The Fairy Tale of ‚What You See Is What You Sign' – Trojan Horse Attacks on Software for Digital Signatures

Adrian Spalka, Armin B. Cremers and Hanno Langweg
Department of Computer Science III, University of Bonn
Roemerstrasse 164, D-53117 Bonn, Germany
Fax: +49-228-734 382, Email: adrian@cs.uni-bonn.de

Abstract. Software for the creation of digital signatures performs a delicate task. The signatory has to trust the manufacturer of the software that it will work in the intended way. Signing a document electronically will have legal consequences in a growing number of countries, therefore the security of the signing software is an important issue. In the past, Trojan horse programs have shown to be of growing concern for end-user computers. Software for digital signatures must provide protection against Trojan horses attacking the legally relevant signing process. In a survey of commercially-of-the-shelf signature software programs we found severe vulnerabilities that can easily be exploited by an attacker.

Keywords: Trojan Horses, Digital Signatures, Security of COTS Software

1 Introduction

Once upon a time, intelligent people proposed digital signatures. The integrity of emails and documents would be secured so no malicious courier would be able to alter the content. Big software manufacturers appeared and announced that they would implement a principle they called 'what-you-see-is-what-you-sign'.[*†] A signatory using her personal computer would not actually understand the bits and bytes her document consisted of, but would digitally sign only what she wanted to sign. Since not every computer could be trusted, eg it could inhibit so-called Trojan horse programs, the software manufacturers called 'trust our software'. It would reveal modifications of text to be signed with a secure viewer component, and shield the secret PIN from eavesdroppers. It turned out that reality is somewhat more complex.

In an attempt to test the protection of current software applications for the creation of digital signatures against attacks of malicious code on the signatory's computer, we found virtually no protection is offered at all.

We tested a couple of signature software programs available in Germany on their susceptibility to Trojan horse attacks. The four programs include both major and minor

* Deutsche Post AG (2001). 'How to Electronically Sign an Email Using eTRUST. [...] This special component ensures that he signs only what he has seen and accepted on screen.' `http://www.signtrust.de/english/etrust/signingemails.htm`

† Utimaco Safeware AG (2001). 'The viewer is a completely independent component for displaying text data and guarantees that only what is seen on the monitor will be signed with the motto: "What you see is what you sign" and "What you see was signed".' *SafeGuard Sign & Crypt FAQ*.

players on the German signing software market. We do not present a complete risk analysis. We focus on some of the most popular and likely attacks that Trojan horse programs might perform on digital signature software. By concentrating on some of the worst problems we show that the vulnerabilites are due to a flawed design.

2 Previous works

Most of the previous approaches to secure the creation of digital signatures focus on the cryptographic algorithms and the secure computation in a smart card. While these problems appear to be solved, the smart card still does not communicate directly with the signatory. The smart card relies on a software that is executed on the signatory's computer to transmit the data that is going to be signed by the card.

Approaches that introduced new and expensive hardware have not proved competitive in the market. Hence, many companies developed signature software products that rely on the secure execution of their software.

We asked manufacturers if and how they protect their software against attacks by Trojan horse programs. The answers we got on trade fairs and in personal communication were that protection would be 'surely interesting but expensive if not impossible'. In fact, the protective measures applied seemed more to decrease ease-of-use than to prevent attacks.

In this case study we verify the manufacturer's statements.

3 Types of conducted attacks

In our study we concentrate on four vulnerabilities a Trojan horse program will be likely to exploit. The first is capturing the PIN code for accessing the signatory's private signing key. Second, we observe if the data can be modified between finishing work in the application software and processing it in the signing software. Some products require the user to review the data before it is actually sent to the signature smart card. In that case a malicious program would have to alter the displayed data for review to avoid detection. Hence, we will treat the second and third vulnerability as a single one. The last problem we focus on is a meet-in-the-middle-attack on the smart card terminal device driver.

We do not present a comprehensive risk analysis and vulnerability report for all signature products available. This is an obligation of the software manufacturers who claim that their products are secure and who keep saying 'what you see is what you sign'. By concentrating on some of the worst problems we show that the vulnerabilites are due to a flawed design.

3.1 Capturing the PIN for access to the PSE

For increased security of the storage of the signatory's private signing key most people advocate the use of a smart card. The card is called a *Personal Security Environment* (PSE) because it is in the possession of the user who usually presents a PIN to the card for authentication purposes. Since it is assumed that the signatory will not lose both her smart card and reveal the corresponding PIN this method is regarded reliable.

In current implementations of this protocol the smart card does not know if the PIN is provided by the user or by a third party, eg an attacker. If an attacker gains access

to the PIN and the user inserts her signature smart card into the smart card reader attached to the computer, the attacker could establish communication with the card, provide the PIN and after successful authentication begin signing messages with the signatory's private key.

Most programmers tend to be lazy when coding security features that are not specified sufficiently. In case of a PIN entry on the Microsoft Windows platform, they use a standard edit dialogue element and set its property *PasswordChar* to '*'. This allows a user to type in her PIN or password while displaying asterisks for every character typed. However, displaying asterisks does not suffice, since the typed characters are still provided by the edit dialogue element. We will show in a later section how the PIN is retrieved in detail.

3.2 Modifying the data to be signed

The user works with her application software to create and modify the data she wants to sign. At some point she decides to finish her work and sign the data. The document is then transferred from the application software to the signing software, sometimes displayed again for confirmation, and finally transmitted 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.

When the manufacturer proposes an additional confirmation step before sending the data to the signature smart card, the signing software displays the data that is going to be signed in a standardised way and prompts the user for acceptance. 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 this case we may ask why they use a smart card at all. If the computer on which most of the vulnerable signing process takes place has to be free of Trojan horses by definition, the need for a tamper-resistant PSE is not understandable.

We will show that modification of the display in a so-called secure viewer component is possible in a commercially-off-the-shelf software product by one of the market leaders for signing software in Germany.

3.3 Interfering with the communication between software and PSE

When the smart card receives data for signing with the signatory's private signing key it does not know if this data originated from the signatory. It requires a PIN to be sent to verify that the signatory is present when the data is received. After authentication the smart card usually receives a hash value of the data that the signature is expected to be computed on.

An attacking Trojan horse program that places itself between the signing software and the signature smart card can observe the communication between the two parties. After the PIN has been sent to the card the malicious program could alter the communication and send different data to the card than that the user is expected to getting

signed.

We show in a later section that the signing software of one of the market leaders accepted a smart card terminal driver that was not provided by the operating system.

4 Surveyed products

We chose the products by a simple selection process. They had to be easily available for purchase in Germany and should be able to work with a smart card. The first requirement turned out to be not so easy to fulfil as we had thought beforehand. Even the products of the presumed market leaders Deutsche Post and Deutsche Telekom took effort to persuade employees that the software was for sell and they needed some weeks to arrive.

According to surveys we conducted on trade fairs (CeBIT 2000 & 2001, Hanover; Systems 2000, Munich), none of the other software manufacturers regarded Trojan horse programs a threat for their signing software solutions and, hence, did not protect the signatory against such attacks. So while the selection is not representative in the strict sense of the word, it comprises products with the typical vulnerabilities.

The tests were conducted mainly on the Microsoft Windows 98 platform, and in part on the Microsoft Windows 2000 platform.

4.1 eTRUST Mail

Deutsche Post AG, the former state-owned postal service, launched it's digital signature solution on the CeBIT 2000 trade fair. Three months after, we bought a starter kit that consists of a signature smart card, registration with the manufacturer's trust centre, a smart card reader, and the signing software *eTRUST* 1.01.

eTRUST is designed to be integrated into the Microsoft Outlook email software. The user writes an email, selects 'sign', then 'send'. The email is then presented to the user (again) for confirmation. After confirmation the user is required to provide the PIN for access to the signature smart card. The email message is sent to the card, signed, and the signature is attached to the email.

There are a couple of Trojan horse program-related problems with this product. We are able to obtain the PIN for the card, can modify the data that is going to be signed without knowledge of the signatory, and *eTRUST* accepts our smart card reader driver for communication with the smart card.

4.1.1 Asking for the PIN - and getting it

Obtaining the smart card's PIN is done with standard Trojan horse methods. A similar attack had received broad attention by the media in 1998. Deutsche Telekom's T-Online ISP service had been shown to be vulnerable against password retrieval. The access password was stored in a standard edit dialogue element and was protected only against visual attacks by displaying asterisks. Nevertheless the password could be made visible or retrieved by another program with virtually no effort. We thought that a newly-developed program two years after the embarrassing T-Online incident would provide at least basic protection against this attack. T-Online meanwhile had modified their password input and is no longer vulnerable to this kind of attack.

Our attack takes place when the user has finished entering her PIN and before she

clicks ‘proceed’. The first step is to obtain a handle to the PIN edit field, then kindly ask for the PIN in it:

```
(relevant code in Delphi)

hWindow:=GetWindow(Self.Handle,GW_HWNDFIRST);
While (hWindow <> 0) do
Begin
   If (hWindow <> Self.Handle) and
      IsWindowVisible(hWindow)
   Then Begin
      GetClassName(hWindow,szClass,SizeOf(szClass));
      GetWindowText(hWindow,szText,SizeOf(szText));
      If (StrPas(szClass) = '#32770') and
         (StrPas(szText) = 'PIN-Eingabe')
      Then Begin         // Found the PIN entry window
         hWindow:=GetWindow(hWindow,GW_CHILD);
         While (hWindow <> 0) do
         Begin
            GetClassName(
               hWindow,szClass,SizeOf(szClass));
            If StrPas(szClass) = 'Edit'
            Then Begin   // Found the PIN edit control
               SendMessage(hWindow,WM_GETTEXT,
                  WPARAM(SizeOf(szPIN)),
                  LPARAM(@szPIN));
               // Show PIN
               editCapturedPIN.Text:=StrPas(szText);
               hWindow:=0;
            End
            Else hWindow:=GetWindow(
                              hWindow,GW_HWNDNEXT);
         End;
         hWindow:=0;
      End;
   End;
   If hWindow <> 0
   Then hWindow:=GetWindow(hWindow,GW_HWNDNEXT)
End;
```

We iterate through the windows on the Windows desktop. Once we have found the window for PIN entry, we iterate through the controls in that window. Luckily, the PIN entry control is the only control of the *Edit* type. We then send a *WM_GETTEXT* message to the control and get the PIN to our *szPIN* buffer. For our purposes we just show the PIN in our application. We can as well start communicating with the signature smart card to create signatures with the signatory’s private signing key.

In Microsoft Windows 2000 the *WM_GETTEXT* message will only get a response if it is sent from within the same application. This is a step forward. Becoming part of the signature application is nevertheless possible. We put a modified *WINSCARD.DLL*

in the *eTRUST* folder. Upon accessing the DLL by *eTRUST* we start a separate thread to capture the PIN.

4.1.2 Modifying the secure viewer's presentation

The next component we targeted was the so-called secure viewer. It is central to the manufacturer's what-you-see-is-what-you-sign concept. Since modification of the data before it is processed by the signing software is not prevented, the data is presented to the user for confirmation. Thus, if a malicious email software or plug-in has altered the data in a way the signatory does not want it to be, she can decline confirmation and thus be prevented from signing unwanted documents.

The data presented in the viewer can be easily modified. Here is how:

```
(relevant code in Delphi)

// Get the handle of the viewer's window
// Class '#32770'
// Title 'Visualisierung der Email'

hWindow:=GetWindow(hWindow,GW_CHILD);
While (hWindow <> 0) do
Begin
   GetClassName(hWindow,szClass,SizeOf(szClass));
   If StrPas(szClass) = 'RICHEDIT'
   Then Begin   // Found the viewer's RICHEDIT control
      SendMessage(hWindow,EM_SETREADONLY,
         WPARAM(false),0);
      // insert for instance text from the clipboard
      // into the viewer
      SendMessage(hWindow,EM_SETSEL,WPARAM(0),
         LPARAM(-1));
      SendMessage(hWindow,EM_PASTESPECIAL,
         WPARAM(CF_TEXT),LPARAM(0));
      SendMessage(hWindow,EM_SETREADONLY,
         WPARAM(true),0);
      hWindow:=0;
   End
   Else hWindow:=GetWindow(hWindow,GW_HWNDNEXT);
End;
```

Like in the PIN capturing example, we iterate through the handles to obtain the handle for the desired control. The protection applied here by the manufacturer helps against the user typing in the viewer, since the *RICHEDIT* control is set to *read-only* state. We send a message to the control and ask for dropping the read-only restriction. Then we select all the text in the control, and override the selection with the text we earlier copied to clipboard. Since we are nice, we set the control's state back to read-only.

4.1.3 Monitoring communication between software and smart card

After the user has entered the PIN for accessing her Personal Security Environment, ie the signature smart card, the data to be signed is sent to the card. We place a library in the installation folder of *eTRUST* to monitor the communication to the card.

We provide the file *WINSCARD.DLL* for PC/SC-compliant smart card communication in the folder where the *eTRUST* program files are stored. Obviously, the signing software does not load the DLL from the Windows system folder but examines the search path in standard order. Hence, we were able to observe and modify communication between the *eTRUST* software and the signature smart card.

The smart card provided by Deutsche Post uses some proprietary commands we have not yet identified. But since the PIN is provided with secure messaging and the hash value that is to be signed is not, we assume that we could have altered the hash value to make the card sign the data an attacker wants to be signed contrary to the signatory. The manufacturer did not deny that.

4.1.4 Comments of the manufacturer

When confronted with the results of our analysis, Deutsche Post neither confirmed nor denied any of the vulnerabilities. They retreated to the position that every software running on the Microsoft Windows platform is susceptible to Trojan horse attacks and it is Microsoft's and the user's responsibility to provide protective measures or a Trojan horse-free computer.

In the manual for *eTRUST Mail* the user is advised to check that her personal computer cannot be manipulated by others and that there are no malicious programs on her computer. In case there are, Deutsche Post will not make any statement concerning the integrity of the signing process. On the other hand they justify their 'secure viewer' component with the possibility that some program might try to sign data different from the one the user wants to sign.

We think this is not sufficient for a product that targets a mass market. A company that wants to become market leader in digital signature software solutions must offer its customers a product they can rely on. Saying that Trojan horse programs may exist but are the signatory's business is not user-friendly.

Deutsche Post told us they would issue a patch but they were unable to provide it in time to be included in this survey.

4.2 Utimaco SafeGuard Sign&Crypt

Sign&Crypt is a product of Utimaco Safeware AG. They were one of the first companies in Germany to offer a signature software with a viewer component. Their homepage named two distributors for their products. One of them was able to ship.

We first tried to retrieve the PIN from the *Edit* control used for input. It displays asterisks to shield the input and it is possible to navigate the cursor with the arrow keys to edit the PIN. Sending a WM_GETTEXT message yields '####', so they prevent this easy attack.

The viewer component gets it input by submitting output to the 'Digital Signature' virtual printer. Thus a Trojan horse knows that this data is going to be presented in the viewer component and signed afterwards. It is possible to alter the display in the viewer component, but the component re-draws the displayed data frequently. Hence, it is difficult to forge a different presentation.

To access the smart card reader Utimaco provides a proprietary service application. Thus, we did not look at the communication between software and card.

4.3 T-Telesec PKSCrypt

Deutsche Telekom is Germany's formerly state-owned telephone company. They are the market leader on the telecommunications market in Germany and have plenty of top-educated and highly-skilled employees. Their security subsidiary T-Telesec offers a digital signature software with smart card integration since 1998 and is presumably one of the first companies to do so. Hence, we expected to easily get a first-class product.

Buying the software was not so simple. We had to persuade some employees that Deutsche Telekom really offers that product for sale and that they surely have the correct forms to fill out for the application for a signature smart card. Some weeks later we actually received the software. An email plug-in is not included, but the software integrates neatly with the Microsoft Explorer to sign files with a click of the right mouse button.

The package comes with a signature smart card that has to be initialised with a user-chosen PIN. The card is initially in a state that it will only accept a first PIN as a command. The first thing we did was to examine the PIN entry. *PKSCrypt* 1.11 offers two ways to enter the PIN. The first is called 'standard input', the second 'secure input'. We checked both.

With 'standard input' the user is supplied with a standard password dialogue window that displays asterisks instead of numbers when she enters her PIN. This sounds familiar, and it takes no effort to kindly ask the software for the PIN like we explained in detail for *eTRUST*. (Did we mention that Deutsche Telekom's subsidiary T-Online got a lot of embarrassing attention three years ago for a similar lazy implementation in another product?) By the way, 'standard input' is the default setting in *PKSCrypt*.

The alternative option of 'secure input' impresses the user with a numeric keypad on the screen that shows a permutation of the ten number keys. The user is supposed to enter the number next to the number her PIN contains. In our example screen shot we would enter '429380' if our PIN was '123456'. Since the permutation is different each time a Trojan horse can ask for the PIN but it will never get the correct one (unless, of course, the permutation is the identity function, but the developers will certainly have thought of that).

We thought this would going to be hard, that we would have to extract the permutation with image recognition techniques from a screen shot. But it is simpler than that. The window consists of the edit control with the permuted PIN, some buttons and a lot of controls of the *Static* type. We get the permutation as easy as the PIN. Yes, we have to write some dozen more lines of

code, but that is it.

```
Static ( 26, 41)-( 52, 59) "7"
Static ( 23, 28)-( 41, 43) "7"
Static ( 77, 41)-(103, 59) "5"
Static ( 74, 28)-( 92, 43) "8"
...
Static ( x1, y1)-( x2, y2) "n"
```

The *Static* controls contain a number each. By their position, which is also provided, we can determine the correct permutation.

Since the permutation of the input aims at an attacker who can access the PIN edit control, it is not understandable why the attacker is provided with the permutation. In a comment of the manufacturer Deutsche Telekom they said that the intent of the 'secure input' was to protect the PIN from attackers who watch the user entering it or who use a camera aimed at the keyboard.

They use the 'CT' API for communication with the smart card reader. This is a standard fairly common in Germany; it is not compatible with the PC/SC standard. Thus, we had to write another dll file to intercept messages sent between application and signature smart card – only to find out that Deutsche Telekom had done a good job and had secured the important messages with secure messaging according to ISO 7816.

4.4 Siemens/SSE TrustedMIME

The steps needed to sign an email with *TrustedMIME* are straight-forward. A user chooses a 'sign' symbol and is asked for a password after clicking 'send'. After password input the message is signed without verification for integrity by the user. Retrieving the PIN works like in *eTRUST Mail* with modified character sequences to determine the correct window.

Siemens provides signature smart cards to their employees only. We used a card for field tests that cannot be obtained regularly. The communication between *TrustedMIME* and the card could not be monitored and altered with the *WINSCARD.DLL* we employed with *eTRUST*. The application refused to work with our modified dll file.

4.5 GDtrust Mail

Manufacturer Giesecke & Devrient was not able to provide a signature smart card, even three months after applying for one. So we excluded smart card communication from our examination.

GDtrust Mail 4.0.2 provides a plugin for Microsoft Outlook. To sign a message, the user selects a 'sign' button. After she clicks 'send', she is asked for the PIN to access her PSE. There is no way to detect if the message has been modified beforehand by Outlook, an Outlook plug-in, or by another process on the computer.

Of course, it was possible to retrieve the PIN like we did with Deutsche Post's *eTRUST*. We only had to modify the character sequences to look for.

4.6 Comparison

Almost all surveyed products use a PIN entry method that can easily be defeated by Trojan horse programs. Even *PKSCrypt's* 'secure input' provides no barrier for an attacker. The only difference is that it is less user-friendly than 'standard input'.

All surveyed products did not prevent modification of the input data before sending it to the signature smart card or processing it internally. *eTRUST* Mail displays the data in standardised format but does not protect this presentation. *Sign&Crypt* is better at this but still not perfect. *PKSCrypt* shows the file name but offers no way to verify that the file has not been modified. The other two products simply disregard this issue.

Product	PIN captured	Input modified	Display altered	Transfer to card intercepted
eTRUST Mail 1.01	yes	yes	yes	yes
Sign&Crypt 2.10	no	yes	yes* (cf. text)	(untested)
PKSCrypt 1.11	yes	yes	n/a	secured
TrustedMIME 2.2.5	yes	yes	n/a	(untested)
GDtrust Mail 4.0.2	yes	yes	n/a	n/a

5 Protection against the presented attacks

We show how the main vulnerabilities that most of the surveyed products revealed can be averted. However, it should be noted that the focus of our paper is to depict the flaws in existing products in an area where you could expect secure implementations. Details of our solution can be found in two earlier papers.*

5.1 Reliable input data

We propose to use a SWORM medium to ensure untampered input to the signing software (*Software-based Write Once Read Multiple*). 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. The data is stored on the SWORM medium and cannot be modified after the file has been closed. It is not necessary to perform an additional presentation of the data in the signing software.

5.2 Secure PIN input

You can achieve a secure PIN input by using a piece of specialised hardware for that purpose. The keyboard manufacturer Cherry offers a smart card reader (G81-8015) that is integrated in a keyboard. After a special command for secure PIN input, key presses on the numerical keypad are transmitted directly to the smart card. The computer operating system receives asterisks instead, so no process, malicious or not, gets to know

* Cremers, A.B., Spalka, A. and H. Langweg (2001). 'Vermeidung und Abwehr von Angriffen Trojanischer Pferd Programme auf Digitale Signaturen'. 7. Deutscher IT-Sicherheitskongress, Bonn, May 2001. [German]
Spalka, A., Cremers, A.B. and H. Langweg (2001). 'Protecting the Creation of Digital Signatures with Trusted Computing Platform Technology Against Attacks by Trojan Horse Programs'. IFIP/SEC conference, Paris, June 2001.

the secret PIN. During secure PIN input a red LED on the keyboard glows to raise user awareness of the secure PIN input mode. This LED cannot be manipulated by other commands.

It is possible to defeat a simple retrieval of the PIN – as *Sign&Crypt* shows. To circumvent the risk of keyboard loggers we favour the PIN input directly at the smart card terminal.

5.3 Detecting tampered input to the smart card

The activation of a computation of a signature must ensure that the card will compute the signature for the intended data. Thus, providing the authentication data has to be linked with the data to be signed.

There are basically two ways to achieve this. You can release the authentication information after the card has proved that it has received the intended data to be signed. On the other hand you can sign the data to be signed with a session key and provide this key together with the authentication information. The recipient can recover the session key only if the authentication information is correct.

Both protocols involve some secret knowledge of the signature application to communicate with the signature smart card. We propose to secure a PIN input in the signature application or to use transaction numbers that can be used only once.

6 Conclusions

With our analysis we only focused on some attacks that every attacker with no inside knowledge of the signing software can perform. This does not rule out more sophisticated attacks by advanced attackers with malicious intent.

In their current implementations the surveyed products offer almost no protection against attacks by Trojan horse programs. Legislation, especially in the European Union, tends to see the responsibility to prove that a signature has been faked on the signatory. A signatory using one of the products our paper deals with cannot be sure that digital signatures will only be computed for the data she intends. The only way to employ the current versions of the programs responsibly would be to use them in a Trojan horse-free environment – which cannot be assumed rationally regarding today's personal computers.

We strongly recommend that the responsible software manufacturers act to incorporate protective measures into their products. The threat of Trojan horse programs is real and at the current stage of shipped products an attacker can just walk through open doors. In earlier papers we proposed easily to implement, user-friendly, and cost-effective methods to dismiss a significant amount of possible vulnerabilities.

Hopefully, one day there will be a software that adheres to the what-you-see-is-what-you-sign principle and they will all live happily together ever after.

References

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

[2] Bundesregierung der Bundesrepublik Deutschland (1997). *Verordnung zur digitalen Signatur (SigV).* Bundesgesetzblatt I, S. 2498. Inkrafttreten 01.11.1997. [German]

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

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

[5] Deutsche Post AG, Ed. (2000). *eTrust Mail. Signatur und Verschlüsselung von eMails nach deutschem Signaturgesetz. Handbuch zur Integration in Microsoft Outlook*. [German]

[6] Deutscher Bundestag (1997). *Gesetz zur digitalen Signatur (Signaturgesetz - SigG)*. Bundesgesetzblatt I, S. 1870, 1872. Inkrafttreten 01.08.1997. [German]

[7] Deutscher Bundestag (2001). *Gesetz über Rahmenbedingungen für elektronische Signaturen und zur Änderung weiterer Vorschriften.* Bundestagsdrucksache 14/5324 vom 14.02.2001. [German]

[8] European Parliament and European Council (1999). 'Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures'. *Official Journal of the European Communities* No. L 13(2000):2. http://europa.eu.int/ISPO/ecommerce/legal/documents/1999_93/1999_93_en.pdf.

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

[10] Fox, D. (1998). 'Zu einem prinzipiellen Problem digitaler Signaturen'. *DuD Datenschutz und Datensicherheit*. 22.7 (1998). 386-388. [German]

[11] Graf, Deutsche Telekom AG (2001). Personal communication.

[12] International standard ISO/IEC 7816-4. *Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 4: Interindustry commands for interchange.*

[13] Kalkreuth, T., Deutsche Post AG (2000). Personal communication.

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

[15] Mackert, A., Giesecke & Devrient GmbH (2000). Personal communication.

[16] Münchmeier, W., Cherry GmbH (2001). Personal communication.

[17] Olfs, D., Siemens AG (2000). Personal communication.

[18] 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*:133-153.

[19] Potzner, R., Utimaco Safeware AG (2001). Personal communication.

[20] Spalka, A., Cremers, A.B. and H. Langweg (2001). 'Protecting the Creation of Digital Signatures with Trusted Computing Platform Technology Against Attacks by Trojan Horse Programs'. IFIP/SEC conference, Paris, June 2001.

[21] Utimaco Safeware AG (2001). SafeGuard Sign & Crypt FAQ. http://www.utimaco.de/internet/uti_know.nsf/51356cb01fbb194b4125666b00482767/3a606701690efe01c12567ae002821ea?OpenDocument