28

Protecting the Creation of Digital Signatures with Trusted Computing Platform Technology Against Attacks by Trojan Horse Programs

Adrian Spalka, Armin B. Cremers and Hanno Langweg
Department of Computer Science III, University of Bonn
Roemerstrasse 164, D-53117 Bonn, Germany
Email: adrian@cs.uni-bonn.de

Key words: Digital Signatures, Trojan Horse Programs, Trusted Computing Platform

Abstract: Digital signatures are a key technology for many Internet-based commercial and administrative applications and, therefore, an increasingly popular target of attacks. Due to their strong cryptographic properties an attacker is more likely to subvert them with malicious software, ie Trojan horse programs. We show that by fusing two techniques, our WORM-supported reliable input method and the Intelligent Adjunct model of the Trusted Computing Platform Alliance, we can achieve a high degree of protection from Trojan horse programs during the process of creating digital signatures. Existing software products immediately benefit from our results. Moreover, we examine three ways of storing and executing the signing software with respect to its susceptibility to Trojan horse programs and identify the most suitable combination.

1. INTRODUCTION

Digital signatures are recognised by the industry, the government and the administration, to name just a few, to be the driving technology for the provision of a wide spectrum of Internet-based services. The permanent success of business-to-consumer e-commerce, e-government, e-administration – one is tempted to say e-everything in view of the recent developments – depends, by and large, on the acceptance of these services

by a substantial number of ordinary private end-users. This will only come if the services are equipped with a sufficient level of security.

Confidentiality of a communication over the Internet is enforced by encryption algorithms. The digital signature ensures its integrity and the identification of the communicating parties or, to be more precise, the digital signature either confirms the integrity of a document and the claimed identity of its sender or uncovers a discrepancy or a lack of correspondence. By their definition, digital signatures employ cryptographic algorithms, usually a hash-function and a public-key system.

The cryptographic algorithms used today are strong in the sense that the computing power necessary to forge a digital signature or to break confidentiality is unavailable to any individual, party or even institution. Since this claim is backed by both theoretical and practical results, an adversary easily recognises that an attack on the cryptographic algorithms is of little avail. Though as such highly positive, for this setting to be invincible one assumes that the adversary has access only to the data travelling over the Internet and, in particular, has no access to the computers of the communicating parties.

While no user would deliberately start a malicious program on her PC, the history of attacks based on viruses, worms, Trojan horses and other rouge programs convincingly demonstrates that an attacker can have various ways to start an attack on the user's PC. The Internet is also a perfect means for distributing such programs. Trojan horse programs often come camouflaged as a nice utility, which offers the user some useful function and, at the same time, performs malicious functions in the background. The user just needs to down-load and install it. Whereas an organisation or a company can establish a security policy that minimises such threats, the end-user at home, ie, the envisaged group of digital signature holders, will consider down-loading and running programs, plug-ins etc. from the Internet a normal operation – or at least their children do so.

The threat posed by Trojan horse programs is real. In their comprehensive investigation of current issues related to electronic commerce Lacoste *et al* (2000)[1] state:

> In spite of sufficiently secure existing algorithms, the technical possibility of obtaining signatures in an underhanded way, aided by such mechanisms as Trojan horse attacks, cannot be ignored. Such attacks might result in an unpredictably high damage for the key holder, especially if he cannot prove that an attack has happened. Hundreds or thousands of transactions can be made within a short time by an attack. [. . .] This means: A software solution for digital signatures might be

[1] Cf Lacoste *et al* (2000), pp 233.

> completely correct and nevertheless cannot prevent malicious Trojan horse attacks.

This statement goes in line with our observation. The digital signature is expected to gain the same level of validity as the human manual signature does any time soon. Due to its universal usability, eg, it can be used for buying a car, casting a ballot in a vote or changing the registration of a car over the Internet, it will be a popular target for attacks. And since the cryptographic algorithms cannot be broken, Trojan horses are likely to subvert the process of digitally signing documents.

We present some of the current approaches of dealing with Trojan horses, or generally, with untrustworthy programs in the subsequent section. One can observe there two extreme views. It is either assumed that all programs on the PC are trustworthy or that any program can be untrustworthy. We believe that, for practical purposes, both are unrealistic. On the one hand, there can always be one untrustworthy program on a PC, on the other, some kinds of programs are more likely to be untrustworthy than others due to the effort on the attacker's side. To give an example, it is much more difficult and complex for an attacker to replace a system internal driver than to write a trendy utility.

On these grounds we have decided to examine the combination of two techniques for strengthening the defence against attacks by Trojan horses on digital signatures. Firstly, the Trusted Computing Platform Alliance, a group of almost 150 companies, including major industry players, promotes an Intelligent Adjunct model presented by Balacheff et al (2000). It aims to ensure that a computer's key components, including major parts of the operating system, are checked for their integrity before being used. This check does not guarantee the absence of Trojan horses, but, if passed, it guarantees that these components have not been modified in an unauthorised manner. Thus, if we can take that the official hardware and system software vendors do not implant malicious functions in their products, then the checked part of the PC constitutes a trusted software subset. Therefore, if all components involved in the creation and verification of a digital signature can use this trusted subset to check their own integrity, then some attacks by Trojan horses can be prevented and some detected. The group of remaining attacks becomes significantly smaller and significantly harder to assemble and to put in place innocuously. Thus, secondly, we propose the inclusion of an inexpensive software WORM medium, which ensures a reliable, ie, unmodified input to the signing software.

Moreover, we address three scenarios of storing and executing the signing software:

- signing software installed on the local hard disk and executed as a regular program
- storage on smart card, execution in secure JVM
- storage and execution on smart card, employing intelligent adjunct technology

We show that, together with our previous findings, the use of a JavaCard as secure software storage and the intelligent adjunct technology we gain higher security and platform-independence.

2. PREVIOUS AND RELATED WORKS

Currently there are three different approaches to facilitate the use of digital signatures in insecure environments. We refer to them as 'secure hardware', 'mental arithmetic', and 'secure software'.

While the first two approaches yield a provably high strength against Trojan horse attacks they are expensive, difficult to use, and not flexible in regard to the data that they are able to sign. The latter 'secure software' approach usually disregards Trojan horse attacks in current implementations. Nearly all surveyed products did not bother to include protective measures against malicious processes on the same computer. On top they are difficult to use and inflexible. All three existing approaches force the signatory to explicitly review the data that is going to be signed before it is processed. The user must do this even if she just finished working with it in her application software. This sharply reduces the ease-of-use.

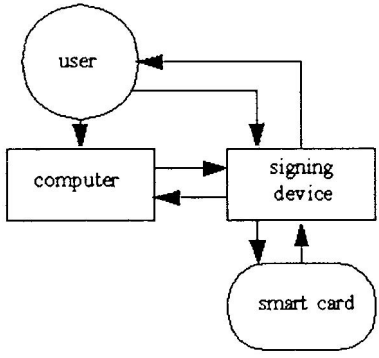
On the Cardis 2000 conference on smart card research and advanced applications Balacheff et al. proposed a technology to use smart cards in a partly secure environment. Their paper combined an 'Intelligent Adjunct Model' with a 'Trusted Computing Platform' approach pursued by some 150 hardware and software manufacturers. We show that a combination of the 'secure software' approach and Balacheff's 'Intelligent Adjuncts' on a 'Trusted Computing Platform' can be used to protect the creation of digital signatures against Trojan horse programs on a computer.

2.1 Using secure hardware devices

This approach is favoured by most academic institutions and pursued by Cryptovision GmbH of Germany. They propose a device that comprises a liquid crystal display (LCD), a smart card reader, and a certified circuit board that contains the operating logic for the signing device; costs are estimated to be less than five thousand Euro each. The flow of information is shown in the sketch.

The computer is used as the provider of the information that shall be signed. Since the computer is assumed as completely insecure the signing device does not get the correct data if this has been manipulated by a Trojan horse on the computer. The signing device and the signature smart card are the only trustworthy components in the eye of the user. So the user has to review the data the signing device received to verify that it is the data she indeed wants to sign.

The data is displayed in a standardized way, eg, rich text format or common word processor file formats without advanced options. This implies that the presentation on the signing device does not always match the presentation on the signatory's computer. The receiver of the signed data may need the same signing device or a software viewer that displays the data like the signing device does. The signatory either confirms or rejects the presentation. After confirmation the data is sent to the signature smart card that actually computes the signature. The signature is then being transmitted through the signing device to the user's computer.



A Trojan horse that modifies the data before it reaches the signature smart card is detected by the user because she will note any modification of the data. Since the presentation and confirmation take place on a trustworthy device the signature is computed for exactly the data the signatory wants to get signed. The weakest component in this approach is a lazy user who does not review the data for reasons of convenience.

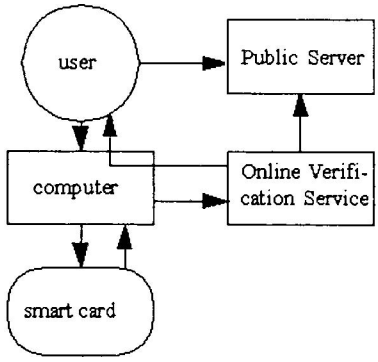
Drawbacks of the system are the small (and fixed) number of accepted file formats, the restriction to data that can be displayed on an LCD (eg, no audio data), the high costs that exceed the price for most personal computers, the high expenditure to roll out updates, and the reduced ease-of-use that diminishes the understanding and the support by the prospective users.

2.2 Neglecting arithmetically-challenged users

To avoid the high costs of additional hardware on the signatory's side T. Stabell-Kulø introduced an approach that forces the signatory to compute simple cryptographical operations by mental arithmetic. The user's computer is regarded as completely insecure and thus can not be trusted. A Trojan horse can alter every communication between the user and components connected to the computer. So Stabell-Kulø proposes to introduce an 'On-

line Verification Service' and a 'Public Server' to the PKI, and a One-Time-Pad and a substitution table for the user.

A signature computed by the smart card is sent to the (trusted) On-line Verification Service. This service verifies that the signature matches the data for which it was computed. It then encrypts the data by applying the substitution table and the One-Time-Pad and transmits this encrypted information through the insecure computer to the user; the signature is sent to the Public Server and marked as 'not yet released'. The signatory applies the One-Time-Pad and the substitution table to the encrypted information and retrieves the decrypted data. She compares if the decrypted data matches the data she wanted to sign. Decryption is done without the untrustworthy computer and takes approximately one minute for 15 characters if the user is a computer science student; otherwise the decryption rate averages 7.5 characters per minute. We did not find a signing method that has a lower ease-of-use. If the user agrees with the signed data she sends a release command to the Public Server that involves the use of a hash value.



While the advantage of obtaining a reliably computed signature in an insecure environment is not bad, the drawbacks are clearly disenchanting: the volume that can be signed is very low, the approach is not suitable for arithmetically-challenged people, you have to produce and securely distribute One-Time-Pads and substitution tables, and you have to introduce an On-line-Verification-Service and a Public Server to the Public Key Infrastructure.

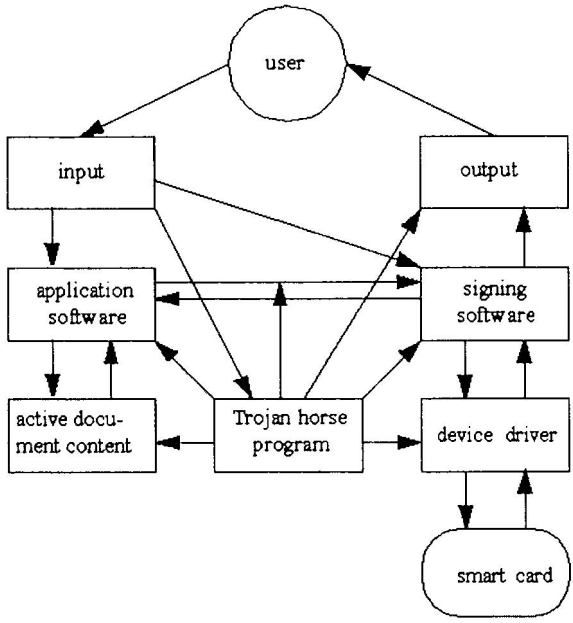
2.3 Implementing 'secure' software

This is the most common approach found in commercial products that are already being shipped to customers. We found that even leading companies disregard the threats posed by Trojan horses. One of the presumed market leaders was vulnerable to basic attacks. When asked why they did not use protection against Trojan horses they responded that it lies in the responsibility of the user to avoid the execution of untrustworthy processes on her computer. That is simply not suitable for a product used by inexperienced people.

The user works with an application software and at some point decides to sign the data. The data is then transmitted from the application software to the signing software, displayed for confirmation by the signatory, and finally sent to the signature smart card that computes the signature.

In most implementations a Trojan horse program has many interfaces it can attack: inside the application software as active document content, between the application software and the signing software, the signing software itself can be a target as well as the device driver between signing software and smart card.

The signing software displays the data that is going to be signed in a standardized way and prompts the user for confirmation. All but a few manufacturers assume that Trojan horses will not attack their software or explicitly put the responsibility for a Trojan horse-free environment in the signatory's domain.



In contrast to the first two approaches this is a low-cost approach without additional hardware devices or PKI components. However, it still has a low ease-of-use since the data has to be reviewed once more even if the user has worked with it for a long time in the application software. And no company has made efforts to effectively block Trojan horse attacks on a conceptual basis.

2.4 Intelligent Adjunct model

In the Intelligent Adjunct model, the adjunct (ie, the smart card) is given control over off-card resources. The card can use all the resources, like peripheral components, the terminal it is connected to can use. Unlike former models that viewed the smart card as a passive component, in the Intelligent Adjunct approach a smart card can initiate transactions.

Connections to peripheral components are channelled through an Intelligent Adjunct service provider. This provider can be integrated in existing standards, eg, the PC/SC standard for accessing smart cards and terminals on the Microsoft Windows platform. Details of the implementation of the Intelligent Adjunct model are provided by Balacheff et al.

2.5 Trusted PC platform

The Trusted Computing Platform Alliance is an industry work group focused on enhancing trust and security on computer platforms. They have developed a specification for verifying the integrity of computer components. The goal is to build secure software applications on top of a verified computing base without the need of additional expensive hardware.

Starting from a hardware module that is resistant against software attacks, the components involved in the boot process of a computer are verified whether their integrity is untouched. The results of these measurements can be reliably retrieved by software that needs certain components to be trustworthy.

At the time this paper was written the Trusted Computing Platform Alliance had almost 150 member companies, including major industry players.

3. PROTECTING THE INTEGRITY OF THE SIGNING PROCESS

We combine the secure software approach with the Trusted PC platform. A negligence of the current secure software implementations is the lack of a check whether the peripheral components that are used can be trusted. We check the integrity of these components employing the integrity measures of the Trusted PC platform initiative. Thus we can rely on a safe input and output assuming that our PC's components do not include a Trojan horse on time of delivery by the clearly identified manufacturer who provides the integrity metrics.

We will elaborate on three scenarios regarding the storage of the signing software. The first scenario is the classic implementation found on computers today; the software is installed on the PC and stored on its local hard disk. The second scenario involves a Java virtual machine (JVM) that receives the code of the signing software from the smart card. This reduces the need of the smart card to verify the integrity of the installed signing software. It only needs to establish trust in the operating system which can include the JVM as a part. Our third scenario eliminates the need of a trusted JVM in the operating system. The signing software is executed on the smart card itself and communicates with the user by use of the intelligent adjunct model.

3.1 Verifying integrity of vital components

The Trusted Computing Platform Alliance (TCPA) proposes to introduce a so-called Trusted Platform Module (TPM) to a PC. It is a hardware component that can not be subverted by software attacks. It acts as the root of the integrity verification process.

When the computer is turned on the TPM verifies that the PC's BIOS conforms to the TCPA specification and thus can be trusted. The BIOS then verifies the operating system loader, the operating system loader verifies the operating system, the operating system verifies its JVM and so on. While the verification of the integrity of the operating system poses a challenge still not solved we are confident that this task will be addressed by the TCPA's member Microsoft.

Software running on a system with a TPM can request the results of integrity measurements to decide which components it will trust. As long as the path from the TPM to the software consists of TCPA-compliant components the software can rely on a trusted computing platform. This does not imply that the platform is free of Trojan horse programs. But it provides the means to identify the manufacturer of a Trojan horse since modifications of third parties can be detected.

3.2 Secure execution of the signing software

The signing software is usually installed on the signatory's computer once and onwards executed by loading it from the local hard disk. To avoid a Trojan horse attack on the signing software we have to check the integrity of the installed software before execution. We propose to store the software on the signature smart card itself. Since the functionality of the software can be kept limited we believe it would be feasible even with today's smart cards' capabilities. The software would be loaded to the operating system's JVM after its integrity has been verified by the card.

A third way would be to execute the signing software on the card itself. Suppose the signature smart card is a JavaCard we already have a trusted JVM available on the card. With the intelligent adjunct model the card gains access to the components needed to interact with the user. The card has previously established trust in the components by issuing a challenge to the TPM to check integrity of the components it is going to use.

Theoretically, a fourth way comes into mind. The signing software could be stored on the local hard disk and be executed in the signature smart card. We do not see advantages in this approach in contrast to storing the software on the card. If we have to load it to the card's JVM anyway, we prefer to store it on the card.

3.3 Trusted input

In the same way that one usually does not sign a blank piece of paper and let someone fill in the text at a later date, without having control, we do this on a computer system. The document that will be signed is being worked on with some application software, eg, a word processor. This application software may not be trustworthy and may represent a Trojan horse program. A big help for a Trojan horse is the labelling of the document as 'to be signed'. Thus it is identified and becomes an interesting target for modification by an attacker.

We withhold this information from a potential Trojan horse by not allowing a direct connection of application and signing software. It may seem convenient to just click 'sign' in your application software, but it reveals the purpose of the document you work with. Instead, every application software has a standard 'save work' function that stores a document for further processing without determining the exact purpose. A document could be opened by the same user with the same application or a different one, by another user on another machine, it could be stored on backup media. In order to not risk being detected a Trojan horse program must abstain from altering a document that is just saved. Thereby we get an unmodified document that we now will transfer unmodified to the signing software.

This approach works with attackers who want to modify specific documents and benefit directly from the signatory signing these data. The assumption does not hold for attackers with the sole aim of vandalism. Here it would be useful to verify the file's contents with a secure viewer, hardened against Trojan horse programs. We state below that the file is protected against modifications after saving and hence can not be altered by a Trojan horse program then.

Our method to ensure a reliable transfer to the signing software is not complicated. We use a WORM medium (Write Once Read Multiple). This medium does not allow modifications of a file after the file is closed. The file can only be read but not modified or deleted. A WORM can be implemented and details on the implementation are presented in an earlier paper (Cremers et. al. (2001)). For reasons of cost-efficiency we prefer a SWORM (=Software WORM), that does not require additional hardware. In the Trusted PC concept we hide the implementation and give the intelligent adjunct access to this secure storage.

Once we have stored the document to be signed in the SWORM as a trusted source we can use it to apply the signature to it. Some signature laws require that the user must be provided with another possibility to display the data the signature is applied to immediately before the signature creation.

This is perfectly possible with our solution. The presentation of the data can be done by opening the read-only file with the application software or by using a different software with the single purpose of displaying documents before signing.

The signing software opens the file stored on the SWORM and sends it to the signature smart card using a cryptographically-secured channel. The smart card is tamper-resistant and computes the signature for the data sent to it with the secret signing key of the signatory. To prove that the signatory is present the card usually requires a PIN input from the user. A secure input can be achieved by using a cheap smart card reader with an attached dedicated keyboard (eg, Cherry G81-8015).

3.4 Reliable presentation of signed data

We have stated that the data that is signed has no semantics in itself. A signed paper document can be interpreted by the parties involved and by a third party, eg, a court. You have a fixed presentation that is not altered by the way you look at it. In computer systems we have to use application software to give the data a meaning. The same binary data will be interpreted more or less different by different software products. While the same binary data has been signed by the smart card the interpretation of the signed document can be different at the site of the receiver without modifying the signed document itself.

A Trojan horse that is transferred with the signed document as active document content, eg, a macro, can alter the presentation on the receiver's computer. So the receiver may have an otherwise Trojan-free system but will nevertheless get a presentation not intended by the signatory. The simplest solution would be to disable active document contents at all. But this may reduce flexibility way too much.

There are two ways we propose to tackle this problem. The first way is a softer approach than just disabling active content. It should be possible to restrict the actions of active contents. So the receiver of a signed document should be able to determine which actions active content could perform on the document that would not alter the semantics. This requires cooperation of application software manufacturers regarding these options for their products. Today you can usually choose between allowing a macro to perform all actions or none.

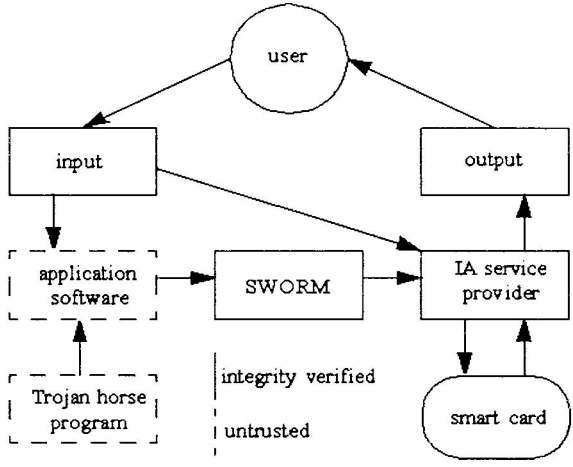
The other more promising method employs that a computer is a deterministic machine. So it is in principle capable of presenting the signed document in the same way as it has been on the signatory's computer. This can be done without cooperation of the application software manufacturers. The idea is to build a "sandbox' around the application that presents the

signed document. This sandbox sets all environment parameters that can be determined by active document content inside the application software and be used against a deterministic presentation. The environment parameters are collected at the signatory's computer and include user name, computer name, network address, application software parameters etc. All parameters are included in an enhanced signature of the document so they can be evaluated by the sandbox on the receiver's computer. The same parameters will lead to the same presentation. This sandbox approach is especially suitable for the verification step.

Sandbox software products already exist. Those tools usually focus on limiting the capabilities of potentially malicious programs by denying access to resources. They could be modified to simulate an environment similar to the one in which the signing took place.

4. IMPLEMENTATION OF A SECURE SIGNING METHOD IN SOFTWARE

Our method identifies components of the computer for which we have verified integrity and thus trust them to behave in the expected manner.



The document the user is going to sign is prepared in application software we do not trust. We transfer the document with a standard operation to a SWORM medium so a Trojan horse will not attempt to alter the data before it is signed. Transfer from the SWORM to the signature smart card to compute the signature can be achieved by one of three techniques depending on the place of storage and execution of the responsible signing software. Traditionally, software stored on a hard disk is used, but we encourage the

use of software stored as a Java application on the signature smart card. The software is then loaded to a verified JVM on the PC or executed in the card's JVM, communicating by Intelligent Adjunct technology with the outside world.

4.1 Trusted PC components

In the Trusted Computing Platform approach there is a path starting at the TPM along which the integrity of the components is verified. For our method this path must lead from the TPM to the operating system and must include input and output peripheral components, the SWORM medium, and the Intelligent Adjunct service provider.

The application software the user runs to create and modify her documents does not have to be TCPA-compliant. We avoid Trojan horse interference by reducing crucial information for an attacker and securing an early input as described in sections 4.4 and 4.6. Vandalism is still possible to a certain extent.

4.2 Intelligent Adjunct functions

The Intelligent Adjunct service provider (IASP) coordinates communication between the signature smart card and off-card resources. An IASP will usually be part of the operating system and thus its integrity will have been verified. If it is not part of the operating system, its manufacturer has to provide TCPA-compliant integrity information.

Components that are addressed by way of the IASP are the Trusted Platform Module, input and output devices, the SWORM medium, and the JVM of the operating system.

4.3 Signing software stored on signature JavaCard

Obviating the need of secure storage of signature software on a signatory's computer is clearly an advantage. Even with TCPA technology the software manufacturer has to provide TCPA-compliant integrity information with each software update. By use of a JavaCard as secure software storage we gain higher security and platform-independence. Execution of the software occurs on a verified JVM and the application uses only components of which the integrity has been verified beforehand.

The signature smart card manufacturer can provide the signatory with a platform-independent solution for signing digital documents. The application is integrated on a single smart card and can be used with every terminal that

implements the proposed methods, regardless whether it is the user's own computer or a public signing terminal.

As long as current signature smart cards are not capable of executing the signature smart card in their own on-card JVM, the software has to be transferred to the (verifiably trustworthy) JVM on the terminal's side.

4.4 Early unchangeable input for signature creation

The proposed use of a SWORM is central to our concept since it is now possible and feasible to keep the input for the signature fixed at a very early stage. In contrast to other approaches we get the input at the earliest time possible before a Trojan horse even knows that the document will be signed and thus become interesting.

To achieve a high level of security, the implementation of the SWORM should look like a non-SWORM medium. This prevents active document content from testing if the target of a standard 'save work' command is a SWORM medium

Details on how we implement the SWORM medium can be found in an earlier paper (Cremers et. al. (2001)).

4.5 Exact data labelling for deterministic presentation

The proposed sandbox method requires that we gather as much input parameters for a deterministic presentation as possible. Since we do not rely on cooperation with the application software manufacturers we have to pick up the parameters at various places.

We think that the following parameters must be included with the signature to make a possible Trojan horse on the receiver's side believe it is executed on the signatory's machine. These parameters are the document format, the application used for working with the document, the version of the application, the parameters for the application (stored in the Windows system registry or in a configuration file), the size and colour depth of the Windows desktop, available fonts on the system, information about the origin and integrity of the fonts, the user name, the machine name, the network address, number and labels of storage media, serial numbers of the machine and application.

The sandbox is built around the application used to present the signed document. Since not every program is capable of being run reliably in a sandbox it may prove necessary to disable active document content entirely on the system in those situations.

Current sandbox software products operate on a simple allow/deny-basis for access to information and computing resources. They have to be

enhanced to provide applications with (misleading) information about the environment in which applications are executed.

4.6 Reducing valuable information for corrupt participants

We rise the risk of detection for a Trojan horse by withholding information it needs for a successful attack. The document is transferred to the SWORM medium without determining the purpose of the transfer. A Trojan horse that modifies the document anyway when it is saved takes a high risk in being detected. By definition, a Trojan horse program has to keep its existence a secret, so it will not risk being detected. Otherwise the origin of it could be traced and the attacker be held liable in a court. So even if a Trojan horse resides on the signatory's system it is not able to catch the right time to interfere with the signing process.

The reduction in information for an attacker is enforced by a user policy. A signatory is advised not to use seamlessly-integrated plug-ins in her application to trigger a signing process. Instead it is necessary to direct the output of the application to the SWORM and making it look like a standard and not suspicious action.

5. CONCLUSION

Trojan horse programs (THP), ie, programs with additional hidden, often malicious, functions, are more and more popular forms of attack. On the one hand, high-level macro programming languages in many office applications make it easy even for inexperienced attackers to write, hide and distribute a THP. On the other, the emerging digital signatures are likely to become a favourite target of attacks. The owner of a digital signature can face considerable damage if a THP is able to sign a document the user did not intend to.

By identifying components of the signatory's computer of which we can check integrity we assemble a trusted computing base. We use this base to execute the signing software in a safe environment.

First of all the signatory determines the data that she wants to sign in the application software way ahead of the actual computation of the signature. It is not necessary to perform an additional presentation of the data in the signing software. Second the transfer of the data through the signing software to the signature smart card is secured by the Trusted PC platform.

Third the solution can be achieved without additional expensive hardware, thus lowering the financial burden of security.

Active document contents still pose a special problem if a user is not willing or able to disable their execution. To ensure that a document is displayed identically on both the sender's and receiver's display, both parties must run the same document processing program with the same profile, ie, the same command-line parameters, options etc. Thus, in addition to a document and its signature, the sender must include the name, version and used profile of her document processing program, which the receiver must use to view the document. This is supported by using a sandbox approach in the signature verification step.

In conclusion, we have shown that the threat of Trojan horse programs attacking a document's integrity can be averted with only a few measures, which – compared with previous approaches – retain a system's flexibility and incur only minor inconveniences on its usability.

References

Balacheff, B., D. Chan, L. Chen, S. Pearson and G. Proudler (2000). 'Securing Intelligent Adjuncts Using Trusted Computing Platform Technology'. *IFIP TC8/WG 8.8 4th Working Conference on Smart Card Research and Advanced Applications.* pp: 177-195.

Bontchev, V. (1996). 'Possible macro virus attacks and how to prevent them'. *Computers & Security* 15(1996):595-626.

CERT Coordination Center (1999). *CERT Advisory CA-99-02-Trojan-Horses.* http://www.cert.org/advisories/CA-99-02-Trojan-Horses.html

Cremers, A. B., A. Spalka and H. Langweg (2001). 'Vermeidung und Abwehr von Angriffen Trojanischer Pferde auf Digitale Signaturen'. *7. Deutscher IT-Sicherheitskongress.* Bonn, May 2001 [German]

Docherty, P., and P. Simpson (1999). 'Macro Attacks: What Next After Melissa?'. *Computers & Security* 18(1999):391-395.

European Parliament and European Council (1998). 'Directive on a Community framework for electronic signatures'. C5-0026/99 - 1998/0191 - (COD).

Ford, R. (1999). 'Malware: Troy Revisited'. *Computers & Security* 18(1999):105-108.

Hoffmeister, A., Cryptovision GmbH (2000). Personal communication.

Lacoste, G. , B. Pfitzmann, M. Steiner and M. Waidner, ed. (2000) *SEMPER – Secure Electronic Marketplace for Europe.* Berlin et al: Springer-Verlag.

Lapid, Y., N. Ahituv and S. Neumann (1986). 'Approaches to Handling "Trojan Horse" Threats'. *Computers & Security* 5(1986):251-256.

Popek, G.J., and C.S. Kline (1977). 'Encryption Protocols, Public Key Algorithms and Digital Signatures in Computer Networks'. R. A. DeMillo (1978). *Foundations of Secure Computation.* pp:133-153.

Pordesch, U. (2000). 'Der fehlende Nachweis der Präsentation signierter Daten'. *DuD Datenschutz und Datensicherheit* 24.2(2000):89-95. [German]

Schmidt, A. U. (2000). 'Signiertes XML und das Präsentationsproblem'. *DuD Datenschutz und Datensicherheit* 24.3(2000):153-158. [German]

Spalka, A., A. B. Cremers and H. Langweg (2001). 'The Fairy Tale of "What You See Is What You Sign". Trojan Horse Attacks on Software for Digital Signatures'. *IFIP Working Conference on Security and Control of IT in Society-II.* Bratislava, June 2001.

Stabell-Kulø, T. (2000). 'Smartcards: how to put them to use in a user-centric system'. *HUC2K The Second International Symposium on Handheld and Ubiquitous Computing.* http://www.pasta.cs.uit.no/publications/HUC2K.html.

Trusted Computing Platform Alliance (2000). *TCPA Trusted Subsystem Specification Version 0.9.* http://www.trustedpc.org.