

SecSigner 5

Benutzerhandbuch

Version 5.05

14. September 2017

Autoren: Tilo Kienitz, Ulrich Heller, Tobias Tiedt
© 2006-2017 SecCommerce Informationssysteme GmbH
www.seccommerce.com

Inhaltsverzeichnis

1 Einführung.....	5
1.1 Anwendung.....	5
1.2 Eigenschaften.....	6
1.3 Benötigte Soft- und Hardware.....	6
1.3.1 Unterstützte Betriebssysteme.....	6
1.3.2 Java Laufzeitumgebung.....	7
1.3.3 Signaturkarten und Kartenlesegeräte.....	7
1.4 Trustcenter Online-Anbindung.....	8
1.5 Initialisierung der Signaturkarte.....	8
2 Auslieferung und Konfiguration des SecSigners.....	9
2.1 Auslieferung: SecSignerDeveloper_[Version].zip.....	9
2.2 Installationsversion.....	11
2.3 Aufruf des SecSigners.....	11
2.4 Funktionsablauf bei der Signatur.....	11
2.5 Konfigurationsdatei.....	12
2.5.1 Sprache.....	12
2.5.2 Systemverhalten.....	12
2.5.3 Signaturerzeugung.....	13
2.5.4 OCSP.....	14
2.5.5 Zeitstempel.....	15
2.5.6 Dateiendungen.....	15
2.5.7 Hilfeseiten.....	16
2.5.8 Schnittstellen zum Kartenleser.....	17
2.5.8.1 Ausschluss einzelner Kartenleser.....	17
2.5.9 Signaturerstellung mit mehreren Kartenlesern.....	18
2.5.10 Software-Schlüssel.....	18
2.5.11 Layout.....	19
2.5.12 Sichere Anzeige.....	19
2.5.12.1 XML-Anzeige.....	20
2.5.13 Externe Vieweranwendungen.....	21
2.5.14 Verschlüsselung.....	21
2.5.15 PDF-Annotation.....	21
2.5.16 Prüfbericht speichern.....	23
2.5.17 Signaturprüfung.....	23
2.5.18 Prüfaufträge an SecPKI senden.....	24
2.5.19 Applet-Darstellung.....	24
2.5.20 Attributzertifikat automatisch einlesen.....	24
2.5.21 Maximale Dateigröße.....	25
2.5.22 Nutzung von Test-Zertifikaten.....	25
2.5.23 Logdatei.....	25
2.5.24 Signaturdatenauswahl durch den Nutzer.....	26
2.5.25 SecSigner-Lizenz.....	27
2.5.26 Signaturformat im Drag&Drop - Dialog.....	27
2.5.27 Zertifikatstapel laden.....	27
2.5.28 Algorithmenkatalog laden.....	28
2.5.29 Firewalls/Proxy-Server.....	28
2.5.29.1 Automatische Proxy-Konfiguration bei Windows.....	28
2.5.29.2 allgemeine Konfiguration.....	28
2.5.29.3 Konfiguration pro Benutzer.....	28

3 Integration des SecSigners in Signaturanwendungen.....	30
3.1 Integration in Java-Anwendungen.....	30
3.2 Integration mittels CallSecSignerDLL.....	30
3.3 Integration in HTML-Seiten mit Java-Web-Start.....	30
3.3.1 SecSignerWebStart.....	31
3.3.2 Mehrfachsignatur mit SecSignerWebStart.....	37
3.3.3 SecVerifierWebStart.....	37
3.4 Integration in HTML-Seiten mit Java-Applets.....	40
3.4.1 SecSignerApplet.....	40
3.4.2 Mehrfachsignatur mit SecSignerApplet.....	47
3.4.3 SecVerifierApplet.....	47
3.5 Aufruf aus der Kommandozeile.....	50
3.5.1 Eingebette Signatur.....	50
3.5.2 Verschlüsselte Signatur.....	50
3.5.3 PDF-Signatur.....	50
3.5.4 PDF-Signatur prüfen.....	51
3.5.5 XML-DSig-Signatur.....	51
3.5.6 XML-DSig-Signatur prüfen.....	51
3.5.7 Daten nur verschlüsseln.....	51
3.6 Lizenzdatei.....	51
4 Prüfung des sicheren Betriebszustandes.....	52
4.1 Selbstprüfung des SecSigners.....	52
4.2 Prüfung der Konfigurationsdatei.....	53
4.3 Prüfung des Zertifikatstapels.....	54
5 Signaturprüfung.....	55
6 Erzeugen einer digitalen Signatur.....	60
6.1 Dokument laden.....	60
6.2 Initialisierung.....	60
6.3 Sichere Dokumentenanzeige.....	62
6.4 Signatur.....	62
6.4.1 Kartenlesegerät mit eigener PIN-Eingabe.....	62
6.4.2 Kartenlesegerät ohne eigene PIN-Eingabe.....	62
6.4.3 Signaturfortschritt.....	63
6.4.4 Zeitstempeldienst.....	64
6.4.5 Bestätigung der Signatur.....	64
7 Weitere Funktionen des SecSigners.....	66
7.1 Online-Zertifikatprüfung.....	66
7.2 Attributzertifikate einbinden.....	66
7.3 Entschlüsselung.....	66
7.4 Verschlüsselung einer Signatur.....	67
7.5 Verschlüsselung einer Datei ohne Signatur.....	68
7.6 Ablage des Verschlüsselungszertifikats als Datei.....	68
7.7 PDF-Signatur-Annotation.....	68
8 SecSigner und Windows.....	70
8.1 Installierbare Version.....	70
8.2 SecSigner als Adobe Plug-In.....	70

8.3 SecSigner als CryptoServiceProvider (CSP).....	71
8.4 Initialisierung der Signaturkarte.....	71
8.5 Installierter SecSigner unter Windows.....	71
8.5.1 Erzeugen einer Signatur.....	71
8.5.2 Signieren und verschlüsseln.....	73
8.5.3 Daten nur Verschlüsseln.....	74
8.5.4 Ablage des Verschlüsselungszertifikats als Datei.....	74
8.5.5 Entschlüsselung.....	76
8.5.6 Zertifikatprüfung.....	76
8.5.7 PDF-Signaturen prüfen.....	77
8.5.8 Signatur mehrerer Dateien mit einer PIN-Eingabe.....	77
 9 SecSigner – Aufruf für Mac OS X, Linux, Windows.....	 79
 10 Impressum.....	 81

1 Einführung

In Deutschland werden die Rahmenbedingungen für rechtlich verbindliche digitale Signaturen durch das **Signaturgesetz** und die **Signaturverordnung** festgelegt.

Nur die darin definierte **qualifizierte elektronische Signatur** ist der persönlichen manuellen Unterschrift gesetzlich gleichgestellt und garantiert somit die Rechtsverbindlichkeit der unterzeichneten Willenserklärung; die Signatur ordnet ein Dokument mit qualifizierter elektronischer Signatur eindeutig seinem Urheber zu. Mit der qualifizierten elektronischen Signatur von Ausgangsrechnungen haben Sie die Möglichkeit, Rechnungen vollelektronisch konform zu §14 UStG zu senden und zu empfangen und auf einen Papierversand zu verzichten.

Für die Handhabung von Dokumenten mit einer elektronischen Signatur benötigen Sie eine bei einer Zertifizierungsstelle erhältliche **Signaturkarte**, auf der ein öffentlicher und ein privater Schlüssel gespeichert sind. Den privaten Schlüssel benötigen Sie, um Dokumente mit einer digitalen Signatur zu versehen. Ihren öffentlichen Schlüssel können andere Nutzer bei der Zertifizierungsstelle abrufen, um Ihre Signaturen zu prüfen.

Die Signaturkarte wird mit einem **Kartenleser** an den PC/Server angeschlossen. SecSigner und unsere anderen Produkte unterstützen eine Vielzahl marktgängiger Kartenleser und Signaturkarten. Für Details verweisen wir Sie auf unsere Webseite.

SecSigner signiert Dokumente oder andere Dateien (Text, HTML, XML, PDF, Office-Dokumente, Bilddokumente, ...) mit qualifizierter elektronischer Signatur und bietet daneben weitere Funktionalität wie:

- Hochsichere **Dokumentenanzeige** (Text-, Bild-, PDF-, HTML-, XML/BMU-Dateien)
- **Mehrfachsignaturen** (mehrere Signaturen pro Dokument: '4-Augen-Prinzip')
- **Massensignaturen** (mehrere Signaturen mit einer PIN-Eingabe) mit geeigneten Signaturkarten.
- **Zeitstempel**-Einholung vom Trustcenter (soweit als Dienst vom Trustcenter angeboten)
- Signatur-**Verifikation** und Online-Überprüfung (OCSP) von Zertifikaten auf Gültigkeit
- **Verschlüsselung** beliebiger Dokumente

Besonders wichtig ist die Prüffunktion, mit der Sie Urheberschaft und Integrität eines signierten Dokumentes verifizieren können.

Der SecSigner ist verfügbar unter:

<https://seccommerce.com/download-secsigner-secsign-id/>

1.1 Anwendung

SecSigner ermöglicht sowohl die Erstellung als auch die Überprüfung digitaler Signaturen.

Das **Unterzeichnen von Dokumenten** und beliebigen Dateien mit qualifizierter elektronischer Signatur nach dem Signaturgesetz erfolgt mit Hilfe der Signaturkarte, die über das Kartenlesegerät mit dem PC verbunden wird. Sofern der Kartenleser über eine Anzeige verfügt, wird der Inhaber der Signaturkarte dort zur PIN-Eingabe über die Tastatur des Kartenlesers aufgefordert.

Auch ohne Signaturkarte und Kartenlesegerät ermöglicht SecSigner die **Überprüfung** von

elektronischen **Signaturen** und **Zeitstempeln**, inkl. Zertifikatstatusabfrage beim ausgebenden Trustcenter.

Signatur und Überprüfung werden dabei unterstützt durch eine **hochsicheren Anzeige** von Text-, PDF-, HTML- und XML-Dokumenten.

SecSigner ist eine vollständige Signaturanwendungskomponente mit Herstellererklärung nach dem deutschen Signaturgesetz.

1.2 Eigenschaften

Merkmale des SecSigners in der Übersicht: Der SecSigner

- ist eine in Version 2.0.0 nach **Signaturgesetz bestätigte** und gemäß ITSEC E2/HOCH zertifizierte vollständige **Signaturanwendungskomponente**, der die Auslieferung über das Internet gestattet ist,
- ist eine Komponente mit HTML-Aufruf-Interface zur einfachen Integration in Portalanwendungen,
- kann als Komponente in JAVA-Anwendungen oder andere Programme eingebunden werden,
- ist eine benutzerfreundliche Anwendung zum Start **direkt aus dem Internet-Browser** mit sicherer Anzeige von Text-, HTML- und XML-Dokumenten,
- signiert Dokumente oder andere Dateien (Text, HTML, PDF, Office-Dokumente, ...) mit CADES, PAdES oder XMLDSig-Signatur,
- kann Mehrfachsignaturen (mehrere Signaturen pro Dokument: '4-Augen-Prinzip') erzeugen,
- kann hochperformante Stapelsignaturen (mehrere Signaturen mit einer PIN-Eingabe) mit geeigneten Signaturkarten erzeugen,
- erfordert **keine** manuelle **Software-Installation**, da die Anwendung direkt über das Internet geladen und über Java-Web-Start ausgeführt werden kann,
- unterstützt eine Vielzahl marktgängiger Kartenleser und Signaturkarten,
- bietet Zeitstempel-Einholung vom Trustcenter und
- Signatur-Verifikation inklusive
- Online-Überprüfung (OCSP) von Zertifikaten auf Gültigkeit.

1.3 Benötigte Soft- und Hardware

1.3.1 Unterstützte Betriebssysteme

Microsoft Windows Betriebssystem:

- Windows 10
- Windows 8.1
- Windows 8
- Windows 7
- Windows Vista

- Windows Server 2012 R2
- Windows Server 2012
- Windows Server 2008 R2
- Windows Server 2008

oder Linux Betriebssystem:

- SuSE
- Redhat
- Debian
- Ubuntu

oder Mac OS X Betriebssystem:

- Mac OS X

Unterstützt werden nur Versionen der Betriebssysteme, die von ihrem Hersteller noch im allgemeinen Support unterstützt werden.

1.3.2 Java Laufzeitumgebung

SecSigner ist eine Java-Anwendung und setzt auf Ihrem PC eine Oracle Java Laufzeitumgebung (JRE) in einer vom Hersteller unterstützten Versionen voraus.

Diese kann von www.java.com geladen werden.

1.3.3 Signaturkarten und Kartenlesegeräte

Zur Signaturprüfung benötigen Sie *keine* zusätzliche Hardware.

Für die Signatur eigener Dokumente wird eine Signaturkarte benötigt, die über ein Kartenlesegerät an den PC angeschlossen wird. SecSigner unterstützt eine Vielzahl marktgängiger Signaturkarten und Kartenlesegeräte, darunter:

Bezeichnung	Herausgeber/Trustcenter
Elektronischer Personalausweis (nPA)	Bundesdruckerei
T-TeleSec ECC, T-Telesec TCOS 3.0 (RSA)	T-TeleSec
D-Trust card V3.0	D-Trust
BNotK Signaturkarte	Bundesnotarkammer
Datev Signaturkarte	Datev
S-Trust Signaturkarte und Massensignaturkarte	S-Trust
Heilberufsausweis	z.B. dgnService
Mitarbeiterkarte der Deutschen Rentenversicherung	Deutsche Rentenversicherung
a.sign premium Karte	A Trust (AT)
ecard der österreichischen Sozialversicherung	Österr. Sozialversicherung (AT)
SwissSign qualifizierte Signaturkarte	SwissSign (CH)
Swisscom qualifizierte Signaturkarte	Swisscom (CH)
QuoVadis qualifizierte Signaturkarte	QuoVadis (CH)

Tabelle 1: Signaturkarten (Auswahl)

<i>Hersteller</i>	<i>Modell</i>
Celectronic	CARD STAR / medic2
Cherry	G83-6744 SmartBoard ST2000U SmartTerminal
Fujitsu Siemens	Tastatur KB SCR eSIG / PRO
German Telematics	GT900 eHealth BCS
Hypercom	medCompact eHealth
Reiner SCT	cyberJack RFID standard und komfort cyberJack e-com plus cyberJack e-com cyberJack secoder
HID Global (Omnikey)	Cardman 3621 / 3821 eHealth 8751 LAN
ORGA Kartensysteme	ORGA 6041
SCM Microsystems	SPR 532 / Chipdrive pinpad 532 eHealth 200 BCS

Tabelle 2: Kartenlesegeräte (Auswahl)

1.4 Trustcenter Online-Anbindung

Für die Prüfung des Zertifikatstatus ist eine Online-Anbindung an das ausgebende Trustcenter notwendig. Es wird eine Kommunikation vom SecSigner über das Internet zu dem Trustcenter, welches das Signaturzertifikat ausgestellt hat, aufgebaut.

1.5 Initialisierung der Signaturkarte

Um eine Signaturkarte **erstmalig** zu nutzen, schalten Sie diese bitte zunächst mit unserem Administrationstool **SecCardAdmin** frei:

www.seccommerce.com → Produkte / SecCardAdmin

2 Auslieferung und Konfiguration des SecSigners

Dieses Dokument enthält Anweisungen zur Installation und Konfiguration der Signaturanwendungskomponente SecCommerce SecSigner. Es richtet sich an Systemadministratoren, die dieses Produkt in ihrer Umgebung installieren, konfigurieren und in ihre Signaturanwendungen integrieren.

2.1 Auslieferung: *SecSignerDeveloper_[Version].zip*

Die Developer-Version des SecSigner wird in einer Zip-Datei *SecSigner-Developer-[Version].zip* auf der Website zur Verfügung gestellt:

<https://seccommerce.com/download-secsigner-secsign-id/>

Neben der Zip-Datei gibt es dort eine Signaturdatei *SecSigner-Developer-[Version].zip.pkcs7*. Dabei handelt es sich um eine elektronische Signatur eines berechtigten SecCommerce-Mitarbeiters. Die Liste der berechtigten Zertifikate findet sich in Abschnitt 4. Mithilfe dieser Signatur kann geprüft werden, ob es sich um eine unverfälschte von SecCommerce stammende Auslieferung handelt.

Die ausgelieferte Zip-Datei enthält ein Verzeichnis *SecSigner* folgenden Inhalts:

- *SecSigner.jar*
Archiv mit SecSigners Java-Klassen
- *SecSignerExt.jar*
Archiv mit Java-Klassen zum Aufruf des SecSignerApplets und des SecVerifierApplets
- *secsigner.properties*
Konfigurationsdatei
- *SecSigner.selfchecksig*
Herstellersignatur über SecSigner.jar
- *CallSecSignerDLL.dll*
32-Bit-Bibliothek für Windows-Anwendungen zum Aufruf des SecSigners. Calling convention `__cdecl`.
- *CallSecSignerDLL64.dll*
64-Bit-Bibliothek für Windows-Anwendungen zum Aufruf des SecSigners. Calling convention `__cdecl`.
- *CallSecSignerDLLStd.dll*
32-Bit-Bibliothek für Windows-Anwendungen zum Aufruf des SecSigners. Calling convention `stdcall`.
- *call-secsigner.command*
MacOS-X-Skript, das den SecSigner startet
- *call-secsigner.sh*
Wind Unix-Shell-Skript, das den SecSigner startet
- *call-secsigner.bat*
Windows-Batch-Datei, die den SecSigner startet
- *secSignerSign.jnlp*
Beispiel-JNLP-Datei zum Aufruf der Signaturfunktion des SecSigners als Java-Web-Start-Applikation.

- *secSignerVerify.jnlp*
Beispiel-JNLP-Datei zum Aufruf der Verifikationsfunktion des SecSigners als Java-Web-Start-Applikation.
- *SecSignerApplet.html*
Beispiel-HTML-Datei zum Aufruf der Signaturfunktion des SecSigners als Java-Applet.
- *SecVerifierApplet.html*
Beispiel-HTML-Datei zum Aufruf der Verifikationsfunktion des SecSigners als Java-Applet.
- *TestCallSecSignerDLL.exe*
Beispiel-Windows-32-Bit-Anwendung in C, die *CallSecSignerDLL.dll* nutzt.
- *TestCallSecSignerDLL.64.exe*
Beispiel-Windows-64-Bit-Anwendung in C, die *CallSecSignerDLL64.dll* nutzt.
- *TestCallSecSignerDLLCSharp.exe*
Beispiel-Windows-Anwendung in C#, die *CallSecSignerDLL.dll* nutzt.
- *TestSecSigner_Api_CSharp_SignData.bat*
Windows-Batch-Datei, die mittels *TestCallSecSignerDLLCSharp.exe* die Signaturfunktion aufruft
- *TestSecSigner_Api_CSharp_SignData_restart.bat*
Windows-Batch-Datei, die mittels *TestCallSecSignerDLLCSharp.exe* die Signaturfunktion aufruft, SecSigner schließt und erneut startet
- *TestSecSigner_Api_CSharp_VerifyData.bat*
Windows-Batch-Datei, die mittels *TestCallSecSignerDLLCSharp.exe* die Verifikationsfunktion aufruft
- *TestSecSigner_Api_EncryptDataOnly.bat*
Windows-Batch-Datei, die mittels *TestCallSecSignerDLL.exe* die Verschlüsselungsfunktion aufruft
- *TestSecSigner_Api_SignData.bat*
Windows-Batch-Datei, die mittels *TestCallSecSignerDLL.exe* die Signaturfunktion aufruft
- *TestSecSigner_Api_SignData_restart.bat*
Windows-Batch-Datei, die mittels *TestCallSecSignerDLL.exe* die Signaturfunktion aufruft, SecSigner schließt und erneut startet
- *TestSecSigner_Api_VerifyData.bat*
Windows-Batch-Datei, die mittels *TestCallSecSignerDLL.exe* die Verifikationsfunktion aufruft

Ferner sind Unterverzeichnisse enthalten:

- *developer*
Enthält die API-Dokumentation (JavaDoc) des SecSigners, sowie Beispielaufufe in mehreren Programmiersprachen, u.a. den Quellcode von *TestCallSecSignerDLL.exe* und *TestCallSecSignerDLLCSharp.exe*.
- *doc*
Enthält dieses Handbuch.
- *html*
Enthält Beispiel-HTML-Dateien, auf die das SecSignerApplet nach dem Ende seiner Funktion den Browser weiterleitet.
- *test*

2.2 Installationsversion

Für das Betriebssystem Microsoft Windows steht eine installierbare Version des SecSigners als *Setup_[Version].exe* auf unserer Website zur Verfügung:

<https://seccommerce.com/download-secsigner-secsign-id/>

Neben der Zip-Datei gibt es dort eine Signaturdatei namens *Setup_[Version].exe.pkcs7*. Dabei handelt es sich um eine elektronische Signatur eines berechtigten SecCommerce-Mitarbeiters. Die Liste der berechtigten Zertifikate findet sich in Abschnitt 4. Mithilfe dieser Signatur kann geprüft werden, ob es sich um eine unverfälschte von SecCommerce stammende Auslieferung handelt.

2.3 Aufruf des SecSigners

Grundsätzlich gibt es drei unterschiedliche Möglichkeiten, SecSigner in eine bestehende Signaturanwendung zu integrieren und mit ihm zu kommunizieren:

- **HTML-Integration**
SecSigner kann zum Erzeugen einer Signatur oder zum Auslesen der Zertifikate als Java-Web-Start-Anwendung oder Java-Applet in eine HTML-Seite integriert werden (Kapitel 3.4.1). Notwendige Parameter werden mit der HTML-Seite, bzw. in der JNLP-Datei als Parameter übertragen. Das SecSignerApplet, bzw. die SecSigner-Webstart-Anwendung interagiert nicht mit dem Anwender; es nimmt lediglich die Parameter entgegen, lädt zu signierende Dokumente und ähnliches und übergibt sie den JAVA-Schnittstellen des SecSigners. Die erzeugten Signaturdaten können auf dem lokalen Computer gespeichert oder mittels HTTP-POST an einen Server zur Weiterverarbeitung geschickt werden. Während das SecSignerApplet und SecSignerWebStart dem Signieren von Daten dienen, werden das SecVerifierApplet und SecVerifierWebStart in ähnlicher Weise zum Überprüfen von Signaturen verwendet.
- **JAVA-Integration.**
SecSigner kann in eine externe Signaturanwendung integriert werden. Durch diese Integration können Funktionen des SecSigner über eine API gezielt angesprochen und Ergebnisse ausgewertet werden. Diese Art der Ansteuerung bietet den flexibelsten Zugriff, erfordert aber einen vertrauten Umgang mit der Programmiersprache JAVA. Es ist eine Integration als Java-Applet wie auch als Java-Applikation möglich.
- **C-/C#-/ .NET-Integration.** SecSigner kann in externe Windows-Anwendung integriert werden. Zu diesem Zweck bietet die Bibliothek CallSecSigner.dll Schnittstellen an, welche den Aufruf, der SecSigner-Funktionen ermöglichen. CallSecSigner.dll wandelt die Parameter in Java-Typen um und ruft die SecSigner-Java-Schnittstelle auf. Beispiele für die entsprechenden Funktionsaufrufe finden Sie im Verzeichnis „developer“.

2.4 Funktionsablauf bei der Signatur

Unabhängig von der Konfiguration des SecSigner kann er vom Anwender nicht in einen unsicheren Zustand überführt werden. Es wird entweder ein korrektes oder kein Ergebnis geliefert. Um ein Dokument mit Hilfe des SecSigner zu unterschreiben, müssen Module in einer festen Reihenfolge aufgerufen werden:

1. Initialisierung Hauptprogramm
 - Laden der Konfigurationsdatei (Properties)
 - Laden der Treiber (nur beim 1. Start)
 - Selbstprüfung
2. Initialisierung der Signaturkomponente
 - Suche der Kartenleser
 - Ermitteln der verwendeten Signaturkarte
3. Signaturprozess
 - optional: Attributzertifikate einbinden
 - Anzeige des zu signierenden Dokuments
 - PIN-Eingabe
 - Anzeige des signierten Dokuments
 - optional: Zeitstempel erzeugen
 - explizites Einverständnis des Benutzers
 - optional: Verschlüsselung

Eine Beeinflussung der Reihenfolge oder der in diesen Modulen enthaltenen Funktionalität ist nicht möglich. Diese Vorgehensweise garantiert dem Endnutzer einen vertraulichen Umgang mit seinen sensiblen Daten und Dokumenten.

2.5 Konfigurationsdatei

Systemverhalten und Layout des SecSigner können über eine Konfigurationsdatei gesteuert werden. Die Konfigurationsdatei `secsigner.properties` enthält alle frei konfigurierbaren Einstellungen des SecSigner. Diese müssen mit äußerster Sorgfalt vorgenommen werden.

2.5.1 Sprache

`seccommerce.language=[de|en|nl]`

Sprache der SecSigner-Oberfläche. Voreinstellung: *de*

`seccommerce.region=[DE|EN|NL]`

Region der Sprache der SecSigner-Oberfläche. Voreinstellung: *DE*

2.5.2 Systemverhalten

`seccommerce.secsigner.autoclose=[on|off]`

Deinitialisiert den Signaturkartentreiber automatisch nach einem Signaturvorgang. Ist dies erwünscht, so muss für jeden Signaturvorgang die Signaturkomponente erneut gesucht werden. Voreinstellung: *off*

`seccommerce.secsigner.smartcardautosearch=[on|off]`

Startet die Suche nach dem Kartenleser und der Signaturkarte automatisch. Voreinstellung: *off*

secommerce.secsigner.autocommitinit=[on|off]

Bestätigt den Dialog mit Informationen über die Kartenleser und die gefundenen Signaturkarten automatisch. Voreinstellung: *off*

secommerce.secsigner.autoswitchtosigndlg=[on|off]

Überspringt den Dialog mit Informationen über die Kartenleser und die gefundenen Signaturkarten beim Signaturaufwurf, falls die Kartenlesersuche bereits durchgeführt wurde und SecSigner seitdem nicht geschlossen wurde. Voreinstellung: *off*

secommerce.secsigner.autocommit=[on|off]

Zeigt nach dem Signaturvorgang optional einen Bestätigungsdialog an ('off'). Voreinstellung: *off*

secommerce.configdir=<Verzeichnisname>

In diesem Verzeichnis werden Dateien lokal gespeichert. Es handelt sich um z.B. um Logdateien, um verwendete Verschlüsselungszertifikate, um temporäre Kopien von Bibliotheksdateien zum Kartenleserzugriff und ähnliches. Voreinstellung: *secommerce* im Home-Verzeichnis des Benutzers.

secommerce.jvm.version=<Versionsnummer>

Legt eine minimal erforderliche Java-VM-Version fest. Voreinstellung: *1.4.2*

2.5.3 Signaturerzeugung

secommerce.cert.requirequalified =[on|off]

Akzeptiert nur Signaturkarten, die ein qualifiziertes Signaturzertifikat enthalten. Voreinstellung: *off*

secommerce.cert.allowpseudonym=[on|off]

Akzeptiert Signaturkarten, deren Personennamen im Signaturzertifikat ein Pseudonym („PN“) enthält. Voreinstellung: *on*

secommerce.cert.expirewarndays=<Tage>

Gibt an, wie viele Tage vor Ablauf des Zertifikates auf der Signaturkarte der Benutzer aufmerksam gemacht werden soll. 0 bedeutet keine Warnung. Voreinstellung: *30*

secommerce.pkcs7.smimecap=[on|off]

Es werden beim Signieren die S/MIME-Capabilities (Liste der unterstützten Cipher) und das Verschlüsselungszertifikat, mit dem der Unterzeichner Nachrichten entschlüsseln kann, in das PKCS#7-Objekt eingefügt. Voreinstellung: *off*

secommerce.secsigner.showattributecertoption=[on|off]

Ermöglicht die Aufnahme von vorhandenen Attributzertifikaten in die elektronische Signatur. Voreinstellung: *on*

secommerce.secsigner.savedocument=[on|off]

Zeigt eine Schaltfläche nach Signatur eines Dokumentes an, um das signierte Dokument und die erzeugte Signatur auf dem lokalen Dateisystem zu speichern. Voreinstellung: *on*

secommerce.secsigner.checkoldsignatures=[on|off]

Enthalten zur Signaturerzeugung übergebene Daten selbst schon Signaturen, so werden diese zunächst geprüft und das Ergebnis der Signaturprüfung angezeigt. Soll auf diese Prüfung verzichtet werden, so kann die Option '*off*' gesetzt werden. Voreinstellung: *on*

secommerce.secsigner.signrightaway=[on|off]

Nach dem Einlesen der Zertifikatdaten des Unterzeichners kann dieser entscheiden, das

Dokument ohne Darstellung in der sicheren Anzeige zu signieren. Eine entsprechende Option im Dialog wird angezeigt, wenn die Option 'on' gesetzt wird. Voreinstellung: *off*

- ! Wird von obiger Voreinstellung abgewichen, befindet sich der SecSigner nicht in einem
- ! Zustand, für den die Herstellererklärung nach Signaturgesetz gilt.

secommerce.secsigner.autosign=off

Signiert die Dokumente automatisch ohne Benutzerinteraktion. Voreinstellung: *off*

- ! Wird von obiger Voreinstellung abgewichen, befindet sich der SecSigner nicht in einem
- ! Zustand, für den die Herstellererklärung nach Signaturgesetz gilt.

secommerce.secsigner.allowsignaturereplacement=[on|off]

Erlaubt SecSigner eine bestehende Signaturdatei mit demselben Namen zu überschreiben. Voreinstellung: *on*

secommerce.secsigner.autosignaturereplacement=[on|off]

Erlaubt SecSigner eine bestehende Signaturdatei mit demselben Namen zu überschreiben, ohne den Benutzer um Erlaubnis zu fragen. Voreinstellung: *off*

secommerce.secsigner.resignfilesindirectory=[on|off]

Legt fest, dass bei der Signatur eines Verzeichnisses auch diejenigen Dateien signiert werden, die bereits signiert sind. Voreinstellung: *on*

secommerce.tcoscard.signaturecertificatechoice=[0,1,2]

TeleSec-Signaturkarten können sowohl über ein qualifiziertes als auch über ein fortgeschrittenes Zertifikat zur Signaturerzeugung verfügen. Der SecSigner verwendet im Normalfall das qualifizierte Zertifikat [0]. Dieser Parameter erlaubt, stattdessen Signaturen mit dem fortgeschrittenen Schlüssel und Zertifikat ("Netkey") zu erzeugen [1], oder das zu nutzende Zertifikat im Dialog auszuwählen [2]. Voreinstellung: *0*

secommerce.secsigner.displaycardnumber=[on|off]

Die auf der Karte abgedruckte Kartenummer kann im initialen Dialog angezeigt werden. Voreinstellung: *off*

secommerce.secsigner.showprevbuttonwhensigning=[on|off]

Bestimmt, ob bei der Anzeige des Dokuments der 'Zurück'-Button angezeigt wird, mit dem der Nutzer in dem vorhergehenden Dialog springen kann. Voreinstellung: *on*

secommerce.secsigner.disableselectwhensigning=[on|off]

Bestimmt, ob während der Signatur des Dokuments das Auswahlménü links deaktiviert wird. Voreinstellung: *off*

2.5.4 OCSP

Das Online Certificate Status Protocol (OCSP) ist ein Internet-Protokoll, dass es Clients ermöglicht, den Status von X.509-Zertifikaten abzufragen.

secommerce.verify.enableocsp=[on|off]

Erlaubt die Online-Sperrabfrage (OCSP) beim Trustcenter im Rahmen der Signaturprüfung. Voreinstellung: *on*

- ! Wird von obiger Voreinstellung abgewichen, befindet sich SecSigner nicht in einem
- ! Zustand, für den die Herstellererklärung nach Signaturgesetz gilt.

secommerce.secsigner.ocspmandatory=[on|off]

Ruft automatisch die OCSP-Prüfung auf. Voreinstellung: *off*

secommerce.ocsp.timetolerance=<Sekunden>

Legt fest, wieviel älter als der Prüfzeitpunkt eine OCSP-Sperrauskunft sein darf. Beispielsweise ist eine Sperrauskunft vom 1. Januar 2009 zur Prüfung einer Signatur vom 1. Mai 2009 nicht geeignet, da das Signaturzertifikat in den dazwischen liegenden Monaten gesperrt worden sein könnte. Die Angabe erfolgt in Sekunden. 0 bedeutet keine Beschränkung. Voreinstellung: 3600

secommerce.ocsp.readtimeout=<Sekunden>

Legt fest, nach wievielen Sekunden eine OCSP-Anfrage an das Trustcenter abgebrochen wird, wenn keine Antwort erfolgt ist. Voreinstellung: 30

2.5.5 Zeitstempel

secommerce.timestamp=[on/off]

Ermöglicht innerhalb der Signierkomponente das Prüfen bzw. Erzeugen amtlicher Zeitstempeln. Voreinstellung: *on*

secommerce.timestamp.mandatory=[on/off]

Vorgabe, ob ein Zeitstempel erzeugt werden muss. Voreinstellung: *off*

secommerce.timestamp.timestamprequestinadvance=[on/off]

Vorgabe, ob eine Zeitstempelabfrage bereits vor der Signaturerzeugung vorgegeben werden soll. Voreinstellung: *off*

secommerce.secsigner.timestampserver.url

URL des Zeitstempeldienstes.

secommerce.secsigner.timestampserver.name

Name des Zeitstempeldienstes für den Auswahldialog (vgl. 6.4.4).

secommerce.secsigner.timestampserver.httpusername**secommerce.secsigner.timestampserver.httppassword**

Benutzername und Passwort für authentifizierte Zeitstempelabfragen.

secommerce.secsigner.timestampserver.keyfilename=[PKCS#8/PKCS#12-Keyfilename]**secommerce.secsigner.timestampserver.certfilename=[PKCS#8-Certfilename]****secommerce.secsigner.timestampserver.keypassword=[PASSWORD]**

Die Zeitstempelanfrage kann per Softwareschlüssel (PKCS#8 oder PKCS#12) authentisiert werden. Bei der Angabe eines PKCS#8-Schlüssels muss auch das Zertifikat angegeben werden.

Weitere Trustcenter können mit

*secommerce.secsigner.timestampserver.1.url**secommerce.secsigner.timestampserver.1.name**...**secommerce.secsigner.timestampserver.n.url*

vorgegeben werden.

2.5.6 Dateiendungen

secommerce.filenameextensionpreset.encryptedsignature=<Suffix>

Vordefiniertes Suffix zum Speichern einer verschlüsselten Signatur. Voreinstellung: *.pkcs7*

seccommerce.filenameextensionpreset.encrypteddocument=<Suffix>

Vordefiniertes Suffix zum Speichern eines verschlüsselten Dokuments. (CMS).

Voreinstellung: *.pkcs7*

seccommerce.filenameextensionpreset.pkcs7signature=<Suffix>

Vordefiniertes Suffix zum Speichern einer CAdES-Signatur. Voreinstellung: *.pkcs7*

seccommerce.filenameextensionpreset.replacesuffixinsignature=[on|off]

Soll das Suffix beim Speichern einer CAdES-Signatur das ursprüngliche Suffix ersetzen [on] oder zusätzlich angehängt werden [off]? Voreinstellung: *off*

seccommerce.filenameextensionpreset.embeddedpkcs7signature=<Suffix>

Vordefiniertes Suffix zum Speichern einer embedded-CAdES-Signatur. Voreinstellung: *.pkcs7*

seccommerce.filenameextensionpreset.pdfsignature=<Suffix>

Vordefiniertes Suffix zum Speichern einer PDF-Signatur. Voreinstellung: *-signed.pdf*

seccommerce.filenameextensionpreset.xmlsignature=<Suffix>

Vordefiniertes Suffix zum Speichern einer XML-Signatur. Voreinstellung: *-signed.xml*

seccommerce.filenameefilter.ocspresponse=<Suffix>

Vordefiniertes Suffix zum Laden einer OCSP-Sperrauskunft. Voreinstellung: **.ors*

seccommerce.filenameefilter.timestamp=<Suffix>

Vordefiniertes Suffix zum Laden eines Zeitstempels. Voreinstellung: **.tsp*

seccommerce.filenameefilter.attribute-cert=*.SU1[;*.SU2[;*.SUn]]

Zum (optionalen) Laden von Attributsertifikaten werden Dateien mit den hier angegebenen Suffixen angezeigt. Voreinstellung: **.crt;*.atz;*.der;*.cry*

seccommerce.filenameefilter.pkcs7=*.SU1[;*.SU2[;*.SUn]]

Ähnlich den Attributsertifikaten, nur für PKCS7-Objekte.

Voreinstellung: **.pkcs7;*.p7;*.p7s;*.p7m;*.pk7;*.cms;*.ads;*.pdf;*.xml*

seccommerce.filenamepreset.signeddata=<Dateiname>

Vordefinierter Name zum Speichern des signierten Dokumentes. Voreinstellung: *Signaturdokument.data*

seccommerce.filenamepreset.ocspresponse=<Dateiname>

Vordefinierter Name zum Speichern einer OCSP-Sperrauskunft. Voreinstellung: *OCSP-Response.der*

2.5.7 Hilfeseiten

seccommerce.help.<suffix>=<Hilfeseite>

Die Namen der verwendeten Hilfe-Seiten. Diese Dateien werden in einem gesonderten Browser-Fenster beim Drücken des Hilfe-Knopfes angezeigt. Als <suffix> sind möglich:

- **init** Hilfe zur Kartenleseereinrichtung. Voreinstellung: *SecSignerHelp_Init.html*
- **sign** Hilfe zum Prüfen einer Signatur. Voreinstellung: *SecSignerHelp_Sign.html*
- **verify** Hilfe zum Signaturvorgang. Voreinstellung: *SecSignerHelp_Verify.html*

seccommerce.secsigner.displayhelp=[on|off]

Legt fest, ob der Hilfe-Knopf angezeigt wird. Voreinstellung: *on*

2.5.8 Schnittstellen zum Kartenleser

`seccommerce.supportreaderpkcs11=[on|off]`

`seccommerce.supportreaderctapi=[on|off]`

`seccommerce.supportreaderpcsc=[on|off]`

Der Zugriff auf die Kartenleser kann prinzipiell über die folgenden Schnittstellen erfolgen: PKCS#11, CT-API und PC/SC. Die Suche erfolgt in dieser Reihenfolge. Der Zugriff kann auch für einzelne Leser eingeschränkt werden (vgl. Abschnitt 2.5.8.1).

Sind bestimmte Zugriffsmöglichkeiten nicht erwünscht (z.B. CT-API unter CITRIX), so können Sie ausgeschaltet werden. Voreinstellung jeweils: *on*

`seccommerce.supportreaderpcscwithoutsecurepin=[on|off]`

Bei Zugriff über PC/SC unterstützt der Kartenleser ggfs. nicht die sichere PIN-Eingabe am Lesegerät. Technisch ist die Signaturerstellung dennoch möglich, aber unsicher. Daher ist diese (unsichere) Möglichkeit der Signaturerzeugung über das Keyboard konfigurierbar. Voreinstellung: *on*

`seccommerce.usepinpad=[on|off]`

In seltenen Fällen kann es wegen technischer Probleme am Kartenleser sinnvoll sein, die sichere PIN-Eingabe am Kartenleser abzuschalten (*off*). Die PIN-Eingabe erfolgt dann auf der Tastatur des PCs. Die PIN kann in diesem Fall von evtl. installierten böswilligen Programmen abgehört werden. Voreinstellung: *on*

! Wird von obiger Voreinstellung abgewichen, befindet sich der SecSigner nicht in einem Zustand, für den die Herstellererklärung nach Signaturgesetz gilt.

`seccommerce.secsigner.collectrandom=[on|off]`

Sollen Dokumente nicht nur signiert, sondern auch verschlüsselt werden, benötigt SecSigner Zufallszahlen zur Erzeugung eines Schlüssels. Diese Zufallszahlen kann die Signaturkarte liefern. Das kostet jedoch einige Sekunden beim Einlegen der Signaturkarte. Voreinstellung: *off*

`seccommerce.pinpad.timeoutfirstdigit=[...]`

Zeitfenster in [s] bis zur Eingabe der ersten PIN-Ziffer am Kartenleser. Dieser Wert wird durch den Kartenleser selbst nach oben beschränkt. Voreinstellung: 99

`seccommerce.pinpad.successivedigits=[...]`

Zeitfenster in [s] bis zur Eingabe der nachfolgenden PIN-Ziffern am Kartenleser. Dieser Wert wird durch den Kartenleser selbst nach oben beschränkt. Voreinstellung: 10

2.5.8.1 Ausschluss einzelner Kartenleser

Der Zugriff für bestimmte Lesertypen für den Zugriff über CT-API bzw. über PC/SC kann ausgeschaltet werden:

Der Zugriff auf die PC/SC-Schnittstelle kann für einzelne Leser unterbunden werden:

`seccommerce.excludedreaderspccsc=[...], [...], [...]`

Die angegebene Liste muss die PC/SC-Lesernamen enthalten, welche am entsprechenden PC gemeldet sind, durch Kommata getrennt.

Der Zugriff auf die CT-API-Schnittstelle kann für einzelne Leser unterbunden werden:

`seccommerce.supportreadercptai.cyberjack=[on|off]`

`seccommerce.supportreadercptai.cyberjack=[on|off]`

`seccommerce.supportreaderctapi.kobil=[on|off]`

`seccommerce.supportreaderctapi.chipdrive=[on|off]`

`seccommerce.supportreaderctapi.orga=[on|off]`

`seccommerce.supportreaderctapi.omnikey=[on|off]`

`seccommerce.supportreaderctapi.cherry=[on|off]`

`seccommerce.supportreaderctapi.scmhealth=[on|off]`

`seccommerce.supportreaderctapi.telematics=[on|off]`

`seccommerce.supportreaderctapi.hypercom=[on|off]`

`seccommerce.supportreaderctapi.celectronic=[on|off]`

Für diese Optionen ist der Default-Wert jeweils 'on'.

Das Fujitsu-Keybord wird als Omnikey-Leser erkannt.

2.5.9 Signaturerstellung mit mehreren Kartenlesern

Für die Stapelsignatur können bis zu vier Kartenlesern parallel genutzt werden. Dazu müssen alle zu verwendenden Lesegeräte konfiguriert werden, etwa:

`secsignerapi.cardreader.0=CT-API Port1 Reiner SCT cyberJack`

`secsignerapi.cardreader.1=CT-API Port2 Kobil Kaan`

`secsignerapi.cardreader.2=CT-API Port3 SCM Chipdrive Pinpad`

`secsignerapi.cardreader.3=PCSC Driver: REINER SCT cyberJack`

Ist kein entsprechender Eintrag in der Konfigurationsdatei enthalten, so wird ein Kartenleser automatisch gesucht (vgl. 2.5.8).

2.5.10 Software-Schlüssel

Alternativ zu Signaturkarten kann ein Software-Zertifikat verwendet werden. Dabei befindet sich der private Schlüssel in einer verschlüsselten Datei. Es ist also keine sichere Signaturerstellungseinheit vorhanden und folglich können keine qualifizierten Signaturen im Sinne des deutschen Signaturgesetzes erstellt werden, sondern lediglich fortgeschrittene.

`seccommerce.secsigner.allowsoftwarekey=[on|off]`

Erlaubt die Nutzung von Softwareschlüsseln. Voreinstellung: *off*

`seccommerce.secsigner.softcertautoselect=[on|off]`

Selektiert die Option zur Verwendung von Softwareschlüsseln automatisch. Die Möglichkeit, Signaturkarten zu nutzen, wird dann übersprungen. Voreinstellung: *off*

`seccommerce.secsigner.continueatsoftkeypinentry=[on|off]`

Es wird automatisch zum nächsten Dialog gesprungen, sobald die PIN für den Softwareschlüssel eingegeben wurde. Voreinstellung: *off*

`seccommerce.secsigner.softwarekey=<Dateiname>`

Dateiname des voreingestellten Softwareschlüssels. Dies kann eine PKCS#12- oder PKCS#8-Datei sein.

`seccommerce.secsigner.softwarecert=<Dateiname>`

Dateiname des Zertifikates zum voreingestellten Softwareschlüssel. Nur erforderlich, wenn der Softwareschlüssel sich in einer PKCS#8-Datei befindet. Im Falle von PKCS#12 befindet sich das Zertifikat bereits mit in der Schlüsseldatei.

`seccommerce.secsigner.softwarecertholder=<Name>`

Angezeigter Name des voreingestellten Softwareschlüssels

2.5.11 Layout

secommerce.secsigner.<suffix>=<Pixel>

Bestimmt die Größe des SecSigner-Applets. Als <suffix> sind möglich:

- *width* Breite des SecSigner-Applets. Voreinstellung: 790
- *height* Höhe des SecSigner-Applets. Voreinstellung: 496

Hier können auch relative Werte bezüglich der Bildschirmgröße angegeben werden:

- *secommerce.secsigner.width=50%*
- *secommerce.secsigner.height=80%*

secommerce.secsigner.resizeable=[on|off]

Legt fest, ob SecSigners Fenster in der Größe geändert werden kann. Voreinstellung: *on*

secommerce.secsigner.parentalignment=[on|off]

Das Fenster des SecSigners kann in der Mitte des Elternfensters ausgerichtet werden.
Voreinstellung: *off*

secommerce.logo.popup=<URL>

Gibt an, welche URL in einem neuen Fenster angezeigt werden soll, wird auf das Logo in der linken, oberen Ecke geklickt. Fehlt dieser Eintrag, so wird kein neues Fenster geöffnet.

secommerce.showdocdialog.height=700

Höhe des Dialoges, welches das Dokument anzeigt. Es handelt sich hier nicht um die interne, sichere Anzeige. Voreinstellung: 700

secommerce.showdocdialog.width=800

Breite des Dialoges, welches das Dokument anzeigt. Es handelt sich hier nicht um die interne, sichere Anzeige. Voreinstellung: 800

2.5.12 Sichere Anzeige

secommerce.secsigner.displaydocument=[on|off]

Falls das zu signierende Dokument eine Grafik ist, kann die Darstellung in der sicheren Anzeige abgestellt werden (*off*). Es wird dann nur der Dateiname angezeigt. Voreinstellung: *on*

- ! Wird von obiger Voreinstellung abgewichen, befindet sich der SecSigner nicht in einem
- Zustand, für den die Herstellererklärung nach Signaturgesetz gilt.

secommerce.secsigner.mustreadbeforesign=[on|off]

Wird ein Dokument in der sicheren Anzeige dargestellt, so ermöglicht diese Option, die Signaturerstellung an das vorherige Lesen des Dokuments zu binden. Voreinstellung: *off*

secommerce.secsigner.mustseealldocsbeforesign=[on|off]

Falls ein Stapel von Dokumenten signiert werden, kann festgelegt werden, dass der Benutzer zunächst alle Dokumente ansehen muss, bevor er die Signatur auslösen kann. Voreinstellung: *off*

secommerce.secsigner.warningifsignaturerequiresviewing=[on|off]

Falls festgelegt wurde, dass der Benutzer vor der Signatur das Dokument vollständig ansehen muss, kann hier entschieden werden, dass ein Dialog den Benutzer auf diesen Umstand hinweisen soll. Voreinstellung: *off*

secommerce.secsigner.defaultfontsize=<Punkte>

Schriftgröße der Anzeige Komponente für den Textmodus. Voreinstellung: 12

secommerce.imagescalefactor=<Faktor>

Voreingestellte Skalierung des Dokumentes, falls es eine Grafik ist. Ein negativer Wert bedeutet, dass das Dokument so skaliert wird, dass seine Breite das Anzeigefenster füllt. Voreinstellung: -1 (Skalieren auf Seitenbreite)

secommerce.smoothingviewer=[on|off]

Entscheidet, ob das Dokument, falls es eine Grafik ist, genauer (*on*) oder schneller (*off*) skaliert werden soll. Voreinstellung: *on*

secommerce.secsigner.checkpdfa=[on|off]

Bei der Anzeige eines PDF-Dokumentes kann geprüft und angezeigt werden, ob es sich um ein PDF/A-Dokument handelt. Voreinstellung: *off*

secommerce.secsigner.printpdfannotations=[on|off]

Bei der Anzeige eines PDF-Dokumentes können dessen Annotationen dargestellt werden. Voreinstellung: *on*

secommerce.secsigner.showprintdocbutton=[on|off]

Legt fest, ob ein Knopf zum Drucken des Dokumentes angezeigt wird. Voreinstellung: *on*

2.5.12.1 XML-Anzeige

secommerce.xmldefaultview=3

XML-Dokumente können in 3 Modi angezeigt werden:

1. Text-Anzeige. Die XML-Tags werden nicht interpretiert.
2. Darstellung als Liste
3. Darstellung als Baumstruktur

Voreinstellung: 3 (Baumstruktur)

secommerce.xmlviewer.attributename.color=<RGB-Wert>

Farbe der Attributnamen. Voreinstellung: 0,128,0 (dunkelgrün)

secommerce.xmlviewer.brace.color=<RGB-Wert>

Farbe der Klammern. Voreinstellung: 64,64,64 (dunkelgrau)

secommerce.xmlviewer.colon.color=<RGB-Wert>

Farbe der Doppelpunkte. Voreinstellung: 64,64,64 (dunkelgrau)

secommerce.xmlviewer.comma.color=<RGB-Wert>

Farbe der Kommata. Voreinstellung: 64,64,64 (dunkelgrau)

secommerce.xmlviewer.comment.color=<RGB-Wert>

Farbe der Kommentare. Voreinstellung: 128,128,128 (hellgrau)

secommerce.xmlviewer.error.color=<RGB-Wert>

Farbe der Fehlermeldungen. Voreinstellung: 255,0,0 (rot)

secommerce.xmlviewer.tag.color=<RGB-Wert>

Farbe der XML-Tags. Voreinstellung: 0,0,255 (blau)

secommerce.xmlviewer.text.color=<RGB-Wert>

Farbe des übrigen Textes. Voreinstellung: 0,0,0 (schwarz)

2.5.13 Externe Vieweranwendungen

secommerce.open.[ext]=[pfad/application]

Dateiformate können während des Prüf- oder des Signaturprozesses an die entsprechend konfigurierten Anwendungen zur Anzeige weitergeleitet werden. Optional können diese Anwendungen entsprechend der Datei-Endung *ext* gesetzt werden:

secommerce.secsigner.showdisplaydocbutton=[on|off]

Die Möglichkeit der Anzeige des Dokuments im externen Viewer kann erlaubt oder unterdrückt werden. Voreinstellung: *on*

secommerce.secsigner.autoshowdocinsign=[on|off]

Das Dokument kann automatisch in einem externen Viewer angezeigt werden. Voreinstellung: *off*

2.5.14 Verschlüsselung

secommerce.secsigner.encryption.mandatory=[on|off]

Mit dieser Konfiguration kann die Erzeugung einer verschlüsselten Signatur (vgl. 7.4) vorgeschrieben werden. Voreinstellung: *off*

secommerce.secsigner.encryption.cipherid=[0|3|4|5]

Die symmetrische Verschlüsselung kann mit folgenden Verfahren erfolgen:

- 3DES 0
- AES128 3
- AES192 4
- AES256 5

Voreinstellung: 5

secommerce.encryptcerts.allowchange=[on|off]

Dem Anwender kann erlaubt werden, die Liste der Empfängerzertifikate im Verschlüsselungsdialog zu ändern. Voreinstellung: *on*

secommerce.encryptcerts.autocommit=[on|off]

Der Verschlüsselungsdialog mit der Anzeige der Liste der Empfängerzertifikate für die Verschlüsselung kann ohne Benutzerbestätigung übersprungen werden. Voreinstellung: *off*

secommerce.secsigner.allowsaveencryptcert=[on|off]

Im Kartenleserdialog kann ein Knopf zum Speichern des Verschlüsselungszertifikates von der Signaturkarte in einer Datei angezeigt werden. Voreinstellung: *on*

secommerce.secsigner.saveencryptcertsinstore=[on|off]

Verwendete Verschlüsselungszertifikate können gesammelt in einer Datei abgelegt werden. Sie stehen dann für weitere Verschlüsselungsvorgänge einfacher zur Verfügung. Voreinstellung: *on*

secommerce.secsigner.encryptcertstore=<Dateiname>

Name und Pfad der Datei, in der verwendete Verschlüsselungszertifikate gesammelt abgelegt werden. Voreinstellung: *<user home>/secommerce/EncryptionCertificates*

2.5.15 PDF-Annotation

secommerce.secsigner.allowpdfannotationdataediting=[on|off]

Mit dieser Konfiguration kann der Dialog zur Annotationsdarstellung bei der PDF-Signatur

(PADES) aus-/eingeschaltet werden (vgl. 7.7). Voreinstellung: *on*

secommerce.secsigner.allowpdfannotationlink=[on|off]

Mit dieser Konfiguration kann die Angabe von Daten für die Einbindung eines externen Links als zusätzliche Annotation bei der PDF-Signatur ein-/ausgeschaltet werden (vgl. 7.7).

Voreinstellung: *on*

secommerce.secsigner.addannotationimagedefault=[on|off]

Falls der Dialog zur Annotationsdarstellung ausgeschaltet ist, wird jede PDF-Signatur als Annotation im Dokument dargestellt (vgl. 7.7). Diese Voreinstellung kann verändert werden.

Voreinstellung: *on*

secommerce.secsigner.enforcepdfannotationdataediting=[on|off]

Wird eine SecSigner-Instanz mehrfach nacheinander aufgerufen, so wird der entsprechende Dialog vom zweiten Aufruf an übersprungen. Der Aufruf des Dialogs für jeden Aufruf kann erzwungen werden. Voreinstellung: *off*

secommerce.secsigner.editpdfannotationdimension=[on|off]

Die Höhe und Breite der Signatur-Annotation kann in der SecSigner-GUI durch den Nutzer eingestellt werden: *on*

secommerce.secsigner.pdfannotationposition=6

Bestimmt die Position der Signatur-Annotation innerhalb des Dokumentes. Voreinstellung: 6
Mögliche Werte sind:

- 1 : 1. Seite, oben links
- 2 : 1. Seite, oben mitte
- 3 : 1. Seite, oben rechts
- 4 : 1. Seite, unten links
- 5 : 1. Seite, unten mitte
- 6 : 1. Seite, unten rechts
- 7 : letzte Seite, oben links
- 8 : letzte Seite, oben mitte
- 9 : letzte Seite, oben rechts
- 10 : letzte Seite, unten links
- 11 : letzte Seite, unten mitte
- 12 : letzte Seite, unten rechts
- 13 : neue Seite, oben links
- 14 : neue Seite, oben mitte
- 15 : neue Seite, oben rechts
- 16 : neue Seite, unten links
- 17 : neue Seite, unten mitte
- 18 : neue Seite, unten rechts

secommerce.secsigner.pdfannotationformfieldname=

Bestimmt die Position der Signatur-Annotation innerhalb des Dokumentes anhand eines bereits im Dokument vorhandenen Formfeldes. Als Wert muss der Name des Formfeldes angegeben werden.

`seccommerce.secsigner.pdfannotationshowdate=[on|off]`

Soll das Datum in der Signatur-Annotation angezeigt werden? Voreinstellung: *on*

`seccommerce.secsigner.pdfannotationshowlabels=[on|off]`

Soll Uhrzeit, Grund und Ort vorgestellt werden? Voreinstellung: *on*

`seccommerce.secsigner.pdfannotationtransparentbg=[on|off]`

Soll die Signatur-Annotation einen transparenten Hintergrund aufweisen? Voreinstellung: *off*

`seccommerce.secsigner.pdfannotationlocation=`

Welcher Ort soll in die Signatur-Annotation eingetragen werden? Voreinstellung: *kein Wert*

`seccommerce.secsigner.pdfannotationreason=`

Welcher Grund für die Signaturerstellung soll in die Signatur-Annotation eingetragen werden? Voreinstellung: *kein Wert*

`seccommerce.secsigner.pdfannotationwidth=220`

Die Annotations-Breite. Standardwert sind 220px.

`seccommerce.secsigner.pdfannotationheight=70`

Die Höhe der Signatur-Annotation. Standardwert sind 70px.

`seccommerce.secsigner.pdfannotationpadding=0`

Padding zwischen Rand der Signaturannotation und deren Text. Standardwert sind 0.

Wenn nur ein Wert angegeben wird, gilt er für alle vier Ränder gleichermaßen, ansonsten kann das Padding auch in der Art 30,30,50,0 angegeben werden. Dabei folgt es dem Schema wie im HTML-Umfeld: Padding links, oben, rechts, unten. Im Beispiel gibt es also links einen Rand von 30px, oben 30px, rechts 50px und nach unten 0px.

`seccommerce.secsigner.pdfannotationbackgroundimage=...`

Hintergrundbild der PDF-Signatur-Annotation.

`seccommerce.secsigner.pdfannotationsignericon=...`

Bild des Unterzeichners, welches rechts unten in der Signatur-Annotation angezeigt wird. Die Höhe und Breite ist auf 80x80px begrenzt. Wenn kein Wert angegeben ist, wird das SecCommerce-Logo aus dem SecSigner-Jar verwendet.

`seccommerce.secsigner.pdfannotationshowsignericon=on`

Soll das Bild des Unterzeichners angezeigt werden: *on*

`seccommerce.secsigner.pdfannotationsignedimage=...`

Unterschriftenbild welches links unten in der Signatur-Annotation angezeigt wird. Wenn kein Wert angegeben ist, wird das SecCommerce-Logo aus dem SecSigner-Jar verwendet.

`seccommerce.secsigner.pdfannotationshowsignedimage=on`

Soll ein Unterschriftenbild angezeigt werden: *on*

2.5.16 Prüfbericht speichern

`seccommerce.secsigner.autosavereport=[on|off]`

Bei der Verifikation einer Signatur kann das Speichern des Prüfberichts automatisch erfolgen. Voreinstellung: *off*

`seccommerce.secsigner.reportdir=[Verzeichnis]`

Es kann ein Verzeichnis für das Speichern des Prüfberichts vorgegeben werden.
Voreinstellung: *(keine Vorgabe)*

2.5.17 Signaturprüfung

secommerce.secsigner.showcommitbuttoninverify=on

Wird der SecSigner in einem Applet genutzt, ist die Web-Anwendung mit der Signaturprüfung evtl. beendet. Es gibt dann keine finish-URL und dementsprechend kann der weiter-Knopf im Prüfdialog ausgeblendet werden. Voreinstellung: *on*

secommerce.secsigner.showverificationdetails=on

Legt fest, ob im Prüfdialog ein Knopf zum Wechseln in einen Dialog mit Details zur Prüfung angezeigt wird. Voreinstellung: *on*

! Wird von obiger Voreinstellung abgewichen, befindet sich SecSigner nicht im SigG-bestätigten Zustand!

secommerce.secsigner.displayprofession=[on|off]

Zeigt zusätzlich den Beruf des Unterzeichners an, sofern er im Signaturzertifikat enthalten ist. Voreinstellung: *off*

secommerce.secsigner.displayprofessioncolor=[RGB-Wert]

Legt die Farbe fest, in der der Beruf des Unterzeichners angezeigt wird. Voreinstellung: *0,0,0* (schwarz)

secommerce.secsigner.showreportinbrowser=off

Der Prüfbericht kann entweder in einem Web-Browser (*on*) oder in einem Fenster des SecSigners (*off*) angezeigt werden. Voreinstellung: *off*

secommerce.secsigner.allowsignafterverify=[on|off]

Bei der Verifikation einer Signatur kann eine weitere Signatur für das selbe Dokument erlaubt werden. Voreinstellung: *off*

secommerce.secsigner.saveextractedordecrypteddocument=[on|off]

Beim Aufruf des SecSigners über die Kommandozeile, bzw. das Microsoft-Windows-Explorer-Kontextmenü kann beim Prüfen einer Signatur das enthaltene Dokument automatisch als Datei gespeichert werden. Voreinstellung: *on*

secommerce.secsigner.showsaveocspbutton=on

Bestimmt ob der Button zum Speichern der OCSP-Antwort angezeigt wird oder nicht. Voreinstellung: *on*

2.5.18 Prüfaufträge an SecPKI senden

SecSigner kann nach Erstellen einer Signatur, die erzeugte Signatur mittel SecPKIApi an einen SecPKIServer zur Prüfung senden. Das ist nur in Ausnahmefällen sinnvoll. Im Allgemeinen sollte der Empfänger einer Signatur den SecPKIApi-Aufruf außerhalb des SecSigners selbst durchführen.

secommerce.secsigner.sendverifyrequest=[on|off]

Schaltet diese Funktion ein. Voreinstellung: *off*

secommerce.secsigner.sendverifyrequest.account=<Login-Name>

Login-Name zum Aufruf der Prüffunktion in SecPKI.

secommerce.secsigner.sendverifyrequest.orgshortname=<Mandant>

Mandantenkurzname in SecPKI.

secommerce.secsigner.sendverifyrequest.pin=<PIN>

PIN in SecPKI.

2.5.19 Applet-Darstellung

secommerce.secsigner.displayinbrowser=[on|off]

Beim Aufruf des SecSigner-Applets im Browser kann dies im aktuellen Browserfenster [*on*] oder einem eigenen Fenster [*off*] angezeigt werden. Voreinstellung: *off*

secommerce.secsigner.postplaintextainverify=[on|off]

Falls der Aufruf des SecSigner-Applets zum Entschlüsseln eines Dokumentes erfolgte, kann das entschlüsselte Dokument per HTTP-Post an einen Web-Server geschickt werden.

2.5.20 Attributzertifikat automatisch einlesen

secommerce.secsigner.autoattributecertsearch=[on|off]

Ein zum Signaturzertifikat gehörendes Attributzertifikat kann automatisch eingelesen werden. Voreinstellung: *on*

secommerce.secsigner.autoattributecertpath=[Pfad]

Ein angegebener Pfad kann Umgebungsvariablen enthalten. Deren Wert wird der Aufrufumgebung von SecSigner entnommen. Sie können wie bei Windows mit einem Prozentzeichen am Anfang und am Ende oder wie bei Linux mit einem Dollarzeichen am Anfang angegeben werden. Eine mögliche Angabe auf einem Windows-System wäre also z.B.

%HOMEPATH%\secommerce

und auf einem Linux-System z.B.

\$HOME/.secommerce

Eine Umgebungsvariable, die keinen Wert besitzt, wird zur leeren Zeichenkette ausgewertet.

Voreinstellung: *[user home]/.secommerce/AttributeCertificates*

2.5.21 Maximale Dateigröße

secommerce.fileopen.maxsizemb=[Größe]

Die maximale Größe für zu signierende bzw. signierter Dokumente ist vorgegeben, um einen Speicherüberlauf zu vermeiden. Voreinstellung: *10*

2.5.22 Nutzung von Test-Zertifikaten

Für die Integration in Testumgebungen kann die Prüfung der Ausstellerzertifikate optional ausgeschaltet werden:

secommerce.secsigner.checkcacert=[on|off]

Ist die Einstellung '*off*', so werden Ausstellerzertifikate nicht geprüft und SecSigner ist keine Signaturanwendungskomponente nach SigG mehr. Ein entsprechender Hinweis wird im Dialog unübersehbar dargestellt. Voreinstellung: *on*

- ! Wird von obiger Voreinstellung abgewichen, kann der SecSigner nicht sicherstellen, dass
- zumindest fortgeschrittene Signaturen erstellt werden!

2.5.23 Logdatei

SecSigner kann Meldungen in eine Logdatei schreiben. Die Meldungen werden immer auch nach stdout (z.B. Java-Konsole) geschrieben. Abhängig von der Art der SecSigner-Einbindung sind die Ausgaben in stdout jedoch nicht unbedingt sichtbar.

log.fileactive=[on|off]

Gibt vor, ob Logdateien geschrieben werden sollen. Voreinstellung: *off*

log.filename=<Datei>

Konfiguration des Dateinamens für die Protokoll-Datei. Voreinstellung: *log.txt*

log.filenamealter=[on|off]

Verwendet für jeden Tag des Monats eine eigene Protokolldatei, wenn auf *on* gesetzt. Voreinstellung: *off*

log.filenameedate=[on|off]

Verwendet für jeden Tag eine eigene Protokolldatei und hängt das Datum im Format *yyyyMMdd* an, wenn auf *on* gesetzt. Bei *off* wird nur der Tag *dd* angehängt. Hat nur eine Wirkung, wenn *log.filenamealter=on*. Voreinstellung: *off*

log.internaltrace=[on|off]

Protokolliert Zugriffe auf die Protokolldatei. Sollte nur auf *on* gesetzt werden, wenn es von SecCommerce vorgeschlagen wird. Voreinstellung: *off*

log.maxlogtype=<n>

Detailstufe der Protokolleinträge. Für *n* sind folgende Werte möglich:

0	<i>Error</i>
10	<i>Warning</i>
20	<i>Standard (PROD)</i>
30	<i>Usage</i>
40	<i>Event</i>
50	<i>Debug</i>
60	<i>Verbose (TEST)</i>

Stellen Sie für den Produktionsbetrieb 20 ein, für Testzwecke kann 50 eingestellt werden; die Log-Dateien werden in diesem Fall sehr umfangreich. Voreinstellung: 20

Die Log-Datei wird im Ordner '*.seccommerce/temp*' abgelegt. Der Ordner '*.seccommerce*' wird grundsätzlich im Nutzerverzeichnis erstellt. Unter Windows kann der Ordner '*.seccommerce*' an einem anderen Ort erstellt werden, der durch die Umgebungsvariable '*USERPROFILE*' bestimmt ist.

2.5.24 Signaturdatenauswahl durch den Nutzer

SecSigner kann zur Signaturerzeugung aufgerufen werden, ohne zu signierende Daten zu übergeben. Ist dies der Fall, so wird der Nutzer aufgefordert, die zu signierende Daten selbst anzugeben. Der entsprechende Auswahl-dialog ist konfigurierbar. Wird keine der nachfolgenden Optionen verwendet, so wird der Nutzer aufgefordert, ein beliebiges Dokument aus dem Dateisystem zu laden.

seccommerce.secsigner.loaddocumentdescription.0=[Dokumentbezeichnung]**seccommerce.secsigner.loaddocumentdescription.1=[Dokumentbezeichnung]**

....

seccommerce.secsigner.loaddocumentdescription.n=[Dokumentbezeichnung]

Der Nutzer wird aufgefordert, *n+1* Dokumente mit den entsprechenden Bezeichnungen aus dem Dateisystem zu laden.

seccommerce.secsigner.alterdescriptionfordocumentloading.0=[on|off]

....

seccommerce.secsigner.alterdescriptionfordocumentloading.n=[on|off]

Der Nutzer darf die Bezeichnung für ein Dokument ändern, wenn der entsprechende Wert '*on*'

secommerce.secsigner.allowemptydocuments=[on|off]

Der Nutzer muss alle geforderten Dokumente aus dem Dateisystem zu laden ('off') oder darf nur einen Teil der geforderten Dokumente laden ('on'). Voreinstellung: 'off'

secommerce.secsigner.deleteoptionfordocumentloading=[on|off]

Hat der Nutzer ein Dokument aus dem Dateisystem geladen, so kann er es wieder löschen ('on') oder nicht ('off'). Voreinstellung: 'off'

secommerce.secsigner.showoptionfordocumentloading=[on|off]

Hat der Nutzer ein Dokument aus dem Dateisystem geladen, so kann er es mit der auf dem PC vorgesehenen externen Anwendung öffnen ('on') oder nicht ('off'). Voreinstellung: 'off'

secommerce.secsigner.replacelabelfordocumentloading=[on|off]

Hat der Nutzer ein Dokument aus dem Dateisystem geladen, so wird die Dokument-Bezeichnung durch den Dateinamen ersetzt ('on') oder in einem eigenen Feld angezeigt ('off'). Voreinstellung: 'on'

secommerce.secsigner.labelborderfordocumentloading=[on|off]

Wird der Dateiname in einem eigenen Feld angezeigt, so wird dieses Feld mit Umrandung dargestellt ('on') oder nicht ('off'). Voreinstellung: 'on'

secommerce.secsigner.showpathfordocumentloading=[on|off]

Hat der Nutzer ein Dokument aus dem Dateisystem geladen, so wird ihm der Verzeichnispfad angezeigt ('on') oder nur der Dateiname ('off'). Voreinstellung: 'off'

secommerce.secsigner.returnemptyrecordsfordocumentloading=[on|off]

Hat der Nutzer nicht alle Dokument aus dem Dateisystem geladen, so werden dennoch alle Dokumente per Post an die angegebene URL übergeben ('on') oder nur die ausgewählten Dokumente ('off'). Voreinstellung: 'on'

secommerce.secsigner.switchlabelpositionfordocumentloading=[on|off]

Der Button zum Laden eines Dokument aus dem Dateisystem steht links ('off') oder rechts der Dateiname ('on'). Voreinstellung: 'off'

secommerce.secsigner.loaddocumenttext=[Text]

Ein erklärender Text für den Dialog zum Laden der Dokumente aus dem Dateisystem st kann vorgegeben werden.

2.5.25 SecSigner-Lizenz

Kundenspezifische Anpassungen für die Lizenzbedingungen oder das Layout werden über eine gesonderte Lizenzdatei abgebildet. Diese Lizenzdatei kann über die folgende Option übergeben werden werden:

secommerce.secsigner.licence=[Lizenzdatei]

Alternativ kann die Lizenz zur Laufzeit über die Java-API oder die DLL übergeben werden. Siehe Abschnitt 3.6 Lizenzdatei.

2.5.26 Signaturformat im Drag&Drop - Dialog

Beim Aufruf des 'SecSigner-Applet für Windows, Mac OS X, Linux' können Dokumente per Drag&Drop in den SecSigner-Dialog gezogen und dann signiert werden (vgl. Abschnitt 9). Die Voreinstellung für das Signaturformat ist 'PKCS7'. Andere Voreinstellung können mit dieser Option eingestellt werden:

secommerce.secsigner.exe.preselectedformat=[0,1,2,3]

Mögliche Werte sind:

0: CAdES detached, : CAdES embedded, 2: PDF-Signatur (PAdES), : CAdES verschlüsselt

2.5.27 Zertifikatstapel laden

`seccommerce.secsigner.certstackdownload.enabled=<on|off>`

Ist diese Option 'on', so kann der Nutzer den Zertifikatstapel von der Hersteller-Website nachladen, falls die Ausstellerzertifikate eines zur Prüfung/Erzeugung verwendeten Signaturzertifikates nicht bekannt sind.

Zertifikatstapel werden im Nutzerverzeichnis abgelegt und nachfolgend berücksichtigt.

`seccommerce.secsigner.certstackdownload.confirm=<on|off>`

Ist diese Option 'on', so muss der Nutzer im Dialog den Zertifikatstapel-Download bestätigen.

2.5.28 Algorithmenkatalog laden

`seccommerce.secsigner.algocatdownload.enabled=<on|off>`

Der SecSigner kann regelmäßig einen signierten Algorithmenkatalog vom SecCommerce-Webserver laden. Dieser Algorithmenkatalog enthält Informationen über die Gültigkeitsdauer und Parameter von kryptographischen Algorithmen für die qualifizierte Signatur in Deutschland und wird nach den entsprechenden Veröffentlichungen der Bundesnetzagentur aktualisiert.

Der signierte Algorithmenkatalog wird im Nutzerverzeichnis abgelegt und nachfolgend berücksichtigt.

Es wird empfohlen, diesen automatischen Update-Mechanismus immer aktiv zu haben. Andernfalls bewertet der SecSigner die Eignung von Algorithmen für die qualifizierte Signatur nach dem Stand der BNetzA-Veröffentlichungen, der bei seiner Herstellung galt. Aufgrund späterer Erkenntnisse über eventuelle Schwachstellen kann es jedoch notwendig sein, die Gültigkeit betroffener Algorithmen zu verkürzen. Darüberhinaus ist es auch wichtig, von den üblichen, jährlichen Verlängerungen der Algorithmengültigkeit zu erfahren, um nach einigen Jahren Betriebsdauer keine gültigen Signaturen abzulehnen.

2.5.29 Firewalls/Proxy-Server

2.5.29.1 Automatische Proxy-Konfiguration bei Windows

Wenn eine Firewall die direkte Verbindung verhindert, kann bei Verwendung von Microsoft Windows im Internet-Explorer ein Proxy konfiguriert werden, der mittels HTTP-POST oder HTTP-CONNECT die Verbindung aufbaut. SecSigner liest die Proxy-Einstellungen, welche der Internet-Explorer in der Registry gespeichert hat.

2.5.29.2 allgemeine Konfiguration

Daneben gibt es die Möglichkeit, einen festen Proxy für SecSigner einzustellen. Folgende Optionen in der Propertiesdatei gelten für alle Benutzer. Beim Start des SecSigner als Applet von über das Internet sind diese Optionen vermutlich nicht sinnvoll, da die Benutzer verschiedene oder keine Proxies nutzen.

`proxy.autoconfurl=<URL>`

URL eines Skriptes zur automatischen Proxyauswahl.

`proxy.https.bypass=<Hosts>`

Liste von Hosts, die ohne Proxy erreicht werden sollen. Die Namen der Hosts sind durch

Komma, Semikolon, Doppelpunkt oder Leerzeichen zu trennen.

proxy.https.list=<Proxy-Liste>

Liste von HTTP-Proxy-Servern. Die Syntax der Proxyliste ist im folgenden Abschnitt definiert.

2.5.29.3 Konfiguration pro Benutzer

Eine feste Proxykonfiguration ist auch pro Benutzer möglich. Dazu muss eine Datei

[user home]\.seccommerce\scProxy.conf

mit diesem Inhalt erstellt werden:

PROXY [IP-Adresse]:[Port]

Es können auch mehrere Proxies konfiguriert werden:

*PROXY [IP-Adresse1]:[Port1]; PROXY [IP-Adresse2]:[Port2]; PROXY [IP-Adresse3]:
[Port3] usw.*

Es gibt Proxies, die nur TLS-Verbindungen über die CONNECT-Methode erlauben und die Verbindung wieder schließen, wenn sie feststellen, dass kein TLS benutzt wird.

3 Integration des SecSigners in Signaturanwendungen

Der SecSigner ist eine Signaturanwendungskomponente, die sich auf verschiedene Arten in Anwendungen integrieren lässt, die mit dem SecSigner eine elektronische Signatur erstellen oder prüfen.

3.1 Integration in Java-Anwendungen

Das Archive *SecSigner.jar* enthält die Java-Klasse *seccommerce.secsignersigg.SecSigner*. Dabei handelt es sich um den Einstiegspunkt in den SecSigner. *SecSigner.jar* muss in den Classpath aufgenommen werden. Im selben Verzeichnis wie *SecSigner.jar* erwartet SecSigner folgende Dateien aus der SecSigner-Auslieferung:

- *SecSigner.jar*
- *SecSigner.selfchecksig*
- *secsigner.properties*

Die Developer-Version des SecSigners enthält im Verzeichnis *developer/javadoc* die Dokumentation (javadoc) aller öffentlichen Klassen.

Im Verzeichnis *developer/javaapplication/seccommerce/secsignersigg/testapplication* findet sich das Aufrufbeispiel *SecSignerExample.java*.

3.2 Integration mittels CallSecSignerDLL

Um die Integration des SecSigners auch in andere Sprachen als Java zu ermöglichen, enthält die Developer-Version die Bibliotheken für Microsoft Windows *CallSecSignerDLL.dll*, sowie *CallSecSignerDLLStd.dll*. Erstere erwartet beim Aufruf ihrer exportierten Funktionen die Parameter in der calling Convention *__cdecl*, zweite in *stdcall*.

Beide DLLs ermöglichen den Aufruf der Method der SecSigner-Java-Klasse. Die jeweilige DLL konvertiert die Parameter und die Rückgabewerte.

Ist die aufrufende Anwendung eine 64-Bit-Applikation, muss die *CallSecSignerDLL64.dll* eingebunden werden. Für 32-Bit-Anwendungen *CallSecSignerDLL.dll* oder *CallSecSignerDLLStd.dll*

Im Verzeichnis *developer/c*, bzw. *developer/csharp* finden sich Aufrufbeispiele in C und C#.

3.3 Integration in HTML-Seiten mit Java-Web-Start

SecSigner kann mit der mitgelieferten Web-Start-Anwendung in bestehende HTML-Seiten integriert werden. Die Web-Start-Anwendung übernimmt die Kommunikation mit den JAVA-Schnittstellen des SecSigner, sodass eine Integration in einen HTML-Workflow (Webanwendung) ohne JAVA-Kenntnisse möglich ist, denn die Web-Start-Anwendung ist im SecSigner bereits enthalten. Die erzeugte Signatur und ggf. der erstellte Zeitstempel werden an eine frei konfigurierbare URL übertragen (HTTP POST oder HTTP GET). Serverseitig kann so die Signatur als CAdES-, PAdES- oder XAdES-Objekt weiterverarbeitet werden.

Der Erstellung von Signaturen und dem Auslesen von Zertifikaten dient SecSignerWebStart, dem Überprüfen SecVerifierWebStart.

Natürlich ist es alternativ auch möglich, selbst eine Java-Web-Start-Anwendung zu schreiben und den SecSigner darin über sein Java-API aufzurufen.

Die Erzeugung von Massensignaturen ist in der kostenfreien Version des SecSigner nicht

3.3.1 SecSignerWebStart

SecSignerWebStart dient der Erzeugung elektronischer Signaturen und Zeitstempel, sowie dem Auslesen von Zertifikaten. Um SecSigner als Web-Start-Applikation aus einer HTML-Seite heraus zu starten, muss der Web-Server eine JNLP-Datei mit allen notwendigen Parameter generieren und den Browser zu einer URL umleiten, unter der die JNLP-Datei abgerufen werden kann.

Beispiel: *secSignerSign.jnlp*

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp
  spec="1.0+"

codebase="http://replaceyourhost.yourdomain.topleveldomain/secSignerWebStart/"
  href="secSignerSign.jnlp">
<information>
  <title>SecSigner</title>
  <vendor>SecCommerce Informationssysteme GmbH</vendor>
  <homepage href="https://www.seccommerce.com/" />
</information>
<security><all-permissions/></security>
<resources>
  <j2se version="1.6+" />
  <jar href="SecSigner.jar" />
  <jar href="SecSignerExt.jar" />
</resources>
<application-desc main-
class="seccommerce.secsigner.webstart.SecSignerWebStart">

<!-- Optional Init-Parameter. If this parameter is set with value 'on', no
document will be signed, but the user's certificates will be posted to
given 'PostURL' -->
<argument>-Init</argument> <argument>off</argument>

<!-- Specify an URL _OR_ the document to be signed _OR_ fileopen -->
<argument>-DocumentURL</argument> <argument>fileopen:*. *</argument>

<!-- <argument>-Document</argument> <argument><html>Dies ist ein
<b>Test</b>, der einen zu unterzeichnenden Text innerhalb eines HTML-
Dokumentes enthaelt.</html></argument>-->
<!-- <argument>DocumentBase64</argument> <argument></argument>-->

<!-- Optional: Specify a document type: text=0, HTML=1, binary=2, image=3,
PDF=4, XML=12, RTF=13, ZIP=14, ZKS=15 (0-14 will be detected
automatically) -->
<!-- <argument>-DocumentType</argument> <argument>0</argument> -->

<!-- Optional: Specify a URL (OldSignatureURL) _OR_ the old signature
(Base64OldSignature) -->
<!-- <argument>-OldSignatureURL</argument>
<argument>http://dev.seccommerce.de/signature.pkcs7 </argument> -->
<!-- <argument>-Base64OldSignature</argument> <argument>MII..</argument>
-->

<!-- ID of the desired signature format: 0 = PKCS#7, 3 = PDF, 4 = XML-DSig,
5 = embedded PKCS#7 -->
<argument>-SignatureFormatType</argument> <argument>0</argument>

<!-- Only for XML-DSig: XPath to the XML node under which the signature
node shall be inserted. If null then the
signature node will be inserted directly below the root node. Value is
Base64 encoded. Example "/descendant::*[local-name()='BGSERZLayer']][1]" -->
<!-- <argument>-XmlDSigNodePath</argument>
<argument>/descendant::*[local-name()='BGSERZLayer']][1]</argument> -->
```



```

<!-- Alternative: <argument>-XmlDSigNodePathBase64</argument>
<argument></argument> -->

<!-- filter method: 0 intersect, 1 subtract, 2 union -->
<!-- <argument>-XmlDSigTransformMethod.0</argument>
<argument>1</argument>-->

<!-- filter expressions to specify the XML nodes to be signed. If null then
such
    a filter will not be created. Example "/descendant::*[local-
name()='BGSERZLayer']" -->
<!-- <argument>-XmlDSigXPathExpression.0</argument>
<argument>here()/ancestor-or-self::*[local-name()='BGSERZLayer']</argument> -->
<!-- <argument>-XmlDSigXPathExpressionBase64.0</argument>
<argument></argument> -->

<!-- filter method: 0 intersect, 1 subtract, 2 union -->
<!-- <argument>-XmlDSigTransformMethod.1</argument> <argument>0</argument>
-->
<!-- <argument>-XmlDSigXPathExpression.1</argument>
<argument>/descendant::*[local-name()='BGSERZLayer']</argument> -->
<!-- <argument>-XmlDSigXPathExpressionBase64.1</argument>
<argument></argument> -->

<!-- namespace mappings to ensure the given xpath expression can be
validated. all namespace mappings belong to a transformation filter
specified
    by the first index. the key-->
<!-- <argument>-XmlDSigNamespace.1.Key.0</argument>
<argument>dsig</argument> -->
<!-- <argument>-XmlDSigNamespace.1.Value.0</argument>
<argument>http://www.w3.org/2000/09/xmlsig#</argument> -->

<!-- Only for XML-DSig: XMLDSig signature ID: This ID is inserted as id
value in signature element node -->
<!-- <argument>-XmlDSigSignatureId</argument> <argument>12345678-
0</argument> -->

<!-- Post signature and additional params to an URL, it posts
"Signature=..." as a base64 coded PKCS7 object -->
<argument>-PostURL</argument> <argument>http://localhost/seccom/</argument>

<!-- (optional) Post signed data: "on" or "off" -->
<argument>-PostSignedData</argument> <argument>on</argument>

<!-- (optional) Post document file name: "on" or "off" -->
<argument>-PostFileName</argument> <argument>off</argument>

<!-- (optional) Post the certificates from the smart card: "on" or "off"
-->
<argument>-PostCertificates</argument> <argument>off</argument>

<!-- (optional) additional params to post (custom specific, e.g. "id" and
"caller") -->
<argument>-PostParams</argument> <argument>id,caller</argument>
<argument>-id</argument> <argument>id-value</argument>
<argument>-caller</argument> <argument>caller-value</argument>

<!-- (optional) call this URL after posting the signature -->
<!-- <argument>-FinishURL</argument>
<argument>https://www.seccommerce.com/</argument> -->

<!-- call this URL on user cancellation -->
<argument>-CancelURL</argument>
<argument>https://www.seccommerce.com/</argument>

<!-- call this URL on fatal errors -->
<argument>-ErrorURL</argument>
<argument>https://www.seccommerce.com/</argument>

</application-desc>

```


</jnlp>

Der Signierprozess wird über die HTML-Seite über eine Umleitung auf die JNLP-Datei angestoßen.

Während der SecSigner grundsätzlich die Möglichkeit bietet, die erzeugte Signatur lokal auf der Festplatte des Kunden zu speichern, sendet SecSignerWebStart alternativ die Daten mittels POST zusätzlich an eine vorgegebene URL.

Nachdem SecSignerWebStart gestartet wurde, gibt es nur drei wohldefinierte Möglichkeiten, wie es beendet wird:

- *Finish*. Das gewünschte Dokument wurde erfolgreich signiert und das Ergebnis konnte an die PostURL gesendet werden.
- *Cancel*. Der Signiervorgang wurde durch den Anwender abgebrochen. Es wurde keine Signatur erzeugt. Es wird die CancelURL aufgerufen.
- *Error*. Es ist ein Fehler bei der Verwendung des SecSigners aufgetreten. Es wird die ErrorURL aufgerufen.

Folgende Kommandozeilen-Parameter können SecSignerWebStart durch die JNLP-Datei übergeben werden. Dabei sind jeweils Kommandozeilenparameter-Paare zu übergeben, wobei der erste der Name des SecSigner-Parameters ist und mit einem „-“ beginnt, während der zweite der Wert des SecSigner-Parameters ist.

Init

Soll kein Dokument signiert, sondern sollen statt dessen nur die Zertifikate des Nutzers von der Signaturkarte ausgelesen, so ist dieser Parameter 'on' zu setzen. Die Nutzerzertifikate werden dann Base64-kodiert per POST an die angegebene 'PostURL' übermittelt.

Modal

Soll das SecSignerApplet zur Signaturerzeugung aufgerufen, so wird es per Default nicht modal angezeigt. Soll das SecSignerApplet modal angezeigt werden, so ist dieser Parameter 'on' zu setzen.

DocumentURL

Document

DocumentBase64

Welche Daten sollen signiert werden? Es kann eine vollständige URL einer HTML-Seite angegeben werden (*DocumentURL*), ein (HTML-)Text innerhalb dieses HTML-Dokumentes (*Document*), oder ein Base64-kodiertes Dokument (*DocumentBase64*).

Soll der Nutzer selbst die Daten lokal aus seinem Dateisystem laden, so können die entsprechenden Optionen als Applet-Parameter übergeben werden (vgl. Abschnitt 2.5.24).

SignatureFormatType

Optional: Bezeichnet das gewünschte Signaturformat. Unterstützt werden derzeit:

0 = CAdES-detached, 3 = PAdES, 4 = XML-DSIG, 5 = CAdES embedded

Wird kein Wert vorgegeben, so erzeugt SecSigner eine Signatur im CAdES-detached-Format.

HashAlg

Optional: Bezeichnet das gewünschte Hashverfahren. Unterstützt werden unter anderem: "autoselect", "SHA1", "SHA256" und "SHA512".

Wird kein Wert vorgegeben, so erzeugt SecSigner eine Signatur im SHA256-Format, sofern die Smartcard es unterstützt

PaddingRsaPss

Optional: Wählt das gewünschte Paddingverfahren für RSA-Signaturen. „on“ wählt RSA-

PSS, „off“ PKCS#1 v1.5. Sicherer ist RSA-PSS, weiter verbreitet ist noch PKCS#1 v1.5.

Bei ECC-Signaturen hat diese Einstellung keine Bedeutung.

Wird kein Wert vorgegeben, so nutzt SecSigner das RSA-PSS-Padding.

OldSignatureURL

Base64OldSignature

Optional: Die neue Signatur wird zu dieser bestehenden Signatur hinzugefügt. Die alte Signatur muss sich auf dieselben Daten beziehen. Es kann eine URL zu der bestehenden Signatur angegeben werden (*OldSignatureURL*) oder die Signatur selbst kodiert in DER und Base64 (*Base64OldSignature*).

PostParams

Kommaseparierte Liste der Parameter P1 bis Pn, die zusätzlich zur Signatur mittels POST durch SecSigner mitgesendet werden.

P1..Pn

Die unter PostParams definierten Parameter.

CharSet

Dieser optionale Parameter gibt an, an welche Codierung für die PostParams genutzt werden soll.

PostURL

Gibt an, an welche URL die signierten base64-kodierten Daten als PKCS7-Objekt mittels HTTP-POST gesendet werden sollen. Die Antwort des Webservers auf das POST wird mittels HTTP-GET gelesen und in einem separaten Browserfenster dargestellt.

Es werden folgende Parameter gesendet:

Parameter	Beschreibung
SignedData[i]	Signierte Daten, wenn <i>PostSignedData=on</i>
Signature[i]	Signaturobjekt
Description[i]	(nur wenn gegeben)
Base64IdentityCert	Zertifikate von der Signanturekarte, wenn <i>PostCertificates=on</i>
Base64UtilityCert	
Base64DecryptCert	
SignCertSerialNumber	Seriennummer des Signaturzertifikates, falls <i>PostSignCertSerialNumber=on</i>
P1..Pn	Siehe PostParams

Hinweis: Ist FinishURL definiert, so wird das Ergebnis von PostURL nicht dargestellt.

PostSignedData [on|off]

Das Applet sendet nicht nur die Signatur des Dokumentes und einen eventuell vorhandenen Zeitstempel an die PostURL, sondern zusätzlich die signierten Daten. Voreinstellung: on

PostFileName [on|off]

Das Applet sendet zusätzlich den Dateinamen des Dokuments an die PostURL.
Voreinstellung: off

CipherCertURL**CipherCertURL.0****CipherCertURL.1****CipherCertURL.n**

Optional: Die signierten Daten werden nach dem Signaturvorgang mit diesem/n Zertifikat/en verschlüsselt.

CipherCertBase64**CipherCertBase64.0****CipherCertBase64.1****CipherCertBase64.n**

Optional: Die signierten Daten werden nach dem Signaturvorgang mit diesem/n Base64-codierten Zertifikat/en verschlüsselt.

EncryptDataOnly [on/off]

Die Daten werden ohne Signatur verschlüsselt. Voreinstellung: off

XmlDSigXPathExpression

Nur für XMLDSig: gibt mittels eines XPath-Ausdrucks an, unter welchem Element/Knoten die Signatur eingefügt werden soll. Ist dieser nicht angegeben, wird die Signatur direkt unter dem Wurzelknoten eingefügt.

XmlDSigXPathExpression.0**XmlDSigXPathExpression.1****XmlDSigXPathExpression.n**

Nur für XMLDSig: gibt einen XPath-Filter-Ausdruck an, der angibt, welche Knoten des Dokumentes signiert werden sollen.

XmlDSigTransformMethod**XmlDSigTransformMethod.0****XmlDSigTransformMethod.1****XmlDSigTransformMethod.n**

Nur für XMLDSig: gibt die Transform-Methode zu XmlDSigXPathExpression an. Mögliche Werte:

- 0=*intersect*
- 1=*subtract*
- 2=*union*

XmlDSigNameSpace.n.Key**XmlDSigNameSpace.n.Value****XmlDSigNameSpace.n.Key.0****XmlDSigNameSpace.n.Value.0****XmlDSigNameSpace.n.Key.1****XmlDSigNameSpace.n.Value.1****XmlDSigNameSpace.n.Key.m****XmlDSigNameSpace.n.Value.m**

Nur für XMLDSig: gibt ein Namespacemapping aus Namespace (Key) und Namespace-URI (Value) an. Die Zahl n gibt dabei an, für welche Transformation das Namespacemapping verwendet werden soll. Das Namespacemapping wird als Namespace-Deklaration im XPath-Transform-Element des Signaturelements eingefügt.

XmlDSigSignatureId

Nur für XMLDSig: dieser String wird in den Signaturknoten als 'SignatureId' eingetragen.

PDFAnnotation.displayAnnotation [on|off]

Nur für PDF-Signaturen: Bestimmt, ob eine Annotationsdarstellung für die Signatur ergänzt werden soll.

PDFAnnotation.imageBase64

Nur für PDF-Signaturen: Optionale JPG-Grafik für Annotationsdarstellung (Base64-kodiert, max. 220 x 70 Pixel).

PDFAnnotation.imageURL

Nur für PDF-Signaturen: Optionale URL der JPG-Grafik für Annotationsdarstellung (max. 220 x 70 Pixel) alternativ zu PDFAnnotation.imageBase64.

PDFAnnotation.scaleImage [on|off]

Nur für PDF-Signaturen: Bestimmt, ob die angegebene JPG-Grafik für die Annotationsdarstellung (max. 220 x 70 Pixel) skaliert werden soll. Größere Grafiken werden in jedem Falle (auf max. 220 x 70 Pixel) skaliert.

PDFAnnotation.displayImageText [on|off]

Nur für PDF-Signaturen: Bestimmt, ob Text in der Annotationsdarstellung ausgegeben werden soll.

PDFAnnotation.sigTextSizeAndPosition

Nur für PDF-Signaturen: Optionale Angabe für Größe und Position des Textes in der Annotationsdarstellung. Bestimmt, wie und wo der Text in der Annotationsdarstellung ausgegeben werden soll. Anzugeben sind 6 Stellen: [ssxxyy], wobei ss die Textgröße, xx den Prozentwert für den linken Rand (innerhalb der Annotationsdarstellung) und yy den Prozentwert für den oberen Rand angibt.

PDFAnnotation.location

Nur für PDF-Signaturen: Ort für die Signatur in der Textausgabe der Annotationsdarstellung.

PDFAnnotation.position

Nur für PDF-Signaturen: Position der Annotationsdarstellung im PDF. Mögliche Werte sind:

LEFT_TOP_FIRSTPAGE = 1, CENTER_TOP_FIRSTPAGE = 2, RIGHT_TOP_FIRSTPAGE = 3

LEFT_BOTTOM_FIRSTPAGE = 4, CENTER_BOTTOM_FIRSTPAGE = 5,
RIGHT_BOTTOM_FIRSTPAGE = 6

LEFT_TOP_LASTPAGE = 7, CENTER_TOP_LASTPAGE = 8, RIGHT_TOP_LASTPAGE = 9

LEFT_BOTTOM_LASTPAGE = 10, CENTER_BOTTOM_LASTPAGE = 11,
RIGHT_BOTTOM_LASTPAGE = 12

LEFT_TOP_NEWPAGE = 13, CENTER_TOP_NEWPAGE = 14, RIGHT_TOP_NEWPAGE = 15

LEFT_BOTTOM_NEWPAGE = 16, CENTER_BOTTOM_NEWPAGE = 17,
RIGHT_BOTTOM_NEWPAGE = 18

Angegeben werden kann auch ein mindestens 5-stelliger Wert ppxyy. Dann bezeichnet pp die Seitenzahl, xx den Prozentwert der Seitenbreite des linken Randes der Annotation und yy den Prozentwert der Seitenhöhe des oberen Randes der Annotation. Der Wert '-' für pp bezeichnet die letzte Seite.

PDFAnnotation.reason

Nur für PDF-Signaturen: Grund der Signatur in der Textausgabe der Annotationsdarstellung.

PDFAnnotation.textAnnotation

Nur für PDF-Signaturen: Alternative Textdarstellung (anstelle der Annotationsdarstellung).

PDFAnnotation.linkImageBase64

Nur für PDF-Signaturen: Optionale JPG-Grafik für einen Link, etwa auf eine Signaturanwendungskomponente zur Prüfung (Base64-kodiert, max. 220 x 70 Pixel).

PDFAnnotation.linkImageURL

Nur für PDF-Signaturen: Optionale URL der JPG-Grafik für einen Link, etwa auf eine Signaturanwendungskomponente zur Prüfung (max. 220 x 70 Pixel) alternativ zu PDFAnnotation.linkImageBase64.

PDFAnnotation.linkURL

Nur für PDF-Signaturen: Optionale URL für einen Link, etwa auf eine Signaturanwendungskomponente zur Prüfung.

FinishURL

URL, die nach einem erfolgreichen Abschluss des Signaturvorgangs im Default-Browser angezeigt wird. Ist FinishURL nicht definiert, so wird die Startseite nicht verlassen.

Hinweis: Ist FinishURL definiert, so wird das Ergebnis von PostURL nicht dargestellt.

CancelURL

URL, die nach einem Abbruch des Signaturvorgangs im Default-Browser angezeigt wird.

ErrorURL

URL, die nach einem Fehler des Signaturvorgangs im Default-Browser angezeigt wird.

3.3.2 Mehrfachsignatur mit SecSignerWebStart

Bei entsprechender Lizenzierung unterstützt SecSigner auch die Mehrfachsignatur. Dazu können die Parameter

DocumentURL**Document****OldSignatureURL****Base64OldSignature****XmlDSigNodePath****XmlDSigXPathExpression****XmlDSigTransformMethod****XmlDSigNameSpace****XmlDSigSignatureId**

jeweils mit einem zusätzlichen nachstehenden Index versehen werden, etwa DocumentURL.0, DocumentURL.1, DocumentURL.2 etc.

Die Aufzählung muss mit dem Index '0' beginnen, kann beliebig fortgesetzt werden und darf keine Lücken aufweisen.

3.3.3 SecVerifierWebStart

Das SecVerifierWebStart ist das logische Gegenstück zum SecSignerWebStart. Es dient ausschließlich dem Prüfen von bereits vorhandenen Signaturen und optional vorhandenen Zeitstempeln.

Um SecSigner als Web-Start-Anwendung aus einer HTML-Seite heraus zu starten, muss der Web-Server eine JNLP-Datei mit allen notwendigen Parameter generieren und den Browser zu einer URL umleiten, unter der die JNLP-Datei abgerufen werden kann.

Der Verifikationsprozess wird über die HTML-Seite über eine Umleitung auf die JNLP-Datei angestoßen.

Beispiel: *secSignerVerify.jnlp*

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp
  spec="1.0+"

  codebase="http://replaceyourhost.yourdomain.topleveldomain/secSignerWebStart/"
  href="secSignerVerify.jnlp">
<information>
  <title>SecSigner</title>
  <vendor>SecCommerce Informationssysteme GmbH</vendor>
  <homepage href="https://www.seccommerce.com/" />
</information>
<security><all-permissions/></security>
<resources>
  <j2se version="1.6+" />
  <jar href="SecSigner.jar" />
  <jar href="SecSignerExt.jar" />
</resources>
<application-desc main-
class="seccommerce.secsigner.webstart.SecVerifierWebStart">

<!-- Specify an URL _OR_ the document to be verified _OR_ fileopen -->
<argument>-DocumentURL</argument> <argument>test/doc1.pdf</argument>
<argument>-SignatureURL</argument> <argument>test/doc1.pdf.pkcs7</argument>
<!--<argument>-TimestampURL</argument> <argument>test/doc1.tsr</argument>
-->
<!--<argument>-OCSPResponseURL</argument>
<argument>test/doc1.ors</argument> -->
<!--<argument>-Base64Document</argument> <argument></argument> -->
<!--<argument>-Base64Signature</argument> <argument></argument> -->
<!--<argument>-Base64Timestamp</argument> <argument></argument> -->
<!--<argument>-Base64OCSPResponse</argument> <argument></argument> -->

<!-- Only for XML-DSig: XPath to the signature node which shall be
verified. If null then all
signature nodes will be checked. Example "/test:testdoc/test:car" -->
<!--<argument>-XmlDSigNodePathBase64</argument>
<argument>L3Rlc3Q6dGVzdGRvYy90ZXN0OmNhcg==</argument> -->

<!-- Post signature and additional params to an URL, it posts
"Signature=..." as a base64 coded PKCS7 object -->
<argument>-PostURL</argument> <argument>http://localhost/seccom/</argument>

<!-- (optional) Post signed data: "on" or "off" -->
<!-- <argument>-PostSignedData</argument> <argument>on</argument> -->

<!-- Save data extracted from signature if not posted -->
<!-- <argument>-SaveIncludedDocument</argument> <argument>on</argument> -->

<!-- (optional) additional params to post (custom specific, e.g. "id" and
"caller") -->
<argument>-PostParams</argument> <argument>id,caller</argument>
<argument>-id</argument> <argument>id-value</argument>
<argument>-caller</argument> <argument>caller-value</argument>

<!-- (optional) call this URL after posting the signature -->
<!-- <argument>-FinishURL</argument>
<argument>https://www.seccommerce.com/</argument> -->

<!-- call this URL on user cancellation -->
<argument>-CancelURL</argument>
<argument>https://www.seccommerce.com/</argument>

<!-- call this URL on fatal errors -->
<argument>-ErrorURL</argument>
<argument>https://www.seccommerce.com/</argument>
```

```
</application-desc>  
</jnlp>
```

Nachdem SecVerifierWebStart gestartet wurde, gibt es nur drei wohldefinierte Möglichkeiten, wie es beendet wird:

- *Finish*. Das gewünschte Dokument wurde erfolgreich verifiziert. Es wird das Ergebnis der PostURL angezeigt oder die FinishURL aufgerufen.
- *Cancel*. Der Verifikationsprozess wurde durch den Anwender abgebrochen. Es wird die CancelURL aufgerufen.
- *Error*. Es ist ein Fehler bei der Verwendung des SecSigner aufgetreten. Es wird die ErrorURL aufgerufen.

Folgende Kommandozeilen-Parameter können SecVerifierWebStart durch die JNLP-Datei übergeben werden. Dabei sind jeweils Kommandozeilenparameter-Paare zu übergeben, wobei der erste der Name des SecSigner-Parameters ist und mit einem „-“ beginnt, während der zweite der Wert des SecSigner-Parameters ist.

Modal

Soll SecVerifierWebStart zur Signaturerzeugung aufgerufen, so wird es per Default modal angezeigt. Soll das SecVerifierWebStart nicht modal angezeigt werden, so ist dieser Parameter 'off' zu setzen.

DocumentURL

Base64Document

Gibt die zu verifizierenden Daten an. Es können eine URL (*DocumentURL*) oder base64-kodierte Daten innerhalb des HTML-Dokumentes (*Base64Document*) angegeben werden.

SignatureURL

BASE64Signature

Gibt die zu verifizierende Signatur an. Es können eine URL (*SignatureURL*) oder base64-kodierte Daten innerhalb des HTML-Dokumentes (*BASE64Signature*) angegeben werden.

TimestampURL

Base64Timestamp

Gibt den zu verifizierenden amtlichen Zeitstempel an. Es können eine URL (*TimestampURL*) oder base64-kodierte Daten innerhalb des HTML-Dokumentes (*Base64Timestamp*) angegeben werden. Zeitstempel, die sich innerhalb der Signatur selbst befinden, werden hier nicht erwähnt.

OCSPResponseURL

Base64OCSPResponse

Gibt die zu verwendende Zertifikatstatus-Auskunft an. Es können eine URL (*OCSPResponseURL*) oder base64-kodierte Daten innerhalb des HTML-Dokumentes (*Base64OCSPResponse*) angegeben werden. OCSP-Antworten, die sich innerhalb der Signatur selbst befinden, werden hier nicht erwähnt.

EvidenceRecordURL

Base64EvidenceRecord

Gibt das zu verwendende Beweisdokument (Evidence Record) gem. RFC 4998 an. Es können eine URL (*EvidenceRecordURL*) oder base64-kodierte Daten innerhalb des HTML-Dokumentes (*Base64EvidenceRecord*) angegeben werden.

DecryptedDocumentPath

Gibt einen Pfad für die Speicherung entschlüsselter Daten an. Die Dateieindung wird jeweils ergänzt.

SaveIncludedDocument

Enthält die Signatur die signierten Daten, so können diese nach der Prüfung im Dateisystem abgelegt werden, falls keine POST URL angegeben ist.

FinishURL

URL, die nach einem erfolgreichen Abschluss des Verifikationsprozesses im Default-Browser angezeigt wird.

CancelURL

URL, die nach einem Abbruch des Verifikationsprozesses im Default-Browser angezeigt wird.

ErrorURL

URL, die nach einem Fehler des Verifikationsprozesses im Default-Browser angezeigt wird.

3.4 Integration in HTML-Seiten mit Java-Applets

SecSigner kann mit den mitgelieferten Applets in bestehende HTML-Seiten integriert werden. Die Applets übernehmen die Kommunikation mit den JAVA-Schnittstellen des SecSigner, sodass eine Integration in einen HTML-Workflow (Webanwendung) ohne JAVA-Kenntnisse möglich ist, denn die Applets sind im SecSigner bereits enthalten. Die erzeugte Signatur und ggf. der erstellte Zeitstempel werden an eine frei konfigurierbare URL übertragen (HTTP POST oder HTTP GET). Serverseitig kann so die Signatur als CAdES-, PAdES- oder XAdES-Objekt weiterverarbeitet werden.

- ! Da die Zahl der Browser, die noch Java-Applets unterstützen immer geringer wird, sollte lieber der Weg über Java-Web-Start gewählt werden. Siehe Abschnitt 3.3

Der Erstellung von Signaturen und dem Auslesen von Zertifikaten dient das SecSignerApplet, dem Überprüfen das SecVerifierApplet. Mit dem SecPKISignOnApplet können die Zertifikate einer Signaturkarte ausgelesen und an einen Server gepostet werden. Diese Funktionalität ist ggf. für eine Erstanmeldung (Registrierung) verwendbar.

Die Erzeugung von Massensignaturen ist in der kostenfreien Version des SecSigner nicht lizenziert.

3.4.1 SecSignerApplet

Das SecSignerApplet dient der Erzeugung elektronischer Signaturen und Zeitstempel, sowie dem Auslesen von Zertifikaten. Um SecSigner als Applet aus einer HTML-Seite heraus zu starten, werden alle notwendigen Parameter mit dieser HTML-Seite übergeben (siehe Beispiel).

Beispiel: *SecSignerApplet.html*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;CHARSET=iso-8859-1">
<TITLE>SecCommerce SecSignerApplet</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<P>
<br><br>

<!-- Codebase and archives of SecSigner -->
```


SecCommerce SecSigner

```

<APPLET codebase="." CODE="seccommerce.secsigner.applet.SecSignerApplet"
archive="SecSigner.jar,SecSignerExt.jar" WIDTH="200" HEIGHT="20"
ALIGN="BOTTOM" MAYSCRIPT>

<!-- Optional Init-Parameter. If this parameter is set with value 'on', no
document will be signed, but the user's certificates will be posted to
given 'PostURL' -->
<PARAM NAME="Init" VALUE="off">

<!-- Specify an URL _OR_ the document to be signed _OR_ fileopen -->
<PARAM NAME="DocumentURL" VALUE="fileopen:*.*)"
<!-- <PARAM NAME="Document" VALUE="<html>Dies ist ein <b>Test</b>, der
einen zu unterzeichnenden Text innerhalb eines HTML-Dokumentes
enthaelt.</html>" -->
<!-- <PARAM NAME="DocumentBase64"
VALUE="PGh0bWw+RGllcyBpc3QgZWluIDxiPlRlc3Q8L2I+LCBkZXIgcZWluZW4genUgdW50ZXJ6
ZWljag5lbmRlbiBUZXh0IGlubmVyaGFsYiBlaw5lcYBIVElMlURva3VtZW50ZXMgZW50aGFlbHQ
uPC9odGlsPgo=" -->

<!-- Optional: Specify a URL (OldSignatureURL) _OR_ the old signature
(Base64OldSignature) -->
<!-- <PARAM NAME="OldSignatureURL"
VALUE="http://dev.seccommerce.de/signature.pkcs7 " -->
<!-- <PARAM NAME="Base64OldSignature" VALUE="MII.." -->

<!-- ID of the desired signature format: 0 = PKCS#7 SignedData, 3 = PDF, 4
= XML-DSig, 5 = Embedded PKCS#7 -->
<PARAM NAME="SignatureFormatType" VALUE="0">

<!-- Only for XML-DSig: XPath to the XML node under which the signature
node shall be inserted. If null then the
signature node will be inserted directly below the root node. Value is
Base64 encoded. Example "/descendant::*[local-name()='BGSERZLayer']" -->
<!-- <PARAM NAME="XmlDSigXPath" VALUE="/descendant::*[local-
name()='BGSERZLayer']" -->
<!-- Alternative: <PARAM NAME="XmlDSigXPathBase64"
VALUE="L2Rlc2NlbnRhbnQ6OipbbG9jYWwtbmFtZSgpcSdCRlNFUlpMYXllciddWzFd" -->

<!-- filter method: 0 intersect, 1 subtract, 2 union -->
<!-- <PARAM NAME="XmlDSigTransformMethod.0" VALUE="1" -->
<!-- filter expressions to specify the XML nodes to be signed. If null then
such
a filter will not be created. Example "/descendant::*[local-
name()='BGSERZLayer']" -->
<!-- <PARAM NAME="XmlDSigXPathExpression.0" VALUE="here()/ancestor-or-
self:*/dsig:Signature" -->
<!-- <PARAM NAME="XmlDSigXPathExpressionBase64.0"
VALUE="aGVyZSgpcL2FuY2VzdG9yLW9yLXNlbnGY6OiovZHNpZzpzTaWduYXRlcmU=" -->

<!-- filter method: 0 intersect, 1 subtract, 2 union -->
<!-- <PARAM NAME="XmlDSigTransformMethod.1" VALUE="0">
<!-- <PARAM NAME="XmlDSigXPathExpression.1" VALUE="/descendant::*[local-
name()='BGSERZLayer']">
<!-- <PARAM NAME="XmlDSigXPathExpressionBase64.1"
VALUE="L2Rlc2NlbnRhbnQ6OipbbG9jYWwtbmFtZSgpcSdCRlNFUlpMYXllcidd" -->

<!-- namespace mappings to ensure the given xpath expression can be
validated. all namespace mappings belong to a transformation filter
specified
by the first index. the key-->
<!-- <PARAM NAME="XmlDSigNamespace.1.Key.0" VALUE="dsig" -->
<!-- <PARAM NAME="XmlDSigNamespace.1.Value.0"
VALUE="http://www.w3.org/2000/09/xmlsig#" -->

<!-- Only for XML-DSig: XMLDSig signature ID: This ID is inserted as id
value in signature element node -->
<!-- <PARAM NAME="XmlDSigSignatureId" VALUE="12345678-0" -->

<!-- Post signature and additional params to an URL, it posts
"Signature=..." as a base64 coded PKCS7 object -->

```

SecCommerce SecSigner

```
<!-- <PARAM NAME="PostURL"
VALUE="http://dev.seccommerce.de/servlet/SecSignerPost"> -->

<!-- (optional) Post signed data: "on" or "off" -->
<PARAM NAME="PostSignedData" VALUE="on">

<!-- (optional) Post document file name: "on" or "off" -->
<PARAM NAME="PostFileName" VALUE="on">

<!-- (optional) Post the certificates from the smart card: "on" or "off"
-->
<PARAM NAME="PostCertificates" VALUE="off">

<!-- (optional) additional params to post (custom specific, e.g. "id" and
"caller") -->
<PARAM NAME="PostParams" VALUE="id,caller">
<PARAM NAME="id" VALUE="id-value">
<PARAM NAME="caller" VALUE="caller-value">

<!-- (optional) call this URL after posting the signature -->
<PARAM NAME="FinishURL" VALUE="html/SecSignerApplet_exit.html">

<!-- (optional) target of FinishURL -->
<PARAM NAME="FinishURLTarget" VALUE="">

<!-- call this URL on user cancellation -->
<PARAM NAME="CancelURL" VALUE="html/SecSignerApplet_cancel.html">

<!-- (optional) target of CancelURL -->
<PARAM NAME="CancelURLTarget" VALUE="">

<!-- call this URL on fatal errors -->
<PARAM NAME="ErrorURL" VALUE="html/SecSignerApplet_error.html">

<!-- (optional) target of ErrorURL -->
<PARAM NAME="ErrorURLTarget" VALUE="">

<font color=red><b>Ihr Browser unterst tzt offenbar noch keine Java-
Anwendungen. Bitte installieren Sie ein aktuelles Java-Runtime-Environment
von <a href="http://java.sun.com/">Sun</a></b></font>
</APPLET>

</BODY>

</HTML>
```

Der Signierprozess wird  ber die HTML-Seite mit dem Aufruf des `seccommerce.secsigner.SecSignerApplet` angesto en.

W hrend der SecSigner grunds tzlich die M glichkeit bietet, die erzeugte Signatur lokal auf der Festplatte des Kunden zu speichern, sendet SecSignerApplet alternativ die Daten mittels POST zus tzlich an eine vorgegebene URL.

Nachdem das SecSignerApplet gestartet wurde, gibt es nur drei wohldefinierte M glichkeiten, wie es beendet wird:

- *Finish*. Das gew nschte Dokument wurde erfolgreich signiert und das Ergebnis konnte an die PostURL gesendet werden.
- *Cancel*. Der Signiervorgang wurde durch den Anwender abgebrochen. Es wurde keine Signatur erzeugt. Es wird die CancelURL aufgerufen.
- *Error*. Es ist ein Fehler bei der Verwendung des SecSigner aufgetreten. Es wird die ErrorURL aufgerufen.

Folgende Applet-Parameter k nnen SecSignerApplet durch die aufrufende HTML-Seite  bergeben werden:

Init

Soll kein Dokument signiert, sondern sollen statt dessen nur die Zertifikate des Nutzers von der Signaturkarte ausgelesen, so ist dieser Parameter 'on' zu setzen. Die Nutzerzertifikate werden dann Base64-kodiert per POST an die angebene 'PostURL' übermittelt.

Modal

Soll das SecSignerApplet zur Signaturerzeugung aufgerufen, so wird es per Default nicht modal angezeigt. Soll das SecSignerApplet modal angezeigt werden, so ist dieser Parameter 'on' zu setzen.

DocumentURL**Document****DocumentBase64**

Welche Daten sollen signiert werden? Es kann eine vollständige URL einer HTML-Seite angegeben werden (*DocumentURL*), ein (HTML-)Text innerhalb dieses HTML-Dokumentes (*Document*), oder ein Base64-kodiertes Dokument (*DocumentBase64*).

Soll der Nutzer selbst die Daten lokal aus seinem Dateisystem laden, so können die entsprechenden Optionen als Applet-Parameter übergeben werden (vgl. Abschnitt 2.5.24).

SignatureFormatType

Optional: Bezeichnet das gewünschte Signaturformat. Unterstützt werden derzeit:

0 = CAdES-detached, 3 = PAdES, 4 = XML-DSIG, 5 = CAdES embedded

Wird kein Wert vorgegeben, so erzeugt SecSigner eine Signatur im CAdES-detached-Format.

HashAlg

Optional: Bezeichnet das gewünschte Hashverfahren. Unterstützt werden unter anderem: "autoselect", "SHA1", "SHA256" und "SHA512".

Wird kein Wert vorgegeben, so erzeugt SecSigner eine Signatur im SHA256-Format, sofern die Smartcard es unterstützt

PaddingRsaPss

Optional: Wählt das gewünschte Paddingverfahren für RSA-Signaturen. „on“ wählt RSA-PSS, „off“ PKCS#1 v1.5. Sicherer ist RSA-PSS, weiter verbreitet ist noch PKCS#1 v1.5.

Bei ECC-Signaturen hat diese Einstellung keine Bedeutung.

Wird kein Wert vorgegeben, so nutzt SecSigner das RSA-PSS-Padding.

OldSignatureURL**Base64OldSignature**

Optional: Die neue Signatur wird zu dieser bestehenden Signatur hinzugefügt. Die alte Signatur muss sich auf dieselben Daten beziehen. Es kann eine URL zu der bestehenden Signatur angegeben werden (*OldSignatureURL*) oder die Signatur selbst kodiert in DER und Base64 (*Base64OldSignature*).

PostParams

Kommaseparierte Liste der Parameter P1 bis Pn, die zusätzlich zur Signatur mittels POST durch SecSigner mitgesendet werden.

P1..Pn

Die unter PostParams definierten Parameter.

CharSet

Dieser optionale Parameter gibt an, an welche Codierung für die PostParams genutzt werden soll.

PostURL

Gibt an, an welche URL die signierten base64-kodierten Daten als PKCS7-Objekt mittels HTTP-POST gesendet werden sollen. Die Antwort des Webservers auf das POST wird mittels HTTP-GET gelesen und in einem separaten Browserfenster dargestellt.

Es werden folgende Parameter gesendet:

Parameter	Beschreibung
SignedData[.i]	Signierte Daten, wenn <i>PostSignedData=on</i>
Signature[.i]	Signaturobjekt
Description[.i]	(nur wenn gegeben)
Base64IdentityCert	Zertifikate von der Signanturekarte, wenn <i>PostCertificates=on</i>
Base64UtilityCert	
Base64DecryptCert	
SignCertSerialNumber	Seriennummer des Signaturzertifikates, falls <i>PostSignCertSerialNumber=on</i>
P1..Pn	Siehe PostParams

Hinweis: Ist FinishURL definiert, so wird das Ergebnis von PostURL nicht dargestellt.

PostSignedData [on|off]

Das Applet sendet nicht nur die Signatur des Dokumentes und einen eventuell vorhandenen Zeitstempel an die PostURL, sondern zusätzlich die signierten Daten. Voreinstellung: on

PostFileName [on|off]

Das Applet sendet zusätzlich den Dateinamen des Dokuments an die PostURL.
Voreinstellung: off

CipherCertURL**CipherCertURL.0****CipherCertURL.1****CipherCertURL.n**

Optional: Die signierten Daten werden nach dem Signaturvorgang mit diesem/n Zertifikat/en verschlüsselt.

CipherCertBase64**CipherCertBase64.0****CipherCertBase64.1****CipherCertBase64.n**

Optional: Die signierten Daten werden nach dem Signaturvorgang mit diesem/n Base64-kodierten Zertifikat/en verschlüsselt.

EncryptDataOnly [on|off]

Die Daten werden ohne Signatur verschlüsselt. Voreinstellung: off

XmlDSigNodePath

Nur für XMLDSig: gibt mittels eines XPath-Ausdrucks an, unter welchem Element/Knoten die Signatur eingefügt werden soll. Ist dieser nicht angegeben, wird die Signatur direkt unter

dem Wurzelknoten eingefügt.

XmlDSigXPathExpression

XmlDSigXPathExpression.0

XmlDSigXPathExpression.1

XmlDSigXPathExpression.n

Nur für XMLDSig: gibt einen XPath-Filter-Ausdruck an, der angibt, welche Knoten des Dokumentes signiert werden sollen.

XmlDSigTransformMethod

XmlDSigTransformMethod.0

XmlDSigTransformMethod.1

XmlDSigTransformMethod.n

Nur für XMLDSig: gibt die Transform-Methode zu XmlDSigXPathExpression an. Mögliche Werte:

- *0=intersect*
- *1=subtract*
- *2=union*

XmlDSigNameSpace.n.Key

XmlDSigNameSpace.n.Value

XmlDSigNameSpace.n.Key.0

XmlDSigNameSpace.n.Value.0

XmlDSigNameSpace.n.Key.1

XmlDSigNameSpace.n.Value.1

XmlDSigNameSpace.n.Key.m

XmlDSigNameSpace.n.Value.m

Nur für XMLDSig: gibt ein Namespacemapping aus Namespace-Name (Key) und Namespace-URI (Value) an. Die Zahl n gibt dabei an, für welche Transformation das Namespacemapping verwendet werden soll. Das Namespacemapping wird als Namespace-Deklaration im XPath-Transform-Element des Signaturelements eingefügt.

XmlDSigSignatureId

Nur für XMLDSig: dieser String wird in den Signaturknoten als 'SignatureId' eingetragen.

PDFAnnotation.displayAnnotation [on|off]

Nur für PDF-Signaturen: Bestimmt, ob eine Annotationsdarstellung für die Signatur ergänzt werden soll.

PDFAnnotation.imageBase64

Nur für PDF-Signaturen: Optionale JPG-Grafik für Annotationsdarstellung (Base64-kodiert, max. 220 x 70 Pixel).

PDFAnnotation.imageURL

Nur für PDF-Signaturen: Optionale URL der JPG-Grafik für Annotationsdarstellung (max. 220 x 70 Pixel) alternativ zu PDFAnnotation.imageBase64.

PDFAnnotation.scaleImage [on|off]

Nur für PDF-Signaturen: Bestimmt, ob die angegebene JPG-Grafik für die Annotationsdarstellung (max. 220 x 70 Pixel) skaliert werden soll. Größere Grafiken werden in jedem Falle (auf max. 220 x 70 Pixel) skaliert.

PDFAnnotation.displayImageText [on|off]

Nur für PDF-Signaturen: Bestimmt, ob Text in der Annotationsdarstellung ausgegeben werden soll.

PDFAnnotation.sigTextSizeAndPosition

Nur für PDF-Signaturen: Optionale Angabe für Größe und Position des Textes in der Annotationsdarstellung. Bestimmt, wie und wo der Text in der Annotationsdarstellung ausgegeben werden soll. Anzugeben sind 6 Stellen: [ssxxyy], wobei ss die Textgröße, xx den Prozentwert für den linken Rand (innerhalb der Annotationsdarstellung) und yy den Prozentwert für den oberen Rand angibt.

PDFAnnotation.location

Nur für PDF-Signaturen: Ort für die Signatur in der Textausgabe der Annotationsdarstellung.

PDFAnnotation.position

Nur für PDF-Signaturen: Position der Annotationsdarstellung im PDF. Mögliche Werte sind:

LEFT_TOP_FIRSTPAGE = 1, CENTER_TOP_FIRSTPAGE = 2, RIGHT_TOP_FIRSTPAGE = 3

LEFT_BOTTOM_FIRSTPAGE = 4, CENTER_BOTTOM_FIRSTPAGE = 5,
RIGHT_BOTTOM_FIRSTPAGE = 6

LEFT_TOP_LASTPAGE = 7, CENTER_TOP_LASTPAGE = 8, RIGHT_TOP_LASTPAGE = 9

LEFT_BOTTOM_LASTPAGE = 10, CENTER_BOTTOM_LASTPAGE = 11,
RIGHT_BOTTOM_LASTPAGE = 12

LEFT_TOP_NEWPAGE = 13, CENTER_TOP_NEWPAGE = 14, RIGHT_TOP_NEWPAGE = 15

LEFT_BOTTOM_NEWPAGE = 16, CENTER_BOTTOM_NEWPAGE = 17,
RIGHT_BOTTOM_NEWPAGE = 18

Angegeben werden kann auch ein mindestens 5-stelliger Wert ppxxyy. Dann bezeichnet pp die Seitenzahl, xx den Prozentwert der Seitenbreite des linken Randes der Annotation und yy den Prozentwert der Seitenhöhe des oberen Randes der Annotation. Der Wert '-' für pp bezeichnet die letzte Seite.

PDFAnnotation.reason

Nur für PDF-Signaturen: Grund der Signatur in der Textausgabe der Annotationsdarstellung.

PDFAnnotation.textAnnotation

Nur für PDF-Signaturen: Alternative Textdarstellung (anstelle der Annotationsdarstellung).

PDFAnnotation.linkImageBase64

Nur für PDF-Signaturen: Optionale JPG-Grafik für einen Link, etwa auf eine Signaturanwendungskomponente zur Prüfung (Base64-kodiert, max. 220 x 70 Pixel).

PDFAnnotation.linkImageURL

Nur für PDF-Signaturen: Optionale URL der JPG-Grafik für einen Link, etwa auf eine Signaturanwendungskomponente zur Prüfung (max. 220 x 70 Pixel) alternativ zu PDFAnnotation.linkImageBase64.

PDFAnnotation.linkURL

Nur für PDF-Signaturen: Optionale URL für einen Link, etwa auf eine Signaturanwendungskomponente zur Prüfung.

FinishURL

URL, die nach einem erfolgreichen Abschluss des Signaturvorgangs angezeigt wird. Ist FinishURL nicht definiert, so wird die Startseite nicht verlassen.

Hinweis: Ist FinishURL definiert, so wird das Ergebnis von PostURL nicht dargestellt.

FinishURLTarget

Target-Frame der URL zur Anzeige nach erfolgreichem Abschluss des Signaturvorgangs.

CancelURL

URL, die nach einem Abbruch des Signaturvorgangs angezeigt wird.

CancelURLTarget

Target-Frame der URL, die nach einem Abbruch des Signaturvorgangs angezeigt wird.

ErrorURL

URL, die nach einem Fehler des Signaturvorgangs angezeigt wird.

ErrorURLTarget

Target-Frame der URL, die nach einem Fehler des Signaturvorgangs angezeigt wird.

3.4.2 Mehrfachsignatur mit SecSignerApplet

Bei entsprechender Lizenzierung unterstützt SecSigner auch die Mehrfachsignatur. Dazu können die Parameter

DocumentURL**Document****OldSignatureURL****Base64OldSignature****XmlDSigNodePath****XmlDSigXPathExpression****XmlDSigTransformMethod****XmlDSigNameSpace****XmlDSigSignatureId**

jeweils mit einem zusätzlichen nachstehenden Index versehen werden, etwa DocumentURL.0, DocumentURL.1, DocumentURL.2 etc.

Die Aufzählung muss mit dem Index '0' beginnen, kann beliebig fortgesetzt werden und darf keine Lücken aufweisen.

3.4.3 SecVerifierApplet

Das SecVerifierApplet ist das logische Gegenstück zum SecSignerApplet. Es dient ausschließlich dem Prüfen von bereits vorhandenen Signaturen und optional vorhandenen Zeitstempeln.

Um SecSigner als Applet, also aus einer HTML-Seite heraus, zu starten, werden alle notwendigen Parameter mit dieser HTML-Seite übergeben (siehe Beispiel).

Der Verifikationsprozess wird mit `secommerce.secsigner.SecVerifierApplet` angestoßen.

Beispiel: *SecVerifierApplet.html*

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;CHARSET=iso-8859-1">
<TITLE>SecCommerce SecVerifierApplet</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<br><br>

<APPLET codebase="." CODE="secommerce.secsigner.applet.SecVerifierApplet"
archive="SecSigner.jar,SecSignerExt.jar" WIDTH="200" HEIGHT="20"
ALIGN="BOTTOM" MAYSCRIPT>

<!-- <PARAM NAME="DocumentURL" VALUE="test/Signaturdokument.html"> -->
<!-- <PARAM NAME="SignatureURL" VALUE="test/Signaturdokument.html.pkcs7">
```



```
-->
<!-- <PARAM NAME="TimestampURL" VALUE="test/Signaturdokument.tsr"> -->
<!-- <PARAM NAME="OCSPResponseURL" VALUE="test/Signaturdokument.ors"> -->
<!-- <PARAM NAME="Base64Document" VALUE=""> -->
<!-- <PARAM NAME="Base64Signature"
VALUE="MIIEIgYJKoZIhvcNAQcCoIIEEZCCBA8CAQExCzAJBgUrDgMCGgUAMC4GCSqGSIb3DQEH
AaAhBB9EYXMGaXN0IG1laW5lIFdpbGxlbnNlcmnts5HJlbnmc6oIICmzCCApwggIDoAMCAQICBC+
u6DgwCgYGKYQDAwECBQAwfTELMakGA1UEBhMCREUxHDAaBgNVBAoUE0RldXRzY2hlIFRlbGVrb2
0gQUcxHzAdBgNVBASUF1Byb2R1a3R6ZW50cnVtIFRlbGVVTZWmXlZAMBgcCggYBCgcUEwExMB8GA
1UEAxQYVGVsZVN1YyBQSlMgU2lnRyBDQSAxOlBOMCIYDzIwMDEwNzAyMDc1MTI0WhgPMjAwNDA3
MDIwNzUxMjRaMDQxMjR1aW5lIFdpbGxlbnNlcmnts5HJlbnmc6oIICmzCCApwggIDoAMCAQICBC+
u6DgwCgYGKYQDAwECBQADgYEAgdDxRkBA6N0hJrJwah/LPL0LVZ7xTXfmjY5uc4s0gg9HQRMc3Hz
e31xux/OtcIB26+Z+lW9FYdum8T2Fi58sKSyrPELw5VayVQItGmjXyl2hJRsdpjNNOAx9XXhJf4
Csh8LW2YbdGWGPabSq7iq7zBY6SYVMUwzsMiBPzL04xggEsMIIBKAIBATCBhTB9MQswCQYDVQQ
GEWJERTECMBoGA1UEChQTRGVldHNjaGUgVGVsZWtvczBBBRzEfMB0GA1UECxxQUWUHVjZHVrdHplbn
RydW0gVGVsZVN1YyEvMAwGBWwKCBgEKBxQTATEwHwYDVQQQDFBhUZWx1U2VjIFBLUyBTaWdHIENBI
DE6UE4CBC+u6DgwCQYFKw4DAhoFADANBgkqhkiG9w0BAQEFAASBgDNPUBT/L6sf5CCScmH/GBtM
fQib+wjZTvdKdnssuMFm3UM3j2maD9v3F+USKA+LFkAo8Ql/5tP0El/zhgz7LzFDm9ZLsb7znY0
LmMeJbr+ri6pEqrVLCac8r+GQVJXtPBjFjUna745pY9vAYnA2dm4x9ceg/HdQ+PcGf2n026mv">
-->
<!-- <PARAM NAME="Base64Timestamp" VALUE=""> -->
<!-- <PARAM NAME="Base64OCSPResponse" VALUE=""> -->

<!-- Only for XML-DSig: XPath to the signature node which shall be
verified. If null then all
signature nodes will be checked. Example "/test:testdoc/test:car" -->
<PARAM NAME="XmlDSigNodePathBase64"
VALUE="L3Rlc3Q6dGVzdGRvYy90ZXN0OmNhcg==">

<!-- Post signature and additional params to an URL, it posts
"Signature=..." as a base64 coded PKCS7 object -->
<!-- <PARAM NAME="PostURL"
VALUE="http://dev.seccommerce.de/servlet/SecSignerPost"> -->

<!-- (optional) Post signed data: "on" or "off" -->
<!-- <PARAM NAME="PostSignedData" VALUE="on" -->

<!-- (optional) additional params to post (custom specific, e.g. "id" and
"caller") -->
<!-- <PARAM NAME="PostParams" VALUE="id,caller" -->
<!-- <PARAM NAME="id" VALUE="id-value" -->
<!-- <PARAM NAME="caller" VALUE="caller-value" -->

<!-- Save data extracted from signature if not posted -->
<PARAM NAME="SaveIncludedDocument" VALUE="off">

<!-- (optional) call this URL after posting the signature -->
<PARAM NAME="FinishURL" VALUE="html/SecVerifierApplet_ok.html">

<!-- (optional) target of FinishURL -->
<PARAM NAME="FinishURLTarget" VALUE="">

<!-- (optional) call this URL on user cancellation -->
<PARAM NAME="CancelURL" VALUE="html/SecVerifierApplet_cancel.html">

<!-- (optional) target of CancelURL -->
<PARAM NAME="CancelURLTarget" VALUE="">

<!-- (optional) call this URL on fatal errors -->
<PARAM NAME="ErrorURL" VALUE="html/SecVerifierApplet_error.html">

<!-- (optional) target of ErrorURL -->
<PARAM NAME="ErrorURLTarget" VALUE="">

<font color=red><b>Ihr Browser unterst&uuml;tzt offenbar noch keine Java-
Anwendungen. Bitte installieren Sie ein aktuelles Java-Runtime-Environment
```

```
von <a href="http://java.sun.com/">Sun</a></b></font>  
</BODY>  
</HTML>
```

Nachdem SecVerifierApplet gestartet wurde, gibt es nur drei wohldefinierte Möglichkeiten, wie es beendet wird:

- *Finish*. Das gewünschte Dokument wurde erfolgreich verifiziert. Es wird das Ergebnis der PostURL angezeigt oder die FinishURL aufgerufen.
- *Cancel*. Der Verifikationsprozess wurde durch den Anwender abgebrochen. Es wird die CancelURL aufgerufen.
- *Error*. Es ist ein Fehler bei der Verwendung des SecSigner aufgetreten. Es wird die ErrorURL aufgerufen.

Folgende Applet-Parameter können SecVerifierApplet durch die aufnehmende HTML-Seite übergeben werden:

Modal

Soll das SecVerifierApplet zur Signaturerzeugung aufgerufen, so wird es per Default modal angezeigt. Soll das SecVerifierApplet nicht modal angezeigt werden, so ist dieser Parameter 'off' zu setzen.

DocumentURL

Base64Document

Gibt die zu verifizierenden Daten an. Es können eine URL (*DocumentURL*) oder base64-kodierte Daten innerhalb des HTML-Dokumentes (*Base64Document*) angegeben werden.

SignatureURL

BASE64Signature

Gibt die zu verifizierende Signatur an. Es können eine URL (*SignatureURL*) oder base64-kodierte Daten innerhalb des HTML-Dokumentes (*BASE64Signature*) angegeben werden.

TimestampURL

Base64Timestamp

Gibt den zu verifizierenden amtlichen Zeitstempel an. Es können eine URL (*TimestampURL*) oder base64-kodierte Daten innerhalb des HTML-Dokumentes (*Base64Timestamp*) angegeben werden. Zeitstempel, die sich innerhalb der Signatur selbst befinden, werden hier nicht erwähnt.

OCSPResponseURL

Base64OCSPResponse

Gibt die zu verwendende Zertifikatstatus-Auskunft an. Es können eine URL (*OCSPResponseURL*) oder base64-kodierte Daten innerhalb des HTML-Dokumentes (*Base64OCSPResponse*) angegeben werden. OCSP-Antworten, die sich innerhalb der Signatur selbst befinden, werden hier nicht erwähnt.

EvidenceRecordURL

Base64EvidenceRecord

Gibt das zu verwendende Beweisdokument (Evidence Record) gem. RFC 4998 an. Es können eine URL (*EvidenceRecordURL*) oder base64-kodierte Daten innerhalb des HTML-Dokumentes (*Base64EvidenceRecord*) angegeben werden.

DecryptedDocumentPath

Gibt einen Pfad für die Speicherung entschlüsselter Daten an. Die Dateieindung wird jeweils

SaveIncludedDocument

Enthält die Signatur die signierten Daten, so können diese nach der Prüfung im Dateisystem abgelegt werden, falls keine POST URL angegeben ist.

FinishURL

URL, die nach einem erfolgreichen Abschluss des Verifikationsprozesses angezeigt wird.

CancelURL

URL, die nach einem Abbruch des Verifikationsprozesses angezeigt wird.

ErrorURL

URL, die nach einem Fehler des Verifikationsprozesses angezeigt wird.

3.5 Aufruf aus der Kommandozeile

SecSigner kann aus der Kommandozeile aufgerufen werden. Dabei ist das zu signierende Dokument bzw. die zu prüfende Signatur als Parameter zu übergeben. Die entsprechende Main-Methodes stellt die Klasse `secommerce.secsignersigg.SecSignerMain` im `SecSigner.jar` zur Verfügung.

Beispielaufruf:

```
java -classpath SecSigner.jar;SecSignerExt.jar  
secommerce.secsigner.ext.SecSignerMain Datei.ext
```

Dateien werden als Signaturen bzw. als zu entschlüsselnde Dokumente angesehen, wenn sie eine der folgenden Endungen tragen: `.pkcs7`, `.p7`, `.p7s`, `.p7m`, `.pk7`, `.cms`, `.ads`, `encrypted`

3.5.1 Eingebette Signatur

Soll eine eingebettete CAdES-Signatur erstellt werden, so muss an den Dateinamen der String „~!!!embedinsd!!!“ angehängt werden.

Beispielaufruf:

```
java -classpath SecSigner.jar;SecSignerExt.jar  
secommerce.secsigner.ext.SecSignerMain Datei.ext~!!!embedinsd!!!
```

3.5.2 Verschlüsselte Signatur

Soll eine eingebettete CAdES-Signatur erstellt und diese für einen oder mehrere Empfänger verschlüsselt werden, so muss an den Dateinamen der String „~!!!encrypt!!!“ angehängt werden.

Beispielaufruf:

```
java -classpath SecSigner.jar;SecSignerExt.jar  
secommerce.secsigner.ext.SecSignerMain Datei.ext~!!!encrypt!!!
```

3.5.3 PDF-Signatur

Wird ein PDF-Dokument übergeben und soll eine PAdES-Signatur erstellt werden, so muss an den Dateinamen der String „~!!!embedinpdf!!!“ angehängt werden.

Beispielaufruf:

```
java -classpath SecSigner.jar;SecSignerExt.jar  
secommerce.secsigner.ext.SecSignerMain Datei.pdf~!!!embedinpdf!!!
```

3.5.4 PDF-Signatur prüfen

Wird eine PAdES-Signatur übergeben und geprüft diese werden, so muss an den Dateinamen der String „~!!!verify!!!“ angehängt werden.

Beispielaufwurf:

```
java -classpath SecSigner.jar;SecSignerExt.jar  
seccommerce.secsigner.ext.SecSignerMain Datei-signed.pdf~!!!verify!!!
```

3.5.5 XML-DSig-Signatur

Zur Erstellung von XML-DSig-Signaturen werden im Allgemeinen weitere Parameter benötigt. Der Aufruf aus der Kommandozeile erlaubt daher keine XML-DSig-Signatur.

3.5.6 XML-DSig-Signatur prüfen

Wird eine XML-DSig-Signatur übergeben und geprüft diese werden, so muss an den Dateinamen der String „~!!!verify!!!“ angehängt werden.

Beispielaufwurf:

```
java -classpath SecSigner.jar;SecSignerExt.jar  
seccommerce.secsigner.ext.SecSignerMain Datei-signed.xml~!!!verify!!!
```

3.5.7 Daten nur verschlüsseln

Soll die Datei nicht signiert, sondern nur verschlüsselt werden, so muss an den Dateinamen der String „~!!!encryptDataOnly!!!“ angehängt werden.

Beispielaufwurf:

```
java -classpath SecSigner.jar;SecSignerExt.jar  
seccommerce.secsigner.ext.SecSignerMain Datei.ext~!!!encryptDataOnly!!!
```

Beispielaufwurf:

3.6 Lizenzdatei

Kundenspezifische Anpassungen für die Lizenzbedingungen oder das Layout werden über eine gesonderte Lizenzdatei abgebildet. Die Lizenzdatei kann auf folgende Weise verwendet werden:

1. Verweis in der Propertiesdatei. Siehe Abschnitt 2.5.25 SecSigner-Lizenz.
2. Beim Java-Web-Start-Aufruf mittels Kommandozeilenparameter:
-seccommerce.secsigner.licence licence-kundenname.pkcs7
3. Beim Java-Applet-Aufruf mittels Appletparameter:
<PARAM NAME="seccommerce.secsigner.licence" VALUE="licence-kundenname.pkcs7">
4. Bei Einbindung in eine Java-Anwendung über die Java-API:
[Instanz von seccommerce.secsignersigg.SecSigner].setLicence(byte[])
5. Bei Einbindung über die SecSigner-DLL:
*CALLSECSIGNERDLL_API int SecSigner_SetLicence(unsigned char * licence, int licenceLen);*

4 Prüfung des sicheren Betriebszustandes

4.1 Selbstprüfung des SecSigners

Das von der SecCommerce GmbH ausgelieferte Programmarchiv muss mit einer qualifizierten Signatur der SecCommerce GmbH versehen worden sein. Dieser Mechanismus verhindert eine nachträgliche Manipulation des Programmcodes und gewährt die Integrität der Anwendung für den Endanwender. Die Prüfung wird bei jedem Programmstart durchgeführt. Wurde eine Veränderung einer Komponente festgestellt, so wird SecSigner nach Anzeige einer Fehlermeldung unwiderruflich beendet.

Beim Einsatz des SecSigners in einem Java-Applet prüft die Java-VM beim Laden der Java-Klassen aus SecSigner.jar die fortgeschrittene SecCommerce-Signatur über das JAR. Die hier beschriebene Prüfung findet davon unabhängig statt und erfolgt auch, wenn SecSigner nicht in einem Browser verwendet wird.

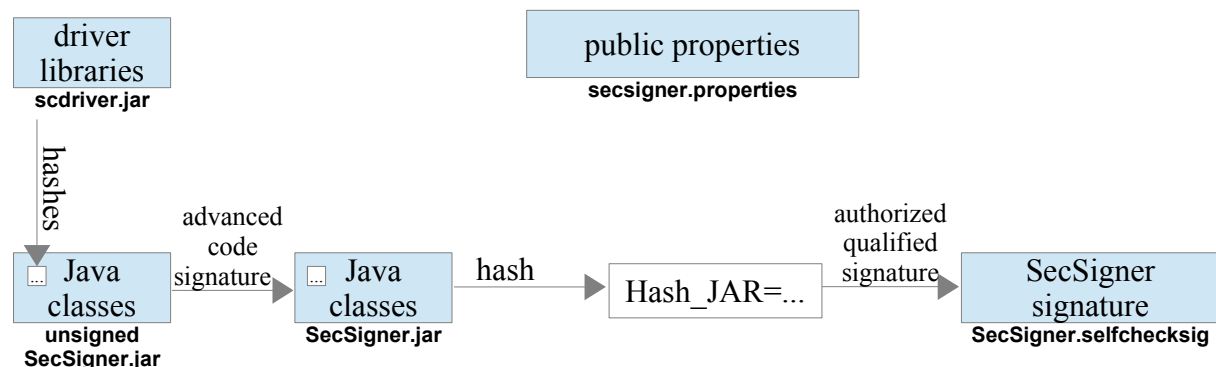


Abbildung 1: von der Selbstprüfung abgedeckte Komponenten

Die Selbstprüfung bezieht sich auf folgende Komponenten:

- Programmarchiv
- Treiberdateien

Es sind sämtliche Hashwerte der an der Selbstprüfung beteiligten Zertifikate sowie aller verwendeter Treiberdateien im SecSigner-Programmcode abgelegt. Die in Abbildung 1 gezeigte Kette wird bei jedem Programmstart verifiziert. Es wird geprüft, dass die qualifizierte Signatur über das Programmarchiv von einem berechtigten Zertifikat eines SecCommerce-Mitarbeiters stammt.

Die zur Signatur des SecSigners berechtigten Mitarbeiter sind auf der Hersteller-Website gelistet: <https://seccommerce.com/download-secsigner-seccsign-id/>

Diese Berechtigung gilt ebenso für die Signaturen (pkcs7-Dateien) über die Zip-Dateien, bzw. die SecSigner-5-Setup.exe, in der SecSigner ausgeliefert wird.

Details über die Selbstprüfung können durch Auswahl des Feldes *Integritätsprüfung OK* links unten in jedem Dialog angezeigt werden:



Abbildung 2: Details zur Selbstprüfung

Die Selbstprüfung befreit den Anwender nicht davon, den SecSigner nur auf einem sicheren System, das frei von böswilliger Software ist, auszuführen! Hat ein Angreifer bereits die Kontrolle über den Computer des Angreifers erlangt, ist es ihm auch möglich, die auf diesem Computer laufende Selbstprüfung des SecSigners zu manipulieren.

4.2 Prüfung der Konfigurationsdatei

Nach erfolgreichem Abschluss der Selbstprüfung, prüft SecSigner, ob die in der Konfigurationsdatei eingestellten Optionen einen sicheren Betrieb entsprechend der Vorgaben in der Herstellererklärung erlauben.

- ! Optionen, die die Sicherheit des Betriebes beeinträchtigen können, sind in Abschnitt 2.5 mit einem roten Ausrufezeichen gekennzeichnet.

So ist beispielsweise die Erstellung qualifizierter Signaturen nur mit SigG-bestätigten sicheren Signaturerstellungseinheiten möglich. Da es eine solche Bestätigung (bislang) nur für Signaturkarten gibt, darf im SecSigner die Verwendung von Schlüsseln aus Dateien (sogenannte Softwarezertifikate) nicht erlaubt sein, wenn gemäß der Vorgaben der Herstellererklärung qualifiziert signiert werden soll.

Es werden vier Betriebszustände unterschieden. Der aktive Zustand wird in jedem Dialog im linken, unteren Bereich angezeigt:

- *Sicherheitseinstellungen hoch*: Die Konfiguration bietet keine Möglichkeiten, die einer qualifizierten Signatur im Wege stehen. Nur für diesen Modus gelten Herstellererklärungen nach Signaturgesetz des SecSigners.
- *Sicherheitseinstellungen mittel*: Die Konfiguration erlaubt z.B. die Eingabe der PIN an der Tastatur des PCs statt direkt am Kartenlesers. Die Konfiguration erfüllt nicht alle Vorgaben aus der Herstellererklärung. Möchte der Anwender trotzdem qualifiziert signieren, muss er „andere geeignete Maßnahmen zur Sicherheit qualifizierter elektronischer Signaturen treffen“¹.

¹Satz 4 des § 17 Abs. 2 Signaturgesetz

- *Testmodus.* Die Konfiguration erlaubt z.B. die Verwendung von Testzertifikaten, d.h. von solchen Zertifikaten, die keiner Person zugeordnet werden können. Damit ist selbst die Erstellung fortgeschrittener Signaturen nicht möglich.
- *Nicht SigG-Funktion aktiv:* Es ist eine Funktion wie etwa Ver- oder Entschlüsselung aktiv, die nicht im Zusammenhang mit dem Signaturgesetz steht.

4.3 Prüfung des Zertifikatstapels

Umfasst das Repository des verwendeten SecSigners die Ausstellerzertifikate eines zur Erstellung oder Prüfung verwendeten Signaturzertifikates nicht, so kann SecSigner einen Zertifikatstapel von der Website des Herstellers nachladen.

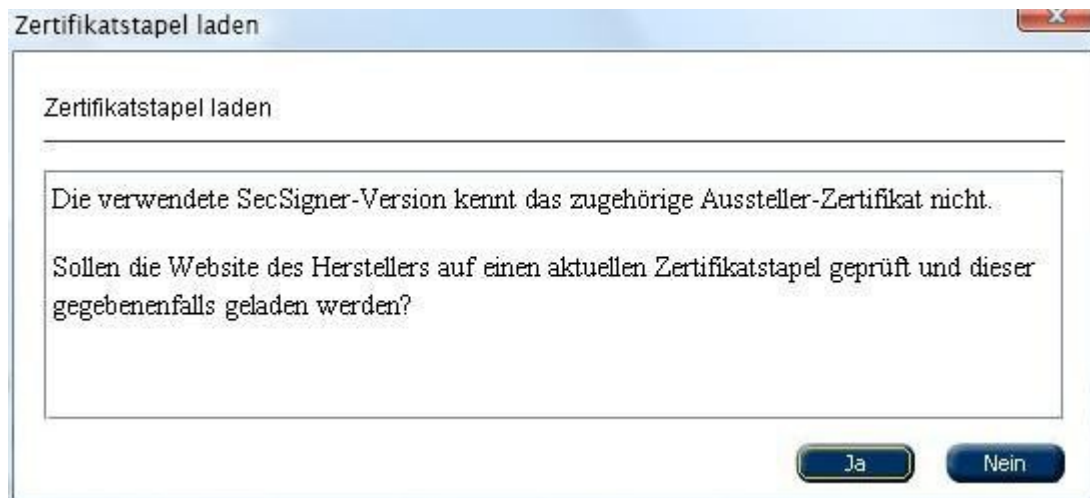


Abbildung 3: Zertifikatstapel laden

Der Stapel ist analog zum Programmarchiv signiert und die Prüfung erfolgt analog zur oben beschriebenen Selbstprüfung (vgl. 4).

5 Signaturprüfung

Wenn Sie keine zu prüfende Signatur übergeben haben, werden Sie zunächst aufgefordert, eine Signaturdatei einzulesen. Wählen Sie dazu bitte die Option 'Signaturdatei laden':



Abbildung 4: Signaturdatei einlesen

Laden Sie anschließend das zugehörige Dokument mit 'Dokument laden'.

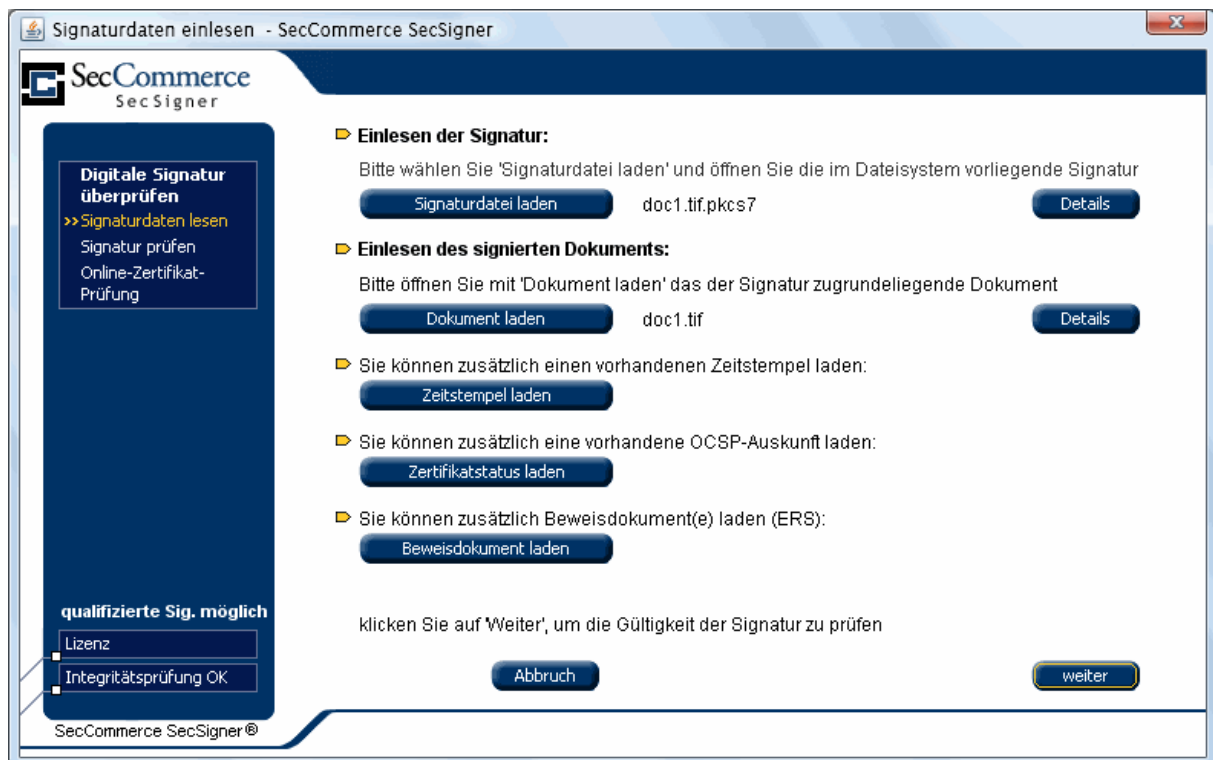


Abbildung 5: Dokument, Zeitstempel, Sperrauskunft und Beweisdokument einlesen

Optional können in diesem Dialog auch folgende weitere Daten geladen werden:

- ein externen Zeitstempel, der sich auf die Signaturdatei bezieht und somit deren Mindestalter beweist, sofern die Siganaturdatei selbst nicht ohnehin schon Zeitstempel enthält,
- ein vom Trustcenter signierter Zertifikatsstatus (Sperrauskunft), der belegt, ob das Signaturzertifikat zum Signaturzeitpunkt oder danach gültig war, sofern die Signaturdatei selbst nicht ohnehin schon Sperrauskünfte enthält,
- ein Beweisdokument, welches nachweist, dass eine auf abgelaufenen Algorithmen beruhende Signatur rechtzeitig übersigniert wurde und damit ihre Beweiskraft erhalten hat.

Anschließend wird Ihnen das Ergebnis der Signaturprüfung dargestellt:



Abbildung 6: Signaturprüfergebnis

Sie können durch Wahl der entsprechenden Optionen nun auch eine Online-Zertifikatprüfung durchführen und/oder einen Prüfbericht erstellen lassen.



Abbildung 7: Sperrabfrage des Zertifikates beim ausgebenden Trustcenter

Handelt es sich beim signierten Dokument um eine Text-, HTML-, XML- oder PDF-Datei, so kann diese mit der Option „Dokument anzeigen“ sicher angezeigt werden:



Abbildung 8: sichere Anzeige eines signierten PDF-Dokuments

Details zur Signaturprüfung sind einem eigenen Dialog verfügbar:



Abbildung 9: Details zur Signaturprüfung

Wurde ein Beweisdokument geladen, kann es hier über den Knopf *Beweisdokument* angezeigt werden. Ein Beweisdokument sichert die Beweiskraft einer Signatur, die Algorithmen verwendet, die inzwischen nicht mehr sicher sind. So wird etwa die Signatur eines 1024-Bit-RSA-Schlüssels heute nicht mehr als ausreichend sicher angesehen. Liegt jedoch ein Beweisdokument vor, welches den Hashwert über die Signatur enthält, sowie einen Zeitstempel eines Trustcenters, welcher beweist, dass die Signatur schon existierte, als 1024 Bit noch sicher waren, dann ist die Signatur weiterhin gültig.

Die Erstellung solcher Beweisdokumente kann mit dem SecCommerce-SecPKI-Server erfolgen. Das Format ist in RFC 4998 spezifiziert.

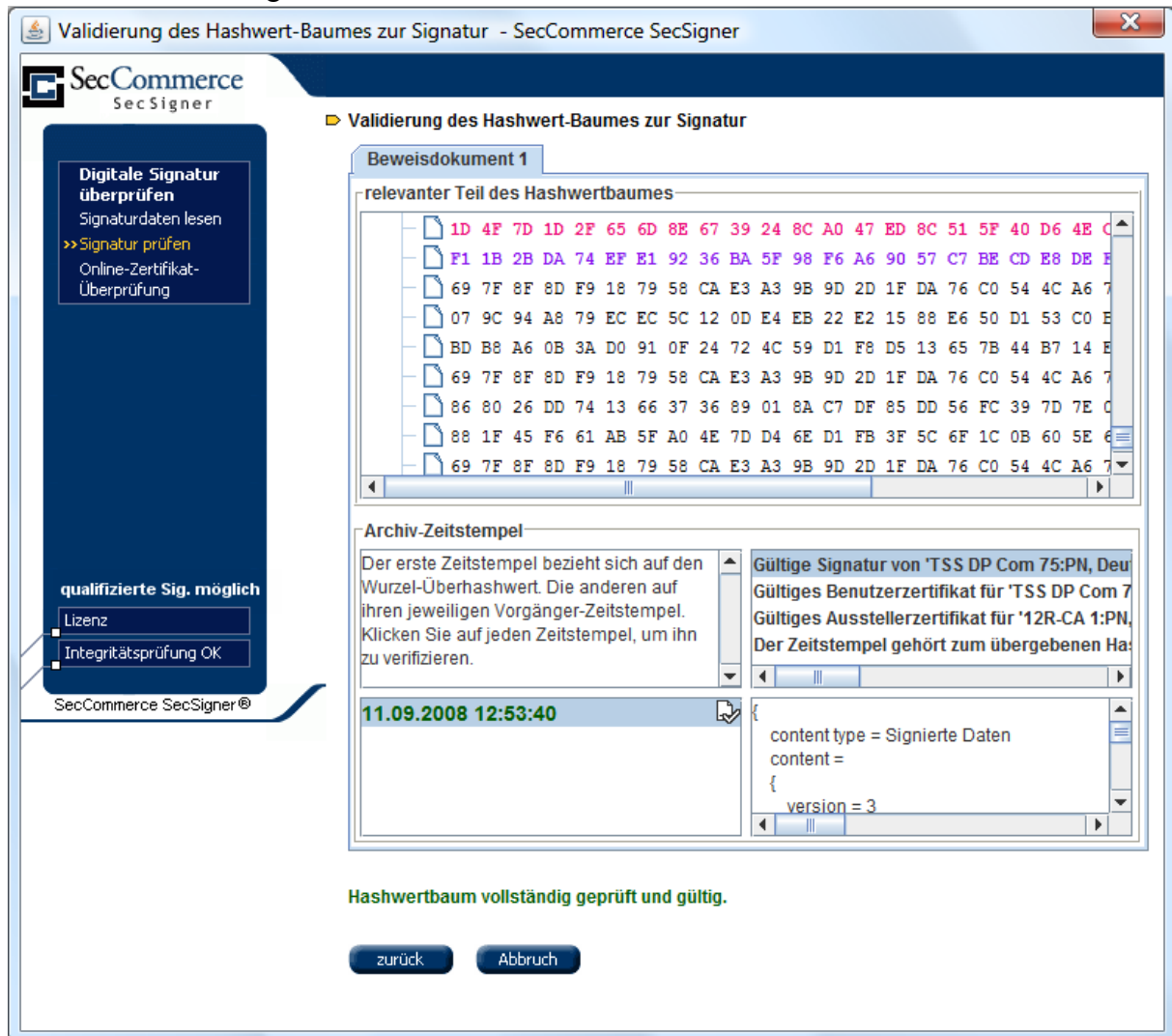


Abbildung 10: Anzeige eines Beweisdokumentes mit vergrößertem SecSigner-Fenster

Der Dialog zeigt oben farblich markiert die Hashwerte über Signatur, Dokument und Zeitstempel. Letztere beiden nur falls vorhanden. Links unten wird die Liste der Archivzeitstempel gezeigt. Eine grüne Anzeige seines Datums symbolisiert, dass der jeweilige Archivzeitstempel gültig ist. Die zugehörigen Prüfdetails finden sich rechts unten.

Die unterste Zeile zeigt das Gesamtprüfergebnis des Beweisdokumentes.

Hinweis: Die Bildschirmfotos zeigen einen Fall, in dem kein Beweisdokument erforderlich gewesen wäre, da die Algorithmen der Signatur noch gültig sind.

6 Erzeugen einer digitalen Signatur

6.1 Dokument laden

Wenn Sie kein zu signierendes Dokument übergeben haben, werden Sie zunächst aufgefordert, ein solches Dokument einzulesen. Wählen Sie dazu bitte die Option 'Dokument laden':

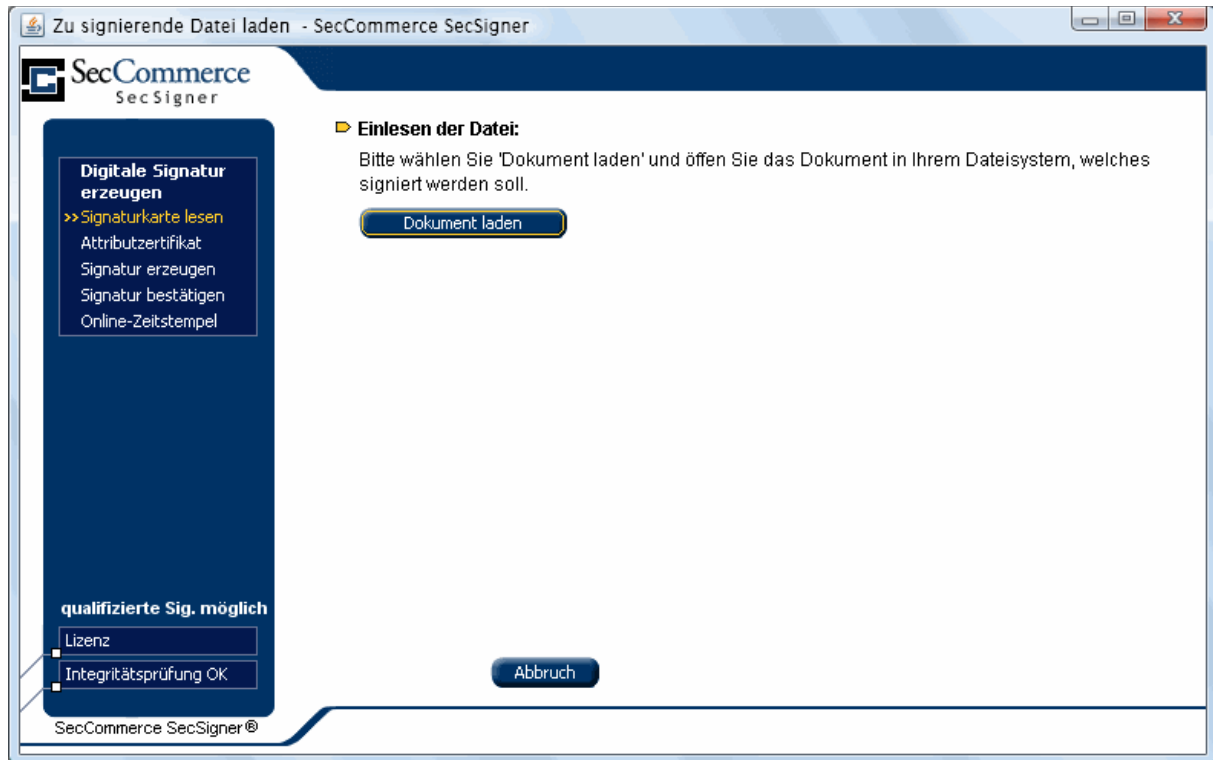


Abbildung 11: zu signierendes Dokument laden

6.2 Initialisierung

Nach Wahl von „weiter“ wird die Dialog zur Kartenleserinitialisierung geöffnet. Sollten in der Konfigurationsdatei bereits die Kartenleser eingetragen worden sein, werden diese Kartenleser nach Auswahl von *Signaturkarte suchen* angesprochen, sowie die eingelegten Karten gelesen. Der Signaturkartentyp (Trustcenter) wird bei der Initialisierung automatisch erkannt.

Enthält die Konfigurationsdatei keine Angaben zu den Kartenlesern, wird genau ein Kartenleser automatisch gesucht.



Abbildung 12: Signaturkarten suchen

In der auf die Initialisierung folgenden Maske können Sie Attributzertifikate (siehe Kapitel 7.2) in die Signatur einbinden oder das Verschlüsselungszertifikat speichern. Werden mehrere Kartenleser verwendet, kann in der Mitte links im Dialog der Kartenleser gewählt werden, dessen Informationen angezeigt werden sollen.



Abbildung 13: Informationen über die Signaturkarte im Kartenleser

6.3 Sichere Dokumentenanzeige

Mit der Option *weiter* gelangen Sie zur sicheren Anzeige. Bei Verwendung mehrerer Kartenleser wird in der Mitte links die Nummer des Kartenlesers angezeigt, der das gerade angezeigte Dokument signieren wird. Die Zuordnung von Signaturkarten zu Dokumenten steht hier bereits fest und kann sich während der Signatur nicht mehr ändern.



Abbildung 14: ein Stapel von 16 TIFF-Dokumenten in der sicheren Anzeige

Prüfen Sie das Dokument und signieren Sie es anschließend.

6.4 Signatur

Nach Wahl von *signieren* werden Sie aufgefordert, die PIN(s) der Signaturkarte(n) einzugeben.

6.4.1 Kartenlesegerät mit eigener PIN-Eingabe

Wenn Sie über ein Kartenlesegerät mit eigener PIN-Eingabe und geeigneten Treibern verfügen, geben Sie die PIN am Lesegerät ein:



Abbildung 15: PIN-Eingabe-Dialog Reiner-SCT

6.4.2 Kartenlesegerät ohne eigene PIN-Eingabe

Wenn Sie über ein Kartenlesegerät ohne eigene PIN-Eingabe verfügen oder Ihr Lesegerät mit eigener PIN-Eingabe ohne geeignet Treiber betreiben, so erfolgt die PIN-Eingabe über die Tastatur des PCs:



Abbildung 16: PIN-Eingabe über die Tastatur des Computers

Auf diese Weise kann keine qualifizierte Signatur erzeugt werden. Dafür ist die sichere PIN-Eingabe direkt am Kartenleser erforderlich.

6.4.3 Signaturfortschritt

Nach rechts wachsende Balken symbolisieren den Fortschritt der Signatur in den Signaturkarten.

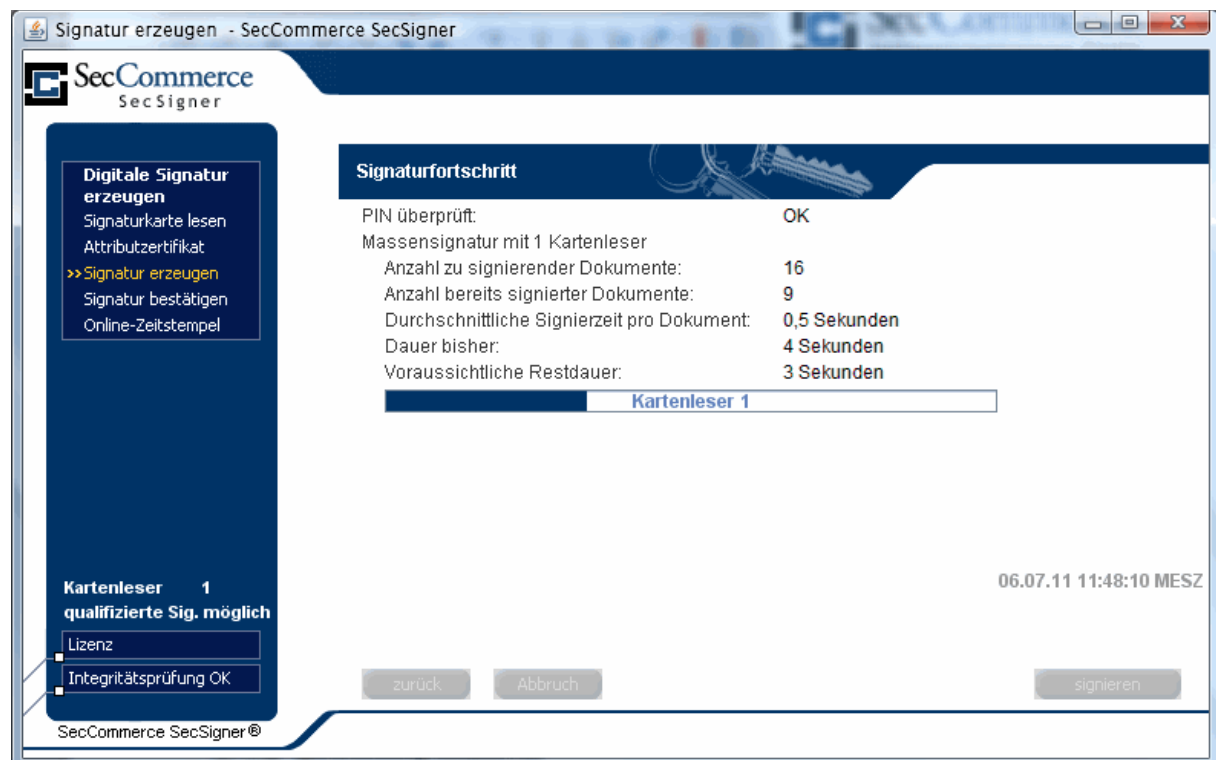


Abbildung 17: Signaturfortschritt am Kartenleser

6.4.4 Zeitstempeldienst

Wenn die entsprechende Option gesetzt ist, können Sie anschließend Ihre Signatur mit einem Zeitstempel versehen. Wählen Sie dazu die Option „Zeitstempel“. Wenn Sie in der Konfigurationsdatei mehrere Zeitstempeldienste vorkonfiguriert haben (vgl. Abschnitt 2.5.5), gelangen Sie in den Auswahldialog. Wählen Sie dort einen Zeitstempeldienst. Dieser Trustcenter muss nicht dasselbe sein, dass Ihre Signaturkarte erstellt hat.

SecSigner fügt den empfangenen Zeitstempel in die erstellte Signatur ein.

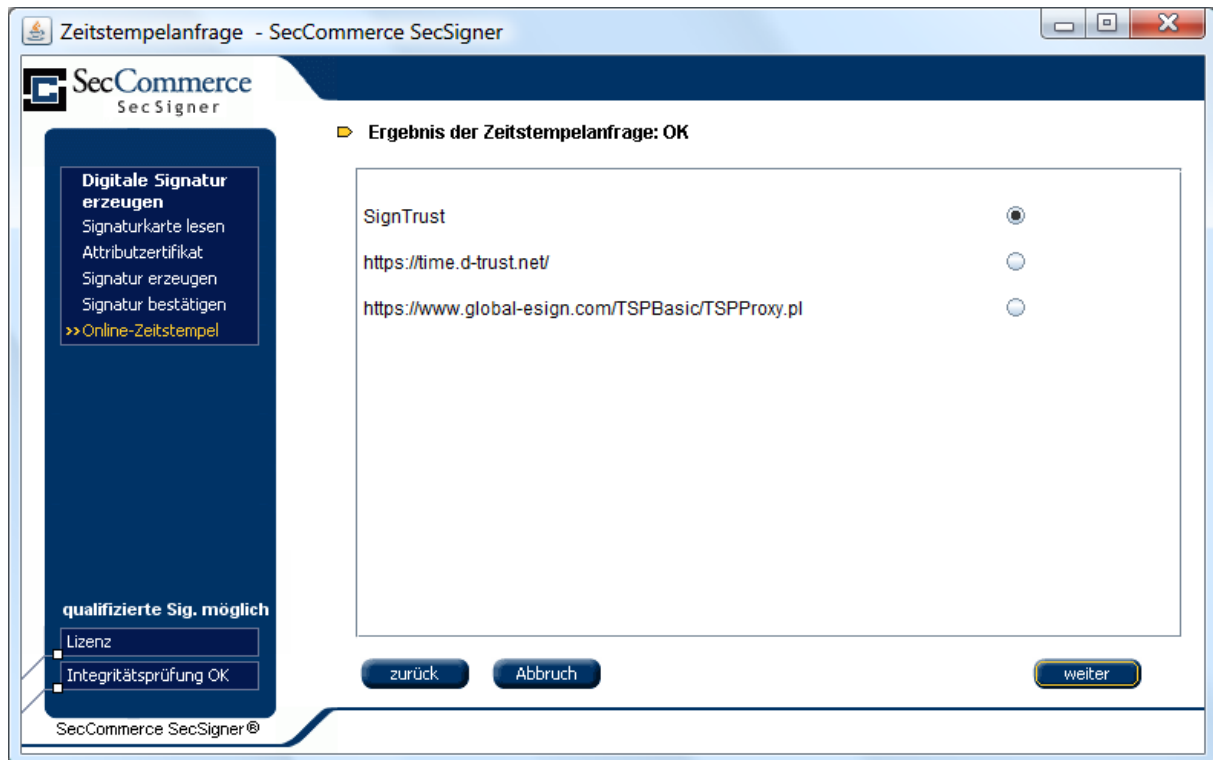


Abbildung 18: Zeitstempelanforderung

6.4.5 Bestätigung der Signatur

Bestätigen Sie abschließend den Signaturvorgang mit der Option *bestätigen*:

**Abbildung 19: Signatur bestätigen**

Je nach Art des SecSigner-Aufrufs finden Sie nach erfolgreicher Signaturerstellung im Verzeichnis des Dokuments zusätzlich die Signaturdatei gleichen Namens mit der Endung *.pkcs7* oder es wird abschließend ein Dateiauswahldialog angezeigt, in dem die Signaturdatei gespeichert werden kann oder die Signatur wird an den Web-Server geschickt, welcher den SecSigner gestartet hatte.

Falls Sie die Signatur mit einem Empfängerzertifikat verschlüsselt haben, erhält die Signatur die Endung *_encrypted.pkcs7*. (Siehe Kapitel 7.4)

7 Weitere Funktionen des SecSigners

7.1 Online-Zertifikatprüfung

Während der Signaturprüfung können Sie die Zertifikatgültigkeit der verwendeten Signaturkarten testen. Es wird geprüft,

- ob die Zertifikate von den vorkonfigurierten Herausgeberzertifikaten abstammen,
- ob das Nicht-Gültig-Nach-Datum der Zertifikate bereits erreicht wurde,
- ob der Zertifikatsstatus (OSCP-Statusabfrage beim Trustcenter) OK ist.

7.2 Attributzertifikate einbinden

Attributzertifikate sind zusätzliche Zertifikate in Dateiform, die zu einem Hauptzertifikat auf der Signaturkarte existieren können, um beispielsweise die Haftung für elektronische Signaturen zu begrenzen oder eine berufsrechtliche Zulassung anzugeben. Das Attributzertifikat kann bei der Erzeugung der Signatur in das erzeugte Signaturobjekt mit aufgenommen werden.

Die Einbindung der Attributzertifikate kann automatisiert erfolgen. Zur Konfiguration vgl. 2.5.20.

7.3 Entschlüsselung

Erhalten Sie ein Dokument, welches mit Ihrem Verschlüsselungszertifikat verschlüsselt worden ist, so wird dieses beim Öffnen im SecSigner entschlüsselt.

Dazu wählen Sie nach der Initialisierung der Signaturkarte den Menüpunkt *Daten entschlüsseln*:



Abbildung 20: Entschlüsselung

SecCommerce SecSigner

Sie werden nun aufgefordert, die PIN der Signaturkarte am Lesegerät einzugeben und verfahren anschließend entsprechend der Anleitung zur Signaturprüfung (Siehe Kapitel 5).

Handelt es sich bei dem entschlüsselten Dokument nicht um eine Signaturdatei, so können Sie dieses im Anschluss an die Entschlüsselung als Datei abspeichern.

Für die Entschlüsselung ist eine entsprechende SecSigner-Lizenzdatei erforderlich.

7.4 Verschlüsselung einer Signatur

Wollen Sie eine Signatur zum Versand verschlüsseln, so benötigen Sie das Verschlüsselungszertifikat des Empfängers. (Siehe Kapitel 7.6)

Sofern dem SecSigner beim Aufruf der Signaturfunktion der Wunsch zur Verschlüsselung mitgeteilt wurde, Im Anschluss an Initialisierung, Anzeige und Signatur gelangen Sie zum Verschlüsselungsdialo:

Bereits verwendete oder beim SecSigner-Aufruf übergebene Verschlüsselungszertifikate werden in der Liste angezeigt.

Wählen Sie den Menüpunkt *Empfängerzertifikat hinzufügen* und laden Sie das Verschlüsselungszertifikat des Empfängers, falls es in der Liste noch nicht enthalten ist.

Bedenken Sie, dass die Verschlüsselung nicht für eine Person, sondern für dessen Zertifikat erfolgt. Es ist möglich, dass eine Person mehrere Zertifikate besitzt. Es ist auch möglich, dass das Zertifikat bereits abgelaufen ist, wenn die verschlüsselte Nachricht den Empfänger erreicht. Er kann die Nachricht dann evtl. nicht mehr entschlüsseln.



Abbildung 21: Laden aus Auswahl der Verschlüsselungszertifikate

Abschließend bestätigen Sie den Verschlüsselungsvorgang und erhalten zur Ausgangsdatei die verschlüsselte Signatur mit der Endung *_encrypted.pkcs7*.

Für die Verschlüsselung ist eine entsprechende SecSigner-Lizenzdatei erforderlich.

7.5 Verschlüsselung einer Datei ohne Signatur

SecSigner bietet die Möglichkeit, Dokumente mit X509-Zertifikaten entsprechend der Vorgaben des CMS-Standards zu verschlüsseln, ohne diese zu signieren. (Vgl. auch 3.4.1)

7.6 Ablage des Verschlüsselungszertifikats als Datei

Wenn Sie Ihr Verschlüsselungszertifikat zur Weitergabe als Datei ablegen möchten, wählen sie nach der Initialisierung des Kartenlesegeräts den Menüpunkt *Verschlüsselungszertifikat speichern*:

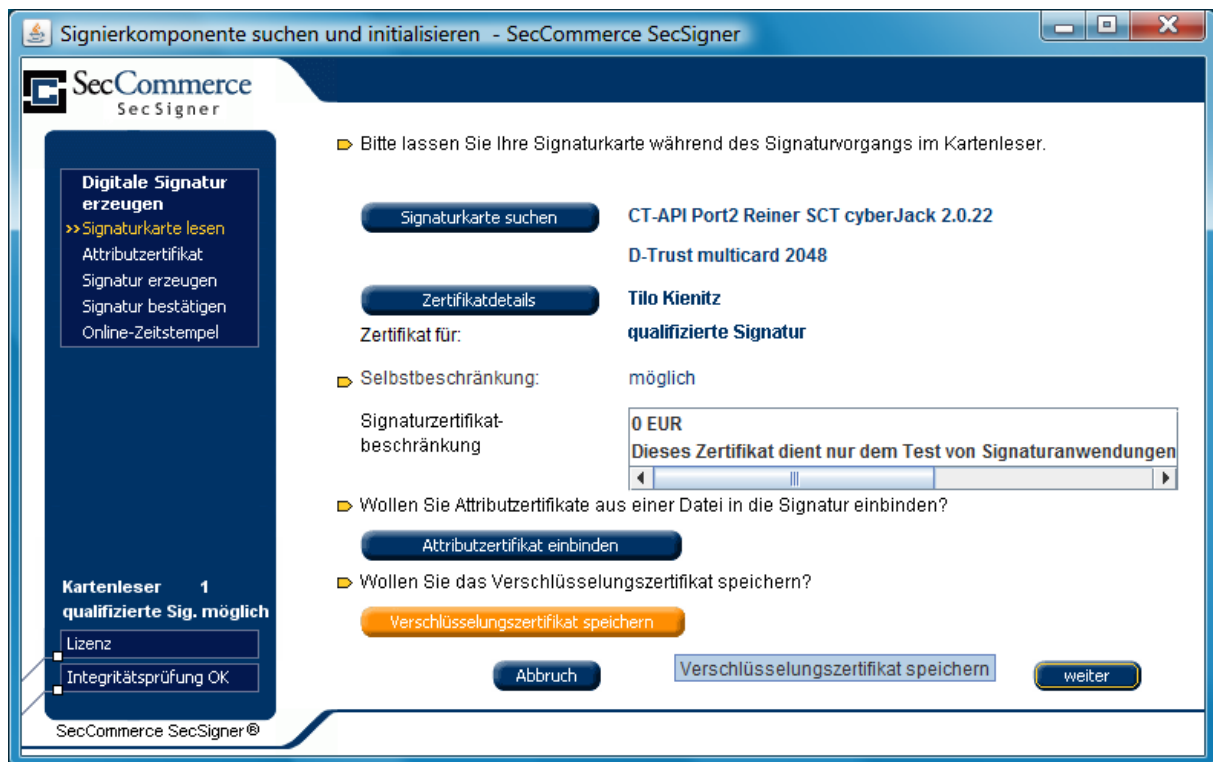


Abbildung 22: Verschlüsselungszertifikat speichern

Nach der Suche Ihrer Signaturkarte haben Sie im Fileauswahldialog die Möglichkeit, Ihr Verschlüsselungszertifikat im Filesystem zu speichern:

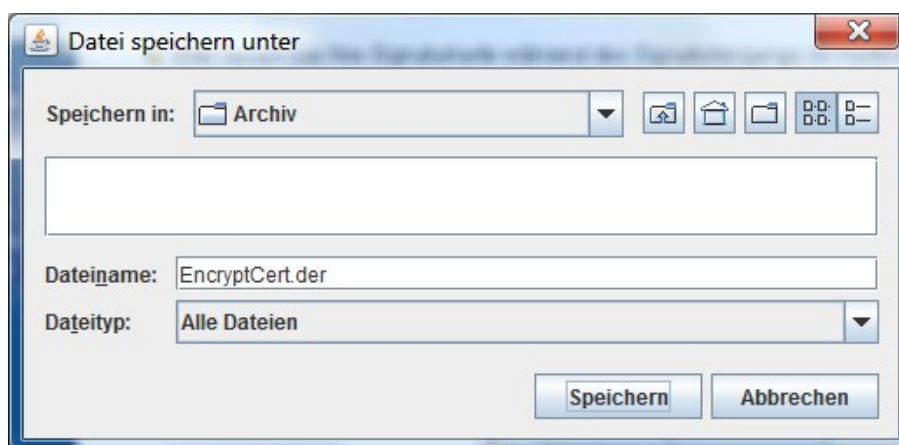


Abbildung 23: Dateiauswahl für Verschlüsselungszertifikat

7.7 PDF-Signatur-Annotation

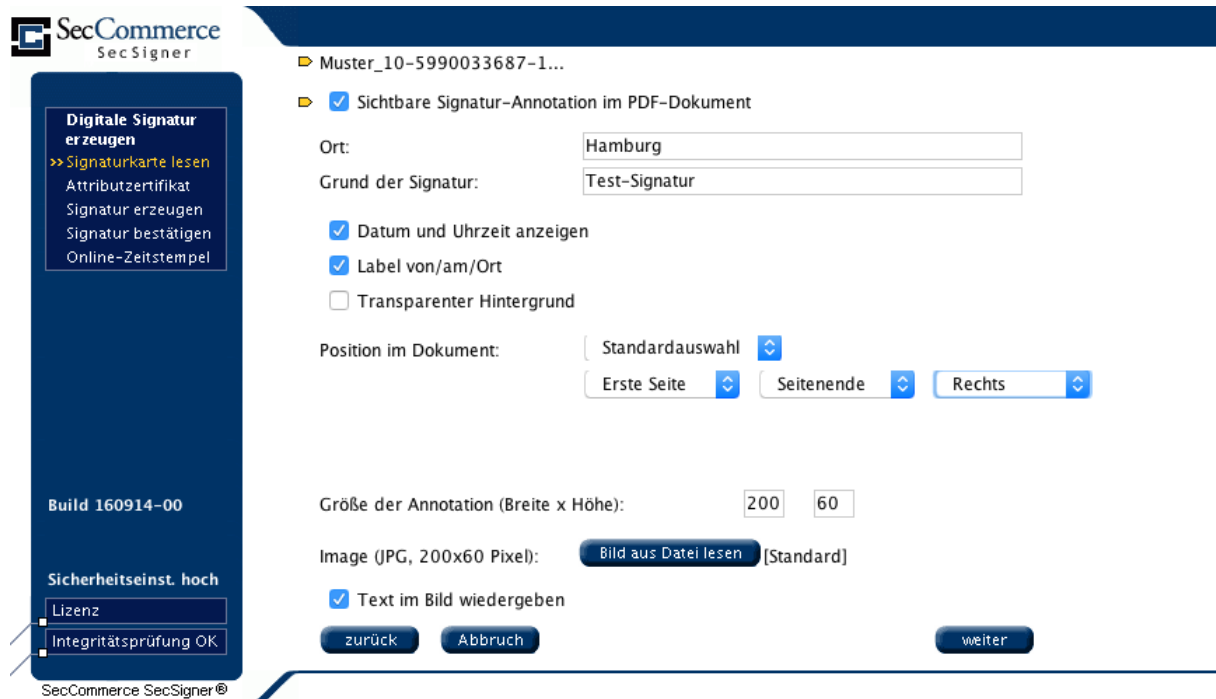
Sofern als Signaturmodus die PDF-Signatur (PADES) gewählt wurde, wird die Signatur wird

SecCommerce SecSigner

in Form einer Annotation ins PDF-Dokument eingebettet. Sie können optional Angaben für diese Annotation ergänzen, darunter

- Ort und Grund für die Signatur
- die Position der Annotation im Dokument
- die Breite und Höhe der Annotation im Dokument
- ein in die Annotation einzubindendes Icon (JPG, max. 200 x 70 Pixel)
- ein Icon (JPG, max. 70 x 70 Pixel) zur Darstellung eines externen Links (z.B. auf eine Signaturanwendungskomponente zur Prüfung der Signatur) mit zugehöriger URL

Nach dem Initialisierungsdialog (vgl. Abschnitt 3.1) können Sie diese Angaben in einem eigenen Dialog hinzufügen:



SecCommerce SecSigner

Digitale Signatur erzeugen

>> Signaturkarte lesen
Attributzzertifikat
Signatur erzeugen
Signatur bestätigen
Online-Zeitstempel

Build 160914-00

Sicherheitseinst. hoch
Lizenz
Integritätsprüfung OK

SecCommerce SecSigner®

Muster_10-5990033687-1...

☒ Sichtbare Signatur-Annotation im PDF-Dokument

Ort: Hamburg

Grund der Signatur: Test-Signatur

☒ Datum und Uhrzeit anzeigen
☒ Label von/am/Ort
☐ Transparenter Hintergrund

Position im Dokument: Standardauswahl
Erste Seite Seitenende Rechts

Größe der Annotation (Breite x Höhe): 200 60

Image (JPG, 200x60 Pixel): Bild aus Datei lesen [Standard]

☒ Text im Bild wiedergeben

zurück Abbruch weiter

Abbildung 24: PDF-Signatur-Annotation

Die Anzeige des Dialogs kann in der Konfigurationsdatei ausgeschaltet werden. (vgl. Abschnitt 2.5.15).

Die Angabe der Daten zum externen Link müssen in der Konfigurationsdatei eingeschaltet werden. (vgl. Abschnitt 2.5.15).

8 SecSigner und Windows

8.1 Installierbare Version

Auf der Website steht der SecSigner als „Setup_[Version].exe“ zum Download zur Verfügung:

<https://seccommerce.com/download-secsigner-secsign-id/>

Zu Installation öffnen Sie die Datei Setup_[Version].exe“ durch Doppelklick. Akzeptieren Sie anschließend den Lizenzvertrag und bestätigen Sie die Installation im gewünschten Ordner.

Während der Installation können Sie SecSigner optional auch als Plug-In für Adobe Acrobat, als CSP für Windows-Anwendungen, oder als Makro für Microsoft Office installieren:



Abbildung 25: Installationsmenü SecSigner

8.2 SecSigner als Adobe Plug-In

Wenn Sie **SecSigner** auch als Adobe Plug-In benutzen wollen, können Sie die Zuordnung während der Installation vornehmen (vgl. Abbildung 25).

Um SecSigner als Standardanwendung für die Signaturprüfung festzulegen, wählen Sie bitte im Acrobat Professional / Acrobat Reader:

Bearbeiten -> Grundeinstellungen -> Sicherheit -> Erweiterte Grundeinstellungen -> Verifikation -> „Immer Standard-Methode verwenden“

bzw.

Edit -> Preferences -> Security -> Advanced Preferences -> Verification -> "Always use the default method (overrides the document-specific method)"

Wählen Sie dann in der Auswahlbox "SecCommerce.SecSignerPDF".

8.3 SecSigner als CryptoServiceProvider (CSP)

Die Signaturkarte wird z.B. unter MS-Outlook nicht erkannt, es werden lediglich die Eigenen Zertifikate aus dem Windows-Zertifikatsspeicher angezeigt.

Um die Zertifikate von der Smartcard in den Windows-Zertifikatsspeicher einzutragen und die Verbindung zu unserem CSP herzustellen, kann etwa das frei verfügbare Tool SecCardAdmin genutzt werden:

<https://www.seccommerce.com/> → Produkte / SecCardAdmin

1. "Automatische Suche"
2. "Zertifikatverwaltung"
3. "Signatur-Zertifikat" - "Details"
4. "Eintragen"

Analog kann für die anderen Zertifikate auf der Karte verfahren werden.

Die Verwendung des CSP wird nicht empfohlen, da diese von Microsoft spezifizierte Schnittstelle sinnvolle Signaturanwendungen nach unserer Ansicht kaum möglich macht.

8.4 Initialisierung der Signaturkarte

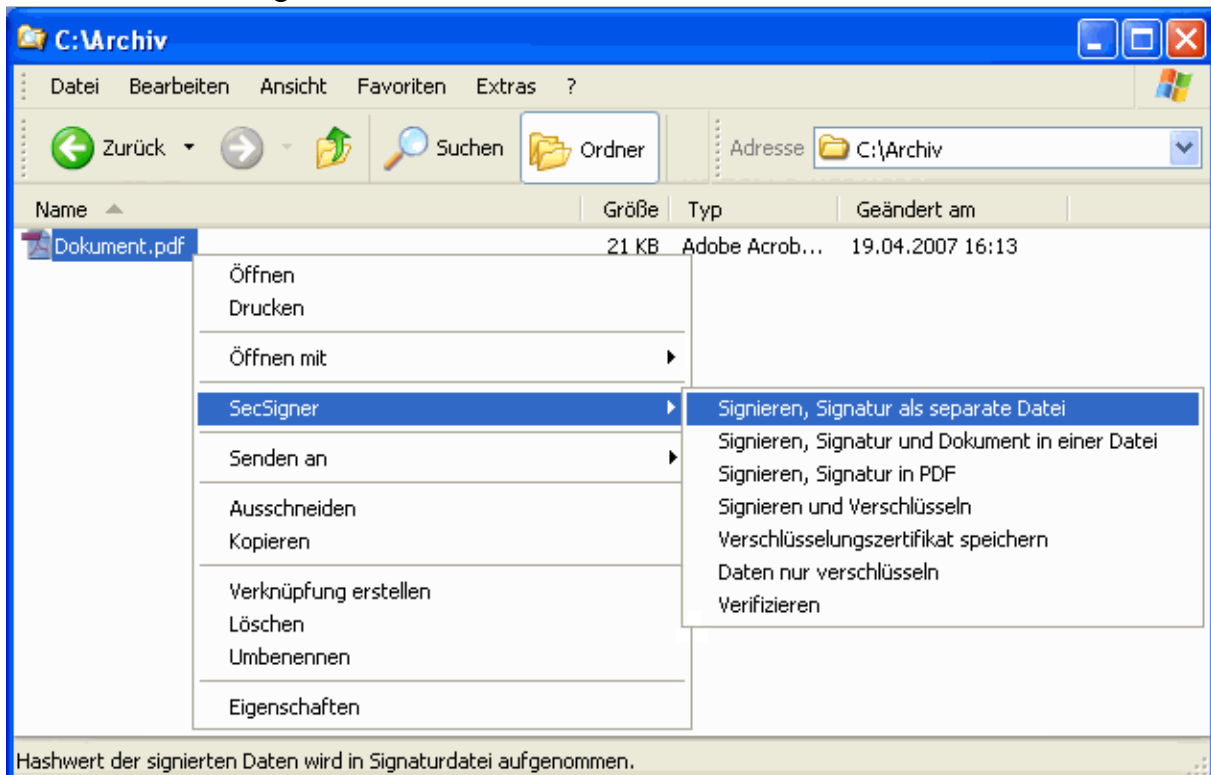
Um eine Signaturkarte erstmalig zu nutzen, schalten Sie diese bitte zunächst mit unserem Administrationstool SecCardAdmin frei:

<https://www.seccommerce.com/> → Produkte / SecCardAdmin

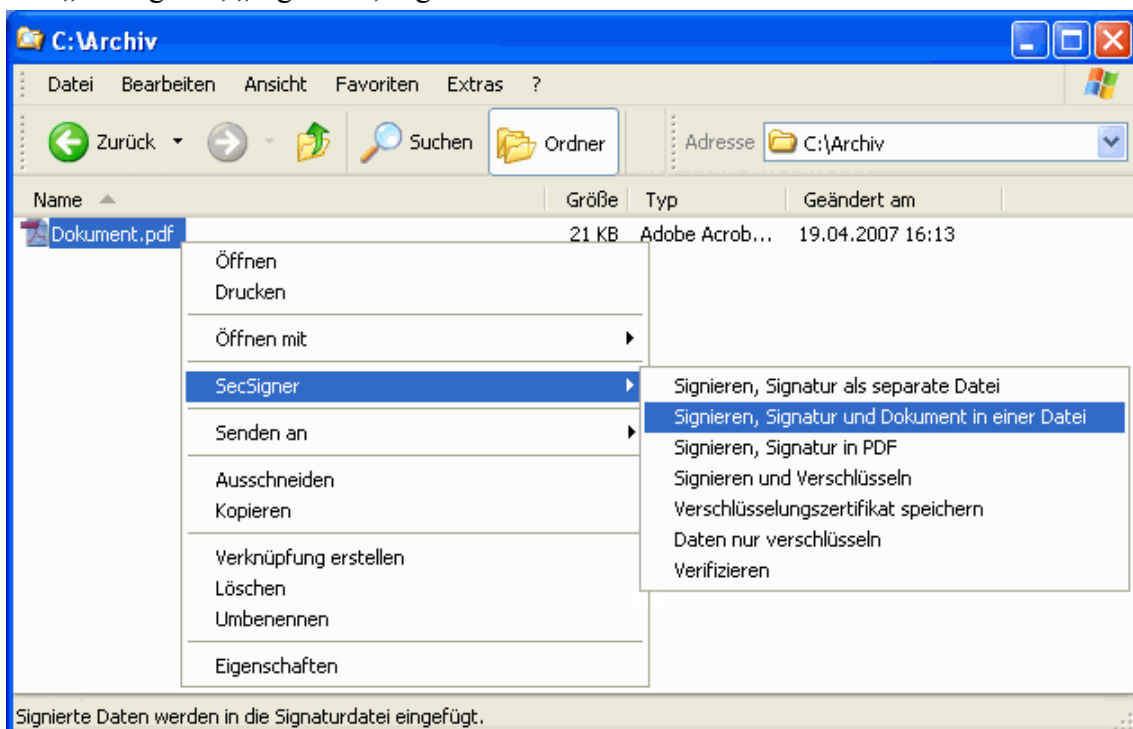
8.5 Installierter SecSigner unter Windows

8.5.1 Erzeugen einer Signatur

Nach der Installation des **SecSigners** können Dokumente beliebigen Typs mit einer digitalen Signatur versehen werden. Der Klick auf ein Dokument im Windows-Explorer mit der *rechten Maustaste* bietet die Optionen „SecSigner“, „Signieren, Signatur als separate Datei“.



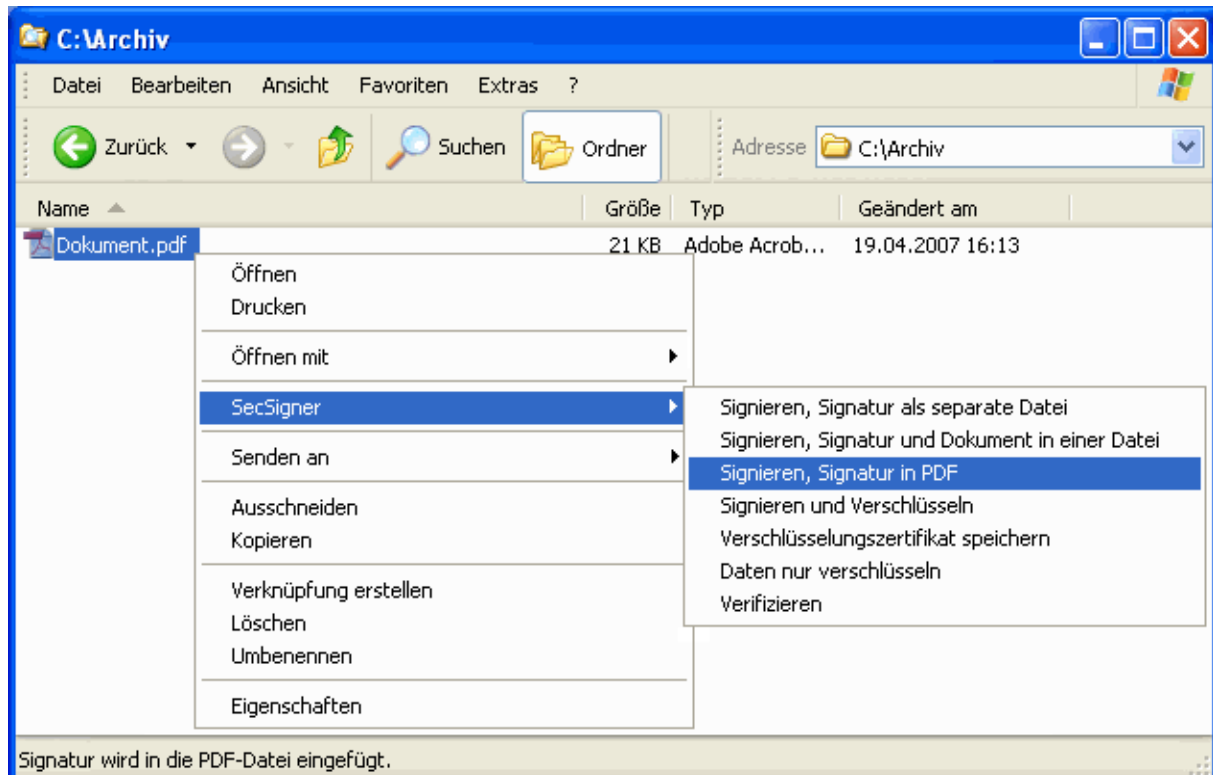
und „SecSigner“, „Signieren, Signatur und Dokument in einer Datei“.



Zum

weiteren Ablauf der Signaturerzeugung siehe Abschnitt 6.

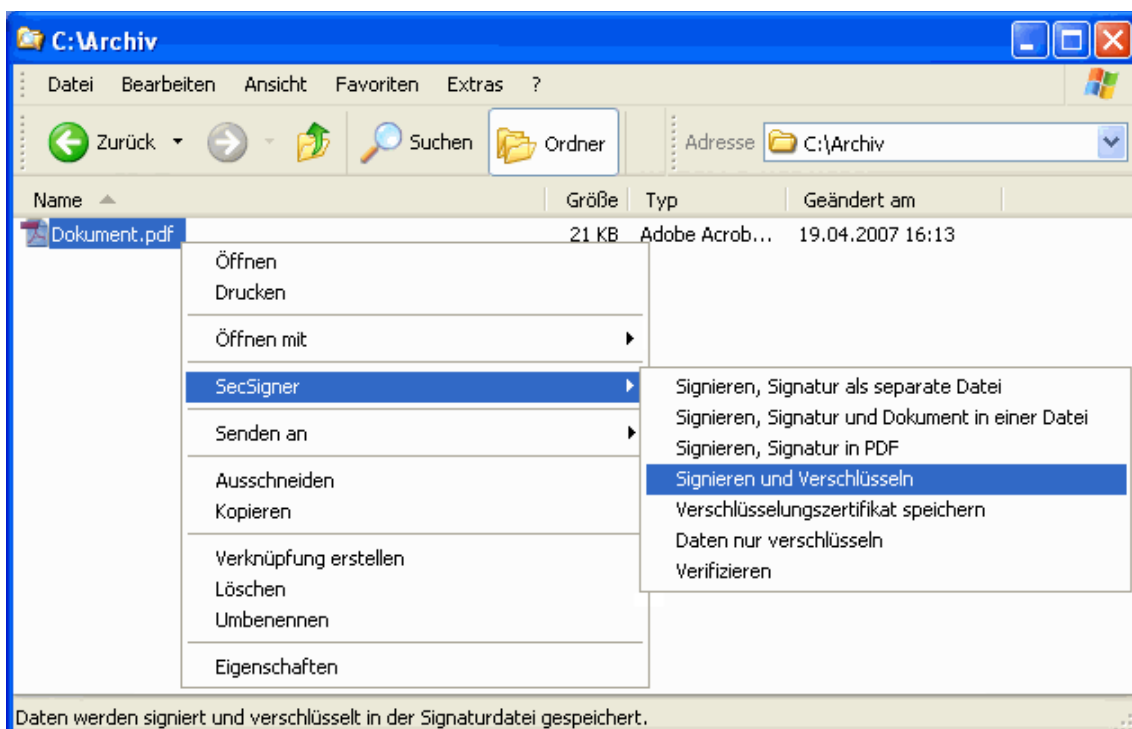
Für PDF-Dokumente können auch PDF-Signaturen (PAdES) erzeugt werden (vgl. auch Abschnitt 7.7). Wählen Sie dazu den Menüpunkt „Signieren, Signatur in PDF“:



8.5.2 Signieren und verschlüsseln

Wollen Sie eine Signatur zum Versand verschlüsseln, so benötigen Sie das Verschlüsselungszertifikat des Empfängers. (Siehe 8.5.4)

Der Ablauf der Signatur mit Verschlüsselung gleicht dem oben beschriebenen: Beim Klick mit der rechten Maustaste auf ein Dokument wählen Sie dazu die Option „SecSigner“, „Signieren und Verschlüsseln“:

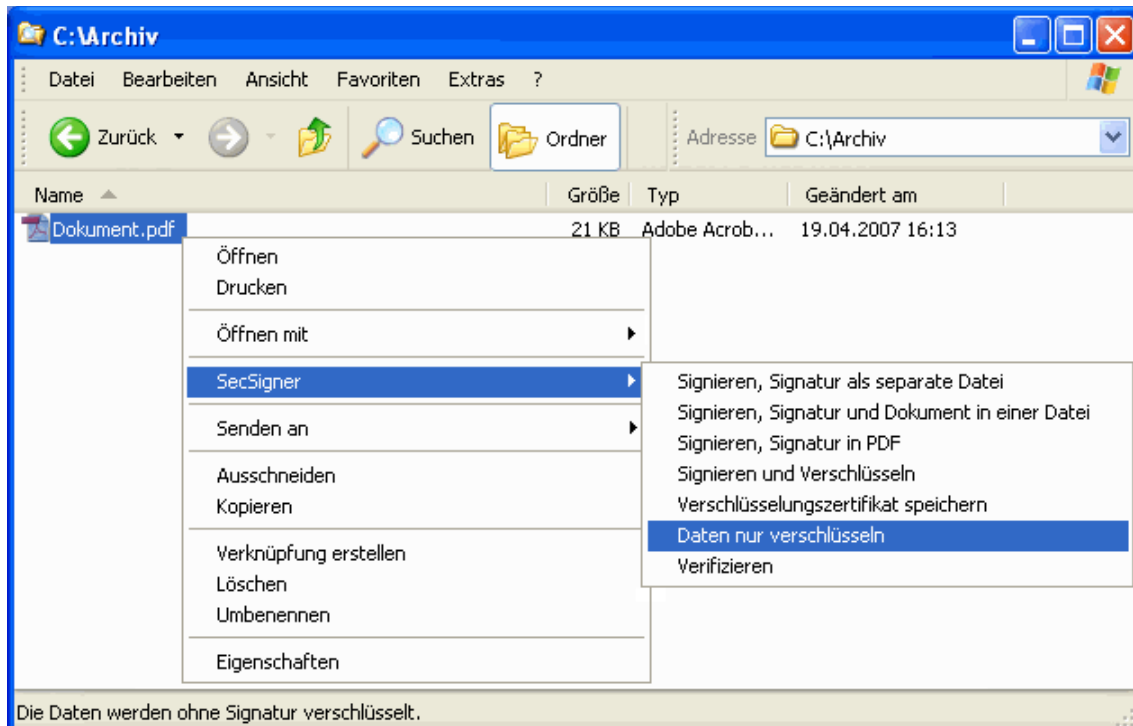


Zum weiteren Ablauf der Signaturerzeugung siehe Abschnitt 7.4.

8.5.3 Daten nur Verschlüsseln

Wollen Sie ein Dokument verschlüsseln ohne es zu signieren, so benötigen Sie das Verschlüsselungszertifikat des Empfängers. (Siehe 8.5.4)

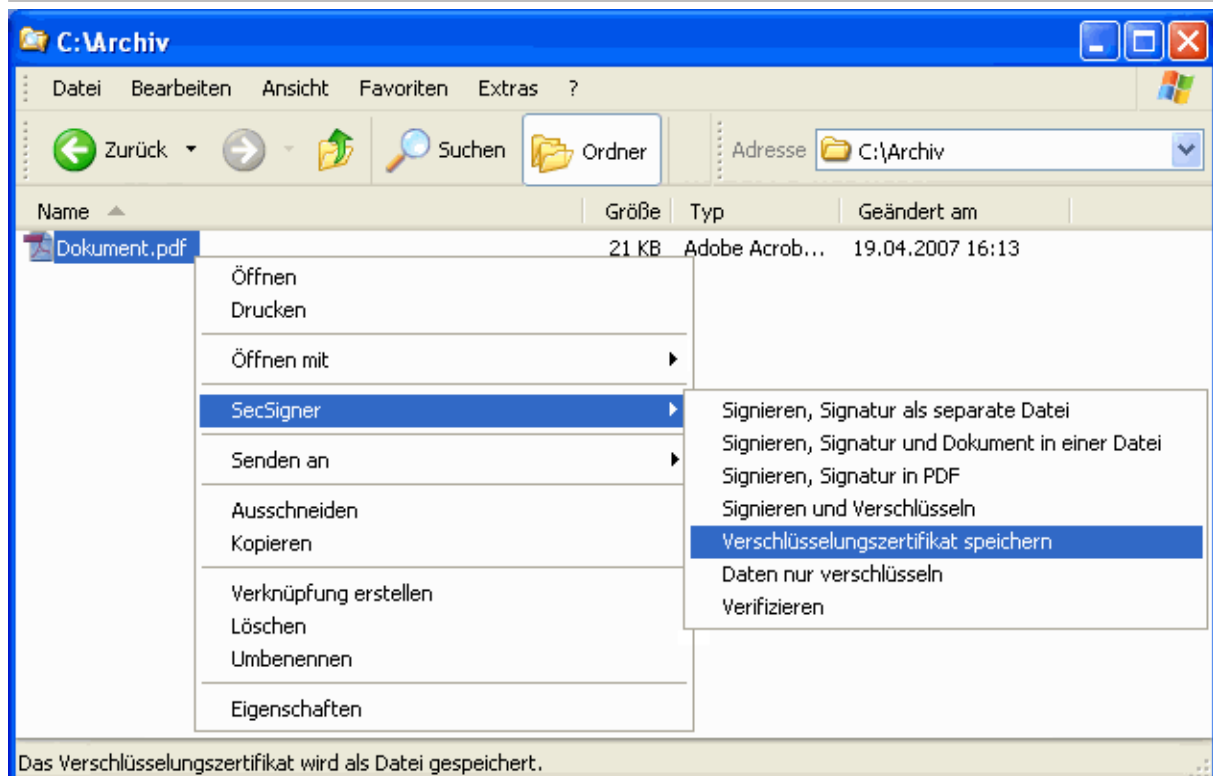
Der Ablauf der Signatur mit Verschlüsselung gleicht dem oben beschriebenen: Beim Klick mit der rechten Maustaste auf ein Dokument wählen Sie dazu die Option „SecSigner“, „Daten nur Verschlüsseln“:



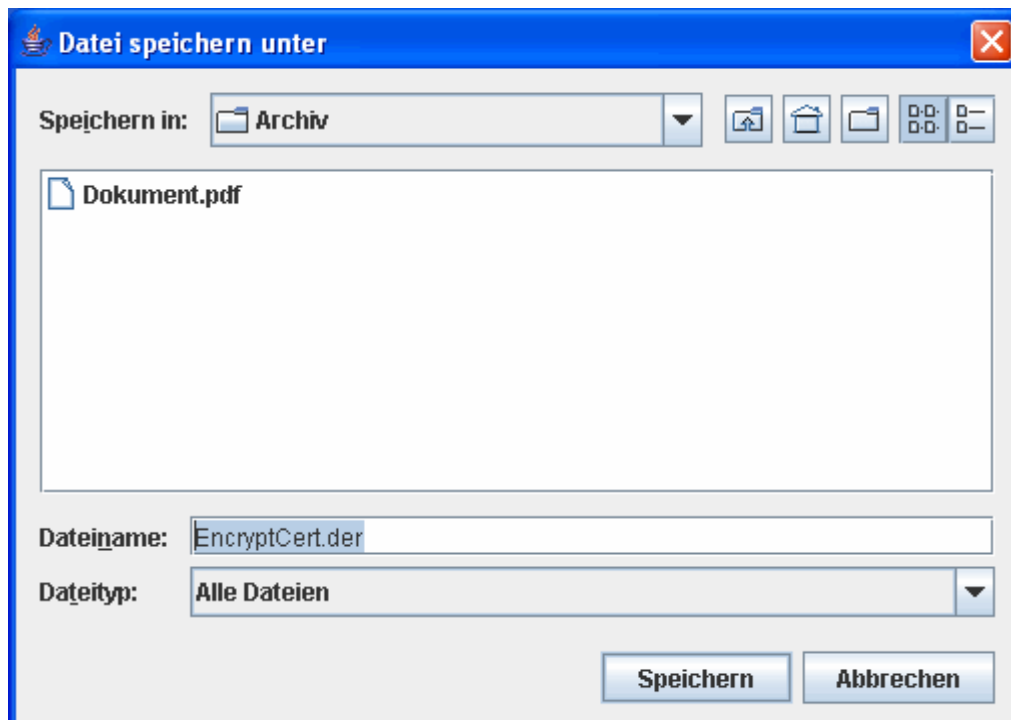
Der Verschlüsselungsvorgang ist analog zum Verschlüsselungsvorgang der Signatur. (Siehe Abschnitt 7.4).

8.5.4 Ablage des Verschlüsselungszertifikats als Datei

Wenn Sie Ihr Verschlüsselungszertifikat zur Weitergabe als Datei ablegen möchten, zeigen Sie auf ein beliebiges Dokument und wählen Sie mit der rechten Maustaste die Option „SecSigner“, „Verschlüsselungszertifikat speichern“:

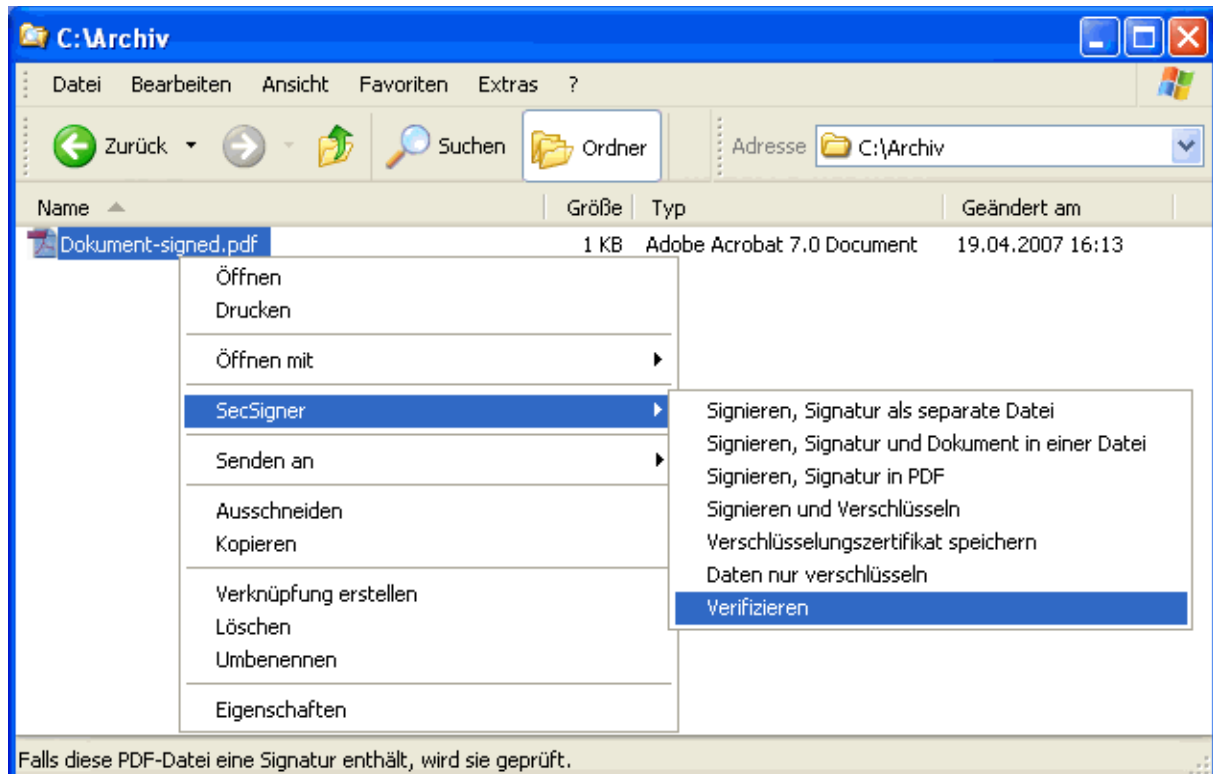


Nach der Suche Ihrer Signaturkarte haben Sie im Fileauswahldialog die Möglichkeit, Ihr Verschlüsselungszertifikat im Filesystem zu speichern:



8.5.5 Entschlüsselung

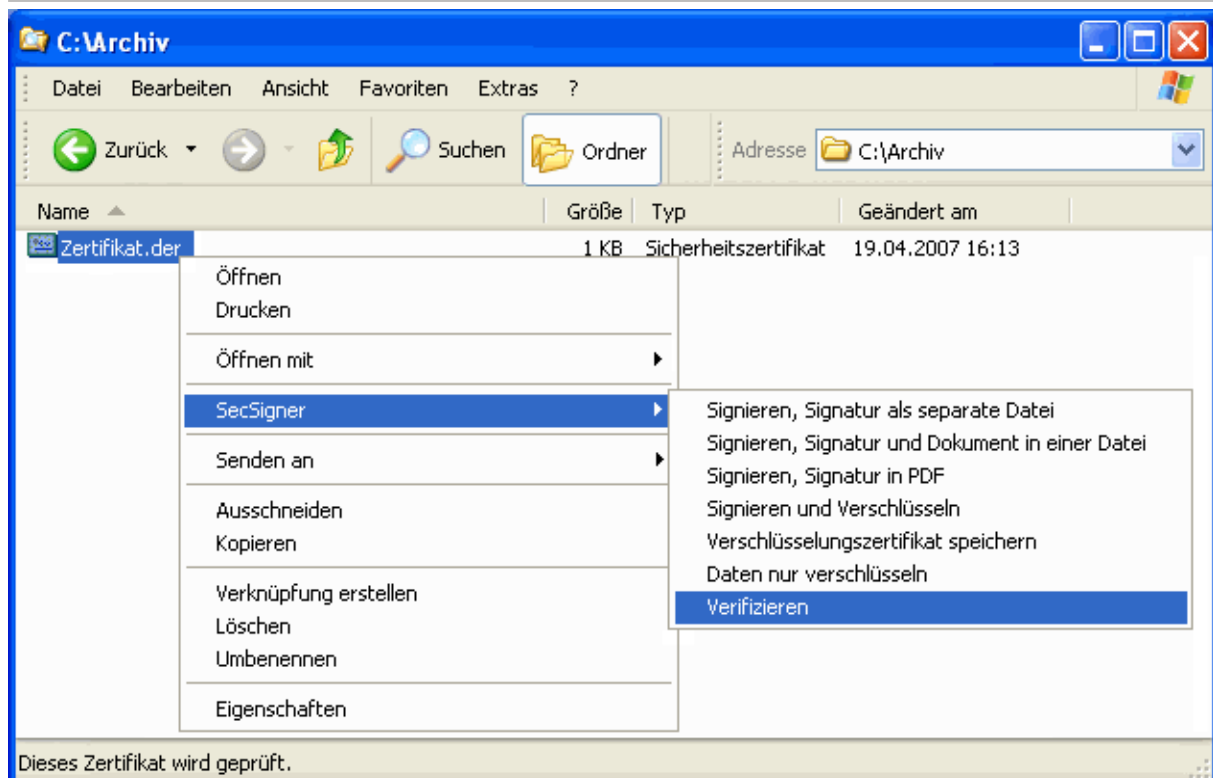
Erhalten Sie ein (Signatur-) Dokument, welches mit Ihrem Verschlüsselungszertifikat verschlüsselt worden ist, so wird dieses beim Öffnen im SecSigner entschlüsselt.



Dazu wählen Sie nach der Initialisierung der Signaturkarte den Menüpunkt “Daten entschlüsseln” (vgl. Abschnitt 7.3).

8.5.6 Zertifikatprüfung

Sie können mit dem SecSigner DER-codierte Zertifikate prüfen. Zeigen Sie dazu auf eine Zertifikat-Datei und wählen Sie mit der rechten Maustaste die Option „SecSigner“, „Verifizieren“:

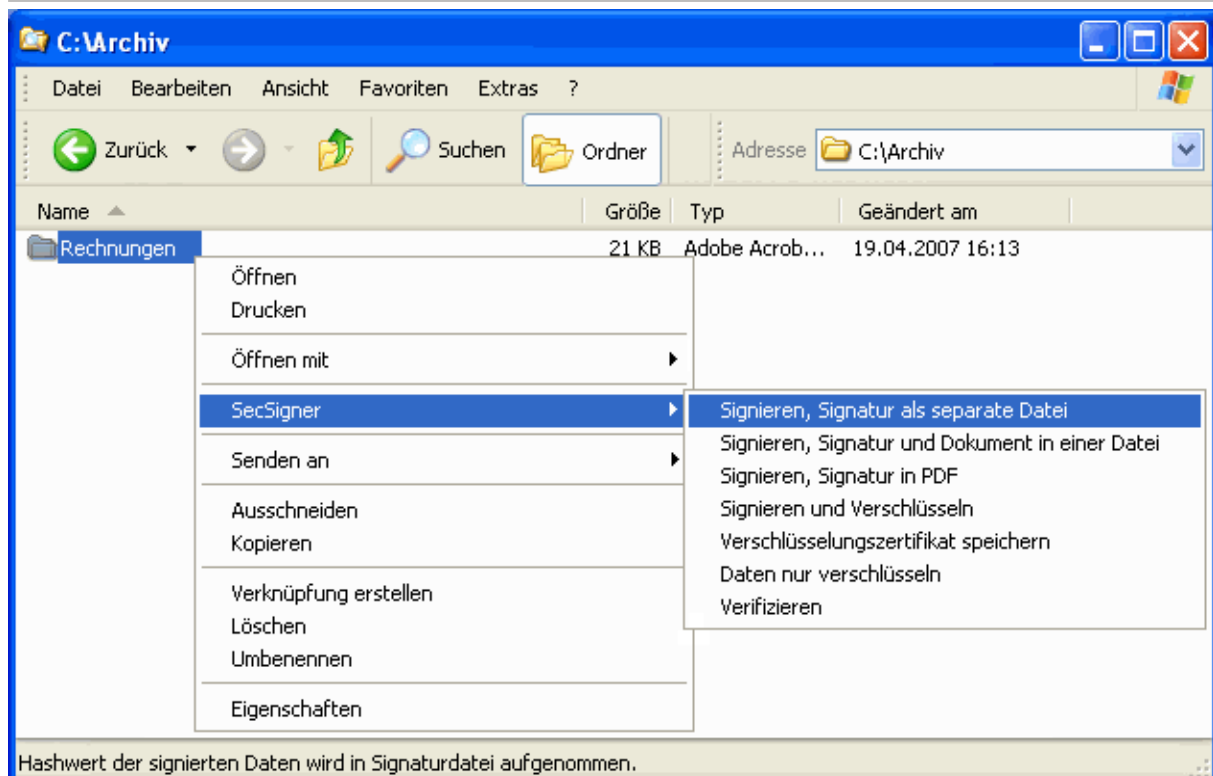


8.5.7 PDF-Signaturen prüfen

Um eine PDF-Signatur (PAdES) im Adobe Signaturformat zu prüfen, zeigen Sie auf ein PDF-Dokument und wählen Sie mit der rechten Maustaste die Option „SecSigner“, „Verifizieren“:

8.5.8 Signatur mehrerer Dateien mit einer PIN-Eingabe

Der Klick auf ein Verzeichnis mit der rechten Maustaste bietet wie bei einer Datei die Optionen „SecSigner“, „Signieren, Signatur als separate Datei“.



Voraussetzung für die Signatur mehrerer Dokumente mit einer PIN-Eingabe ist die Unterstützung dieser Funktionalität durch die Chipkarte!

Entsprechend dem in Kapitel 6 geschilderten Vorgang können durch diesen Aufruf alle in diesem Verzeichnis befindlichen Dateien signiert werden. Im weiteren Verlauf des Signaturvorgangs können dabei alle Dateien angezeigt werden:



9 SecSigner – Aufruf für Mac OS X, Linux, Windows

Für SecSigner steht auf der Website <https://seccommerce.com/download-secsigner-secsign-id/> auch das 'SecSigner-Applet für Windows, Mac OS X, Linux' zum Download zur Verfügung. Das entsprechende Zip enthält Aufrufmöglichkeiten für Mac OS X, Windows und andere Betriebssysteme:

- callSecSigner.command : Aufruf für Mac OS X
- callSecSigner.sh : Aufruf für Linux
- callSecSigner.bat : Aufruf für Windows
- index.html : Aufruf im Browser für beliebige Betriebssysteme

Mit diesem Aufruf wird jeweils der 'Drag & Drop'-Dialog des SecSigners gestartet:



Abbildung 26: SecSigner: Drag&Drop-Dialog

Der Nutzer kann Signaturen zur Prüfung in das grau hinterlegte Feld im Dialog ziehen. SecSigner ruft dann automatisch die im Abschnitt 5 beschriebene Signaturprüfung auf.

Zur Erstellung von Signaturen kann der Nutzer Dokumente das grau hinterlegte Feld im Dialog ziehen. Nachdem dort mindestens ein Dokument angezeigt wird, kann der Nutzer die in der Auswahl-Box ein Signatur-Format wählen und den Signaturprozess mit 'signieren' starten (vgl. Abbildung 27).

Nicht gewünschte Dokumente können aus dem grau hinterlegten Feld gelöscht werden. Dazu ist der angezeigte Verzeichnispfad mit der Maus zu markieren und die 'Entfernen'-Taste zu betätigen.

Die Voreinstellung für das Signaturformat ist 'PKCS7 (CAdES)'. Wenn Sie eine andere Voreinstellung verwenden möchten, können Sie die entsprechende Option in der SecSigner-Konfigurationsdatei anpassen (vgl. Abschnitt 2.5.26).



Abbildung 27: SecSigner: Signaturerzeugung im Drag&Drop-Dialog

10 Impressum

Hersteller des SecSigners und verantwortlich für diese Dokumentation:

SecCommerce Informationssysteme GmbH
Obenhauptstraße 5
22335 Hamburg
Bundesrepublik Deutschland

E-mail: info@seccommerce.com

Web: <https://www.seccommerce.com/>