e-EMV: Emulating EMV for Internet Payments using Trusted Computing Technology

Shane Balfe and Kenneth G. Paterson

Technical Report
RHUL-MA-2006-10-v2
07 March 2008

Royal Holloway
University of London

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England
http://www.rhul.ac.uk/mathematics/techreports

Abstract

The introduction of EMV-compliant payment cards, with their improved cardholder verification and card authentication capabilities, has resulted in a dramatic reduction in the levels of fraud seen at Point of Sale (PoS) terminals across Europe. However, this reduction has been accompanied by an alarming increase in the level of fraud associated with Internet-based Card Not Present (CNP) transactions. This increase is largely attributable to the weaker authentication procedures involved in CNP transactions. This paper shows how the functionality associated with EMV-compliant payment cards can be securely emulated in software on platforms supporting Trusted Computing technology. We describe a detailed system architecture encompassing user enrollment, card deployment (in the form of software), card activation, and subsequent transaction processing. Our proposal is compatible with the existing EMV transaction processing architecture, and thus integrates fully and naturally with already deployed EMV infrastructure. We show that our proposal, which effectively makes available the full security of PoS transactions for Internet-based CNP transactions, has the potential to significantly reduce the opportunity for fraudulent CNP transactions.

1 Introduction

The use of magnetic stripe cards for Point Of Sale (PoS) transactions is slowly being phased out in favour of Integrated Circuit Cards (ICCs) compliant with the Europay-Visa-Mastercard (EMV) specifications [14, 15, 16, 17]. This process is essentially complete in Europe, well underway in the Far East, and under active consideration for North America. This major technology change is motivated by the susceptibility of magnetic stripe cards to cloning, and the projected fraud losses associated with the continued use of such cards. ICC cards, having chips that are actively engaged in the transaction authorisation process, are much less easily cloned. The U.K., which began the transition process in 2004, has seen a 47% post-migration reduction in PoS fraud [6]. However, with this reduction in PoS fraud has come an accompanying increase in Internet-based Card Not Present[1] (CNP) fraud. For example, CNP transaction fraud is now the predominant means through which payment card fraud is committed in the U.K. [5]. Specific figures are harder to come by for other countries, but we believe the U.K. experience to be illustrative of the global trend.

[1] For the remainder of this paper all references to CNP transactions refer to Internet-based CNP transactions.

The level of fraud seen with CNP transactions has raised significant concern amongst the card processing community. Over the years a number of proposals have been put forward to address this problem. The most noteworthy of these are SET [37] and 3-D Secure [49]. However, neither of these proposals have gained widespread adoption. Indeed, the vast majority of CNP payments are protected today just using SSL/TLS to secure data in transit and to provide a limited form of merchant authentication. Customer authentication is provided through the customer's ability to provide relevant card details such as the Personal Account Number (PAN) and the corresponding Card Security Code (CSC) over the established SSL/TLS session.

Thus, from the merchant's perspective, there is no guarantee that the customer is actually in possession of the card corresponding to the details being proffered in a payment transaction. This problem is exacerbated by the perpetual increase in "phishing" attacks [22]: through a combination of social engineering and technical subterfuge, customers may be tricked into revealing their card account details to an attacker. In the past, these attacks have required some active participation from the user. However, as attacks become more sophisticated, the human element is becoming removed, with malicious software (malware) residing on a customer's platform being used to capture customer account details and manipulate customer transactions (including possibly instigating new transactions). We use the term Transaction Generator (TG), introduced in [24], to describe such malware in the remainder of this paper.

On the other hand, from the customer's perspective, there is little, if any, assurance that a merchant will endeavour to protect sensitive cardholder data stored on the merchant's servers. In a PoS environment, a customer's suspicion may be aroused by environmental cues, such as damage to the housing of the payment terminal or the demeanor of the sales assistant. Based on this physical evidence, a customer may decide not to engage in a transaction. However, in an on-line setting, the environmental (browser-based) cues that are available are often either poorly interpreted, or not heeded [12]. In an attempt to instill greater confidence in customers, the Payment Card Industry Data Security Standard (PCI-DSS) [11] has been proposed. This standard details 12 mandatory requirements that merchants and third party processors must satisfy. A customer then trusts that a (legitimate) merchant is in compliance with this standard and has adopted best practice procedures to protect their credit card details. However, PCI-DSS-compliance levels are still low, and there have been instances of companies passing a PCI-DSS conformance audit, only to be later shown to be non-compliant with the standard at the time of a breach [35]. Recently, concerns over merchants running vulnerable payment applications have become so great that beginning in January 2008, Visa will begin implementing a series of mandates to eliminate the use of non-secure payment applications from the Visa payment system [51]. Visa will only accept payments from merchants using payment applications that adhere to, and have been validated against, Visa's Payment Application Best Practices (PABP) [50].

In summary, current CNP transaction processing cannot make use of the robust security features available from EMV-compliant ICC cards, and simply reverts to pre-EMV card authentication procedures. This weakness is now being ruthlessly and increasingly exploited by fraudsters, and closing this attack vector represents a significant challenge to the payment card industry.

1.1 Our Work

To combat the threats posed by malware TGs (and by merchants that are non-conformant with the PCI-DSS), we propose e-EMV, a system that makes use of Trusted Computing technology to securely emulate EMV for CNP transactions. We describe a system architecture encompassing user enrollment, deployment of software cards to customer platforms, card activation, and subsequent transaction processing. Our e-EMV proposal uses a combination of application software, a Trusted Platform Module (TPM) [45], a processor (with chipset extensions) [23] and Operating System (OS) support [34, 1] to securely emulate the functionality of a standard EMV-compliant card in software. We provide a detailed description, at the level of individual TPM commands, showing how this emulation is achieved. We also explain how the security features provided by Trusted Computing are used to obtain an appropriate level of security for our system.

Our approach of emulating EMV on Trusted Platforms for CNP transactions provides the following benefits:

1. It is possible to demonstrate e-EMV card ownership and authentication for CNP transactions as in standard EMV card authentication procedures. Thus a merchant can ensure that a customer claiming to present a particular e-EMV card is the legitimate owner of that card.

2. A merchant can obtain a payment guarantee through being able to demonstrate customer authorisation of a transaction.

3. A customer can gain an assurance prior to transaction initiation that a merchant will endeavour to protect sensitive cardholder information. Such a feature is is absent from EMV's use at PoS terminals, a fact which is now allegedly being exploited by criminal gangs [48].

4. An executing e-EMV application can avoid the threat posed by malware TGs, by hosting e-EMV cards in their own isolated memory partition, free from observation and interference.

5. Our proposal supports direct migration to EMV cards that are more powerful and offer enhanced authentication and transaction authorisation procedures, relative to cards used in current EMV deployments for PoS transactions. While these enhanced security features (Dynamic Data Authentication (DDA) and Combined DDA and application cryptogram generation (CDA)) are specified in the EMV standards [15], they are not ubiquitous on today's EMV ICCs because of their cost implications; instead, some banks find it more economical to work with cheaper cards and accept the higher level of risk implied by the use of Static Data Authentication (SDA) [15]. However, our proposal, involving only software running on mass-market consumer computing platforms, is not restricted in this way.
6. In e-EMV, modifications and enhancements (resulting from, for example, changes to the EMV specifications) can be realized through relatively straightforward software update processes. In contrast, traditional EMV requires complicated and expensive card upgrades to achieve the same thing. This lends our e-EMV approach an inherent degree of "future proofing."

The motivation for emulating EMV, rather than designing a new protocol from scratch, comes from real-world, practical constraints. By creating an environment where EMV transaction flows can be mapped directly to e-EMV transactions we can avoid any expensive re-engineering of the back-end financial network. Indeed, our proposal is compatible with the existing EMV transaction processing architecture, and thus integrates fully and naturally with already deployed EMV infrastructure. Additionally, as a consequence of EMV having been developed and deployed over many years, many of the protocol bugs should already have been ironed out.

The relative applicability of our approach is obviously dependent on the ubiquity of Trusted Computing on commodity computing platforms. This is, however, not an unreasonable assumption, and once made, allows a number of interesting solutions to a whole host of security problems plaguing CNP transactions. Currently available sales figures for 2005 [41] showed estimates of 32% of all notebook systems shipped that year being TPM-enabled. This figure is expected to nearly triple by the end of 2007, with processor and OS support to follow soon after [23, 34, 1]. In addition to this, the recently released Trusted Mobile specifications [43] hold particular promise for our architecture, as market penetration of mobile devices typically occurs at an expedited rate compared to that of Personal Computer (PC) clients.

Our e-EMV proposal could result in a mutually beneficial arrangement for TPM manufacturers and card issuers. Indeed, our e-EMV proposal could be a "killer application" for Trusted Computing in the consumer space, something that seems to be currently lacking.

Organisation: In Section 2, we present an overview of related work. In

Section 3, we provide a brief introduction to EMV and Trusted Computing. In Section, 4 we describe the payment model used for CNP payments. Section 5 provides a high-level overview of our e-EMV architecture. Section 6 explains in detail the procedures and processes involved in establishing an e-EMV card within a Trusted Computing enhanced platform. Following on from this, Section 7 highlights how a normal EMV transaction flow can be mapped to an e-EMV transaction. Section 8 examines how the threats posed to CNP transactions are mitigated by e-EMV. We conclude with Section 9.

2 Related Work

Herreweghen and Wille [21] present a detailed evaluation of the security requirements for Internet-based payments. These include: payment guarantee for merchant, mutual authentication between customer and merchant, customer privacy and anonymity, and transaction authorisation.

The real world roll-out of EMV and the accompanying desire to replicate the fraud reduction seen with PoS transactions has resulted in a number of proposals that utilize EMV's functionality for Internet-based payments [21, 27, 31, 13, 2]. However, these approaches have not seen any real traction in the market place. A possible explanation for this is these proposals' underlying assumption that customers would make use of card readers connected to their PC platforms. This assumption engenders an additional cost in the form of distributing card readers to end-users. Even if the cost issue could be surmounted, these approaches alone offer only limited security gains. This is due to the lack of a trusted path between the card-reader and host, as well as a lack of OS support for application isolation. Without these, a TG could passively observe Personal Identification Number (PIN) entries, actively modify transaction data and possibly generate new transactions, enabling criminals to remotely take control of users' payment cards. There has, however, been a recent development in integrating EMV with CNP transactions that aims (albeit indirectly) to address the TG issue. This is the proposed use of "unconnected" card readers [29]. Unfortunately, this approach also suffers with respect to the additional costs associated with distributing card readers to end-users and, once deployed, cannot be updated to address new threats as they emerge.

The use of Trusted Computing to combat phishing has been proposed in [3] and [18]. The primary threat considered in [3] is external attack, whereby a credential needs to be extracted from the client's platform in order for it to have any value to an attacker. No consideration is given to the ever-increasing threat posed by malware TGs. The threat from malware is examined in greater detail in [18, 25, 24]. Here, much like in our approach, Virtual Machines (VMs) are used to constrain the use of malware to an individual VM "compartment". These authors also suggest using visual cues as to the trustworthiness of the VM with which an end-user interacts, an idea originating in [7]. Such work is complementary, but orthogonal, to our own: these cues could be adopted in e-EMV and might help enhance customer protection from TGs.

3 Overview of EMV and Trusted Computing

In this section, we provide a high-level overview of the main features of EMV and of Trusted Computing, providing references for readers who require more detail. This section introduces many acronyms on which we will later rely, unfortunately this is somewhat unavoidable given the two technologies (EMV and Trusted Computing) involved.

3.1 EMV

EMV, as described in the current iteration of the specifications defines the full spectrum of interactions, from the physical to the logical, between an ICC and an ICC-enabled PoS terminal [14]. A readable introduction to the technical content of the EMV specifications can be found in [31]. EMV supports both cardholder authentication, through new Cardholder Verification Methods (CVMs), and ICC authentication through either SDA, DDA or CDA. SDA cards allow the terminal to determine if the data on the card has been modified since card personalisation. DDA cards are more complex in that they allow for the same checks as SDA but also enable the terminal to determine if the card is genuine or not through a challenge-response mechanism. This involves the ICC generating a signature (using a card-specific private key, with the card storing a certificate issued by the card's issuer for the corresponding public key) over the terminal-supplied challenge. CDA cards are similar to DDA cards except the message signed by the ICC includes an additional Application Cryptogram (AC). ACs are used to protect transaction messages generated by the ICC using AC Session Keys (derived from a long-term master key, the ICC AC Master Key, that is shared between the ICC its card issuer). This ICC AC Master Key is unique per card and is derived from a customer's PAN, PAN sequence number and an issuer Master Key. Here the PAN sequence number helps to identify a card amongst several cards belonging to the same customer with the same PAN.

EMV defines a number of different types of transaction messages. Depending on the reconciliation of card and terminal risk management routines one of the following will be generated by the card: an Application Authentication Cryptogram (AAC), an Authorisation Request Cryptogram (ARQC), an Authorisation Response Cryptogram (ARPC) or a Transaction Certificate (TC). If the transaction is approved, the ICC will generate a TC which will be passed to the terminal and can be used to claim payment during the clearing process. If a transaction is declined, the ICC will generate an AAC. In the event that the transaction needs to be approved on-line the ICC generates either an AAC or an ARQC. which will be forwarded to the issuer. The issuer will then reply with an ARPC indicating whether the transaction should be approved or declined, in which case a TC or an AAC will be generated by the ICC.

3.2 Trusted Computing

Trusted Computing relates directly to the types of systems espoused by the Trusted Computing Group (TCG). Namely, a *trusted system* is one which will behave in a particular manner for a specific purpose. Trusted Computing is based on the inclusion a hardware "root of trust", the Trusted Platform Module (TPM), within a platform, that aims to allow users (and third parties) to assess the "trustworthiness" of the devices with which they interact. To achieve this, a Trusted Platform will be able to attest to its current operating environment, or any sub-component thereof. Interested second parties can make requests for this information, and any divergence from an intended operating state can be detected by the second party, allowing them to make informed decisions as to whether to continue to interact with the current system. We assume the reader is familiar with Trusted Computing and the TPM command set, literature for which can be found in [30] and [47] respectively. We now briefly outline some of the Trusted Computing functionally upon which we depend in the remainder of this paper.

Measuring events on a platform is a two-stage process that begins with an *extend* command. This command, more commonly referred to as 'extending the digest', appends a hash of the event being measured to one of a number of Platform Configuration Registers (PCRs) located internally to the TPM. These PCRs store a representative hash of all the events generated so far to form a picture of the current platform state. The second stage in this process is the writing of events, reflected in a PCR, to permanent storage. This logging of integrity-altering events within a platform occurs in the Stored Measurement Log (SML). The SML maintains sequences of events to which new events are appended. For example, such an event might be the launch of an application. The measurement of an application can be performed by computing the cryptographic hash over the current PCR value (to which this event will be recorded) concatenated with the application's instruction sequence, its initial state (i.e. the executable) and its input. This hash is then written to the PCR and a description of the event recorded in the SML.

When a verifier wishes to inspect a host platform's configuration, it requests (a portion) of the requestee platform's SML and asks the requestee's TPM to produce a signature over a specific set of PCR values describing a portion of the platform's operating state. The TPM uses the private component of a key pair called an Attestation Identity Key (AIK) to perform the signing operation over the requested PCR values. However, in order for the values being attested by a platform to have meaning outside of the confines of a challenger's platform, it is necessary for the challenger's platform to first obtain a credential for the signing AIK from a trusted third party recognised by the verifier. How this credential is obtained differs between version 1.1b and version 1.2 of the TCG specifications. Version 1.1b uses what is referred to as the "Privacy CA" model (see [40]) whilst version 1.2 introduced a new (optional) model in the form of Direct Anonymous Attestation (DAA). We refer readers to [10] and [47] for the technical details of DAA.

A TPM can manage an unlimited number of cryptographic keys. Every TPM_Key [46] has an assigned attribute designation as well as a defined key type. Attribute designations help to define key mobility. A key can be designated as being either migratable, Certified Migratable or non-migratable. Migratable keys are unrestricted and are capable of leaving a TPM, Certified Migratable Keys (CMKs) require authorisation from a third party in order to be migrated from one TPM to another, whilst non-migratable keys are inextricably bound to a single platform. Key types are used to define what particular operations a TPM_Key is capable of performing. For example, a key could have a control policy that says a TPM_Key is of signing type or storage type depending on its intended use. In addition, individual private keys may require explicit authorisation data to be entered by a user before the key can be used, or require a specific set of platform metrics to be present before the key can be loaded and used.

To allow a second party to inspect the properties of a CMK or non-migratable TPM_Key, the private component of an AIK (for which certification of the public component has been obtained) may be used to sign the public component of a TPM_Key to produce a TPM_Certify_Info(2) structure [46]. This structure contains a digest of the corresponding public TPM_Key and information that describes the control policy for the private portion of this key. This structure can be sent (with the corresponding AIK credential and public key) to a second party for verification.

In addition to the features provided by the TPM, both Operating System (OS) and processor support represent integral components in the realisation of Trusted Platforms. As well as providing access to TPM functionality, a Trusted OS will be capable of launching sandboxed Virtual Machines in which applications can run. A Dynamic Core Root of Trust for Measurement (D-CRTM) [39], as defined by Intel in their Trusted Execution Technology (TXT) and AMD in their Secure Virtual Machine (SVM) system architectures [23, 4], refers to the instantiation of one or more protected Virtual Machines (VMs) running on-top of a Measured Virtual Machine Monitor (MVMM) [1]. These VMs run in parallel to the standard OS partition without requiring a system reboot. The following are generic security services for a protected VM based on an MVMM; we assume the presence of such functionality in our later discussions.

No interference: Ensures that a program is free from interference from entities outside its execution space.

Trusted path: Assumes a trusted path between a program and an input device.

Secure inter-process communication: Enables one program to communicate with another, without compromising the confidentiality and integrity of its own memory locations.

Non-observation: Ensures an executing process and the memory locations it is working upon are free from observation.

4 Payment model

The processing model used for most card payment systems (including EMV) is typically referred to as the four-corner-model. Within this model, a number of steps are necessary to complete a given transaction (see Figure 1). Prior to a customer being able to interact with a merchant, it is necessary that they follow some issuer-specific enrollment procedure in order to obtain a physical payment card. A merchant, likewise, can only accept payments from a customer if they have preregistered to accept payments for that customer's particular card type with their acquirer. The dashed line in Figure 1 represents the boundary of the financial network domain. Payment processing occurs as follows:

Step 1: The process begins with a customer signaling their intent to purchase goods by forwarding a payment record to a merchant. In this instance, the actual characteristics of a payment record differ depending on the environment in which it was created. For an on-line purchase, a payment record typically includes the information embossed on the customer's physical

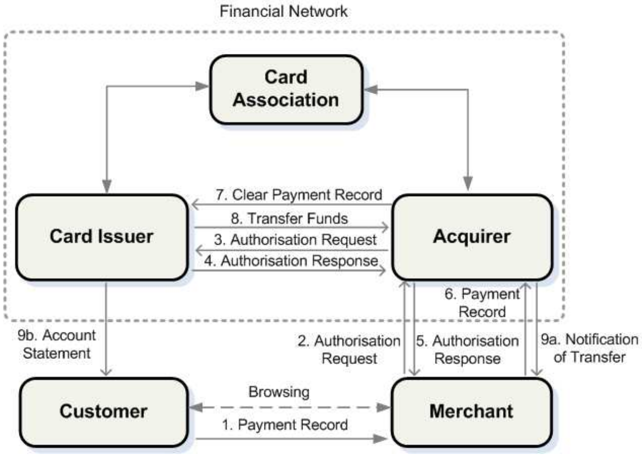

Figure 1: Generic model for card processing.

payment card in conjunction with certain merchant supplied information (such as the invoiced amount).

Steps 2-5: These steps occur immediately after receiving the customer's payment record. They consist of a merchant submitting the transaction details to their acquirer which will either authorise or reject the transaction based on their interactions with the customer's card issuer. After this, the merchant will either confirm payment or inform the cardholder that their transaction has been rejected.

Steps 6-9: Based upon the transaction being approved, either as a result of a successful outcome from steps 2-5 or merchant risk management routines, steps 6-9 represent the account settlement process through which funds are debited from a customer's account and credited to the merchant's.

Perhaps the most surprising feature of this model is that a positive transaction authorisation does not guarantee payment for a merchant. It is merely an indication that the card account details being proffered have not been reported stolen and that the customer has sufficient funds to cover the transacted amount. Indeed, unless the card has been reported stolen, it is difficult for a card issuer, and by extension a merchant, to ascertain whether a particular transaction is fraudulent or not.

Our e-EMV architecture closely follows the generic four corner model presented above. The only differences in our approach are that the communication channel between the customer and the merchant is now Internet-based (instead of being based on physical proximity of card and terminal), and that both card issuers and acquirers must provide enrollment facilities for their e-EMV clients, separate to the facilities provided for the physical issuance of an EMV card. For a card issuer this means providing a mechanism through which customers can establish their e-EMV cards on their platforms. Similarly, for an acquirer this means providing a facility whereby a merchant can download an application that can interface with an customer's e-EMV application. In doing so, we assume the presence of a Public Key Infrastructure (PKI), which can be an extension of the one that currently exists for EMV. In this regard, enrollment facilities should be able to be authenticated by customers and merchants via public key certificates issued by a particular card association.

5 An overview of e-EMV

This section aims to present a high-level overview of our e-EMV proposal. Our overall aim is to provide functionality akin to that of a standard EMV card by replicating that functionality through procedures and capabilities natively supported by a host that is augmented with Trusted Computing. This allows us to provide a secure and extensible architecture for CNP transactions.

The procedure for establishing an e-EMV card on a Trusted Computing platform is a two-stage process consisting of an account activation stage and an application delivery stage. In describing this architecture we are making the following important assumptions, reiterated from Section 4: we assume the presence of a PKI extending the one currently in existence for EMV, we assume that Trusted Computing Platforms are ubiquitous within the merchant/customer domain, and additionally, that both processor and OS support are available to all platforms within this domain. Furthermore, we have the underlying assumption that a card issuer has already made the decision to extend credit or debit facilities to a particular customer.

5.1 Enrollment

Enrollment in our e-EMV architecture involves a customer formally registering as a legitimate cardholder, allowing them to obtain an e-EMV card. Much like enrollment in the traditional EMV architecture, a card issuer within our system is responsible for enrolling cardholders as well as later authenticating their transactions (and possibly the cardholders themselves). Acquirers can be seen as providing similar functionality to their merchant customers. The stages for establishing an e-EMV card on a TPM-enabled platform are as follows.

Account Activation: Account activation is the process through which a

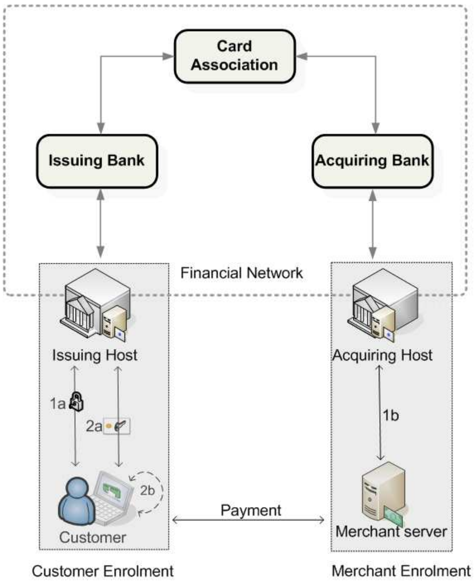

Figure 2: Enrollment procedure.

customer becomes a member of a card-issuer-controlled group. In this case, group membership is indicative of a customer activating their account within the system (Figure 2, Step 1a). In the context of Trusted Computing, this means enrolling with a Privacy CA (see [40]) or optionally becoming a member of a card-issuer-controlled Direct Anonymous Attestation (DAA) group (see [10, 47]). This is achieved by the customer demonstrating the presence of a non-migratable TPM-controlled secret over which certification is requested, either an Attestation Identity Key (AIK) in the case of the Privacy CA or a secret value f for DAA.

At this point, the actual binding between the customer and their platform can be established through a mechanism of the issuer's choosing. For example, the customer might provide information supplied to them in the pre-enrolment stage (in which the card issuer agreed to extend debit/credit facilities), communicated using an out-of-band mechanism. In addition to this authentication information, a platform will send various platform credentials (describing the binding of the TPM to the platform) as well as evidence of the existence of a non-migratable TPM-controlled secret. After receiving this information, the card issuer performs due diligence in satisfying itself as to the relationship between a customer and their platform. This is achieved through a reconciliation of the provided authentication information with an examination of evidence supporting the existence of a TPM-controlled secret.

If the card issuer is satisfied by this evidence, the customer's platform will receive certification on their TPM-controlled secret. This certification will later be used to demonstrate the customer's membership of a particular card scheme.

Enrollment for the merchant (Figure 2, Step 1b), by contrast, involves satisfying the requirements for payment processing as laid down by the relevant acquirer's Merchant Operator Guidelines (MOGs). During the merchant enrollment procedure, the merchant's acquirer also becomes a certificate issuer, again in the context of the Privacy CA or DAA models.

Secure Application Delivery: The customer downloads a small secure application bundle (Figure 2, Step 2a) that fulfills the role of an e-EMV card as well as acting as a guide through the process of creating/installing the requisite TPM managed keys (Figure 2, Step 2b) [15]. This bundle, once installed will enable a platform to perform electronic transactions analogous to those carried out by an EMV card at a PoS terminal. Our e-EMV application contains all the keys required to perform CDA authentication. However, unlike an EMV card, our e-EMV application will need to provide some of the functionality typically seen in PoS terminals, particularly when it comes to cardholder verification (see Section 7).

The merchant will need to download and install a plug-in, similar in applicability to a 3-D Secure merchant plug-in [49], but capable of emulating certain EMV terminal functionalities. The most important of these will be the authentication of customer-supplied credentials. Much like terminals in the physical setting, merchants will require card issuer's public key certificates in order to verify customer transactions.

5.2 Transaction Architecture

In order for an e-EMV application (card) to be launched, a Trusted Platform's OS sends an instruction sequence to the D-CRTM to create a new isolated memory partition. The D-CRTM launches a VM into this newly created memory area, which in turn executes our e-EMV application, free from observation and influence of TGs. For the details of launching a secure VM, we refer readers to [23]. With the exception of Step 1 (see Figure 3), the basic flow for e-EMV transaction processing follows EMV's transaction processing at PoS. The steps are as follows:

Step 1 represents the browsing phase in which a customer peruses a particular merchant site. During this step, the customer verifies that the merchant is a member of a valid group of merchants, corresponding to a particular Privacy CA/DAA issuer (acquirer). Additionally, the customer may verify that the merchant is in a state that exemplifies an adequate policy

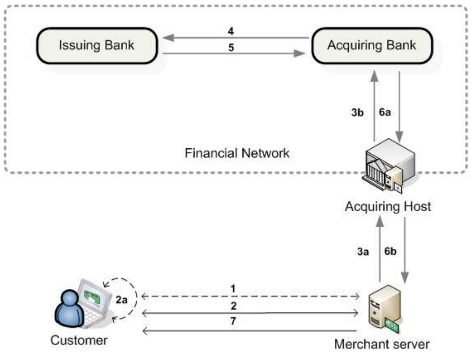

Figure 3: Transaction architecture for e-EMV payments.

for addressing privacy and confidentially concerns (for example, conformance with the PCI-DSS or Visa's PABP) via Trusted Computing's attestation procedures.

Step 2 represents the typical EMV ICC and terminal interaction, except that the communication channel is now exclusively Internet-based.

Step 2a represents the creation of an AC. As part of this step the customer's platform generates a signature over its AC (using the certified non-migratable TPM-controlled secret established during the account activation phase) as well as the Platform Configuration Register (PCR) values that provide evidence as to the platform's current state at the time of transaction authorisation. These data items are communicated over a secure and authenticated channel to the merchant server.

Steps 3 to 7 are executed if the optional decision to go "on-line" has been exercised. As a result of either terminal or card risk management procedures, the AC (possibly in conjunction with the information adducing the current platform state) is forwarded to the customer's card issuer (Steps 3a–4). After examining the received data, the card issuer returns an AC of its own (Steps 5–7). This AC informs both the card and the merchant as to whether the request is to be approved or declined, in which case the card application will either proceed with the transaction or reject it. It is important to note that all e-EMV messages exchanged in Steps 2–7 are standard EMV message flows that are exchanged between a PoS terminal and EMV card.

Where additional Trusted Computing platform state information has been added to EMV's AC messages, it is optional for an card issuer to examine this state information as part of their decision making process. The Tag Length Value (TLV) encoding mechanism used in EMV makes it easy for an card issuer to ignore any extraneous information from a transaction referral request. Significantly, if the card issuer does decide to take a customer's platform state into consideration, the customer's card issuer would not need any Trusted Computing facility to examine these characteristics. It would only need to be capable of hashing some supplied records and verifying a signature (as we shall see in Section 6). Indeed, such functionality could be provided by a third party facility or performed by the merchant plug-in.

6 Installing and Instantiating an e-EMV card

This section explains in greater detail the process involved in establishing an e-EMV card within a Trusted Platform.

6.1 Account Activation

In order for a customer's e-EMV account to be activated, the customer's Trusted Platform must become a member of a card issuer controlled group. This process is mirrored for the merchant server account activation procedure with respect to a merchant's acquirer. In both cases, account activation is achieved by successfully obtaining/generating a credential issued for an AIK public key. This credential could be in the form of X.509 certificate issued by a Privacy CA or a credential issued by a DAA Issuer [47].

In discussing the use of a Privacy CA or a DAA Issuer here, we are not suggesting that our approach should actually benefit from the privacy-enhancing features available from either of these choices. Rather, we see the card-issuer playing the role of either a Privacy CA or DAA Issuer as providing a convenient mechanism for achieving client-side certification within the limitations of the TCG specifications.

In either case, the choice of a Privacy CA or DAA Issuer will end with a similar result: the customer will have their account activated, later allowing them to demonstrate physical/logical possession of an active card within the system. Through the establishment of a customer-centric credential embedded within a platform, the customer will be able to attest to the existence of a key capable of replicating EMV's SDA/DDA/CDA functionality.

In obtaining this credential, the credential issuer needs to ensure that a request is coming from a particular customer. This can be achieved, for example, through a customer demonstrating knowledge of a shared secret created in the pre-enrollment stage over a secure channel. However, the actual mechanism used is orthogonal to our discussion.

6.1.1 Privacy CA Approach

Here the process involves a customer's TPM generating an attestation key pair (S_{AIK} and P_{AIK} for the private/public portions respectively) and having P_{AIK} incorporated into an ICC Public Key Certificate (AIK credential). When creating this key, the customer specifies a password for S_{AIK}, which will be required every time the key is loaded for use. The process involved in the generation of a new attestation key within a platform maps to the generation of an AIK key within a TPM, that is, as a result of performing the TPM_MakeIdentity command [47, pp.146]. The P_{AIK} portion of this AIK, along with various platform credentials and customer authentication data, are encrypted with the issuing host's public key and sent to the card-issuer-controlled Privacy CA. After authenticating a particular customer and satisfying itself that the request is coming from a genuine TPM, the Privacy CA will issue an AIK certificate to the customer's TPM-enabled platform. This credential can then be used to provide evidence of card activation within the system. The AIK certificate could be further enhanced through X.509v3 extensions. For example, it is possible to add key and policy information to the credential, such as setting a private key usage period under which the AIK signing key will operate.

6.1.2 DAA Approach

The DAA enrollment procedure occurs exactly as laid out in [10]. However, here the role of the DAA Issuer is provided by the card-issuer-controlled issuing host. After a successful completion of the DAA_Join command [47, pp.255], the customer will instruct their TPM to create a new AIK pair (S_{AIK} and P_{AIK} for the private/public portions respectively). As part of this key generation process, the customer specifies a password for S_{AIK}, which will be required every time the key is loaded for use. The customer then instructs their TPM to sign the P_{AIK} component using their newly enrolled DAA key. This will later allow the customer to be able to demonstrate that their account has been activated with respect to a particular card issuer.

Subsequent to a successful completion of either the DAA join procedure or upon receipt of an AIK certificate from a Privacy CA, the card issuer activates the customer's account within their systems. This enables the soon-to-be-downloaded e-EMV application to be used in ensuing financial transactions.

6.2 Secure Application Delivery

The delivery of the e-EMV application to a customer's platform involves an interactive process between a customer and their card issuer. The delivery of the software itself necessitates a number of checks to ensure the binding between a customer and their platform. The software set-up utility requires the inclusion of a unique EMV ICC AC Master Key per card issuance, and this key cannot be generated by the platform itself as it is derived from an issuer Master Key. Instead the ICC AC Master Key needs to be injected into the platform as part of the application delivery process. In addition to the secure delivery of the application, there is an underlying customer-driven requirement to ensure the authenticity and the "behaviour" of the application once delivered. The dual concerns of authenticity and "behaviour" can typically be addressed using what is termed a validation credential in the Trusted Computing literature [44]. In this way, load-time metrics can be compared against known good values to assure customers (and merchants) that their application will perform as intended.

The issue of secure delivery of an application using Trusted Computing has previously been examined in the context of conditional access in mobile systems [19]. However, the methods of [19] are inadequate for our purposes as we require the card issuer to be able to specify the state on the customer's platform to which the e-EMV application will be sealed. Sealed messages in the context of Trusted Computing are messages that are bound to a set of platform metrics (specified by one or more PCR values) and can only be opened when the platform is in a certain state. This is normally achieved with the TPM_Seal [47, pp.58] or the TPM_Sealx command [47, pp.78]. However, the TPM_Seal and TPM_Sealx commands only allow messages to be sealed locally to a customer's platform. Instead we require a message to be sealed by the customer's card issuer on their servers, and sent to customer.

In our proposal, the delivery of an application to a customer's platform, as well as the corresponding requirement of securely storing the application upon receipt, is handled in three stages. In the first stage, the customer generates an asymmetric key pair, specifying security constraints for the private key. In the second stage, the customer sends the public key of this key pair to the card issuer. The card issuer will then encrypt the customer's e-EMV application with the customer's public key. In the third stage, the customer decrypts their e-EMV application (provided the security constraints specified for the private key are satisfied by the platform). The details of these stages are as follows.

In the first stage, the customer's TPM generates either a CMK or a non-migratable key pair. The set-up phase of this is illustrated in Algorithm

1.

Algorithm 1 Set-up Phase

```
1: $CI \rightarrow C : SealedState \parallel \mathrm{S}_{CI}(SealedState) \parallel \mathrm{CI}_{Cert}$
2: IC_Key := TPM_CreateWrapKey (TPM_Auth_Always, digestAtCreation, digestAtRelease)
3: TPM_Certify_Info := TPM_CertifyKey(IC_Key)
```

Notation: Here CI is the card issuer and C is the customer. $\mathrm{S}_X(\mathrm{Y})$ denotes a signature with the private key of entity X over data item Y and X_{cert} is the public key certificate for entity X.

In **step 1**, the customer's card issuer specifies a state to which it would like the customer to "seal" a private key.

In **step 2**, the customer creates a new asymmetric key pair, IC_Key. The customer specifies a password for the private component of this key, which will be required every time the key is loaded (TPM_Auth_Always). The customer also specifies the current platform state at the time the key is created (digestAtCreation) as well as the future platform state required for key usage (digestAtRelease). Here digestAtRelease is specified to be the *SealedState* provided by the card issuer.

In **step 3**, the newly generated IC_Key is certified by S_{AIK} using the TPM_CertifyKey command [47, pp.128]. Here the customer's TPM is effectively producing a signature over the public IC_Key using a signature key that the card issuer knows to be bound to a particular customer. In this setting IC_Key can provide the same level of assurance as results from the TPM_Seal command. This is because the "digestAtRelease" parameter is specified during the generation of the private component of the IC_Key. In this case specifying digestAtRelease is semantically equivalent to sealing.

Once the set-up stage is complete, the download of a customer's e-EMV card application can proceed as follows:

Algorithm 2 Downloading the e-EMV Application

```
1: $CI \rightarrow C : R_{CI} \parallel \mathrm{CI}_{Cert} \parallel$ [Platform Attestation]
2: $C \rightarrow CI : \mathrm{E}_{CI}(\mathrm{TPM\_Certify\_Info} \parallel \mathrm{ICC\_Key}_{pub} \parallel \mathrm{Platform\ Attestation} \parallel R_{CI} \parallel R_C \parallel S_{ICC\_Key_{priv}}(\mathrm{C} \parallel \mathrm{CI} \parallel R_{CI}))$
3: $CI \rightarrow C : R_C \parallel \mathrm{E}_C(\text{e-EMV application})$
```

Notation: Here R_x is a random number, $\mathrm{ICC\_Key}_{pub}$ is the public key of the ICC key generated in the set-up stage and $\mathrm{ICC\_Key}_{priv}$ is the private key of the ICC key generated in the set-up stage. $S_{ICC\_Key_{priv}}(\mathrm{Y})$ denotes a signature with the ICC_Key private key over data item Y, E_X(Z) denotes encryption with the public key of entity X over data item Z, and X_Cert(C_{pub}) is a public key certificate issued by entity X.

In **step 1**, the card issuer sends a challenge, its public key certificate and optionally its own platform state.

In **step 2**, the customer verifies this platform state (if present) as well as their card issuer's public key certificate. The customer returns the TPM_Certify_Info structure (generated in the set-up phase) which contains a hash of the public key of IC_Key and a description of the security constraints (of the private component) of IC_Key. In addition to this it sends the public component of IC_Key, signed platform metrics, responds to the challenge of the card issuer and sends a challenge of its own, all encrypted with the public key of the card issuer.

In **step 3**, the card issuer examines the received data. Provided the storage semantics for the private key (IC_Key) match the card issuer's security policy and the customer correctly responded to the earlier challenge, then the card issuer sends a reply to the customer's challenge and the customer's e-EMV application encrypted with the customer's public IC_Key.

The final stage of the protocol for secure application delivery is application retrieval. The basic algorithm is as follows:

Algorithm 3 Application Retrieval

```
if (Platform state == digestAtRelease) && (incomingAuth == objAuthData) then
    Retrieve application
else
    Fail
end if
```

Providing that the current platform state matches the requirements for the sealed data (as specified by the card issuer) and the incoming authorisation data matches the authorisation data set for the private key during the set-up phase, then the application is unsealed. If one or both of these two conditions is not met then the application should remain sealed until both conditions can be fulfilled. Following a successful delivery of the e-EMV application, the customer is now able to utilise their card on their TPM-enabled platform.

7 e-EMV in Operation

Once our e-EMV application (card) is launched into its own isolated VM, transaction processing proceeds as follows.

7.1 Customer to Merchant Interaction

Customer-to-merchant interaction in our e-EMV system uses the same transaction processing used for EMV at PoS terminals. Transaction flows for e-EMV transaction are mapped to EMV transaction flows as follows (see Figure 4):

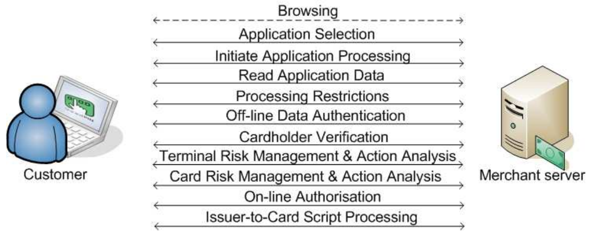
Browsing
Application Selection
Initiate Application Processing
Read Application Data
Processing Restrictions
Off-line Data Authentication
Cardholder Verification
Terminal Risk Management & Action Analysis
Card Risk Management & Action Analysis
On-line Authorisation
Issuer-to-Card Script Processing
Customer
Merchant server

Figure 4: e-EMV Customer-Merchant Interface

Application Selection: Typically, application selection occurs in response to a terminal-initiated command to select the appropriate EMV application from a multi-application card. In this case we assume a single application instance in which customer/merchant application matching is dependent on finding a suitable set of validation credentials supported by the platform or a reported set of one or more PCR values in the form of a platform attestation, representing a valid application execution.

Initiate Application Processing: The customer's application next commences application processing. In the physical world this would yield a response in which the card returns its application interchange profile and its application file locator. However, in this setting as the cards are virtualised and free from the constraints of typical ICC cards, all e-EMV cards would represent CDA capabilities. Additionally, as part of this step the merchant terminal plug-in provides to the e-EMV application any terminal-related information pertaining to the business environment at the point of service. An example of such information would be terminal capabilities or the country in which the terminal is operating.

Read Application Data: The steps involved in issuing one or more READ RECORD commands [16, pp.69] in this setting is relatively redundant as we assume the application is executing solely in main memory on a customer's platform.

Processing Restrictions: This mandatory step is performed by the merchant and doesn't require any direct interaction with the customer's execution environment. This step is primarily concerned with judging the compatibility of the e-EMV application with that of the terminal application. This process can be handled in an e-EMV environment through a process of reconciliation between terminal supported applications and customer supplied validation credentials/platform attestation.

Off-line Data Authentication: The three options for off-line data authentication available in EMV-compliant cards are SDA, DDA and CDA. These all involve a PoS terminal issuance of an INTERNAL AUTHENTICATE command [16, pp.65]. We achieve this functionality in our e-EMV application through the platform's attestation mechanism. We now discuss how this is achieved for each of the three options.

SDA: SDA is relatively simple process whereby a validation credential (generated by a card issuer) providing metrics for a correctly functioning e-EMV card application is compared against load-time PCR metrics reported in an attestation challenge to a merchant. An outline of this process is as follows (here M is the merchant and Req is a request for one or more PCR registers:

1: $M \rightarrow C$: M_{Cert} $\|$ S_M(Req) $\|$ Req $\|$ Platform Attestation
2: $C \rightarrow M$: C_{Cert} $\|$ E_M(P_{AIK} $\|$ Platform Attestation)

The merchant server requests one or more PCR values corresponding to the PCRs to which the e-EMV application is bound. In addition to this request the merchant attests its own platform metrics for examination by the customer's platform.

The customer's platform examines the attested merchant metrics and determines the suitability of the merchant for adherence to certain desirable policies, such as the PCI-DSS standards for card processing or Visa's PABP. If satisfied, a customer's platform agent culls its SML for the events responsible for generating the requested PCR values.

The TPM loads the AIK signing key using customer-provided authorisation data. The e-EMV application, using the now loaded AIK key, calls the TPM_Quote command [47, pp.157] to sign the requested PCR registers. This will either be the AIK for which a certificate was obtained during the e-EMV account activation stage (as per the Privacy CA model) or an AIK over which a DAA signature has been generated (as per the DAA model). This information is returned to the merchant server for verification.

The merchant examines the AIK credential, checks signatures[2] and compares a hash of the SML entries to the attested PCR values. If they match the merchant can be sure as to the current state of the VM in which the e-EMV application is executing. In effect, this procedure achieves the same function as SDA does in standard EMV.

DDA: DDA requires a slight modification to the SDA mechanism as outlined above. During the operation of TPM_Quote in which the PCR registers are signed, the customer incorporates a merchant-supplied 160-bit random challenge to the attestation. This challenge takes the place of the operand *externalData* which is of type TPM_NONCE [46, pp.27] in the PCR attestation process.

CDA: In the EMV PoS environment this would consist of a ICC private key signature over the dynamic application data of which an AC (specifically a TC or an ARQC) and a random challenge (as per DDA) are integral parts. Replicating this in e-EMV environment is not substantially more complicated than either the SDA or DDA approaches. The e-EMV application carries out the AC generation process. The AC, along with its data output (TC/ARQC) can be integrated into a PCR register. The actual examinable output, that is the TC or ARCQ, is then correspondingly appended to the SML and a signature is generated over the representative PCR values. In this way the merchant, upon receipt of the attestation bundle can examine the SML for the inclusion of an AC and verify its veracity through a signature verification.

Card-holder Verification: In the typical PoS EMV operation, this step allows the terminal to verify the authenticity of a cardholder. This authentication is based on the card and terminal both supporting a particular Card-holder Verification Method (CVM). In addition to the CVM being used to verify a customer, some CVMs, particularly PIN authentications, are used as a means of authorising transactions.

Through the use of Trusted Computing functionality we can make PIN authentication and authorisation intrinsic to transactions. This is achieved through the TPM key authorisation mechanism, whereby certain keys require 20 bytes of authorisation data to be supplied prior to being loaded. The AIK key's operation requires authorisation data to be securely communicated to a TPM. In our architecture, this authentication data will be communicated over a secure Object-Independent Authorisation Protocol (OIAP) session [45, pp.62]. The use of an OIAP session allows authorisation information to be sent TPM without revealing the data on the channel over which it is sent.

[2] There may be a situation in which a merchant is unfamiliar with certificate authority referred to in an ICC public key certificate. However, provided each card issuer is itself a link in a chain traversing back to globally recognised root CA, for example Visa or Mastercard, then this verification can be as a result of certificate chain traversal.

The availability of such a secure channel is an adjunct to our assumption of a Trusted Path provided by the OS.

The ability to actually use a key as part of a transaction demonstrates to a merchant that a CVM has occurred successfully. By verifying the state of the platform, a merchant can be assured that only a valid customer would be capable of using the AIK/DAA key generated in the account activation stage.

To prevent malware from launching a dictionary attack against key authorization data, it is important that the TPM provide some form of resiliency to such an attack. The current iteration of the TPM specifications [45] detail an example mechanism where a count of failed authorization attempts is recorded. If this count exceeds a threshold, the TPM is locked and remains non-responsive to further requests for a predetermined time-out period. However, at present, the implementation of a dictionary attack resistant authorisation mechanism is vendor-specific and not always implemented securely [32].

Terminal Risk Management and Action Analysis: The purpose of terminal risk management and terminal action analysis is to protect the issuer, acquirer and payment system from fraud. The existing variety of measures used by EMV in terminal risk management, such as floor limits, random transaction selection and velocity checking can all be replicated in the merchant terminal plug-in in our e-EMV architecture.

Card Risk Management and Action Analysis: Details of card risk management are proprietary to card issuers and are outside the scope of the EMV specification and as such would remain proprietary to individual card issuers within our system.

On-line Authorisation: During EMV transaction processing the merchant terminal may decide to proceed with an on-line check; this again can be replicated in our e-EMV environment.

Issuer Script Processing: To allow updates to EMV cards in the field, the card issuer can return scripts via the terminal to the ICC for processing. These scripts are not necessarily relevant to the current transaction and may be used to update applications during the utilisation phase of the ICC's lifecycle, or to transition the card into a blocked or unblocked state. Updates to e-EMV cards can be performed in a similar manner. An issuer can send update scripts via the merchant plug-in to the customer's e-EMV card. The issuer can then distribute new state measurements for updated e-EMV cards to merchants via new validation credentials. Merchants should only allow e-EMV cards whose attestation matches the values contained in the latest validation credential to perform e-EMV transactions. Card blocking scripts are a more difficult EMV feature to replicate in an e-EMV setting. It may be illegal in certain jurisdictions for a card issuer to force an update on a customer's platform without gaining customer consent.

7.2 Payment

As we saw in Section 3.1, in EMV, an ICC uses session keys derived from the ICC Master Key to protect the transaction messages. In our e-EMV architecture, in conjunction with the information typically sent in the AC message, the e-EMV application sends the current platform state as witnessed by its issuer-validated signing key. After examining the AC message as well as the supplied state, the verifier (a merchant or a card issuer) can make a decision as to whether or not to proceed with a transaction. In the instance where an AC is an ARQC, as mentioned in Section 5.2, the TLV encoding scheme used in EMV allows the issuer to ignore extraneous information if they so choose. The card issuer can then respond with an ARPC indicating whether the transaction should be approved or declined, in which case a TC (which can be signed, see CDA in Off-line Data Authentication) or an AAC will be generated by the customer's platform. The transition to Internet-based communications in e-EMV requires that ACs be protected in transit between the customer and merchant. Such protection can be achieved by establishing a TPM-centric SSL tunnel between customer and merchant, as per [8].

7.3 Migration

Enabling the migration of an e-EMV card from one Trusted Platform to another would be useful in our architecture. We could achieve such a feature by using the TPM's certifiable migration functionality. If the customer creates a CMK during the application delivery process (see Section 6.2), a customer may later migrate their application bundle to another TPM-enabled platform. However, as neither DAA secrets nor AIK private keys obtained in the enrollment phase are migratable from a TPM, the customer would need to rerun the account activation phase (see Section 6.1) in order for their card to be usable on the new platform. In this instance, the cost of providing additional infrastructural elements to support a trusted migration service [42] may be somewhat prohibitive for issuers, and it may just be simpler in practice for customers to enroll their new platform with their card issuer from scratch.

8 Security Analysis

The semantics of trust enforced by Trusted Computing functionality enables both parties, the merchant and the customer, to obtain certain guarantees that were hitherto unrealizable in past proposals [37, 49]. Both merchant and server can be sure as to the integrity of their communicating peer's platform, i.e. that each peer will behave in the expected manner (in this case, adhere to, and faithfully adduce state characteristics corresponding to legitimate transaction states). It is important to point out that we do not expect the customer to be able to recognize or validate software states within our system. This function can be fulfilled by the application software itself and should be reported to the customer in a way that is understandable. This section analyses the effectiveness of our e-EMV architecture as a means of securing electronic payments. This is achieved by examining how well it satisfies the security requirements of Herreweghen and Wille [21] described in Section 2.

Mutual authentication of customer and merchant: In our architecture both customers and merchants are authenticated via their card issuer/acquirer supplied credentials, providing a much stronger form of authentication than that currently employed for CNP payments. Balfe *et al.* [8] have shown how such credentials can be used to authenticate the establishment of SSL/TLS sessions, thus providing an additional layer of protection for the transport of e-EMV transaction messages over the Internet. Their proposal can also be adopted here.

An added benefit of our approach is allowing the customer to examine a merchant's platform state prior to transaction authorisation. This enables the customer to satisfy themselves that the merchant will behave in a manner that will protect their sensitive card data. Such a feature is lacking with current PoS transactions, a fact which is now allegedly being exploited by criminal gangs [48].

Transaction authorisation & payment guarantee for merchant: In our architecture, the use of the private transaction authorising key is contingent on a particular platform state being present on the customer's platform. Much like PIN entry at a PoS, the completion of a transaction is conditioned on the input of correct authorisation data, ensuring the physical presence of the cardholder in the remote transaction. Additionally, as part of the transaction authorisation process, a customer's platform must attest to the VM in which our e-EMV application is executing. Any divergence from intended operating state (due to unwanted memory resident applications) will be picked up in the attestation, allowing merchant risk management routines to terminate the transaction.

Assuming a transaction goes ahead, payment guarantee for the merchant is provided by CDA. CDA provides a signature over an e-EMV card's Transaction Certificate, providing the merchant with non-repudiable evidence of payment. Cardholder non-repudiation is dependent on the degree to which the cardholder's private signing key is securely generated and stored in the cardholder's Trusted Platform.

Customer privacy and anonymity: Customer privacy is somewhat problematic in any CNP transaction as the customer will typically provide copious amounts of personally identifiable information, such as name and billing address. However, our DAA approach does provide a degree of pseudonymity for the actual transaction. Provided the customer selects a fresh DAA identity for each transaction, a customer's different transactions should be unlinkable by a merchant.

9 Conclusions

The use of the Internet as an avenue for electronic commerce, in the form of CNP transactions, has seen something of an explosion in recent years. However, CNP transactions are currently far from secure. This paper proposed a new security architecture for securing CNP transactions. By creating software-based EMV cards running on Trusted Platforms, our e-EMV proposal replicates many of the features of standard EMV-compliant cards for use in CNP transactions. Through our account activation and secure application delivery procedures, we established cards can be remotely provisioned within our system. We showed how card ownership can be demonstrated by the customer through the use of an OIAP session enabling secure PIN entry. We also showed how EMV transaction messages can be mapped to e-EMV transaction messages. We demonstrated how EMV keys can be generated and bound to a particular TPM-enabled platform. Through these various measures, we can achieve a significant improvement in the level of security afforded to CNP transactions.

Our work raises a number of areas for further research. In this paper, we have focussed on describing architectural and technical aspects of our e-EMV proposal. Our future work will examine security and system management issues. A prototyping activity based on the OpenTC framework[3] with SVM [4], L4 μ-Kernel[4] and Xen [9] support is also likely to be useful in terms of revealing unforseen practical issues, operational problems, and the like. At present, our approach is reliant on MVMMs and Trusted Computing

[3] http://www.opentc.net/

[4] http://os.inf.tu-dresden.de/L4/

augmented processors being present in commodity platforms. Unfortunately, such support is not widely available today. A more immediate avenue for adoption would be Trusted Computing enhanced mobile phones [43] which do not require MVMM support. However, at present only preliminary details of the TCG's trusted mobile architecture are available [43]. We believe that Trusted Computing can provide an enhanced level of security over EMV's deployment at PoS. However, given the perpetual increase in attacks (both in terms of number and sophistication) targeting end-user systems, we see an increased role for both terminal and card risk management routines to control transaction risk in our e-EMV architecture.

Whilst outside of the immediate scope of this paper, in developing our architecture we note two weaknesses in the current attestation mechanisms adopted by the TCG. Firstly, as noted in [20], the current TCG attestation is static, inexpressive and has poor handling of patches and upgrades to system software. Alternative approaches to attestation have been proposed in [33, 38, 20]. Secondly, the TCG attestation mechanism only concerns itself with load-time measurements of applications. It is very difficult to obtain any guarantees of the run-time behaviour of the application, and as such, applications may suffer from a time-of-check-to-time-of-use problem. Recent approaches that attempt to address this issue have been proposed in [26, 28, 36, 38]. Investigating these proposals in the context of e-EMV will be of interest. In the absence of more expressive attestations we must rely on the properties of the MVMM to ensure that our executing e-EMV application cannot be interfered with by outside applications.

We further note the possibility of extending our system by exploiting additional features of the DAA protocols to support pseudonymous payment cards. In such an extension, we envisage the information identifying individuals being removed from the merchants' view of transactions, with acquirers still being able to obtain the necessary payment guarantees. We leave the details of such a system to the full paper.

References

[1] M. Abadi and T. Wobber. A logical account of NGSCB. In David de Frutos-Escrig and Manuel Nú nez, editors, *Proceedings of the 24th International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2004)*, volume 3235 of *LNCS*, pages 1–12. Springer Verlag, 2004.

[2] M. Al-Meaither and C. J. Mitchell. Extending EMV to support Murabaha transactions. In *Proceedings of the 7th Nordic Workshop on Secure IT Systems (NordSec 2007)*, pages 95–108, Gjovik University College, Norway, October 2003. Department of Telematics, NTNU, Trondheim, Norway.

[3] A. Alsaid and C. J. Mitchell. Preventing phishing attacks using trusted computing technology. In *Proceedings of the 6th International Network Conference (INC 2006)*, pages 221–228, July 2006.

[4] AMD. *AMD64 architecture programmer's manual: Volume 2: System programming*, AMD Publication no. 24594 rev. 3.11 edition, May 2006.

[5] APACS. Card fraud the facts 2006. `http://www.apacs.org.uk/resources_publications/documents/FraudtheFacts2006.pdf`, April 2006.

[6] APACS. Card fraud losses continue to fall. `http://www.apacs.org.uk/media_centre/press/07_14_03.html`, March 2007.

[7] B. Balacheff, D. Chan, L. Chen, S. Pearson, and G. Proudler. Securing intelligent adjuncts using trusted computing platform technology. In *Proceedings of the 4th working Smart Card Research and Advanced Applications (CARDIS 2001)*, pages 177–195. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[8] S. Balfe, A.D. Lakhani, and K.G. Paterson. Securing peer-to-peer networks using trusted computing. In C.J. Mitchell, editor, *Trusted Computing*, pages 271–298. IEE Press, 2005.

[9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauery, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003)*, pages 164–177, The Sagamore, Bolton Landing (Lake George), New York, 19–22 October 2003. ACM Press, Bolton Landing, New York, USA.

[10] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and Communications Security (CCS 2004)*, pages 132–145, Washington DC, USA, 2004. ACM Press, New York, NY, USA.

[11] PCI Security Standards Council. Payment Card Industry Data Security Standard – Version 1.1. https://www.pcisecuritystandards.org/tech/download_the_pci_dss.htm, 2006.

[12] R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In *Proceedings of the SIGCHI Conference on Human factors in computing systems (CHI 2006)*, pages 581–590, Montreal, Qubec, Canada, 2006. ACM Press, New York, NY, USA.

[13] EMVCo. *Book 3 - Application Specification*, 4.0 edition, December 2000.

[14] EMVCo. *Book 1 - Application independent ICC to Terminal Interface requirements*, 4.1 edition, May 2004.

[15] EMVCo. *Book 2 - Security and Key Management*, 4.1 edition, May 2004.

[16] EMVCo. *Book 3 - Application Specification*, 4.1 edition, May 2004.

[17] EMVCo. *Book 4 - Cardholder, Attendant, and Acquirer Interface Requirements*, 4.1 edition, June 2004.

[18] S. Gajek, A-R. Sadeghi, C. Stüble, and M. Winandy. Compartmented security for browsers–or how to thwart a phisher with trusted computing. *ARES*, 0:120–127, 2007.

[19] E. Gallery and A. Tomlinson. Conditional access in mobile systems: Securing the application. In *First International Conference on Distributed Frameworks for Multimedia Applications (DFMA 2005)*, pages 190–197. IEEE, 2005.

[20] V. Haldar, D. Chandra, and M. Franz. Semantic remote attestation: A virtual machine directed approach to trusted computing. In *USENIX Virtual Machine Research and Technology Symposium*, pages 19–41. USENIX, May 2004.

[21] E.V. Herreweghen and U. Wille. Risks and potentials of using EMV for internet payments. In *Proceedings of the 1st USENIX Workshop on Smartcard Technology*, pages 163–174. USENIX, May 1999.

[22] IBM-Global-Services. IBM Global Business Security Index Report, February 2005.

[23] Intel-Corporation. *LaGrande Technology Preliminary Architecture Specification*, intel publication no. d52212 edition, May 2006.

[24] C. Jackson, D. Boneh, and J. Mitchell. Attack of the transaction generators. `http://crypto.stanford.edu/SpyBlock/spyblock.pdf`.

[25] C. Jackson, D. Boneh, and J. Mitchell. Spyware resistant web authentication using virtual machines. `http://crypto.stanford.edu/antiphishing/spyblock.pdf`.

[26] T. Jaeger, R. Sailer, and U. Shankar. PRIMA: policy-reduced integrity measurement architecture. In *Proceedings of the 11th ACM Symposium on Access Control Models And Technologies (SACMAT 2006)*, pages 19–28, Lake Tahoe, California, USA, 2006. ACM Press, New York, NY, USA.

[27] V. Khu-Smith and C.J. Mitchell. Using EMV Cards to Protect E-commerce Transactions. In *Proceedings of the 3rd International Conference on E-Commerce and Web Technologies (EC-WEB 2002)*, volume 2455, pages 388–399. Springer-Verlag, London, UK, January 2002.

[28] J.M. McCune, B. Parno, A. Perrig, M.K. Reiter, and A. Seshadri. Minimal TCB Code Execution. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 267–272. IEEE Computer Society, Washington, DC, USA, 2007.

[29] P. Meadowcroft. Combating card fraud. `http://www.scmagazine.com/uk/news/article/459478/combating+card+fraud/`, January 2005.

[30] C.J. Mitchell, editor. *Trusted Computing.* IEE Professional Applications of Computing Series 6. The Institute of Electrical Engineers (IEE), London, UK, April 2005.

[31] C. Radu. *Implementing Electronic Card Payment Systems.* Artech House, Inc., Norwood, MA, USA, 2002.

[32] A-R. Sadeghi, M. Selhorst, C. Stüble, C. Wachsmann, and M. Winandy. TCG inside?: a note on TPM specification compliance. In *Proceedings of the 1st ACM workshop on Scalable trusted computing (STC 2006)*, pages 47–56, Alexandria, Virginia, USA, 2006. ACM Press, New York, NY, USA.

[33] A-R. Sadeghi and C. Stüble. Property-based attestation for computing platforms: caring about properties, not mechanisms. In *Proceedings of the 2004 workshop on new security paradigms (NSPW 2004)*, pages 67–77, Nova Scotia, Canada, 2004. ACM Press, New York, NY, USA.

[34] A-R. Sadeghi, C. Stüble, and N. Pohlmann. European Multilateral Secure Computing Base: Open Trusted Computing for You and Me. http://www.prosec.rub.de/Publications/SaStPo2004Web.pdf, 2004.

[35] U.S. Securities and Exchange Commission. Form 10-K – The TJX Companies, INC. http://www.sec.gov/Archives/edgar/data/109198/000095013507001906/b64407tje10vk.htm, 2007.

[36] A. Seshadri, M. Luk, N. Qu, and A. Perrig. SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes. In *Proceedings of 21st ACM SIGOPS symposium on Operating Systems Principles (SOSP 2007)*, pages 335–350, Stevenson, Washington, USA, 2007. ACM Press, New York, NY, USA.

[37] SETCo. SET Secure Electronic Transaction 1.0 specification — the formal protocol definition. http://www.cl.cam.ac.uk/research/security/resources/SET/, May 1997.

[38] E. Shi, A. Perrig, and L.V. Doorn. BIND: A Fine-Grained Attestation Service for Secure Distributed Systems. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 154–168. IEEE Computer Society, Washington, DC, USA, 2005.

[39] TCG. *TCG PC Specific Implementation Specification*, 2003. https://www.trustedcomputinggroup.org/downloads/specifications.

[40] TCG. *TCG Specification Architecture Overview*, 1.2 edition, 2004. https://www.trustedcomputinggroup.org/downloads/specifications.

[41] TCG. Trusted computing: Opportunities and challenges. https://www.trustedcomputinggroup.org/downloads/tcgpresentations/, 2004.

[42] TCG. *Interoperability Specification for Backup and Migration Services*, 1.0 revision 1.0 edition, 2005. https://www.trustedcomputinggroup.org/specs/IWG/.

[43] TCG. *TCG Mobile Trusted Module Specification*, .09 draft edition, 2006. https://www.trustedcomputinggroup.org/specs/mobilephone/.

[44] TCG. *TCG Specification Architecture Overview Revision 1.2*, 1.2 revision 93 edition, 2006. https://www.trustedcomputinggroup.org/downloads/specifications.

[45] TCG. *TPM Main: Part 1 Design Principles*, 1.2 revision 93 edition, 2006. https://www.trustedcomputinggroup.org/downloads/specifications.

[46] TCG. *TPM Main: Part 2 Structures of the TPM*, 1.2 revision 93 edition, 2006. https://www.trustedcomputinggroup.org/downloads/specifications.

[47] TCG. *TPM Main: Part 3 Commands*, 1.2 revision 93 edition, 2006. https://www.trustedcomputinggroup.org/downloads/specifications.

[48] The Sunday Times. Don't use cards at petrol stations. http://business.timesonline.co.uk/, Febuary 18 2007.

[49] Visa. 3-D Secure^{TM} Protocol Specification: System Overview. http://international.visa.com/fb/paytech/secure/main.jsp, May 2003.

[50] Visa. Cardholder information security program – list of validated payment applications. http://usa.visa.com/merchants/risk_management/cisp_payment_applications.html, October 2007.

[51] Visa. Cardholder information security program bulletin 102307–visa announces new payment application security mandates. http://usa.visa.com/merchants/risk_management/cisp_payment_applications.html, October 2007.