A Novel Consumer-Centric Card Management Architecture and Potential Security Issues

Raja Naeem Akram and Konstantinos Markantonakis

Information Security Group, Smart Card Centre, Royal Holloway, University of London, Egham, United Kingdom.
Email: r.n.akram@rhul.ac.uk

Damien Sauveron

XLIM (UMR CNRS 7252 / Université de Limoges) Département Mathématiques Informatique, Limoges, France.

Abstract

Multi-application smart card technology has gained momentum due to the Near Field Communication (NFC) and smart phone revolution. Enabling multiple applications from different application providers on a single smart card is not a new concept. Multi-application smart cards have been around since the late 1990s; however, uptake was severely limited. NFC has recently reinvigorated the multi-application initiative and this time around a number of innovative deployment models are proposed. Such models include Trusted Service Manager (TSM), User Centric Smart Card Ownership Model (UCOM) and GlobalPlatform Consumer-Centric Model (GP-CCM). In this paper, we discuss two of the most widely accepted and deployed smart card management architectures in the smart card industry: GlobalPlatform and Multos. We explain how these architectures do not fully comply with the UCOM and GP-CCM. We then describe our novel flexible consumer-centric card management architecture designed specifically for the UCOM and GP-CCM frameworks, along with ways of integrating the TSM model into the proposed card management architecture. Finally, we discuss four new security issues inherent to any architecture in this context along with the countermeasures for our proposed architecture.

Keywords: Smart Card, GlobalPlatform, Java Card, Multos, User Centric Smart Cards, Card Management Architecture.

1. Introduction

Existing multi-application smart card platforms (e.g. Java [39], Multos [35]) support the installation of applications remotely (post-issuance). Standardisation efforts that enable an application to be managed remotely — for example, GlobalPlatform [21] have been effective in the Issuer Centric Smart Card Ownership Model (ICOM) field [4]. The advent of NFC technology and the growing convergence of different services on mobile phones have prompted GlobalPlatform and the GSMA[1] to propose a new application management architecture (e.g. TSM) [24, 26, 20, 30]. Similarly, Multos has a strong card and application management architecture that is heavily issuer-centric and it can be argued that it can easily be adapted to the Trusted Service Manager (TSM) architecture. The GlobalPlatform and Multos cards along with their associated application management architectures provide two contrasting views of the smart card industry. We limit our analysis of traditional card management architectures to these two examples. The Java Card does not have any associated management architecture and in most commercial deployments it is coupled with the GlobalPlatform management architecture. In the ICOM, the issued smart cards are under the complete control of the card issuer. By contrast, in the User Centric Smart Card Ownership Model (UCOM) [4] the consumers (end-users) get "freedom of choice" which allows them to install or delete any application as they desire from their smart cards. The UCOM supported smart cards can hold multiple heterogeneous applications from various organisations. The only limitation on the number of applications a UCOM supported smart can have is based on its store capacity. The GlobalPlatform Consumer-Centric Model (GP-CCM) [27] provides consumers the ability to choose the applications they want. Although the finer details for the GP-CCM are still to be articulated, we consider that the overarching aim of both UCOM and GP-CCM are similar. In this paper, the expression Consumer-Centric Model is used for conciseness to indicate both UCOM and GP-CCM. As the card management architecture in the Consumer-Centric Model has to consider the contrasting needs of both the administrative authority (i.e. TSM) [7] and the consumers (end-users), it has to remain flexible. It must determine the ownership requirements of each of

[1]The GSM Association (GSMA) is an association that represents the interests of mobile operators worldwide, along with developing and prompting the Global System for Mobile Communication (GSM) specification.

these entities and then articulate how a Consumer-Centric Model framework will manage them. In addition, the management architecture proposed in this paper deals with application issuance (lease), application domain provision on the smart card, installation, deletion, and application/domain management. Furthermore, this paper an arhitecture for the proposed application download/installation protocols [8, 7, 9]. As the Consumer-Centric card management architecture brings forward conflicting views smart card management architectures, it also highlights new security issues. These issues relate to the card and its rightful owner. They include simulator problems, user ownership issues, parasite application problems, and platform insider attacker problems.

1.1. Contributions

The salient contributions of this paper are to:

1. Describe and compare traditional application management architectures (Multos and GlobalPlatform) and justify why they might not be considered as adequate for the Consumer-Centric Model
2. Specify a flexible card management architecture for the Consumer-Centric Model
3. Provide a flexible card management architecture for the integrated TSM and Consumer-Centric Models
4. Discuss the security issues and related countermeasures for the proposed architecture

1.2. Structure of the paper

The GlobalPlatform card management architecture is discussed in section 2, followed by the Multos card management architecture in section 3. The proposed architecture of the Consumer-Centric card management is described in section 4, along with different types of relationships between a user and a Service Provider[2] (SP). In section 5, we discuss the issues that are inherent to any architecture in this context along with providing related countermeasures for our proposed architecture.

2. GlobalPlatform Card Management Architecture

In this section we discuss the GlobalPlatform card management architecture along with the mechanics by which it supports TSM-based card management.

2.1. Architecture Overview

The GlobalPlatform card security requirement specification [21] specifies eight entities (excluding the smart card and the cardholder) that perform various tasks in the overall card management architecture. The overall architecture is depicted in figure 1, which is a simplistic representation of the architecture described in [21]. The shaded entities in figure 1 have different titles and roles, but together they form the card issuer. The term "card issuer" as defined by GlobalPlatform in [21] is restrictive, so that issuers only have the responsibility to acquire the smart cards, set a security policy, and issue cards to individual cardholders. The card administrator is responsible for managing the cards once they are issued to individual customers. If application provider would like to issue its applications, these must first be verified by the verification authority. The verification authority performs an off-card application code verification to ascertain whether the given code conforms to the security policy set by the card issuer. Once the verification is performed, the application provider requests the controlling authority to give permission to load the application. The controlling authority checks the verification authority's verification and issues the permission to load the application. If the application is going to be loaded at the pre-issuance stage then the domain keys and data will be sent to the card issuer [22] through the card enabler. Otherwise, the domain keys and data will be sent to the application loader. In figure 1, we opt for the pre-issuance model. Finally, the application provider will send its application, keys and application personalisation data to the application loader, which will load them onto the smart cards of individual customers. The keys in aforementioned message, used to secure the application download to a smart card, are security domain keys. In figure 1 the security domain keys that are used by the application provider to manage its domain are loaded onto the smart card through the card enabler (i.e. card issuer) [23]. However, a later addendum to the GlobalPlatform card specification [23] permitted the generation and loading of keys without the active involvement of a card issuer [25]. This extends the role of the Controlling Authority (GP-CA)[3] by provisioning an on-card controlling entity (i.e. Controlling Authority Security Domain: CASD) that will be

[2] A Service Provider (SP) is an entity which is also referred as Application Provider (AP) that develops smart card applications in order to facilitate its customers to securely access their services. The terms SP and AP are used interchangeably hereafter.

[3] A Controlling Authority is an off-card entity (e.g. card issuer) that has a security domain on the GlobalPlatform smart card. Its role as defined in the GlobalPlatform card specification [23] is to enforce the card issuer's security policy. In the GlobalPlatform card specification, the GP-CA has the power to sanction or evict any application from a smart card.

responsible for generating and/or loading the application provider's cryptographic keys. The GlobalPlatform specification supports two models: the push model in which the cryptographic keys are sent to the CASD by the application provider, and the pull model that generates the cryptographic keys on the card and then sends them to the application provider. The GlobalPlatform card specification [23] and its amendment [25] provides a secure and reliable way to load the application provider's keys onto a smart card in a confidential way. In all fairness, the CASD proposal is not that different to the original GlobalPlatform card specification (including the GlobalPlatform card specification amendment A [25]) where keys could be loaded by the card enabler at pre-issuance. Note that the CASD (post-issuance loading) is under the control of the GP-CA. Both roles, card enabler and GP-CA, are predominantly played by the card issuer. Nevertheless, the amendment provides a base to accommodate the TSM architecture.

2.2. Support for Trusted Service Manager Architecture

To provide a standardised architecture and facilitate the adoption of NFC-enabled mobile phones for various services, GlobalPlatform proposed an architecture for the management of secure elements (these secure elements can be smart cards in NFC mobile phones [24]). The TSM role specified by GlobalPlatform [24, 26] includes managing the relationships between various actors in the smart card ecosystem. It does not handle any key management or provide any trusted services [24]. GlobalPlatform proposes a new entity termed the Confidential Key Loading Authority (CKLA) that will loads the initial key set onto the smart cards in a confidential way. It does not specify who will take the role of the CKLA. Additionally, it breaks down the role of GP-CA so it can be performed by two different (independent) actors. This role includes managing the CKLA and the Mandated Data Authentication Pattern (Mandated DAP) Authority. The CKLA will facilitate the generation and loading of keys using an Over-the-Air (OTA) framework. The DAP allows the application provider to sign its applications before they are loaded onto the smart card. The Mandated DAP Authority will verify the signature and notify the application provider. Note that in any architecture, whether it is pre-issuance or post-issuance application loading in the GlobalPlatform card specification [23] or application loading via OTA in the NFC mobile phone [24], the loading of cryptographic keys is dependent on either the GP-CA or the CKLA entities. These entities should be trusted by the application provider and their aim is to provide the key material for confidential application loading and management to the respective application providers. Therefore, such entities (e.g. GP-CA or CKLA) which in most cases belong to an off-card actor (i.e. card issuer), cannot be accommodated in the UCOM proposal.

3. Multos Card Management Architecture

In this section, we discuss the Multos card management architecture and we explore how it can be adapted into the TSM architecture for NFC mobile phones.

3.1. Architecture Overview

The card management architecture for Multos is more straightforward than GlobalPlatform (section 2.1). An overview of the Multos card management architecture is illustrated in figure 2 and discussed below. The shaded entities in figure 2 represent various roles, but traditionally they reside with a single entity, for example the card issuer [36, 37]. An application provider will generate a signature key pair and application code; the private key of the application provider along with the application code is securely communicated to the application load unit generator. The signature key is used to generate a cryptographically protected application load unit (i.e. downloadable application). The application provider will also send its signature verification key and application header to the card issuer, which forwards it to the Multos Certification Authority (M-CA). The application header is a data structure that contains information regarding the application, which includes the application identity, hash of the application code, code and data size [36]. The M-CA can be either the card manufacturer, or an authorised entity of the Multos consortium [35]. The role of the M-CA is to issue an application load or delete certificate to the card issuer for the specific application. In addition, the M-CA also provides the list of public keys for individual Multos cards that have been issued by the card issuer. This list of public keys remains with the application load unit generator as the keys are used to encrypt individual applications. This transfer of public keys is marked as user personalisation data in figure 2. The application load unit generator will create application load units for individual smart cards only if the load unit has to be encrypted. Finally, the load unit include the application code, digital signature on the application using application provider's signature key. Optionally, the whole load unit may be encryptec by the public key of the smart card. The application load facility will have the application load units and associated M-CA's certificates that will can be use for loading the application onto individual smart cards. It is evident from figure 2 that application providers have to communicate their application (in plaintext) and their private keys to the application load unit generator (i.e. card issuer). Furthermore, the application management tasks (i.e. installation, deletion, and updating etc.) have to be performed through the card issuer and/or the M-CA. Unlike the GlobalPlatform,

Multos specifications do not provide independent application management architecture. The process of deleting a Multos application is similar to the Multos application installation except that there is no need to generate an application load unit but rather an application delete certificate from the M-CA.

3.2. Support for Trusted Service Manager Architecture

To the best of our knowledge, there are no official proposals on how to incorporate Multos into a proposed TSM architecture for NFC mobile phones. However, in a centralised environment a TSM can take the role of the M-CA and application load unit generator.

4. Proposed Flexible Consumer-Centric Card Management Architecture

In our proposal, the card management architecture is divided into two categories depending on whether the card is under administrative control or not. Therefore, these two categories are referred to as administrative and user card-management, corresponding to the Coopetitive Architecture for Smart Cards (CASC) architecture [7] and the UCOM architecture [4], respectively.

4.1. Administrative Card Management Architecture

In the administrative card management architecture, a smart card is under the shared ownership of an administrative authority and the respective cardholder. The architecture is shown in figure 3 and the dotted lines in this figure represent optional messages. The card manufacturer gets its product evaluated by a third party that issues an evaluation certificate. The smart cards are then acquired by the administrative authority that takes administrative control and issues the cards to individual cardholders. The cardholder then obtains the card ownership, under certain terms and conditions. The cardholder has to register with the relevant SP to gain access to the corresponding application. The registration process generates customer credentials that are issued by the SP and used by cardholders to download the application(s) onto their smart cards. The cardholder then provides these credentials to the smart card along with the details of the SP's application server [2]. Before the SP leases its application, it requests the smart card to provide a security assurance, which is furnished by providing the evaluation certificate and a validation proof [3]. The SP then sends the application identity to the smart card, which will check whether the application belongs to the administrative authority's partner: partners are referred to as syndicated members. The smart card has a list of syndicated members that is provided by the administrative authority. If the SP is registered as a syndicated member of the administrative authority, it will then reveal the identity of the administrative authority. Under the scenario in which the SP is a syndicated member, the SP will then contact the administrative authority to authorise the installation. Upon successful authorisation, the application is leased to the smart card and installed in the administrative authority's space [7]. If the SP is not a syndicated member, then the application is installed under the authorisation of the cardholder and she might be charged for it, which is represented by a "use charge" message sent from the smart card to the administrative authority. Subsequently, the administrative authority processes the request and issues an application "installation authorisation" message to the smart card. On receipt of this message, the smart card will allow the application to execute. It will generate an "application download certificate" that acts as a contract between the smart card and the SP. The contract signifies that the application was downloaded succesfully onto the smart card and it is enabled to communicate with the SP. The administrative card management architecture can easily be adapted into the TSM architecture by replacing the administrative authority with the TSM. However, the shared ownership principle has to be accommodated by the TSM architecture to comply with the CASC. In paper [7], the Application Acquisition and Contractual Agreement Protocol supports the basic design of the administrative management architecture.

4.2. User Card Management Architecture

In the user card management architecture there is no administrative authority. The user card management architecture is depicted in figure 4 and is described below. The smart card establishes a secure communication channel with the SP and this is in order to provide smart card security and reliability assurance to the SP, facilitate the generation of domain management credentials, and download the application. In the Consumer-Centric Model, whether it is under administrative or user management, each SP gets its own domain. The SP's domain management credentials are mutually generated by the SP and the smart card without involving any off-card entity (e.g. including the card manufacturer). Applications are downloaded directly to the SP's domain using the cryptographic keys mutually generated by the smart card and the SP, in contrast to GlobalPlatform that uses either a push or pull model for key sharing (section 2.1), or Multos that require an application provider to reveal its application code and signature key to Application Load Unit Generator . Note that the two variants of the Secure and Trusted Channel Protocol (STCP), referred to as STCP_{SP} [9] and STCP_{SC} [10] support the user card management architecture.

4.3. Types of Application Leases

An application lease refers to the issuance of an application to the requesting smart card under some terms and conditions that are stipulated by the Application Lease Policy [2]. In this section, we discuss the various types of application leases that an SP can issue.

1. Card Bound Application Lease: In this lease, an SP issues its application to a specific smart card and lease is only bound to the particular smart card. Therefore, an SP will only issue one lease per user, which can be associated with any of her smart cards; examples of such a lease may be credit card and (U)SIM card applications.

2. User Bound Application Lease: This lease is bound to the user, not to her smart card. She can install the given application on any number of her smart cards. Examples of such a lease may be Internet Identity applications [19, 14].

3. Open Application Lease: The open application lease does not bind the lease to either a user or a smart card. Any smart card, and any user can download this application. Examples of such applications may be pre-paid mobile telephone cards, pre-paid calling cards, hotel room access cards, and transport cards. One thing to note is that these examples are only valid if they do not require any user registration before and after application is issued.

4.4. Possible Relationships between a Cardholder and an SP

The lease issued to a cardholder discussed in the previous section would be based on a relationship that an SP has with a particular cardholder. In this section, we discuss various possible relationships that can exist between an SP and a cardholder.

1. Pre-Registration: This scenario deals with applications that are only issued to registered and pre-authorised customers. Such applications can be banking, health centres, identity, travel documents, and telecom (e.g. post-pay accounts) that require proof that the requesting user is actually the current owner of the smart card. This relationship is valid for the card- and user-bound application leases discussed in the previous section.

2. Post-Registration: The post-registration relationship allows a cardholder to download an application without being a registered customer. However, the application does not go into service unless the user registers herself with the application (or its respective SP). This type of relationship can be valid for all three types of application leases. There are two possible cases:

   a) the SP is only concerned with the security assurance and validation of the smart card platform and does not require user registration (anybody can download and use their application) or

   b) at least during the application lease process, the SP is not concerned with user registration. However, once the application is downloaded the SP can initiate the user registration process.

   The latter option is like a user registering for a service for the first time. Examples of applications that can be downloaded in this scenario include pre-paid telecom applications, transport, and hotel room access applications.

3. No-Registration: This option does not require any registration, before or after the application is issued. It is suitable for the open application lease category. Examples include hotel room access cards, fixed pre-paid calling cards, and pre-paid gift cards.

4.5. Application Installation

In this section, the processes that support the secure transmission and installation of an application are discussed. The installation process discussed in this section builds additional checks around the application installation protocols (discussed in [9, 7, 10]). The installation request will initiate the process of acquiring an application from an SP's application server and installing it on a smart card. The entire process can be divided into three sub-processes:

i) Downloading,

ii) Localisation, and

iii) Application Registration.

These sub-processes are explained below.

1. Downloading: The downloading of an application is initiated by the smart card, through a secure channel protocol [9, 10]. At the conclusion of the secure channel protocol, both entities generate a set of keys for application download and domain management. The smart card then generates an SP's domain, provided it has enough space to accommodate it. The SP and smart card will then start the application downloading process. The SP will first generate a signature on the application, then encrypt and MAC it before sending it to the smart card. The smart card checks the generated MAC, decrypts the application, and verifies the signature. A decrypted application is not a fully installed application — it is the equivalent of copying an application to a memory location. The next step is to verify whether the application complies with the smart card's operational and security policy. For this purpose an on-card byte code verification is performed [16], which is already mandated by the Java Card 3 [39]; this can be based on the well-defined on-card byte code verification proposals [15, 33, 16, 34]. Furthermore, additional runtime checks are performed that are discussed by [1]. The Consumer-Centric Model, especially the UCOM, does not mandate the security evaluation of an application. However, certain applications require evaluation due to government or industry regulations (e.g. EMV applications). In these cases, an SP's application(s) provides an evaluation certificate (e.g. Application Assurance Certificate: AAC). To verify the certificate the smart card would have to calculate the hash of the downloaded application and compare it with the AAC.

2. Localisation: First, the application will be personalised by the SP. Depending upon the relationship between the cardholder and the SP, at the SP's discretion the personalisation can include acquiring user details (in post and no-registration scenarios), and cryptographic key generation. Furthermore, if the SP is issuing a card-bound lease then it will make sense to generate on-card cryptographic keys as they will automatically become device identifiers because each lease of the application will have a different set of keys. After personalisation, the downloaded application establishes connections with various on-card services (i.e. shareable resources) that are provided by partner applications. To access a partner's application services, the downloaded application will establish an application sharing relationship that is discussed in detail in [5].

3. Application Registration: The final stage of an application installation is the application registration by the SP. The registration will allow the particular instance of the application to access sanctioned services. Once the SP registers (sanctions) the downloaded application, the smart card will also make it selectable to an off-card entity. By making an application selectable, the smart card allows the application to execute and access on-card services, and communicate with off-card entities.

4.6. Application Deletion

The application deletion process has similar steps to the application installation but they are taken in the opposite direction. The installed application will first establish a connection with the SP and signal the deletion. It will initiate the de-registration process that will restrict the leased application's access to the SP's services. The smart card will also make it un-selectable for off-card entities; in addition, any interdependencies will be resolved. As most of the interdependencies that the deleted application might have are the result of the application sharing mechanism, the smart card firewall mechanism will cascade the deletion event to the related (partner) applications [5]. The interdependencies that might exist between various applications on a smart card may end up creating a feature interaction problem. Finally, the SP's domain key material, and registration with various card services are deleted by the Card Security Manager [23].

5. Consumer-Centric Card Management-Related Issues

In this section, we discuss the potential issues related to card management architecture introduced by the Consumer-Centric Models. We also propose countermeasures to the issues discussed in this section.

5.1. Simulator Problem

In the context of the Consumer-Centric Models, the simulator problem refers to a possible scenario in which a malicious user could remotely install an application onto a smart card simulator. One thing to note is that the simulator problem is only related to remote installation and not to on-site installation. In remote installation, a smart card is not present at an SP's site, and the application is downloaded over the internet. Therefore, an SP needs a way of making sure that its application is not installed on a simulated card. It can be asserted that simulators are used in a number of different environments, especially mobile application development, and do not present a substantial security issue in the mobile application environment. Nevertheless, the nature of an application installed on a smart card is different to an application on a mobile phone. The smart card application might represent the identity of the user, along with serving as a security token to access some services (including financial services). Furthermore, the service or business environment that a smart card

application deals with is substantially different from that of a mobile phone application. In the ICOM, the simulator problem is not relevant, as applications are predominantly installed by the card issuer before the smart cards are issued to individual users. This stage in the smart card lifecycle is also referred as the pre-issuance stage. The GlobalPlatform card specification provides the architecture for application installation under the application provider's control, after the smart cards are issued to customers. GlobalPlatform defines a secure entity on the smart card referred to as the Card Manager, along with associated domains [23]. The SP requires symmetric/asymmetric keys in order to gain access to the domains (application domains) and install applications. The assumption in the ICOM is that malicious users cannot access or retrieve these keys. The basis of this assumption is the tamper-resistance properties of the smart card hardware — that is, the assumption is based on trust in the card manufacturer or a third party evaluator (e.g. Common Criteria [17] evaluation laboratory). In the Consumer-Centric Model, the problem is not only verifying the existence of a smart card, but also validating that it is in a secure and reliable state. In a simulator attack, a malicious user has a stand-alone simulator that enables him to simulate the Consumer-Centric Model environment. To do so, the adversary has to have knowledge of the cryptographic keys and any related attestation mechanism (regardless of whether it is based on PRNGs [13] or PUFs [12]). If the attestation mechanism is based on PUFs then the adversary should be able to emulate the PUF for a genuine smart card. He can then try to acquire an application from an SP, install it on the simulator, and potentially attack the application; that may include reverse engineering the application, retrieving the sensitive user and application data (e.g. cryptographic keys).

5.1.1. Countermeasure to the Simulator Problem

The countermeasure to the simulator problem has to deal with physical and side-channel attacks along with the risk of compromising the communication protocol(s). The countermeasure for consumer-centric card management architecture is based on three aspects of the UCOM architecture that are listed as below:

1. Security evaluation of the smart card platform, which certifies that the smart card is tamper-resistant, and effective against state-of-the-art attacks [3].
2. Self and remote attestation mechanism [13, 12].
3. A secure and reliable entity authentication and key sharing protocol [9, 10].

The evaluation and validation of the smart card provides (time-limited) assurance against the simulator attack. If during the lifetime of the smart card, an attack vector is discovered that can compromise the card's security and make a simulator attack feasible, the evaluation authority (and the card manufacturer) can revoke the certificates, and the SP can blacklist the affected smart cards. Because of such attacks, smart cards can be rendered useless. Therefore, card manufacturers, even today, are compelled to build a strong product or otherwise security issues would damage the reputation of their brand, and the same would be true in the case of the UCOMs. Finally, a secure channel protocol should be designed in a way that would make it impossible for an adversary to retrieve the keys shared between a smart card and an SP. To provide protection against simulator attacks, SPs can request device attestation as described in [6]. The device attestation is based on a protocol, which involves the card manufacturer at the time of application lease attesting that its smart card is secure and reliable as stated by the evaluation certificate [3]. Therefore, an evaluation certificate from an independent third party will confirm that the smart card is secure against complete and partial simulations. The online attestation mechanism provides a validation that the smart card is still in conformance with the state in which it was evaluated. Finally, by integrating the smart card assurance and validation mechanism into the secure channel protocols, the UCOM can avoid simulator problems.

5.2. User Ownership Issues

In this section, we discuss an issue that is related to the identity of the smart card owner and the authorised user who can download an application from an SP. This issue arises in the pre-registration relationship (section 4.4) between an SP and a cardholder. During an application installation, a cardholder will provide her credentials to the SP that leases the application. In this situation, the SP requires that the application can only be downloaded to a smart card that is under the ownership of the authorised user. The aim of an adversary may be to acquire the credentials of an authorised user for a particular SP in order to use them to download the application onto his smart card. If the SP does not issue card-bound leases (section 4.3), both entities (the authorised user and the adversary) can download the application.

5.2.1. Countermeasure to User Ownership Issues

To avoid this situation, the SP could require proof of ownership to be produced by the smart card for the given user during the application download process. The proof of ownership can be based on a signature key pair belonging to the user, which is certified by the smart card itself or by its card manufacturer. The issue is that a similar certificate can also be requested by an adversary for the given authorised user, if the

malicious user knows enough personal information relating to the genuine user. Having the smart card signing the certificate is easy, and it also allows the user the independence to sell/give her smart card to other users. On the other hand, including the card manufacturer in the user identification and issuance of certificates to individual users provides similar protection as the previous proposal, without effectively increasing the overall security. Therefore, we prefer the smart card to sign the certificate rather than the card manufacturer. Another possibility is that the smart card user can register her smart card physically (offline) with the SP. The smart card can later access the SP server through the internet to download the application. If there are no adequate checks during the offline registration, an adversary could perform the same process and then use the credentials associated with the user and request the application lease. Furthermore, this solution also complicates the relationship between the user and the SP as there may not always be the possibility for physical access to the SP's office (e.g. online gaming website). An optimum solution can be based on one-time credentials issued by SPs. We assume that an SP issues a one-time credential (e.g. password) to a user to download its application to her smart card. Therefore, even if an adversary was to gain access to the credentials, he cannot use them to download the application. Furthermore, if an application is already leased to a user, an SP can reject any subsequent requests for an application lease unless the user either deletes the previous lease or loses the smart card. Therefore, in the case that the application is deleted and the user wants to install the application on her new smart card, the SP will issue a new one-time credential to the user, to download the relevant application. A malicious user cannot fake the deletion of the application, as in the UCOM deletion process an application communicates with the SP in order to notify it of the deletion and also to perform any required housekeeping tasks. Therefore, the SP knows beforehand that the application is deleted so for a malicious user it is difficult to fake application deletion. Furthermore, if a user loses her smart card then she can use an authorisation token (issued by the SP and discussed in [11]) to acquire an application. An authorisation token is a short encrypted structure issued by an SP and it acts as an authorisation credential to download an application. If she does not have the authorisation token, then the user can contact the SP and request the issuance of new credentials. This is similar to the process after a user loses her smart card in the ICOM, where she has to contact the card issuer to get a new smart card.

5.3. Parasite Application Problem

In the parasite application problem, an installed application masquerades as the UCOM-based smart card referred as a User Centric Smart Card (UCSC) on which it is installed. This is possible because a UCSC allows an installed application to request the platform/application state validation from the Trusted Environment & Execution Manager (TEM) [1]. Regarding GP-CCM, the architecture has not been specified enough to assess if this problem is relevant. In this attack as shown in figure 5, an adversary ($\mathcal{A}$) has installed his malicious application ($App_{Malicious}$) on a UCSC. The $App_{Malicious}$ implements the application download protocols that are discussed in [9, 10]. The adversary ($\mathcal{A}$) then requests the installation of an application from an SP through the $App_{Malicious}$, represented by message one in figure 5. In response, the SP asks the UCSC for the security validation to avoid a simulator attack and this request is sent to the $App_{Malicious}$. The $App_{Malicious}$ then asks the TEM for the security validation [12]. At the successful conclusion of the security validation process, the card manufacturer produces a certificate (as part of the online validation proof [12, 13]) that for privacy reasons does not include the identity of the requesting SP as described in the online attestation protocol [13]. Therefore, the $App_{Malicious}$ can communicate this certificate to the requesting SP as validation in message five depicted in figure 5) and may be able to start the application download process. The $\mathcal{A}$ might have designed the application as if it will communicate the downloaded application off-card, which will enable $\mathcal{A}$ to retrieve the application code and data.

5.3.1. Countermeasure to Parasite Application Problem

To avoid this problem, there are four possible solutions: 1) restrict the security assurance validation request to off-card entities and any installed applications should not be allowed to request it, 2) include the identity of the requesting SP in the security assurance validation certificate during the application installation process, 3) include the identity of the application requesting the security assurance validation during the application installation: the request is initiated by the card security manager [21], or 4) avoid generating the signature (as validation proof) as part of the security validation if it is requested by an application, but use the shared symmetric keys between the TEM and the application [5]. From the above listed options, we consider that option four is appropriate for the UCOM environment to prevent the parasite application problem. In option four, the TEM will only sign the security validation proof if it is requested by the card security manager and not by an application. Therefore, an application installed on a UCSC cannot request (message 3 in figure 5) a (signed) security validation proof from the TEM, which would prohibit the application from effectively masquerading as the respective UCSC.

5.4. Platform Insider Attacker Problem

In the multiapplication context, the fear of the smart card platform owners is related to the insider attacker problem. Indeed a successful attack can damage their businesses and the reputation of their brands [40]. To avoid this problem in ICOM, GlobalPlatform and Multos card management architecture are used to lock and control trust in platform (authorisation to load are granted only after a trust establishment: e.g. in Multos architecture the application is installed after the whole code is evaluated; similarly, in GlobalPlatform architecture this is achieved by off-card verification with code signature and some business agreements including potential damages between card issuer and SP). The card locking and control mechanism, which is under a centralised authority (e.g. card issuer) guarantees that a malicious application would not be installed on the smart cards. In the Consumer-Centric Model, the paradigm is shifted and the user is the owner of the platform and he/she may act maliciously.

5.4.1. Countermeasure to Platform Insider Attacker Problem

To enforce card security, as stated in section 4.5, an embedded byte code verifier in addition to the firewall is required (GP-CCM also mentions them as on-board platform security mechanisms). However some attacks may still be possible. For this reason, but also for instance to avoid the whole certification of the platform and all embedded applications, GlobalPlatform is working on a Composition Model [28, 29] to allow post-issuance application loading while maintaining trust. To make the Consumer-Centric Model attractive even for the small and medium scale organisations as a target architecture of their applications, SPs are not required to get a third party evaluation certificate for their application. Thus it may limit the potential of the market to only big companies and this is not satisfying. An additional countermeasure can be to assume that the UCSC are supported with state-of-the-art security mechanisms and thus they might be resistant against attacks like:

1. the use of installed applications to identify power consumption or electromagnetic leakage patterns [18, 41]
2. combined attacks (software and physical attacks like optical fault or voltage glitches) [42]
3. software only attacks that may bypass the embedded byte code verification [33], like those on transaction mechanism [38] which took advantage of the bug in the implementation of the transaction mechanism to create "ill-typed code". Note that the attacks on the shareable interface mechanisms cited in [38] were only working on cards without byte code verification.

Protection against these attacks can be achieved through efficient shield and defensive Virtual Machines [32, 31]. In [38], authors stated that runtime type checking offered an excellent protection against ill-typed code. Therefore, the proposal for the TEM [1] in the UCOM ecosystem has introduced a runtime security manager that goes beyond the traditional notion of a trusted and secure execution environment. It oversees the execution of the application and manages different aspects of the runtime environment. The aim of the runtime security manager is to ensure the security of the UCSC platform and provide a trusted execution environment to individual applications. If an application behaves maliciously the UCSC will prohibit the application from executing and to a further extend delete or block the application.

6. Conclusion

In this paper, we discussed the widely accepted and deployed card management architectures: GlobalPlatform and Multos. GlobalPlatform is more open to independent application management by application providers, whereas Multos is a hardcore ICOM architecture that requires an authorisation from a centralised authority (i.e. Multos Certification Authority) before an application can be installed or deleted. Furthermore, Multos also requires that application providers should reveal their application codes to the Multos application load unit generator. We then described a flexible consumer-centric card management architecture followed by the application lease types, and the various kinds of relationships a consumer can have with an SP. Next, we discussed the application installation and deletion approach. Finally, we discussed new security issues (including simulator, user ownership, parasite application problems and platform insider attacker problems) in the general context of the Consumer-Centric Model. We suggested that the TSM can co-exist with the Consumer-Centric Model: in fact one can argue that the merger of TSM with the Consumer-Centric Model might resolve many of the issues that have plagued the multi-application smart card initiative from the beginning. The scope of discussion related to the security issues was intentionally kept short. We have only included the issues raised due to the changes in the management of a smart card and not included issues raised by Consumer-Centric Model in general. For future multiapplication smart card deployments, we consider that the application management architecture proposed by the UCOM could provide a robust, scalable and flexible architecture. Adoption of the UCOM-based ecosystem even in TSM-based service deployments will be beneficial to all stakeholders. Consumer-centric ecosystems have now matured to such a level that they should be adopted for smart cards. This may in fact increase their adoption in diverse fields and specifically for applications that were not traditionally part of smart card services due to the stringent requirements placed by card issuers in the ICOM.

References

[1] Raja Naeem Akram and Konstantinos Markantonakis. Rethinking the Smart Card Technology, Invited Paper. In Theo Tryfonas and Ioannis Askoxylakis, editors, *16th International Conference on Human-Computer Interaction*. Springer-Verlag, June 2014.

[2] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. Location Based Application Availability. In R. Meersman, P. Herrero, and T. Dillon, editors, *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, volume 5872/2009 of *LNCS*, pages 128–138, Vilamoura, Portugal, November 2009. Springer-Verlag.

[3] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. A Dynamic and Ubiquitous Smart Card Security Assurance and Validation Mechanism. In Kai Rannenberg and Vijay Varadharajan, editors, *25th IFIP International Information Security Conference (SEC 2010)*, IFIP AICT Series, pages 161–171, Brisbane, Australia, September 2010. Springer-Verlag.

[4] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. A Paradigm Shift in Smart Card Ownership Model. In Bernady O. Apduhan, Osvaldo Gervasi, Andres Iglesias, David Taniar, and Marina Gavrilova, editors, *Proceedings of the 2010 International Conference on Computational Science and Its Applications (ICCSA 2010)*, pages 191–200, Fukuoka, Japan, March 2010. IEEE Computer Society.

[5] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. Firewall Mechanism in a User Centric Smart Card Ownership Model. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application, 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010*, volume 6035/2010 of *LNCS*, pages 118–132, Passau, Germany, April 2010. Springer-Verlag.

[6] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. Simulator Problem in User Centric Smart Card Ownership Model. In Helen Y. Tang and Xinwen Fu, editors, *6th IEEE/IFIP International Symposium on Trusted Computing and Communications (TrustCom-10)*, HongKong, China, December 2010. IEEE Computer Society.

[7] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. A Privacy Preserving Application Acquisition Protocol. In Geyong Min and Felix Gomez Marmol, editors, *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-12)*, Liverpool, United Kingdom, June 2012. IEEE Computer Society.

[8] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. Building the Bridges – A Proposal for Merging different Paradigms in Mobile NFC Ecosystem. In Shengli Xie, editor, *The 8th International Conference on Computational Intelligence and Security (CIS 2012)*, Guangzhou, China, November 2012. IEEE Computer Society.

[9] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. Coopetitive Architecture to Support a Dynamic and Scalable NFC based Mobile Services Architecture. In K.P. Chow and Lucas C.K. Hui, editors, *The 2012 International Conference on Information and Communications Security (ICICS 2012)*, Hong Kong, China, October 2012. Springer-Verlag.

[10] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. A Secure and Trusted Channel Protocol for the User Centric Smart Card Ownership Model. In *12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-13)*, Melbourne, Australia, July 2013. IEEE Computer Society.

[11] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. Recovering from Lost Digital Wallet. In F. G. Marmol Y. Xiang and S. Ruj, editors, *The 4th IEEE International Symposium on Trust, Security, and Privacy for Emerging Applications (TSP-13)*, Zhangjiajie, China, November 2013. IEEE Computer Society.

[12] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. Remote Attestation Mechanism based on Physical Unclonable Functions. In C. Ma J. Zhou and J. Weng, editors, *The 2013 Workshop on RFID and IoT Security (RFIDsec'13 Asia)*, Guangzhou, China, November 2013. IOS Press.

[13] Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes. Remote Attestation Mechanism for User Centric Smart Cards using Pseudorandom Number Generators. In S. Qing and J. Zhou, editors, *5th International Conference on Information and Communications Security (ICICS 2013)*, Beijing, China, November 2013. Springer-Verlag.

[14] Ross Anderson. Can We Fix the Security Economics of Federated Authentication? In James A. Malcolm, editor, *SPW 2011, 19th International Workshop on Security Protocols*, London, UK, March 2011. Springer-Verlag.

[15] David A. Basin, Stefan Friedrich, and Marek Gawkowski. Verified Bytecode Model Checkers. In *TPHOLs '02: Proceedings of the 15th International Conference on Theorem Proving in Higher Order Logics*, pages 47–66, London, UK, 2002. Springer-Verlag.

[16] David A. Basin, Stefan Friedrich, Joachim Posegga, and Harald Vogt. Java Bytecode Verification by Model Checking. In *CAV '99: Proceedings of the 11th International Conference on Computer Aided Verification*, pages 491–494, London, UK, 1999. Springer-Verlag.

[17] CCMB. Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, Part 2: Security Functional Requirements, Part 3: Security Assurance Requirements,, August 2006.

[18] Serge Chaumette and Damien Sauveron. Some Security Problems Raised by Open Multiapplication Smart Cards. *10th Nordic Workshop on Secure IT-systems: NordSec 2005*, pages 20–21, October 2005.

[19] European Commission. Future Networks and the Internet: Early Challenges Regarding the Internet of Things. Commission Staff Working Document SEC(2008) 2516, Commission of the European Communities, Brussels, September 2008.

[20] EPC and GSMA. EPC-GSMA Mobile Contactless Payments Service Management Roles Requirements and Specifications. Technical Report EPC 220-08, European Payments Council (EPC) and GSM Association, October 2010.

[21] GlobalPlatform. GlobalPlatform Card Security Requirement Specification 1.0. Specification, Redwood City, USA, May 2003.

[22] GlobalPlatform. GlobalPlatform Guide to Common Personalization. Specification 1.0, Redwood City, USA, May 2003.

[23] GlobalPlatform. GlobalPlatform: GlobalPlatform Card Specification, Version 2.2,, March 2006.

[24] GlobalPlatform. GlobalPlatform's Proposition for NFC Mobile: Secure Element Managment and Messaging. Specification, GlobalPlatform, April 2009.

[25] GlobalPlatform. GlobalPlatform Card: Confidential Card Content Management. Card Specification v2.2 - Amendment A. Specification 1.0.1, Redwood City, USA, January 2011.

[26] GlobalPlatform. GlobalPlatform Device: Secure Element Remote Application Management. Specification, GlobalPlatform, February 2011.

[27] GlobalPlatform. GlobalPlatform A New Model: The Consumer-Centric Model and How It Applies to the Mobile Ecosystem. Whitepaper, GlobalPlatform, March 2012.

[28] GlobalPlatform. GlobalPlatform Card Composition Model. Specification Ver 1.1, GlobalPatform, June 2012.

[29] GlobalPlatform. Security Evaluation of Trusted Execution Environment: Why and How? Whitepaper, Trusted Labs, 2013.

[30] GSMA. Global Systems for Mobile Communication (GSM), Visited August, 2010.

[31] Michael Lackner, Reinhard Berlach, Michael Hraschan, Reinhold Weiss, and Christian Steger. A Defensive Java Card Virtual Machine to Thwart Fault Attacks by Microarchitectural Support. In *International Conference on Risks and Security of Internet and Systems (CRiSIS)*, pages 1–8, 2013.

[32] Michael Lackner, Reinhard Berlach, Wolfgang Raschke, Reinhold Weiss, and Christian Steger. A Defensive Virtual Machine Layer to Counteract Fault Attacks on Java Cards. In *Workshop in Information Security and Practice (WISTP)*, pages 82–97, 2013.

[33] Xavier Leroy. On-Card Bytecode Verification for Java Card. In *E-SMART '01: Proceedings of the International Conference on Research in Smart Cards*, pages 150–164, London, UK, 2001. Springer-Verlag.

[34] Xavier Leroy. Bytecode Verification on Java Smart Cards. *Softw. Pract. Exper.*, 32(4):319–340, 2002.

[35] MAOSCO. Multos: The Multos Specification,. Online.

[36] MAOSCO. Multos: Guide to Generating Application Load Units. Technical Report MAO-DOC-TEC-009 v2.52, MAOSCO, 2006.

[37] MAOSCO. Multos: Guide to Loading and Deleting Applications. Technical Report MAO-DOC-TEC-008 v2.21, MAOSCO, 2006.

[38] Wojciech Mostowski and Erik Poll. Malicious Code on Java Card Smartcards: Attacks and Countermeasures. In Gilles Grimaud and Fran?ois-Xavier Standaert, editors, *Smart Card Research and Advanced Applications*, volume 5189 of *LNCS*, pages 1–16. Springer-Verlag, 2008.

[39] Oracle. Java Card Platform Specification: Classic Edition; Application Programming Interface, Runtime Environment Specification, Virtual Machine Specification, Connected Edition; Runtime Environment Specification, Java Servlet Specification, Application Programming Interface, Virtual Machine Specification, Sample Structure of Application Modules, May 2009.

[40] Damien Sauveron. Multiapplication Smart Card: Towards an Open Smart Card? *Inf. Secur. Tech. Rep.*, 14(2):70–78, 2009.

[41] Dennis Vermoen, Marc Witteman, and GeorgiN. Gaydadjiev. Reverse Engineering Java Card Applets Using Power Analysis. In Damien Sauveron, Konstantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater, editors, *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, volume 4462 of *LNCS*, pages 138–149. Springer-Verlag, 2007.

[42] Eric Vétillard and Anthony Ferrari. Combined Attacks and Countermeasures. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application, 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010*, volume 6035/2010 of *LNCS*, pages 133–147, 2010.

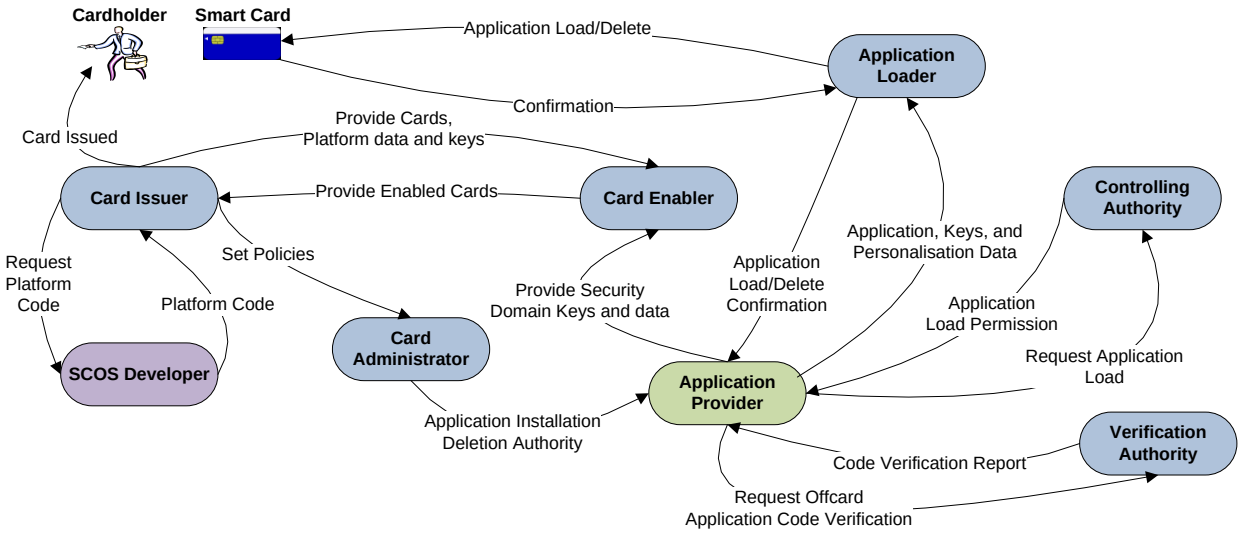

Figure 1: GlobalPlatform card management architecture [21]

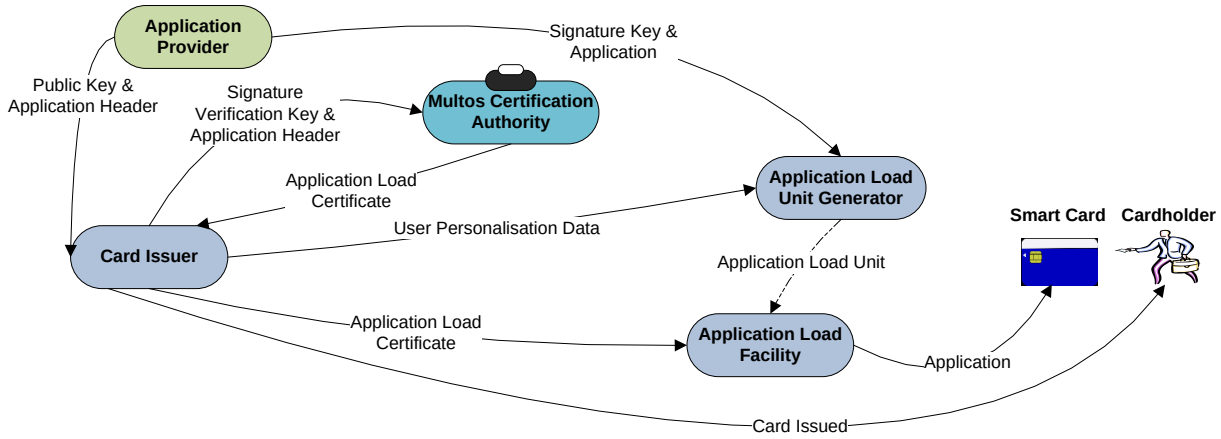

Figure 2: Multos card management architecture

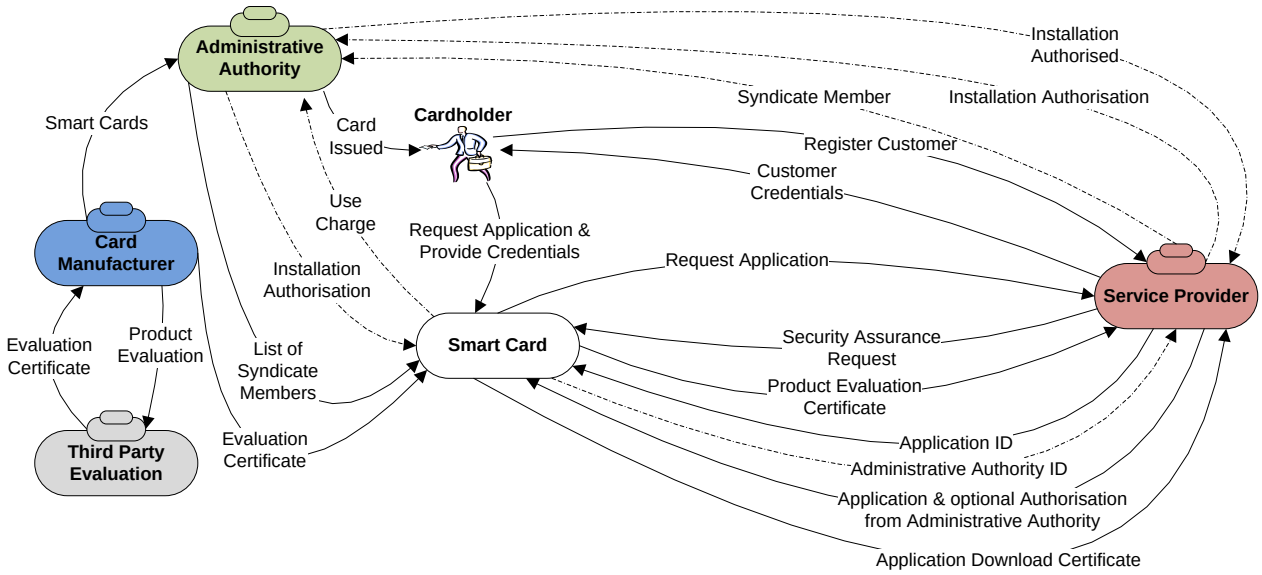

Figure 3: Administrative card management architecture (CASC architecture [8])

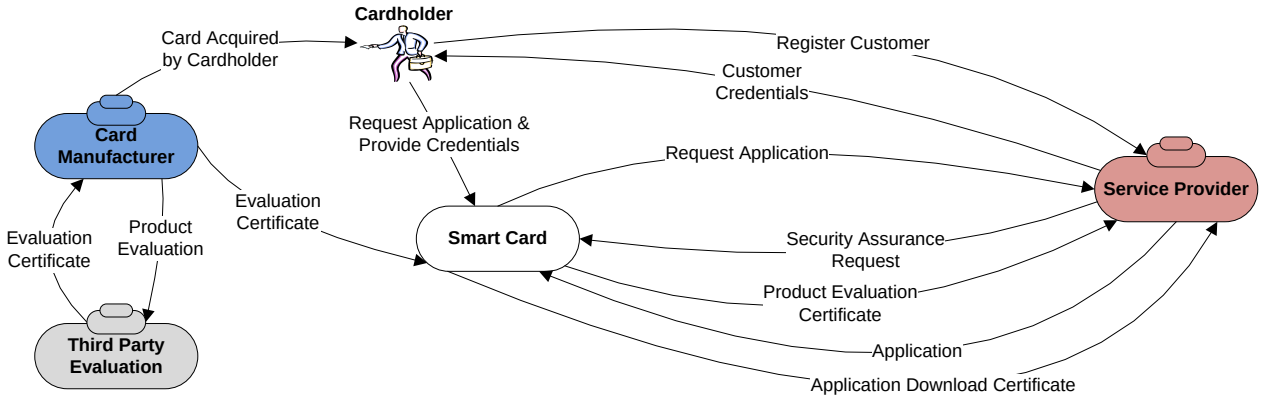

Figure 4: User card management architecture (UCOM [4])

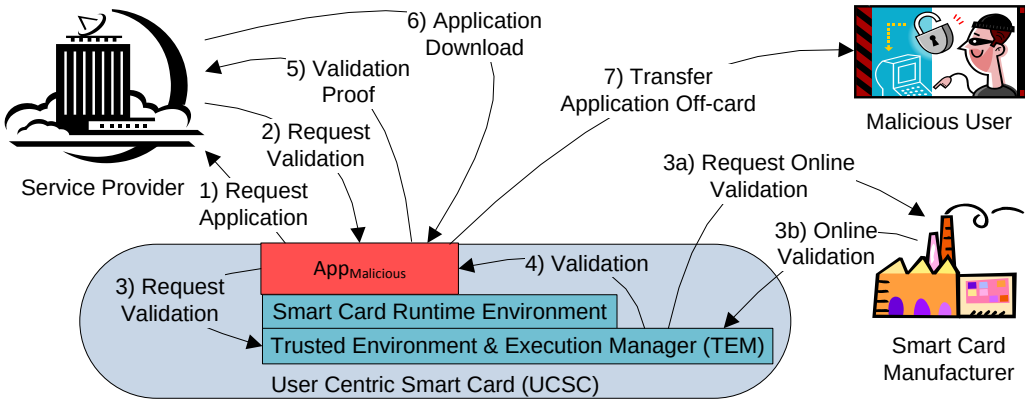

Figure 5: Illustration of parasite application problem