SECURING INTELLIGENT ADJUNCTS USING TRUSTED COMPUTING PLATFORM TECHNOLOGY

Boris Balacheff, David Chan, Liqun Chen, Siani Pearson, and Graeme Proudler
Hewlett-Packard Laboratories, Filton Road, Stoke Gifford, BRISTOL BS34 8QZ, UK
{boris_Balacheff, d_chan, liqun_chen, siani_pearson, graeme_proudler}@hp.com

Abstract In [1], Balacheff et al described a new paradigm for smartcard usage called the Intelligent Adjunct model. The current increasing programmability of smartcards and development of the Internet is enabling new flexible and dynamic platforms for electronic commerce and services. In particular, the Intelligent Adjunct model combined with the use of a Trusted Computing Platform enables more flexible and more reliable network-based service development. This paper describes such a method using a hardware-based component in a computing platform to enable the establishment of a trust relationship between a smartcard and the terminal to which it is connected.

We will first give a brief overview of the Intelligent Adjunct model as described in [1]. We will look at the technology required from what we call a Trusted Computing Platform. Then we will describe how the latter helps to secure Intelligent Adjunct technology. Finally, we will look at a specific application that enables a user to establish some trust in his or her computing platform.

Keywords: Smartcards, Trusted Measurement, Trusted Reporting, Integrity, Trusted Computing Platform, Intelligent Adjunct, Trust.

1. INTRODUCTION

The Intelligent Adjunct paradigm, presented in [1], described a new approach to the use of smartcards in a computing environment. Evolving from the traditional Master/Slave interaction to a Peer-to-Peer model between a smartcard and its terminal, this new model allows for smartcards to have access to off-card resources through the terminal to which they are connected. From the user's perspective, the Intelligent Adjunct model enables the use of smartcards to dynamically personalise the environment of the computing terminal and to automate user-defined tasks.

Throughout this paper a smartcard that implements Intelligent Adjunct Technology is referred to as an IA.

To make this new smartcard usage model useful in today's electronic world, it is necessary to address the need for security in the implementation of applications that use this technology. For example, a user's IA can be programmed to check a website for share prices and perform various investment tasks accordingly. This assumes that the security required by the provider of such a web-based stock manipulation service can be implemented.

In the case of services where the IA communicates directly through its terminal with a service provider's server, it is sufficient for security to be implemented both on the server side and on the IA side. However, for a user's IA to be able to take advantage of resources local to the terminal to which it is connected, there is a need for a means by which the IA can establish trust in the local terminal.

Enabling a smartcard to have a degree of trust in the terminal to which it is connected makes it possible to use IA functionality in the context of e-commerce applications, and more generally electronic services.

This paper describes how it is possible to enable an IA to establish trust with what we call a Trusted Computing Platform. Several existing technologies aim to increase the security and trustworthiness of computer platforms. In this paper we will describe a Trusted Platform, but will concentrate on the functionality therein relevant to our particular problem. More details on Trusted Platform Technology can be found in [2] and [3].

The rest of this paper is organised as follows. The following section provides further information about the Intelligent Adjunct model as described in [1]. The next section looks at the technology required for a Trusted Computing Platform. We then describe in section 4 how the latter helps to secure Intelligent Adjunct technology, and how trust can be established between an IA and the local terminal. Finally, in section 5, we will look at a specific application that enables a user to establish some trust in his or her computing platform.

2. INTELLIGENT ADJUNCTS

In the Intelligent Adjunct model, the smartcard is a general-purpose computing device that carries application logic. It carries out programmed tasks using off-card resources, with as much control over their execution as the terminal it is connected to would have. This model makes the IA an independent processor that will cooperate with a terminal to fulfil these tasks. Contrary to the usual model of the master/slave relationship

between terminals and a standard smartcard, in the Intelligent Adjunct model smartcards and their terminals are peers: each of them can make equal use of the other's resources. In other words, the IA carries the logic to some specific applications that can access multiple off-card resources. These resources can be either local to the smartcard terminal, or network resources.

The key features of the Intelligent Adjunct model are:

- to migrate some application logic to the smartcard,
- to have card initiated actions,
- to make terminal and network resources available to the smartcard, and
- to make sure this model can be integrated to existing infrastructures.

With this model, a new range of services can be made available to the end-users with unprecedented portability. Once all potential smartcard terminals are upgraded to enable this technology there will be no more limitations on what an IA can achieve, other than the limitation of the terminal's capabilities.

Let us take the example of a dial-up service. An IA could be programmed to control the dial-up procedure for a user's computer. It could contain fixed parameters as well as configurable ones, and modifiable scripts to carry out a number of tasks that the user would normally do manually upon connection. In this way, a dial-up connection between the user's computer and his Internet Service Provider could be set-up by the IA. Furthermore, upon connection the IA could check a number of things: new email, specific email, the number of friends connected to the chat-rooms, the last modified date of a given web-page and so on. The IA could carry out any task that the user would normally do manually and systematically. Furthermore, once the IA model becomes more widespread, the user's IA will be able to do this from any terminal, without the terminal being configured to carry out these operations.

Today's technology (such as Windows for smartcards and JavaCards) addresses the issue with end-user programmability of smartcards. But given the very limited resources on the card and the severe restriction of the oncard processing capability, more needs to be provided in order to make the paradigm useful.

[1] described how Intelligent Adjunct functionality can be integrated with the PC/SC architecture. This enables the provision on every PC platform of access to off-card resources for IAs.

A key limiting aspect of this model as presented in [1] was that of the trustworthiness of the terminal. We will discuss this issue later in the paper, and see how it can be resolved, and how it impacts the types of off-card resources that an IA can access.

3. A TRUSTED COMPUTING PLATFORM

Much work has recently been carried out on the concept of a Trusted Computing Platform that we introduce here. More detailed information on this technology can be found in [2] and [3]. We are also pleased to see that the industry is looking at the concept of a Trusted Platform in the context of an industry alliance formed in October 1999 [4].

3.1. WHAT IS A TRUSTED COMPUTING PLATFORM?

To be a "trusted platform", a computing platform must be trusted to behave in the expected manner for the intended purpose. The basis for this trust is a declaration by a known authority that the platform will always behave in the expected manner for the intended purpose. This authority must be trusted by anyone who needs to build a trusted relationship with the computing platform, such as the owner of the platform, the user of the platform and other entities which communicate with the platform. In other words, someone whom the owner/user/entity trusts says with certainty that the platform is exactly what it says it is, and that the owner/user/entity can trust the platform.

In order to provide such a trusted platform, we need a mechanism that reliably identifies that platform as a trusted platform and reliably measures and reports integrity information about that platform to a party that requires it (we will call such a party a "challenger").

In particular, this mechanism is provided for identity verification and integrity checking of the platform both in local and remote mode, using authentication, measurement and reporting of platform integrity information, such as hardware, BIOS, Master Boot Record, and Operating System. The challenger can decide whether to trust the platform based on the identity and integrity information and the authorities vouching for this information. Furthermore, in order to protect privacy, computer owners maintain control over information contained in the system.

In order to implement such a mechanism in a computing platform, it is necessary that the architecture of an ordinary platform be changed in a fundamental way.. Existing security technology has been developed as far as possible without security being embedded into a platform's architecture: for example, even the most robust PC Operating System (OS)

cannot prevent unauthorised software to be loaded before the OS does. Once such unauthorised software has been loaded, it can gain access to areas of the system that should not be accessible in a trusted environment.

To protect the main computing platform against software attacks, a processing engine separate from the main CPU is required, and that separate engine must be protected from prying and/or interfering. This engine is hardware-based in order to provide the cryptographic potential for storing secrets and using them without exposure. In this paper, we call such a hardware-based engine a Trusted Subsystem (TS). It is tamper-resistant and forms the root of trust for measurement and reporting of Integrity Metrics relating to the platform components.

The Trusted Subsystem provides security primitives that can be used to enhance service provision. In particular, support is given for a platform identity, platform integrity measurement and reporting of these measurements, enhanced cryptographic capabilities (including random number generation, public and private key pair creation, digital signatures, data encryption and hash function), and protected storage of confidential information.

This can be done using technology such as described in [2] and [3], whereby a tamper-resistant hardware component is tied to the platform to provide these functionalities. We will not go into details as to how the integration of such a component can be achieved. We will simply describe the functionality provided by such a Trusted Subsystem, and refer to [2] and [3] for details.

3.2. PROVING TRUSTED PLATFORM INTEGRITY

Upon request, a trusted platform can supply the challenger with results of measurements (called "Integrity Metrics") taken on the platform, plus the details of the measurement processes that produced those metrics. The platform may also supply the challenger with the values that should be expected if the platform has not been tampered with. The challenger checks the appropriateness of the measurement processes, and compares the supplied metrics with the expected values (whether these values were provided by the platform itself, already known to the challenger, or retrieved from a third party). Then the challenger may use his particular policies to determine whether the platform may be trusted for the intended purpose. Typically, these Integrity Metrics will be some measurements (such as a hash) on a number of components of the platform such as: BIOS, OS loader, Option Roms, OS modules, etc.

A trusted bootstrapping process is used by which each component that is loaded measures an Integrity Metric of the next component to be loaded, and stores this in the TS. If the next component is malign, it will only potentially be able to store false Integrity Metrics for the components still to be loaded, but will not be allowed to alter its own Integrity Metric, which has already been stored. Therefore, when a challenger requests a platform's Integrity Metrics, it will be able to decide whether or not a platform has had an unexpected software component loaded.

The challenger needs to verify both the authenticity of the TS and the authenticity of the expected values. Since all trust can ultimately be traced to a trust in people (an individual or an organization), trust in the TS and trust in the expected values must derive from trusted people who vouch for that TS and those values. Those trusted people publish data that attests to the integrity of the Subsystem and integrity of the expected values, and a conventional Public Key Infrastructure attests to the cryptographic identity of those trusted people. Once a challenger has decided the names of the people it will trust, and has a trusted Certification Authority to provide access to the PKI, it can obtain attestation to the Subsystem and to the expected values.

The response to a challenge will therefore consist of:

- Measured Integrity Metrics that are digitally signed by the TS
- The certificate on the asymmetric key pair used by the TS to compute digital signatures And optionally,
- The Integrity Metrics to be expected
- A certificate (in the PKI sense) on the Expected Metrics for each separate component measured

The proving of Identity is essential to this technology, as it provides the means for the TS to prove it is valid, and vouched for by a Certification Authority. This identity is not linked to a user; it is issued to a TS with a certificate that vouches for that TS's behaviour.

With this information a challenger can then check the Digital Signature and the corresponding certificate to establish trust in the TS (assuming the certificate is issued by a Certification Authority that the Challenger decides to trust). Then the challenger can trust the measured metrics reported by the TS. Finally, for each of these metrics, the challenger can check the certificate for the Expected Metrics value and trust its integrity (once again assuming the Certification Authority used for this particular component is trusted by the Challenger). These Expected

Metrics could be fetched by the challenger itself from a public directory on the network, provided that they are digitally certified by their supplier.

The challenger must be satisfied that the Expected Metrics can be trusted to describe a platform configuration that he is prepared to interact with for his intended purpose. Then, by checking the measured Integrity Metrics against these Expected Metrics, he can decide whether the result of this challenge gives him sufficient trust in the computer platform for the intended purpose.

3.3. TS IDENTITY AND PRIVACY CONSIDERATIONS

As mentioned above, an identity is required in order to prove that the TS can be trusted. This identity is called the TS identity. In a mechanism for creating the identity, the TS generates a public/private key pair, and stores the private key in the TS's tamper-resistant storage, which it will never leave. Then the public key is sent to a trusted Certificate Authority (CA), which is chosen by the owner of the platform. The CA issues an identity certificate on the TS's identity public key after verifying that the TS is a genuine TS, and that the platform has been built so that it correctly uses that TS.

This identity is the TS identity; it is not meant to be a user identity. The privacy of the platform owner must be maintained. This implies that the platform owner must have full control over the identity, including the disclosure of that identity. But the owner is not able to create and to use such an identity without involvement of the genuine TS.

One way to satisfy privacy requirements is to allow the TS to have multiple anonymous cryptographic identities. Each of these identities is completely independent of the other identities, and is determined by the platform owner, but nevertheless attests that the identity describes a genuine TS. In this way the platform owner can choose to use different TS identities for different applications, while each identity remains unique to his or her particular TS.

3.4. SECURE STORAGE

Another functionality that can be provided by a Trusted Platform's TS is the secure storage of data. This may exist internal to the TS in order to protect information relating to integrity challenges, but it could also provide an external entity with such functionality. Either of the following can be used: secure storage within the TS's tamper-resistant hardware (for cryptographic private keys for example); secure off-TS storage

whereby the TS can encrypt data (under a symmetric key that only it can decrypt) and store this data on the local platform.

3.5. SIGNING OF DATA

A Trusted Computing Platform's TS should also provide a means for applications to digitally sign data that will be sent to other parties. This functionality shall be provided to the local OS once the TS is satisfied with the integrity measured during the boot process. The TS will therefore trust the OS not to provide this functionality to applications running on other platforms.

Trusted Communication Path. When communicating with a remote Trusted Platform it is important that beyond Integrity Verification, all data exchanges with the remote machine are digitally signed. More importantly, all data received from the remote machine must be signed by its TS.

This mechanism is necessary in situations that require protection against a straightforward man-in-the-middle attack. When challenging a remote machine for integrity, there is no guarantee that that machine is not routing the Integrity Verification protocol to another machine. This could result in trusting that machine, but believing it has the identity and the configuration of another. The second mechanism ensures that all communications subsequent to an integrity challenge can be authenticated by the TS, and proves that they come from the same machine that has proved its integrity.

4. SECURING INTELLIGENT ADJUNCT FUNTIONALITY

This section describes how a Trusted Computing Platform can be used to secure IA technology.

As mentioned above, the Intelligent Adjunct model describes a protocol that allows a smartcard to request access to resources available on the terminal to which it is connected. For this protocol to be available to the IA, it needs to be implemented as a service on the terminal to which the smartcard is connected.

An IA can then access off-card resources through the terminal. On a traditional PC, the degree of trust the IA can have in accessing these resources is only as high (or as low) as that of the host PC. Therefore, for this technology to be useful in the context of network-based services it is necessary to build in some security mechanism. This will allow a service provider to use Intelligent Adjunct technology to enhance the level of

services it provides, while keeping the level of security it requires for the implementation of the technology used to provide them.

An Example. A web-based stock-broker service provider might want to roll out IA to its customers to allow them to flexibly and automatically check their stock rates. In order to allow customers to use their IA to automatically initiate transactions (for example when some threshold is reached), the security of the IA's interaction with the service must be enhanced.

4.1. NEED FOR SECURITY

As mentioned in the original description of Intelligent Adjunct technology [1], the off-card resources for which access can be provided to an IA can be categorized as follows:

- Internal to the host machine: examples of this include displaying facilities, user input and file storage. The components providing Intelligent Adjunct technology are system-level application modules on the host PC, and will thus enjoy the same level of security as other system components; their security level is therefore as high (or as low) as that of the host PC.

- Local to the host machine: this mostly refers to peripherals attached to the host PC, such as a printer or a modem. As for internal resources, the security of the host machine limits the security level of accessing the resource.

- External resources: when the IA accesses resources somewhere on the network, cryptographic techniques can help to make the intermediate communication channel secure. In some cases, even the security level of the connection endpoint is unimportant: for example a networked file server could store encrypted files that only the card can access.

For an IA to be able to use off-card resources reliably, it is necessary that it can decide whether or not to trust the platform to which it is connected. Once this decision is made, it will also be important that the IA remains assured that communications subsequent to the establishment of this trust relationship still take place with this same terminal..

4.2. INTERACTING WITH A TRUSTED PLATFORM

This section describes how an IA can interact with a Trusted Platform, and more specifically with the Trusted Platform's Trusted Subsystem (TS).

4.2.1 Enabling IA/TS Communications. For an IA smartcard to take advantage of Trusted Platform technology, and to be able to interact with a Trusted Platform's Trusted Subsystem, a service needs to be provided on that platform that will allow IA smartcards to request to establish a communication with the TS.

Intelligent Adjunct technology as described originally in [1], requires that terminal-side software be implemented in order to allow for card-initiated actions. As in that paper, we have focused on the PC as a specific computing platform for discussions on implementation.

In the specific case of a PC, it was suggested in that paper that this software be integrated into the existing PC/SC stack that has become a de facto standard for PC-to-smartcard communications on Microsoft Windows(operating systems. The way that this can be done was described in [1], such that the PC/SC Resource Manager would manage the proactive protocol that enables card-initiated actions to take place. Access to off-card resources could then be implemented partly in the PC/SC Resource Manager, but mostly as separate modules (PC Service Providers, or PCSPs) related to a particular resource made available to the card.

Following the same architecture, access to a Trusted Platform's TS can be implemented as a PCSP. The PC/SC Resource Manager manages requests from an IA to use this particular PCSP, and routes communications between the IA and the TS's PCSP (see Figure 1).

The PCSP is necessary to implement interfaces to the TS's functionality, but it is not necessary for it to be trusted by the IA for the purpose of interactions with the TS. Because it simply routes communications between an IA and the TS, communications can be secured (i.e. encrypted and authenticated) end-to-end between the two parties.

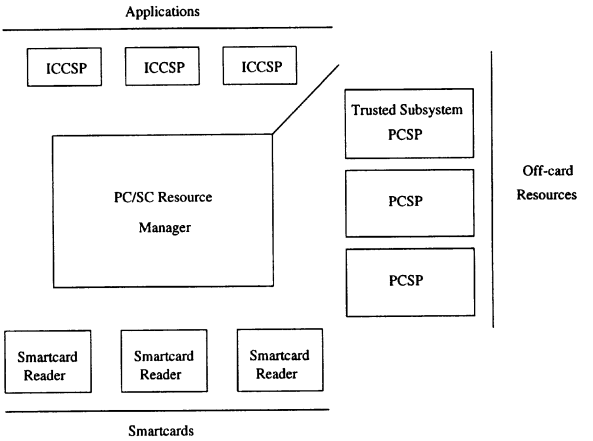

Figure 1 - A TS's PCSP in an IA-enabled PCSC architecture

We will now look at the functionality that must be available for an IA to interact with the local platform's TS.

4.2.2 TS Functionality Available to IAs. The TS's PCSP interfaces TS functionality to the IA smartcard. This functionality allows the IA to initiate:

- An integrity verification protocol,
- An authentication request or
- A request for secure storage.

These will be considered in turn.

INTEGRITY VERIFICATION. The following protocol for integrity verification will allow the IA to receive the Integrity Metrics measured by the TS on the local PC, and optionally values of these metrics that should be expected for the platform to be trusted and certificates on these values.

The protocol is carried out as follows:

1 The TS makes integrity measurements on the platform before other programs have had the opportunity to subvert the measurement process (cf. 3.2 Proving Trusted Platform Integrity). These measured Integrity Metrics are stored securely in the TS. The TS also stores information describing the measurement process and which components of the platform where measured.

2 When the platform's TS receives an "integrity challenge" containing a nonce, the nonce is concatenated to the measured Integrity Metrics, and the result is signed. The TS then collects the data describing what measurement where performed, the signed Integrity Metrics, and a certificate attesting to the cryptographic identity of the TS, and inserts the data into an "integrity response". The TS sends the integrity response to the challenger.

3 In order to be able to trust the Integrity Response, the challenger IA first validates the certificate attesting to the identity of the TS, using the cryptographic identity (public key) of the corresponding Certification Authority. If the IA trusts the TS's CA, and this verification is positive, the IA can now trust that the TS is a genuine TS.

4 Then the challenger IA validates the TS's signature on the measured Integrity Metrics using the TS's public key.

If this last step was positive, the IA can now trust that the Integrity Metrics measured and reported by the TS are genuine.

What remains is for the challenger IA to be able to decide whether the Integrity Metrics reported by the TS correspond to components (hardware and software) it is prepared to trust.

In order to do so, the challenger IA needs to get access to Integrity Metrics certified by the supplier of the components that were measured. The suppliers of individual components will provide a digital signature for these metrics (we call them Expected Metrics)

These Expected Metrics (along with the digital signature by their supplier) can either be provided by the TS along with the Integrity Response, or else the challenger IA can fetch them from a server on the network, based upon information describing what measurements were performed.

Once the IA has access to these Expected Metrics, it will decide whether or not to trust them, based on the cryptographic identity of the supplier of these metrics. For example, a BIOS vendor would provide a signature on the Expected Metrics for their BIOS, and use a CA to certify their cryptographic identity.

Finally, once the challenger IA is satisfied that it can trust the Expected Metrics for the components measured by the TS, it can proceed to compare the reported measured Integrity Metrics against the Expected Metrics.

If any of the untrusted or unprotected parts of the platform have lied, at least one of the above tests will fail. If any of the measurements do not match with the expected value, at least one of the above tests will fail. The IA can therefore determine whether the information gathered

through the metrics and certificates is to be trusted. If all tests succeed, the IA can be confident that the components measured by the TS on the platform are exactly the ones attested to by their suppliers.

The challenger IA can then decide whether or not to trust the platform, based on its trust policies (depending whether it trusts the Certification Authorities involved, for the intended purpose).

Alternative Implementations. In this protocol, some of the processing required from the challenger IA might be impossible or induce poor performance on a smartcard (i.e. Certificate Verification, fetching Expected Metrics if they are not provided by the TS). The challenger IA may therefore have other options in order to validate the steps of the protocol that it cannot carry out by itself:

1 The TS itself can provide the facility to do some of the integrity verification processing on behalf of the IA, and simply report the results to the IA. Then it is left to the IA to draw conclusions based on these results. This is possible as long as the IA has a means of validating the TS's identity against the certificate provided for its public key, in order to decide whether it can trust it.

2 The IA may have access to a trusted third party networked server (such as the server of the entity that issued the IA for use in a particular context). It can then use this server as an off-card resource to have some of the integrity verification processing carried out. Again, it will be required that the IA is aware of the cryptographic identity of such a server, in order to be able to authenticate and trust it.

In the two alternatives above, different levels of granularity can be implemented in the reporting of the metrics. This leaves more or less work to be done by the IA in order to obtain information from which it can make a decision on whether to trust the platform for a specific purpose. This granularity is left as a parameter to be tuned according to the context of each specific application in which the IA is involved.

Options 1 and 2 should be made accessible to the IA through the platform's TS PCSP. Here are some examples of how some of the processing of an integrity response can be delegated (these are valid whether the trusted party is a networked server or the TS itself).

- The IA delegates the verification of certificates (and possibly the fetching of Expected Metrics if necessary) to a specific trusted party. It will then receive the result of these verifications, and decide whether it should proceed and compare the measure Integrity Metrics with the Expected Metrics.

- The IA delegates the verification of certificates AND the comparison of measured and expected Integrity Metrics to a trusted server.

In all cases, the IA will obtain a final result of that comparison, and follow its own policy to decide whether or not to trust the platform to which it is connected.

AUTHENTICATION. Authentication of the TS is part of the integrity verification mechanism. But it is also present as a separate functionality for the IA to verify the identity of the TS. This is of particular benefit when the IA wants to delegate most of the integrity verification interpretation processing to a trusted party, but still wants to authenticate the TS itself.

SECURE STORAGE. As mentioned in the previous section, this function is available on the TS. It allows an IA to use the TS to store information securely off-card.

Depending on the implementation, the TS will enforce access control to data store in secure storage based on a more or less reliable authentication mechanism (i.e. cryptographic authentication, shared secrets, etc)

From the point of view of using secure storage by an IA, the way the TS authenticates the IA may or may not matter. When an IA wishes to store data off-card, it should only do so once it has established that it can trust the platform. Depending on how much it trusts the software reported to be running on that platform, the IA may or may not trust that no rogue system-level software could eavesdrop the data sent to the TS.

The IA can always decide to encrypt the data itself before sending it to the TS for off-subsystem storage.

4.3. SECURING ACCESS TO OFF-CARD RESOURCES

This subsection discusses how Trusted Platform technology for an IA terminal helps secure access to off-card resources.

In our model it is essential that the challenger IA be aware of the identity of at least one root of trust (i.e. a specific TS, a Certification Authority, a Trusted Third Party Server).

This root of trust must allow the IA to establish trust in the identity of the TS of the platform to which it is connected.

This allows for the IA to make sure that when it verifies the integrity of a platform by querying its TS, it will be able to authenticate this specific TS, and have the option to subsequently require all data sent by the platform to be digitally signed by its TS.

With these new functionalities (of a Trusted Computing Platform), an IA can be programmed to verify the integrity of the platform to which it is connected. It can therefore carry some policy logic relating to platform configurations it will accept to interact with.

Communications with a Service Provider (in the sense of the PCSPs described in 4.2.1), in order to make use of an off-card resource, can then require authentication. The IA should be able to request that all communications from the PCSP be digitally signed by the TS. This will give the IA confidence in that the PCSP it is communicating with is indeed a piece of software on the platform whose TS it checked for integrity.

An IA is now able to allow communications only with the platform it has checked for integrity.

Once an IA has established that the local terminal can be trusted, it will be able to trust the behaviour of the terminal software and hardware. It will thus be comfortable in accessing local or networked resources and making use of them. This may include access to an off-card file storage facility, access to the terminal's display, using the terminal for user input to the card, etc. In the case of networked resources, the IA also has the option to use end-to-end encryption to protect its communications.

An IA can therefore establish a trust relationship with its terminal's TS, and securely make use of the facilities provided on that platform to access off-card resources.

5. AN APPLICATION: ESTABLISHING USERS' TRUST IN THEIR PCS

This section describes how the technology presented in the previous section can be used to establish trust between a user and his/her Trusted Platform. This is done through a two-fold process whereby the user's smartcard is able to establish trust in the local platform (as described in the previous section), and through the user trusting his/her smartcard.

Once these conditions are satisfied, it remains for the smartcard to convey its conclusions about the platform's trust to the user, and there are at least two ways of doing this.

These aspects are considered in more details below.

5.1. USER TRUST IN THE IA

First of all, the user has to know how his/her smartcard (namely IA) works and has to trust that the smartcard has proper Intelligent Adjunct functionality. This trust may come from the user getting the IA from a trusted party, such as a corporate IT department, or under contract with a Service Provider. Alternatively, the user could have been involved in

creating the private key of the IA so that he/she is able to verify the cryptographic function in a trusted manner. For example, a purchaser of an IA may be able to configure his/her own smartcard to operate according to user preferences.

5.2. IA TRUST IN THE LOCAL PLATFORM

For the purpose of allowing the user to trust a computing platform, the IA will interact with the user's platform's TS in two ways in order to determine its own trust in the local platform:

Firstly, the IA authenticates the TS of the platform. One way to do this is for the IA to be aware of the identity of the TS in advance, such as a public key of the TS, for the purpose of authenticating the TS. Alternatively, the IA might simply rely on a Certification Authority that it trusts to certify TS identities. Or the IA might trust a Third Party server to which it will delegate authentication of the TS identity.

Secondly, the IA checks the integrity of the platform, including the integrity of the boot process, the integrity of the OS loader, and so on. This follows the process described in Section 4.

Once the IA has established trust in the local platform, it is essential that the IA requires that all communications with the local platform are signed by the TS (after authentication). This is the only way to ensure that the IA carries on communicating with the same TS that it has authenticated and challenged in order to verify the integrity of the platform.

Finally, the IA may need a means of establishing that the target TS is indeed the TS on the local platform. How this is done, and whether it is necessary will depend on the application domain in which the IA is involved. Ways to achieve this include:

1 The IA will consent to interact with an application on the platform it has checked for integrity, only if it is satisfied that the software it has identified to be running on this machine operates with a smartcard inserted in a reader local to the platform. Let us consider the example of a smartcard logon. The IA can be required to verify integrity of the platform before consenting to release credentials for smartcard logon. It may itself require that the integrity verification allows it to identify that the version of the software on the platform it is checking for integrity will only accept logon credentials from a smartcard that is local to the system.

2 The IA will encrypt all data it provides to applications on the local platform under the public key of the TS it has authenticated. This

ensures that this data will only be available to applications on the platform that it has established it can trust.

3 The IA will use the mechanism described below in 5.4 Explicit Feedback subsection to ask the user to enter a PIN number to confirm that the user has seen the "secret" image displayed on the local platform.

These mechanisms should be built into the functionality with which the IA is designed to provide its user. The above methods are examples of how the IA can trust that the platform it has established it can trust is indeed the local platform to which it is connected.

5.3. USER TRUST IN THE LOCAL PLATFORM

As it has been mentioned earlier, this paper uses the following meaning of trust: if a user trusts a platform, it means that the user believes that the platform always behaves in the expected manner for the intended purpose. So, a user's trust in a computing platform is dependent upon whether he/she is able to find out in a trusted manner whether the BIOS, the OS-loader, the Operating System, etc. in the platform have been properly installed and operate correctly. If the user trusts the IA, and if the IA is able to verify the integrity of the platform and convey the result to the user in a trusted fashion, then the user is able to directly infer whether the platform may be trusted or not for the intended purpose.

5.4. CONVEYING THE IA'S FINDINGS TO THE USER IN A TRUSTED MANNER

It is obvious that a smartcard lacks a direct interface to the user to do any kind of reporting. So after the IA completes authentication and integrity checking of the platform, how can it convey the result of whether the platform can be trusted or not to the user?

There are a number of different methods to let the IA tells its owner the result of authentication and integrity checking. The following are two examples.

- IMPLICIT FEEDBACK

  The IA can be programmed in such a way that it will never accept an interaction with an application, such as for the purpose of authentication or for provision of cryptographic services, unless it can first authenticate and verify the integrity of the computing platform to which it is connected. More precisely, when a user uses his/her IA to log on to a platform or a service, he/she inserts the IA

into the smartcard reader. Then the IA communicates with the TS inside the platform, authenticates the TS and checks the integrity of the platform, as described earlier.

If the IA is satisfied with the result of authentication and integrity verification, it will accept to present its credentials to the platform or the server, and the user will be logged in or the service provided successfully. From this the user can infer that the IA was satisfied with the platform's integrity.

If however the IA is not satisfied with the result of authentication and integrity verification of the platform's TS, it will refuse to present the user's credentials for logon or service provision. In this case the expected login or service provision will not occur and the user can infer that the IA does not trust the platform.

This technology may benefit a Service Provider. The Service Provider may provide customers with an IA that only operates correctly when it can verify that the computing entity to which it is connected has passed various integrity checks specified by the Service Provider on the IA. This can be used in a number of applications. It provides the Service Provider with a number of functionality, such as making sure the IA can protect the user from connecting to a spoof service, or for other purposes such as providing a user with functionality to be used off-line that can still be protected through the use of the IA.

- EXPLICIT FEEDBACK

  Alternatively, the IA can use the platform's display in order to present the user with information describing the computer platform's state. The IA simply uses the platform's display as an off-card resource.

  This can be enhanced using an image specific to the IA that is displayed on the visual display unit. For instance, the IA may contain a difficult-to-recreate image data, preferably known only to the user. Only the platform's TS should be able to retrieve this image data from the IA. The platform should have no other way of obtaining the image data except by satisfying the user's IA integrity verification and authentication requests.

  The user can visually identify with a high degree of accuracy that the image is genuine (by visual inspection). The user then has increased confidence that the computing entity has in fact interacted with the IA, and satisfied the IA's requirements to trust the TS - for otherwise the image would not be obtainable.

With either of these two methods, the trust relationship between the IA and the platform can be conveyed to the user. As a result, the user can reliably infer whether or not the platform is to be trusted for the intended operation.

6. CONCLUSIONS

Together with the functionality of the Trusted Computing Platform ([2],[3]), the Intelligent Adjunct model provides much greater functionality to the world of online service provision. Users can now be issued with tokens that will increase the flexibility of use of online services with higher assurance of their security.

As we have shown, IA coupled with a Trusted Computing Platform also provides a new means for a user to establish trust in their computing platform.

Intelligent Adjunct technology now provides a brand new smartcard usage model that becomes usable, and can integrate perfectly in today's world of Internet-based electronic services. We believe it will allow for the provision of whole new types of services that current technology would not permit, due to both a lack of client-end security and user interaction flexibility.

References

[1] B. Balacheff, B. Van Wilder, and D. Chan. Smartcards - From Security Tokens to Intelligent Adjuncts. in Proceedings of Cardis98, Springer-Verlag, In publication.

[2] B. Balacheff, D. Chan, L. Chen, S. Pearson, and G. Proudler. How can You Trust a Computing Platform?. To appear in the Proceedings of Information Security Solutions Europe Conference (ISSE 2000), Barcelona, Spain, September 2000.

[3] D. Chan, L. Chen, S. Pearson, and G. Proudler. Computing Platform Security in Cyberspace. vol. 5, no. 1, pp 54-63, Elsevier Information Security Technical Report, 2000.

[4] Trusted Computing Platform Alliance. founded October 11, 1999. www.trustedpc.org.