised as follows. Section 2 presents our methodology for finding the vulnerabilities, including the outline of the process, and the resulting formal abstract model, from which we derive our attack. Section 3 provides an overview of the attack, which is composed of two stages: collection of fraudulent transactions, and converting these transactions into money. Section 4 outlines existing safeguard to protect EMV transactions, while Section 5 looks into the EMV functionality exploited by the attack. Section 6 outlines the experimental software implementation to carry out the attack, including an Android app and a rogue merchant server. Section 7 presents some results from executing the attack, demonstrating the feasibility of such attack. In Section 8 we offer potential methods for preventing the attack and Section 9 concludes our paper.

2. METHODOLOGY

Our work focuses on the analysis of the EMV payments protocol and specifically the security impact of the introduction of contactless and mobile payments functionality into the protocol.

Analysis of the protocol is non-trivial due to the complexity of the EMV payment protocol specification. EMV is a global payment system, the protocol therefore has to incorporate competing (and sometimes conflicting) requirements from each of the credit card issuers (MasterCard, Visa, Amex, JCB, Diners, Discover, UnionPay) and from the financial regulators in each of the countries in which EMV operates. In addition, the introduction of contactless / mobile payments has significantly increased the complexity of the EMV specifications. The EMV specification for contact (Chip & PIN) credit / debit cards describes a single unified payment protocol sequence (kernel) for all card types. The specification for contactless / mobile payments contains seven protocol sequences (kernels), one for each card issuer. The complexity and page count has expanded, from four books comprising 765 pages for contact transactions, to fourteen books containing 2,392 pages for both contact and contactless.

To address this complexity, we have developed a systematic approach which combines formal and informal techniques. At the centre of our approach are *UML sequence diagrams*, an example of which can be seen in Figure 6, which we use as the informal but precise description of the protocol fragments. Each UML diagram is accompanied by a table listing the references in the EMV specification which were the diagram's information source. Creating the UML diagrams takes input from three main sources: (i) the EMV specification documents, (ii) feedback from insights gained by the developers coding the emulator, and (iii) feedback from insights gained by the designers constructing a formal model. Essential to our process is the systematic line-by-line documentation of the linkage between EMV specification, UML diagram, abstract formal model, emulator code and test cases.

The formal aspects of our approach are inspired by the Praxis methodology [2], tailored to our needs. It focuses on the construction and proof of an abstract model using the Z notation [13]. This abstract model is used to investigate the consistency of the requirements, expose descriptive errors, and ultimately be used to generate test cases for the emulator code. Ultimately, if our abstract formal model correctly characterises the EMV requirements, then our test cases will be both minimal and wide-reaching, given they come from the mathematical characterisation of the EMV requirements for NFC.

2.1 The Process

Figure 1 shows our analysis process. The rounded boxes are activity nodes within the process e.g. [A1]. The square boxes are object nodes e.g. [O1.0]: these are the data sources that drive the activities. Connecting edges, represented as black solid-arrows, indicate the default order in the flow of activities. The red dashed-arrows are connecting edges, which indicate feedback, creating an iterative process of refinement of the UML diagrams [O1.1], the abstract model [O2.1] and the emulator code [O4.1].

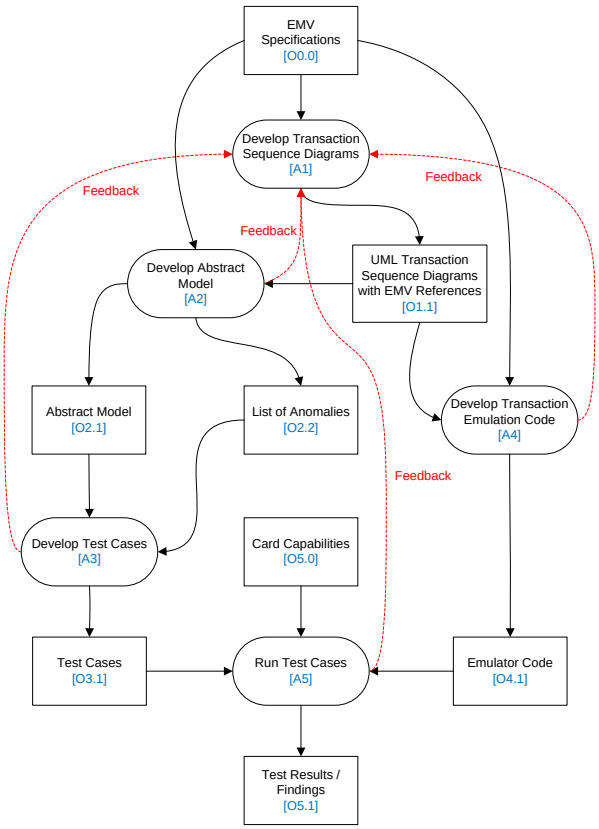

Figure 1. Protocol analysis process

At the centre of our approach is the construction [A1] of UML sequence diagrams [O1.1] with accompanying reference lists. Much of the process is about constructing these sequence diagrams as accurately as possible. To achieve this, we use a detailed analysis of the EMV requirements and a detailed working knowledge of the structure of the various specifications contributing to a single transaction. Moreover, we use feedback from the formal model construction [A2], the derivation of test cases [A3] and the coding [A4].

The EMV specifications [O0.0] are the originating source of all of the data in the process. Any data or assumption made in the

emulator code or in the abstract model should be traceable back to its origin (i.e. the book/section/page within the EMV specifications). The EMV specifications are structured so that the complete description for a single transaction protocol sequence is split across multiple sections and multiple books. The UML sequence diagrams [O1.1] collate these multiple sources into a single easy to follow description of the transaction sequence. These transaction sequence diagrams are the initial stage of the iterative process that we used to create the concrete software implementation of the emulator [O4.1].

At each stage of the process, if additional information is found about the working of EMV it is fed back into the UML transaction sequence diagrams [O1.1]. The feedback is essential to refine our understanding of the EMV specifications and document it. Each time the diagrams are updated, this drives the improvement of the emulator code [O4.1]. The completed emulator code is used in practical experiments [A5], running full or partial transaction protocol sequences against real bank cards.

2.2 UML Protocol Sequence Diagrams

The role of the UML protocol sequence diagrams is to collate information from multiple sources in the EMV specification, creating a single description of the payment protocol sequence (kernel).

There are eight payment protocol sequences (kernels) in the EMV specification, one for contact transactions and seven for contactless transactions. There is a single UML diagram for each of the eight kernels. Each diagram is accompanied by a table of references detailing the EMV specification sections from which the diagram was derived. Each reference details the EMV book, section number, page number and a section of text describing the functionality.

Table 1. Snippet of UML diagram references table

Descriptive Text	References
7.1 Transaction Setup Data including PDOL list If the Visa application is successfully selected the card will return the data that the terminal requires to set up the transaction including the PDOL list. The Processing Data Objects List (PDOL) is a list of data fields the card requires to complete the transaction, the terminal returns the populated PDOL data in the Get Processing Options command. Typically the data fields requested by the card will include the transaction amount, currency, date, country and POS terminal capabilities (TTQ).	EMV v2.2 Book C-3 2.4.1 Initiate Application Processing, page 12 EMV v4.3 Book 3 10.1 Initiate Application Processing, page 91 EMV v2.2 Book B 3.5 Outcome Processing (3.5.1.5 Other), page 33 EMV v4.3 Book 4 Annex A - Coding of Terminal Data Elements, page 115

Table 1 shows a snippet of the references table for Figure 6, which provides the details of one of the 26 steps in the Visa fDDA [5] protocol sequence (kernel 3).

It is these reference tables that provide the documented link between the UML diagrams and the EMV specification documents.

2.3 Protocol Emulator

The protocol emulator is a concrete software implementation of the EMV payments protocol. It is both an end product of the analysis process and the test-bed used to validate the findings of our analysis process; for instance the protocol emulator was used to confirm the existence of the foreign currency flaw in UK issued credit / debit cards.

To maintain the linkage between the protocol emulator code and the UML diagrams / EMV specification, we insert comments into the Java code. These comments contain the same descriptive text and references as per Table 1. In this way, each line of Java code can be traced back to its origin in the EMV specification and can also be understood as part of the overall protocol sequence thanks to the references to the UML diagrams.

2.4 Formal Abstract Model

In this work, we studied the EMV requirements documents [4][5] to produce a formal abstract model of its properties and functionalities, specifically for the Visa fDDA contactless transaction protocol (summarised in Figure 6). The motivation is to capture these requirements mathematically, enabling checking that the properties of interest hold (i.e. the requirements documents are consistent), and to produce test cases for our EMV emulator derived from formal proof of operational feasibility of each protocol stage (i.e. by proving the stage is feasible, we expose both abstract behaviours: normal and exceptional).

2.4.1 Implementation of the Abstract Model

Our abstract model uses the Z notation [13]. Proof obligations in Z are usually of three kinds: *well-formedness of models*, where partial functions are applied within their domains, and unique existential quantifiers are sound; *operational feasibility*, where specified operations have (implicitly defined) preconditions strong enough to establish (explicitly defined) post-conditions; and *data reification* via (usually forward) simulation, where the use of (concrete) data structure representations in operations closer to an implementation language are shown to respect the abstract representation and operations.

Our models have 49 type definitions, 61 Z schemas representing the NFC operations of the protocol, and 79 proofs in total, of which 49 are theorems representing properties of interest for the whole model [7]. Feasibility proofs are useful in deducing formal model-based test cases, as they characterise the complete space of behaviours for all operations of interest, including successful and all possible error cases, both determined by mathematical predicates representing disjoint behaviours of the protocol. That is, feasibility proofs characterise a set of disjoint predicates with (in EMV's case) non-overlapping conditions that when accumulated lead to true (e.g. pre-condition of an operation being $x < 0$ or $x > 0$ or $x = 0$). Thus, each disjunct represents a unique class of behaviours for the functionality being proved. Moreover, we also prove that these disjunct predicates amount to true, hence we guarantee all behaviours are accounted for.

The formal model follows the methodology advocated in [2], which enumerates requirements realised by each piece for formal specification. Thus, if all elements of the requirements are accounted for within the abstract mathematical model in a way that conveys the intended behaviour described in English, then proofs about the abstract model (or rather, proof failure) will lead (as our experiments show) into potential attacks and vulnerabilities discovered through proof investigation. Once validated by EMV experts, such formal model becomes a more

accurate representation of the EMV protocol than the EMV books [4][5].

These efforts correspond to the POS terminal side of Figure 6. The mechanisation of a formal concrete design, together with a proof of refinement indicate that these designs faithfully satisfy the abstract model linked to the requirements. Refinement proofs are perhaps the most costly aspect of a proof exercise, as it needs to establish that the implementation details do not breach any of the contractual requirements established by the abstract model. This concrete model can then serve to annotate the Java (or any other implementation) with formal specification for code-level functional correctness as done by tools such as VeriFast [11].

Furthermore, we derive a set of test cases from this abstract model that is the smallest with highest coverage possible. We also derive a systematic code-annotation technique, using the same principle to enumerate what aspect of the requirements each piece of code within the emulator is realised. These test cases represent a test-oracle based on requirements testing, rather than testing for any implementation issues. Together, the test cases and systematic code annotation are useful for capturing potential (major) errors. Errors from the concrete design are more likely to expose problems with implementation choices, and it is our aim in the future to annotate the emulator code with formal specification amenable to static analysis of the properties corresponding to the behaviour of the code.

2.4.2 Abstract Model for Foreign Currency Transaction Limits

EMV specifies the transaction currency as one of the data fields for mandatory inclusion in the Application Cryptogram (AC) [4]. This indicates the importance of the currency as it is one of the fields which is cryptographically protected against alteration. Nevertheless, the EMV books do not specify the process required when the terminal and the card have different currencies. This omission was discovered as part of the process to formulate the pre-conditions for the abstract model that currency exchanges were consistent. It was clear that the currency was one of the pre-conditions that should be included in the model, but we could not establish the correct process or outcome when the terminal currency was different from the card's currency.

The abstract model has identified the following pre-conditions relating to currency: (i) the native currency of the card; (ii) the native currency of the POS terminal; and (iii) the currency of the current transaction. For instance, when assembling the fDDA Processing Data Objects List (PDOL) for a Visa NFC transaction we get the following Z schema (from [7]):

$$
\begin{array}{|l}
\hline
FDDANFCVisaPDOL \\
NFCVisaPDOL!;\ Transaction!;\ cardCurrency? : CURRENCY \\
\hline
pdolAmount! = convertCU\ ((tcurrency!, cardCurrency?), amount!) \\
pdolCashback! = cbamount! \\
pdolUpno! = tunpredictableNumber! \\
pdolCountry! = tcountry! \\
pdolCurrency! = cardCurrency? \\
pdolDate! = tdate! \\
pdolTrType! = type! \\
\hline
\end{array}
$$

It creates the `NFCVisaPDOL!` with the adequate fields from both the card's and transaction's data. The PDOL amount, however, needs to be corrected for the card's target/preferred currency. For that we use a bijective function linking currencies and countries, as well as the agreed transaction currency (returned as `tcurrency!`), and the given `cardCurrency?` input for the given amount. This PDOL is then used to produce the AC and the Signed Dynamic Authentication Data (SDAD) for the validation of the transaction, by the bank and the POS terminal respectively.

We could satisfy all requirements when `cardCurrency?` is equal to `tCurrency!`; however we could not do the same when they are not equal. This prompted us to run foreign currency transaction experiments on real credit cards using the emulator, revealing the vulnerabilities leading to the attack.

3. OVERVIEW OF THE ATTACK

Figure 2 shows the key elements of the attack and how they interact with the EMV payment system.

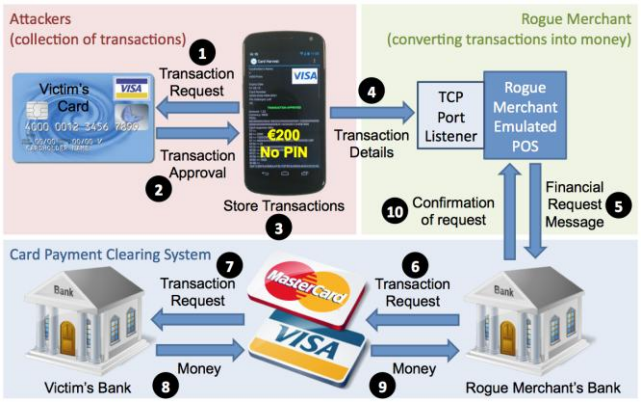

Figure 2. Transaction harvesting attack

The attack consists of two stages:

- *Attackers (collection of fraudulent transactions)*: attackers using Near Field Communication (NFC) enabled Android mobile phones can collect fraudulent transactions from unsuspecting cardholders. This can be done whilst the contactless card is still in the cardholder's pocket (see steps 1 to 3 of Figure 2).
- *Rogue merchant (converting transactions into money)*: a rogue merchant converts the collected transactions into money in their bank account by sending the transaction data to a bank (steps 4 to 5 of Figure 2).

Finally the transaction request enters the *Card payment clearing system* where the rogue merchant's bank acts innocently to transfer the transactions into the card payment system, which transfers the money from the victim's bank account into the rogue merchant's bank account (see steps 6 to 10 of Figure 2).

3.1 Collecting fraudulent transactions

Transactions are collected using a malicious app written for NFC-enabled Android mobile phones. The app automatically initiates and collects a transaction immediately upon detection of a contactless credit / debit card in the phone's NFC field. This process takes less than 500 milliseconds from card detection to transaction completion.

It is imagined that attackers will operate in a similar way to pickpockets, hiding their activity in crowded situations such as on public transport or in the crowd at an event. When a credit / debit card is detected, the app gives the attacker an audible signal through their headphones; a second audible signal is given when the transaction collection is complete. This will allow the attacker to operate without attracting too much attention.

3.1.1 Hardware

An Android mobile phone is chosen as the attack platform for the following reasons:

- Android mobile phones have a built-in NFC reader.
- An Android phone is an innocuous item for the attacker to carry in a crowded place; for example, it will not raise attention if the attacker is stopped by the police, since everyone carries mobile phones these days.
- The mobile phone platform provides portability, Internet connectivity and good battery life, making it a very capable attack platform.

3.1.2 The transaction collecting app

The attack starts when the NFC-enabled Android phone identifies a contactless credit / debit card which is vulnerable to this attack in the victim's wallet. The app sends a transaction request to the vulnerable card.

The app plays an audible alert to the attacker to signal that a vulnerable card has been found.

When the victim's card receives the transaction request message, it can approve or decline the transaction. If the card approves the transaction it generates the AC and the SDAD, this proves to the bank and POS terminal respectively that the card that approved the transaction was genuine (see Section 4.3 for more detail).

The cryptographic algorithms used to generate the AC and SDAD also ensure that the transaction details cannot be changed subsequent to the card authorising the transaction.

When the attack is complete the app plays a second audible alert.

3.1.3 Storage of approved transactions

The app was designed to operate in locations where an Internet connection is not always available, for example on underground public transport. Therefore the app will initially just store the transaction authorisation data returned by the victim's card. When a reliable Internet connection is available, the app will send the stored transaction data to the rogue merchant who will convert the transaction data into money.

The ability to capture fraudulent transactions offline and store them for later transmission is one of the novel features of this attack. This allows the attack to be operated on a large scale without the need for synchronisation.

Furthermore, storing the transactions minimises the time required to collect fraudulent transactions as the app does not have to wait for a connection. It also allows the attackers to operate in victim-rich crowded places that are normally without an Internet connection such as on subway trains, on buses and at large events.

3.2 Converting transaction data into money

The criminals would set up a rogue merchant account with an acquirer bank in one of the 76 countries that accept EMV payments. This rogue merchant will receive the fraudulent transactions collected by the attackers and convert them into money by sending the transaction data to the bank.

The rogue merchant consists of three elements:

- An Internet-based listening service, which will receive collected transaction data from attackers.
- A data format conversion process, which converts the fraudulent transactions collected by the attackers into the format required by the bank.
- A rogue Point of Sale (POS) terminal, which must imitate the actions of a legitimate POS terminal so that it does not raise the bank's suspicion. To achieve this, the rogue POS takes the previously converted data, adds the merchant data and sends that data to the bank using an Internet Protocol (IP) connection.

3.2.1 Internet-based listening service

The rogue merchant provides an Internet-based listening service on a pre-arranged IP address and port number, to receive the fraudulent transactions from the attackers. The transactions are initially stored to be processed later, once the merchant details have been added to the transaction and the connection to the acquirer bank is available.

3.2.2 Data format conversion process

Financial presentment request messages are used to transmit EMV credit / debit card transactions between the merchant (who captured the transaction) and the acquirer bank (who will process the transaction).

Merchant-related data such as merchant ID, terminal ID and the merchant's bank account details are added to the transaction to complete the data required by the EMV card clearing system. The fraudulent transaction is now ready for transmission to the acquirer bank.

The exact format of the message will differ slightly between different acquirer banks. However, there are a number of mandatory fields that are the same for every acquirer bank. Standard 70 [12] in the UK and ISO 8583 [8] in other EMV countries define the mandatory data fields which must appear in the financial presentment request message and the optional fields which may differ between the acquirer banks.

The software for our attack prototype implements a Standard 70 message format, complete with all of the mandatory fields and a number of optional fields (see Section 6).

3.2.3 Rogue POS terminal process

Once correctly formatted, the financial presentment request message is sent to the bank. The acquirer bank returns a financial presentment response message, to which the merchant responds with a financial presentment confirmation message that acknowledges receipt of the acquirer's response message.

The supported communication options for this message exchange are PSTN, X25 over ISDN, IP over ISDN, and IP over public networks (i.e. the Internet) for transmission of messages between the merchant and the acquirer bank. The software implementation presented in this paper uses IP over the Internet.

Our software implements data format conversion (Section 3.2.2) and implements the sending of the financial presentment request message over an IP connection protected by SSL/TLS encryption.

For obvious reasons we were not willing or able to check against a real bank. Of course, one approach to defeating the attack is to try to detect rogue POS behaviour at the bank, but it is not clear how well this can be done. A simple solution would be to have the payment card reject any contactless foreign currency transaction immediately, but is just not practical. As we will argue in Section 8, a more effective solution can be implemented by either forcing

foreign currency contactless transactions to be carried out in online mode only, or where that is not possible, to switch the transaction to "Chip & PIN".

4. EMV TRANSACTION SAFEGUARDS

In the UK, EMV credit / debit cards can perform two different transaction types: contactless "tap and go" transactions, and contact "Chip & PIN" transactions.

4.1 Contactless "tap and go" transactions

Contactless transactions are intended to be a quick and convenient replacement for small cash purchases. In a contactless payment, the credit / debit card is placed on the POS terminal's contactless reader for less than 1 second and the payment is approved.

There are two significant differences between a contactless transaction and a contact "Chip & PIN" transaction. First, the contact transaction requires the cardholder to enter their PIN, whereas the PIN is not required for contactless transactions. Second, contact transactions require the card to be removed from the wallet and inserted into the POS terminal, whilst contactless transactions is completed wirelessly by placing the card on the POS terminal, this can be done whilst the card is still in the wallet.

PIN entry provides one of the key safeguards in "Chip & PIN" transactions. The PIN ensures that only the cardholder, who knows the PIN, can use the card. Contactless transactions are not protected by PIN entry. EMV have therefore implemented the following safeguards to limit the potential loss from lost or stolen contactless cards:

- In the UK, each contactless transaction is limited to £20; any transaction above this value will require a Chip & PIN transaction.
- EMV cards are limited to five consecutive contactless transactions, after which the PIN must be entered in a "Chip & PIN" transaction.

These safeguards ensure that the maximum loss due to a lost or stolen contactless card is £100.

4.2 Contact "Chip & PIN" transactions

The majority of EMV card transactions are "Chip & PIN" transactions. "Chip & PIN" transactions allow purchases up to the balance of a debit card or the credit limit of a credit card.

"Chip & PIN" transactions are protected by the following safeguards. First, the cardholder must enter their PIN to authorise the transaction. This is used to ensure that the person making the payment is the authorised cardholder.

Second, if the value of the transaction is greater than the card's *offline* transaction limit, the card will request that the POS terminal makes an *online* connection to the bank to perform additional authorisation checks. The POS terminal must connect to the bank to provide the card with the *online* authorisation code (Authorisation Response Cryptogram (ARPC)). The bank will respond with the authorisation code only if the card has not been reported lost or stolen, and the account has sufficient funds to pay for the transaction. The card will only authorise the transaction if it receives a valid online authorisation code from the POS terminal.

4.3 Cryptographic protection of transactions

The EMV payment system utilises cryptography to ensure that (i) only genuine EMV credit / debit cards can authorise transactions (ii) the transaction details approved by the card cannot be altered.

4.3.1 Application Cryptogram (AC)

The AC contains a Message Authentication Code (MAC). The MAC utilises a symmetric algorithm, either Triple DES or AES, to encipher the transaction data fields detailed below:

- amount authorised (value of the purchase)
- amount other (cashback amount if required)
- terminal country code (UK - 0826, USA - 0840 etc.)
- terminal verification results (POS status code)
- transaction currency code (UK£ - 0826, US$ - 0840 etc.)
- transaction date
- transaction type (purchase - 00, cash - 01, refund - 20)
- POS terminal unpredictable number (prevents cloned cards)
- application interchange profile (card's security capabilities)
- application transaction counter (card's transaction counter)

The AC is sent to the bank as part of the Financial Presentment message (see Table 2). This allows the bank to verify that the transaction details supplied by the merchant are the same as the transaction approved by the EMV card.

4.3.2 Signed Dynamic Authentication Data (SDAD)

The SDAD is a RSA digital signature on a SHA1 hash of the transaction data. In the Visa fDDA protocol the transaction data included in the SDAD are:

- POS terminal unpredictable number
- amount authorised
- transaction currency code
- card unpredictable number
- card transaction qualifiers

The SDAD is used by the POS terminal to verify that the card is genuine in an offline transaction.

5. EMV FUNCTIONALITY EXPLOITED BY THE ATTACK

The attack circumvents the safeguards built into EMV credit / debit cards by exploiting some EMV functionality that has been made vulnerable due to the introduction of contactless payment interface. In particular, there are three features that are exploited in our attack scenario:

- Contactless foreign currency transactions. As described in Section 4.1, the safeguards built into EMV will limit the maximum value allowed for each contactless transaction to £20. Any amount over £20 will require the cardholder to enter their PIN, and any amount above the *offline* transaction limit (e.g. £100) will require the POS terminal to connect to the bank to perform additional checks before the transaction is approved. Our research has found that EMV credit and debit cards can be tricked into approving contactless

transactions of much higher value than £20, simply by requesting the transaction in a foreign currency. In our experiments, EMV cards have been found to approve contactless transactions up to €999,999.99 without requesting the PIN, and without requesting that the POS terminal goes online to perform additional checks. This sidesteps the usual safeguards employed by EMV payments system.

- Wireless interaction with card. This attack exploits the wireless interface on contactless cards to collect transaction authorisations whilst the card remains in cardholder's wallet. This means the cardholder remains unaware that they have been exploited until their card statement arrives, thereby allowing the attack to operate for longer and be more lucrative to the attackers.
- The merchant ID and terminal ID can be added later by the rogue merchant, as these data are not included in the AC generated by the card. The AC cryptographically ensures that the transaction data approved by the card is the same as that received by the issuing bank (see Section 4.3).

6. IMPLEMENTATION

To validate our research, we have implemented a number of software elements which demonstrate the viability and practicality of the attack. The software consists of three separate applications:

- An Android mobile phone app which captures transactions from the cards. Transactions are stored on the Android phone to be transmitted to the rogue merchant later.
- A rogue merchant Internet listening service which waits to receive the captured transactions from attackers using the Android mobile phone app.
- A rogue merchant bank communications module which packages the transactions into financial presentment request messages for transmission to the bank. This module handles all of the communication with the bank, which involves sending the financial presentment request messages and receiving acknowledgement messages.

6.1 Android transaction capture app

We have implemented the attack platform on an NFC enabled Android mobile phone as this would be an innocuous device for an attacker to carry around in a crowd.

6.1.1 Attack platform

For implementation and testing, we selected the Google Nexus 5 mobile phone. Implementing on a mobile phone platform limits the effective range to approximately 1 cm. However in testing the Nexus 5 was capable of extracting transactions from an EMV contactless card which was located in a leather wallet in the pocket of a pair of jeans worn by our "unsuspecting" test victim.

6.1.2 Android app operation

The attacker starts by pre-setting the amount and currency for all the transactions which will be captured from the victims cards. Figure 3 shows the attacker setting the amount to 999,999.00 and setting the currency to 0978 which is the code for Euros. In testing we have also obtained transaction approvals in US Dollars for $999,999.99 (currency code 0840).

The app is now ready and will automatically collect a transaction from every EMV contactless card that it detects, without any further interaction from the attacker. This will minimise the chance of the attacker being detected, as they are not constantly interacting with their phone.

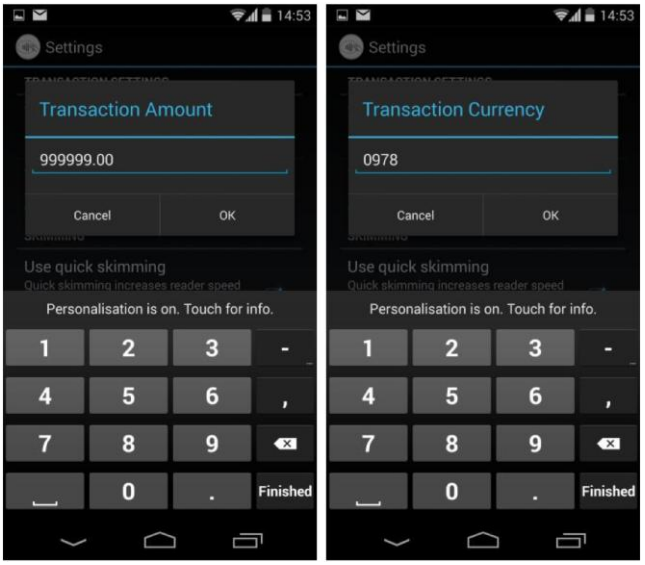

Figure 3. Capture transaction settings

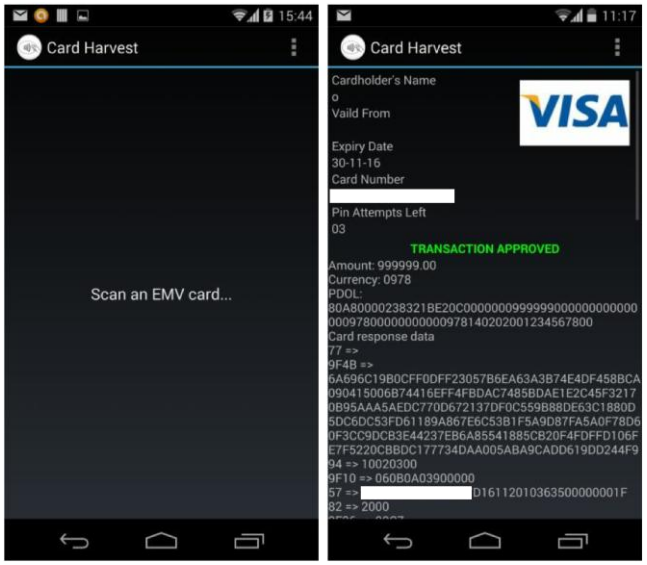

Figure 4. Capturing the transaction

In Figure 4 the screen on the left shows the app waiting to detect an EMV contactless card. The screen on the right shows the €999,999.99 transaction being captured from the card.

When the app detects an EMV contactless card, it sounds an audible alert in the attacker's headphones; a second alert is given once the transaction has been successfully collected. This takes less than 500 milliseconds. Once the transaction has been captured the app stores the transaction data for transmission to the rogue merchant later. As soon as the app has collected a transaction, it automatically returns to waiting to detect another EMV card; it is now ready to collect the next transaction.

Figure 5 shows the data fields as captured by the app, this includes all of the data and cryptographic authorisation codes required by the bank to accept the transaction as genuine.

The mobile app stores transaction data until it has an Internet connection, at which point the app transmits the data to the rogue merchant.

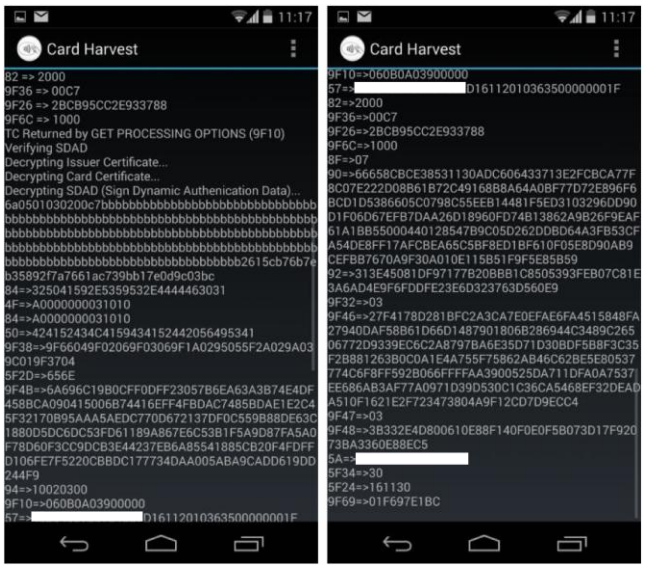

Figure 5. Captured transaction data

6.1.3 Transaction protocol

The code implements the Visa fDDA [5] contactless transaction protocol sequence (depicted as Figure 6) as this is an *offline* only contactless protocol. This allows the attack to be performed in less than 500 milliseconds and avoids additional validation by the bank.

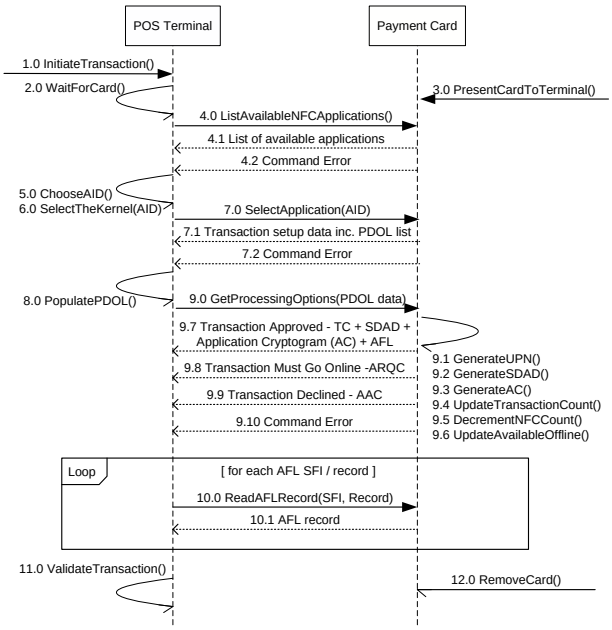

Figure 6. Visa fDDA transaction protocol sequence

6.1.4 Storing the transaction data

The transaction data is sent by the card in TAG / Length / Value (TLV) format. The Android application stores all of the data fields returned by the card for later transmission to the rogue merchant.

6.1.5 Transmission to the rogue merchant

Our software can collect and store multiple offline transactions, without a connection to the Internet. The stored transactions can then be transmitted once a suitable connection is available. The transaction details will include all of the data fields required by the bank. The Application Cryptogram (AC) and the clear text equivalent fields listed in Section 4.3 are arguably the most important, as together they are used by the bank to verify and thereby approve the transaction.

6.2 The rogue merchant application

The rogue merchant application consists of three processes:

- an Internet listening service to receive the transactions from the Android transaction capture app
- a data conversion module which converts the EMV data in TLV format into the ISO 8583 / Standard 70 format required by the bank
- a POS terminal emulation which sends the formatted data to the bank to collect the money from the fraudulent transactions

6.2.1 Internet based listening service

This is a simple Internet based service which listens to a pre-agreed IP address and port number. The Android transaction capture app (Section 6.1) connects to the pre-arranged IP address and port number to send all of the collected transactions to the rogue merchant. The listening service stores the transactions for later processing.

6.2.2 Data conversion process

The data conversion process accepts TLV data as captured from the EMV credit / debit card and converts it into ISO8583 / Standard 70 format required by the bank.

To request the money from the victim's account, the rogue merchant must send a financial presentment message (in ISO8583 or Standard 70 format) to the acquirer bank that holds their merchant account.

Table 2 shows the data fields required by the ISO 8583 financial presentment message and shows how the rogue merchant will complete the data fields from the data generated by the EMV card during transaction approval.

6.2.3 POS terminal emulation

Once the financial presentment request message has been generated, it is sent to the acquirer bank to complete the transaction and transfer the money from the victim's bank account into the rogue merchant's account.

In the UK, communications with the acquirer bank over a public IP network must be protected using Secure Sockets Layer/Transport Layer Security (SSL/TLS) or IPSec [12].

The use of standard encryption such as SSL/TLS and/or IPSec allows the rogue terminal to be implemented in Java on a PC platform; no specialist hardware is required.

Table 2. Financial presentment message data requirements

Item	Name	Description and mapping to EMV card data
1	bit map extended	List of fields included in the message
2	primary account number	**0x5A** – 16-digit card account number
3	processing code	Constant 00 for goods and purchases
4	amount, transaction	**0x9F02** – the transaction amount
5	amount, reconciliation	Transaction amount **0x9F02** converted into the currency to be applied to the victim's card, this value is calculated by the rogue POS terminal
7	date and time, transmission	Date and time the rogue POS transmits the transaction to the bank
9	conversion rate, reconciliation	Conversion rate for the reconciliation amount, calculated by the rogue POS terminal
10	conversion rate, cardholder billing	As above; this value is calculated by the rogue POS terminal
11	systems trace audit number	Transaction sequence number generated by the rogue POS terminal
14	date, expiration	**0x5F24** – Expiry date of the card (YYMM)
16	date, conversion	Date / time of the currency conversion (same as 7)
19	country code, acquiring institution	Country code of the rogue POS terminal (e.g. 0826 for UK, 0840 for USA, 0036 for Australia)
20	country code, primary account number	**0x5F28** – Country code for the card i.e. 0826 – UK
21	country code, forwarding institution	**0x5F28** – Country code for the bank that issued the card i.e. 0826 – UK
22	point of service entry mode	Type of POS terminal, constant value "051" for Chip & PIN / EMV contactless terminals
23	card sequence number	**0x5F34** – Identifies subsidiary EMV cards issued on the same 16-digit account number
25	point of service condition code	Constant "00" normal card presentment
26	point of service PIN capture code	Constant "x8xx" indicates a POS terminal that accepts up to 8 digits
27	approval code length	Constant set by acquirer bank
32	acquiring institution identification code	Constant set by acquirer bank
33	forwarding institution identification code	Constant set by acquirer bank, indicates the institution that will provide the card payment clearing (steps 6 to 9 in Figure 2)
34	primary account number, extended	Not applicable to Visa – used only when the primary account number begins with "59"
39	action code (was response code)	Constant "0xx" for financial transaction request messages
43	card acceptor name/location	Constant string name and location of the merchant
49	currency code, transaction	**0x5F2A** – Transaction currency code
50	currency code, reconciliation	Currency code for reconciliation, see item 5
51	currency code, cardholder billing	**0x9F42** – Currency Code from the card.
66	country code, receiving institution	**0x5F28** – Country code for the bank that issued the card i.e. 0826 – UK
100	receiving institution identification code	Code that identifies victim's bank – ISO 7812
102	account identification 1	Information contained in 16-digit card account number **0x5A**
103	account identification 2	Information contained in 16-digit card account number **0x5A**

In the above table, data fields from the EMV card data are denoted by their EMV reference number e.g. **0x5A**.

Table 3 shows the communication sequence required for the POS emulation to transmit a transaction to the acquirer bank.

Table 3. POS / acquirer communication sequence

Message	From → To	Purpose
financial presentment request message	POS → Acquirer	Requests approval and money transfer by the acquirer
financial presentment response	Acquirer → POS	Contains the answer to the request
financial presentment confirmation	POS → Acquirer	Confirms that the response was received

7. TEST RESULTS

The attack software has been tested against various UK-issued credit / debit cards. Table 4 shows the vulnerability of several different card types.

Table 4. Vulnerability of UK-issued contactless card types

Card Type	Max Value	Comment
Visa credit cards (UK currency)	£85.00	Visa credit cards will approve multiple transactions until offline limit reached
Visa credit cards (foreign currency)	€999,999.99 $999,999.99	Visa credit cards will approve foreign currency transactions up to the maximum value possible in EMV
Visa debit cards (UK currency)	£45.00	Visa debit cards will approve multiple transactions until offline limit reached
Visa debit cards (foreign currency)	€0.00 $0.00	Visa debit cards decline foreign currency contactless transactions
MasterCard	N/A	MasterCard is not affected by this attack as the cards request online completion of transactions in local currency and foreign currencies

7.1 Transaction capture timings

The Android transaction capture app is designed to operate as quickly as possible, thereby reducing the risk of detection for the attacker. The software automatically collects the fraudulent transaction as soon as it detects a Visa contactless credit or debit card. Table 5 shows analysis of protocol timings from 20 captured fraudulent transactions.

Table 5. Fraudulent transaction capture timings

Statistics	Time (in milliseconds)
Average transaction duration (card discovery to transaction approval)	478ms
Standard deviation	36ms
Fastest transaction	452ms
Slowest transaction	527ms

8. POTENTIAL SOLUTIONS

The key weakness exploited in this paper is that Visa credit cards will authorise unlimited value transactions in a foreign currency. This makes the attack described in this paper both scalable and very lucrative.

The solution is relatively simple. This can be done by changing future Visa credit cards to implement one or both of the following:

- the cards will request *online* completion of contactless foreign currency transactions; making the transaction subject to the additional *online* verification steps.
- the cards will force "Chip & PIN" completion of all foreign currency transactions; this will eliminate the possibility of high value transactions without the added security of cardholder's PIN.

9. CONCLUSION

In this paper we have demonstrated that it is possible to collect high value transactions from contactless Visa credit cards whilst the card is still in the victim's pocket. The attack exploits a previously undocumented flaw in the cards, in which the cards will approve transactions of unlimited value in a foreign currency. Combined with the lack of POS terminal authentication and the threat of contactless payment card skimming, this vulnerability poses a real risk that allows high value fraudulent transaction to be harvested and converted into money.

Our experimental results show that the attack could be implemented in the "real world" because:

- it takes less than 500milliseconds to collect a transaction
- NFC enabled Android phones are cheap and readily available
- the phone looks innocent if the attacker is challenged by the police or a member of the public

We have also outlined a scenario by which the captured fraudulent transactions could be exploited by a rogue merchant to access the money in the victim's bank account. The rogue merchant receives the transactions and passes them off as genuine transactions to their bank. It should be noted that although we have implemented the rogue POS terminal software, we have not tested it against a live acquirer transaction clearing system.

From this we can conclude that this attack represents a plausible threat to contactless Visa credit cards. We can also see that it can be easily remedied.

We have proposed two simple changes in the operation of Visa credit cards that would eliminate the risk posed by this attack. Both of which use the existing functionality of the cards and would therefore be relatively inexpensive to implement.

10. ACKNOWLEDGMENTS

Our thanks to Feng Hao and Dylan Clarke for proof reading some sections of this paper. The work presented here is partly supported by the UK RCUK *Social Inclusion through the Digital Economy (SiDE)* EP/G066019/1 project and the UK EPSRC *Cybercrime Network* EP/K003410/1.

11. REFERENCES

[1] Bond, M., Choudary, O., Murdoch, S.J., Skorobogatov, S., Anderson, R. 2014. Chip and Skim: cloning EMV cards with the pre-play attack. *35th IEEE Symposium on Security and Privacy*. http://arxiv.org/pdf/1209.2531.pdf

[2] Cooper, D. and Barner, J. 2008. Tokeneer ID station EAL5 demonstrator. Technical Report S.P1229.81.1, Altran Praxis.

[3] Drimer, S. and Murdoch, S.J. 2007. Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks. *16th USENIX Security Symposium*, Boston, MA, USA. http://www.cl.cam.ac.uk/~sjm217/papers/usenix07bounding.pdf

[4] EMVCo. 2011. EMV Integrated Circuit Card Specifications for Payment Systems – Version 4.3. http://www.emvco.com/specifications.aspx?id=223 [Accessed: 22 August 2014]

[5] EMVCo. 2014. EMV Contactless Specifications for Payment Systems – Version 2.4. http://www.emvco.com/specifications.aspx?id=21 [Accessed: 22 August 2014]

[6] Francis, L., Hancke, G., Mayes, K., Markantonakis, K. 2012. Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. *The 2012 Workshop on RFID and IoT Security (RFIDsec 2012 Asia)*, Nai-Wei, L., Yingjiu, L. (editors). Vol. 8, IOS Press (Cryptology and Information Security Series), pp. 21-32. http://eprint.iacr.org/2011/618.pdf

[7] Freitas, L. and Emms, M. 2014. Formal specification of EMV protocol. School of Computing Science Technical Report Series 1429, Newcastle University.

[8] International Organization for Standardization. 1995. ISO 8583:1995 – Financial transaction card originated messages – Interchange message specifications.

[9] Murdoch, S.J., Drimer, S., Anderson, R., Bond, M. 2010. Chip and PIN is Broken. *IEEE Symposium on Security and Privacy*, pp. 433-446. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5504801&isnumber=5504699

[10] Reason, J. 1990. *Human Error*. Cambridge University Press.

[11] Smans, J., Jacobs, B., and Piessens, F. 2013. VeriFast for Java: A Tutorial. In: Clarke, D., Noble, J., Wrigstad, T. (eds.) *Aliasing in Object-Oriented Programming*. LNCS, vol. 7850, pp. 407– 442. Springer, Heidelberg.

[12] The UK Cards Association Limited. 2013. Standard 70 – Card Acceptor to Acquirer Interface Standards.

[13] Woodcock, J. and Davies, J. 1998. *Using Z*. Prentice Hall.