



Technische Dokumentation (TDoc)

Einheitliche Elster-Datenschnittstelle: Spezifikation ELSTER-Token

Bezeichnung	<i>Verfahren: ELSTER</i> <i>Projekt: Elster-Datensatzbeschreibung</i> <i>Auftragnehmer: BayLfSt - Dst. München -</i>	
Verfahrensmanager	Roland Krebs	
Projektmanager	Christine Randlkofer	
Erstellt am	Donnerstag, 16. Juli 2009	
Zuletzt geändert	11.07.2017 15:27	
Bearbeitungszustand	<input type="checkbox"/> in Bearbeitung <input type="checkbox"/> vorgelegt <input checked="" type="checkbox"/> fertig gestellt	
Dokumentablage	<i>Bitte einfügen!</i>	



Änderungsnachweis

Änderung			Geänderte Kapitel	Beschreibung der Änderung	Autor	Zustand
Nr.	Datum	Version				
1	19.05.2005	00.10	Alle	Erster Entwurf	M.S.	In Bearbeitung
2	15.06.2005	00.50	Alle	Ausarbeitung	M.S.	In Bearbeitung
3	20.06.2005	00.90	Alle	Überarbeitung	C.S.	In Bearbeitung
4	21.06.2005	01.00	Alle	Freigabe	C.S.	Fertiggestellt
5	23.04.2007	01.10	Alle	Details aktualisiert	M.R.	Fertiggestellt
6	07.01.2009	01.20		Anpassungen für <ul style="list-style-type: none">• neuen Sicherheitsstick• neue Schlüssellänge• neue Hash Algorithmen	M.S.	Fertiggestellt
7	20.03.2009	01.21	Alle	Übernahme in neues Format	Franz Widholm	Fertiggestellt
8	16.07.2009	01.22	Alle	Verweise auf Referenzliste und Absatzformatierung	Wolfgang Christ	Fertiggestellt
9	24.07.2009	01.23	Alle	Kleine Anpassungen bzgl. Schlüssellängen und CN	Mathias Rühl	Fertiggestellt
10	08.03.2010	01.24	Alle	Kleine Anpassungen bzgl. Schlüssellängen, CN und XML-Signatur, sowie zum Schlüsselwechsel im Softtoken (Kap. Fehler! Verweisquelle konnte nicht gefunden werden.)	Mathias Rühl	Fertiggestellt
11	06.10.2010	01.25	Alle	Kleine Anpassungen bzgl. Schlüsselwechsel Softtoken.	Stephan Wehr	Fertiggestellt
12	12.09.2013	01.26	6.3	Erwähnung des neuen Sicherheitssticks	Matthias Wurm	Fertiggestellt
13	11.07.2017	01.27	5.2, 6.2, 6.3	Ergänzungen für den neuen Sicherheitsstick „G&D StarSign CUT S“	Wolfgang Christ	Fertiggestellt

1 Referenzliste

1	PKCS12: Personal Information Exchange Syntax, V1.0, June 24, 1999, RSA Laboratories
2	PKCS #12 v1.0 Technical Corrigendum 1, V1.0, February 25, 2000, RSA Laboratories
3	PKCS #11: Cryptographic Token Interface Standard, v2.20, RSA Laboratories June 28, 2004
4	EBA - Elster Datenschnittstelle, Bay.Landesamt für Steuern, München
5	ElsterOnline-Portal: Beschreibung Sicherheitseigenschaften, V 1.1.6 / 10.10.2006
6	Spezifikation : Applet für SCE6/7, v0.5 vom 08.06.2017



2 Inhaltsverzeichnis

1	Referenzliste.....	3
2	Inhaltsverzeichnis	4
3	Abbildungen und Tabellenverzeichnis.....	4
4	Einleitung	5
5	Übersicht	6
5.1	Architektur	6
5.2	Der ElsterOnline-Client.....	7
6	Spezifikation ELSTER Schlüsselspeicher	9
6.1	Datencontainer	9
6.1.1	Private Schlüssel	9
6.1.2	Zertifikate	9
6.2	Softtoken	10
6.2.1	Definition der PKCS#12 Parameter	11
6.2.2	Schlüssellängen, Schlüsseltypen und Hashalgorithmen	12
6.2.3	Sicherheitsdiskussion	12
6.2.4	Alternative Verwendung des Softtoken.....	13
6.3	Sicherheitsstick	13
6.3.1	Verwendete Hardware	14
6.3.2	Definition der PKCS#11 Parameter	14
6.3.3	Sicherheitsdiskussion	16
7	Schnittstellenbeschreibung	17
7.1	Verwaltungsfunktionen	17
7.1.1	PIN Änderung	17
7.1.2	Schlüsselerzeugung und Erzeugung der Zertifikatsanfrage im PKCS#10 Format.....	17
7.1.3	Zertifikatsimport	18
7.1.4	Zertifikatsaustausch	18
7.1.5	Schlüsseltausch	18
7.2	Operative Funktionen	19
7.2.1	Signaturerstellung zwecks Authentisierung	19
7.2.2	Entschlüsselung.....	19
8	Glossar	21
9	Anhang:	27
9.1	Beispiel eines PKCS12 Softtoken (ASN1 dump):.....	27
9.2	Beispielcode zur Generierung des PKCS12 Keystore	31
9.3	Beispielcode zur PIN Änderung des PKCS12 Keystore	32
9.4	Beispielcode zur PIN Änderung des PKCS11 Keystore	33
9.5	Beispielcode zur Signatur mit PKCS12 Keystore.....	34
9.6	Beispielcode zur Signatur mit PKCS11 Keystore.....	35

3 Abbildungen und Tabellenverzeichnis

Abbildung 1: Systemüberblick	6
Tabelle 2: Systemüberblick ElsterOnline-Client	8

4 Einleitung

Die Finanzverwaltung hat zu Beginn des Jahres 2006 das ElsterOnline-Portal in Betrieb genommen. ElsterOnline bietet den Anwendern wie zum Beispiel Bürgern, Unternehmen und Selbständigen eine Vielzahl von Online-Dienstleistungen der Steuer- oder Finanzverwaltung. Diese Dienstleistungen werden durch mehrere Komponenten auf Anwender- und Finanzverwaltungsseite realisiert. Anwender übermitteln über diese Komponenten sensible Daten wie z. B. Steueranmeldungen und Lohnsteuerbescheinigungen. Weiter besteht die Möglichkeit der Einsichtnahme in Steuerkonten. Alle Daten unterliegen dabei u. a. dem Steuergeheimnis und sind vor unberechtigter Einsichtnahme zu schützen. Die für ElsterOnline realisierte Sicherheitsinfrastruktur gewährleistet sichere Authentisierung, Verschlüsselung, Authentizität und Integrität in der Datenverarbeitung über alle Prozesse hinweg.

Technologisch gesehen werden alle Dienste dem externen Anwender über einen Portal-Server via Internet bereitgestellt. Mittels eines Clients kann der Anwender auf die Dienste des ElsterOnline-Portals zugreifen. Zudem bietet der Client Zugriffsmöglichkeiten auf Authentisierungs-, Signaturerzeugungs- und -Verschlüsselungskomponenten, die durch ELSTER in einer Sicherheitskomponente des Clients gekapselt sind. Als Client selbst ist ein Internet-Browser verwendbar. Dieser muss ein Java-Applet ausführen können, welches automatisch vom Portal-Server in den Browser geladen wird.

Alternativ ist als Client auch eine Software bzw. Systemumgebung verwendbar, welche die von der Finanzverwaltung bereit gestellte ELSTER-Funktionsbibliothek oder die entsprechenden Funktionen selbst einbindet. Hersteller, die aus verschiedenen Gründen aber auf die Integration der ELSTER-Funktionsbibliothek verzichten und stattdessen selbst die Funktionalität dieser Bibliothek nachbilden wollen, können dies tun. Das vorliegende Dokument beschreibt die verwendete Schnittstelle des ElsterOnline-Clients und stellt somit die Grundlage für die Nachbildung der Funktionen.

5 Übersicht

5.1 Architektur

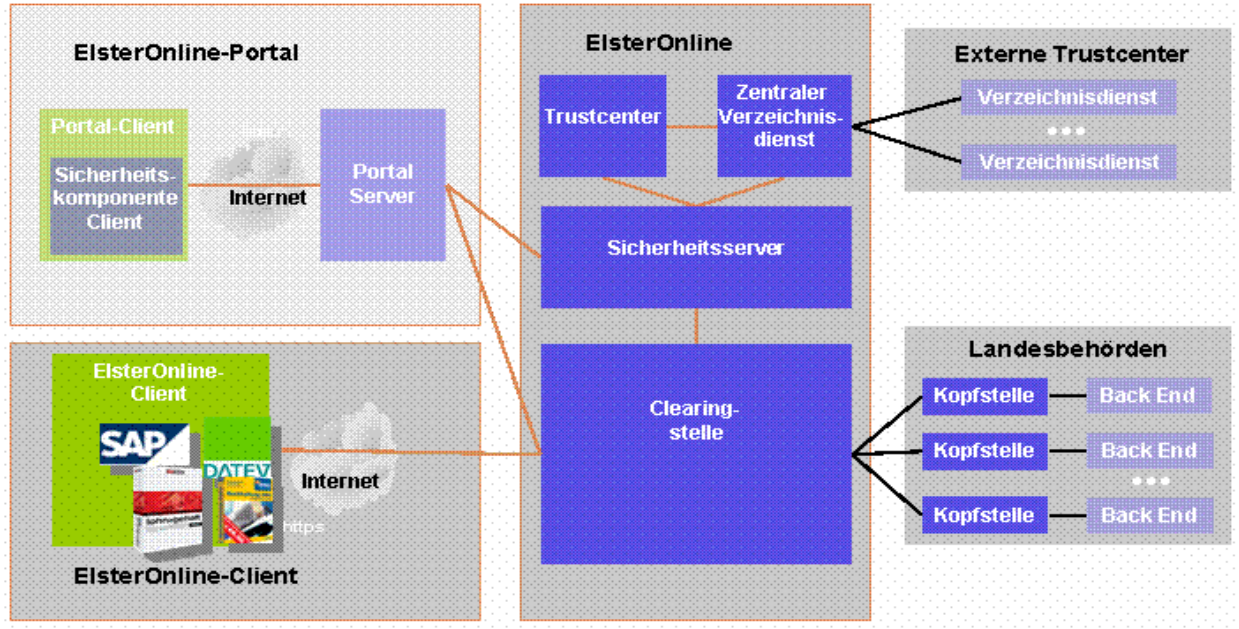


Abbildung 1: Systemüberblick

Abbildung 1 zeigt die beiden Zugänge zur ElsterOnline-Infrastruktur über das ElsterOnline-Portal und den ElsterOnline-Client. Sämtliche Authentisierungs- und Signaturprüfungsanfragen werden vom Portal-Server und einer Clearingstelle an einen Sicherheitsserver weitergeleitet. Der Verschlüsselungsdienst, welcher ebenfalls vom Sicherheitsserver bereitgestellt wird, wird von der Clearingstelle zur Verschlüsselung von Nachrichten an den Anwender genutzt. Die Prozesse der Authentisierung, Signatur und Verschlüsselung basieren auf persönlichen, asymmetrischen Schlüsselpaaren und Zertifikaten der Anwender. Das ELSTER-Trustcenter dient der Erzeugung von Zertifikaten für die Sicherheitskomponente auf Seite des Anwenders sowie der Veröffentlichung der Zertifikate und entsprechenden Sperrlisten in einem Zentralen Verzeichnisdienst.

Die Clearingstellen sind Ein- und Ausgangsrechner, welche die Übertragungsschnittstellen von der Anwendungsseite zu den Hintergrundrechnern der Länder darstellen. Es gibt insgesamt 2 Clearingstellen, eine in Bayern und eine in Nordrhein-Westfalen. Welche Clearingstelle jeweils von einem Programm angesprochen wird, wird mittels eines Algorithmus zur Lastverteilung ermittelt.



5.2 Der ElsterOnline-Client

Eine ElsterOnline-Client-Implementierung gibt dem Anwender die Möglichkeit, Steuerklärungen und Steuermeldungen abzugeben oder Einsicht in Steuerkontodaten zu nehmen. Um die Sicherheit des Verfahrens zu gewährleisten, sind Dokumente und Anfragen mit Signaturen zu versehen und bei Bedarf zu verschlüsseln. Für die Durchführung dieser Operationen wird möglichst spezielle kryptografische Hardware (Signaturkarte, Sicherheitsstick) eingesetzt. Wer nicht über die entsprechende Hardware verfügt, kann bestimmte Verfahren auch mit einem Software-Schlüssel (Soft-PSE, d.h. gesicherter Schlüsselspeicher) nutzen. Die Daten werden vom Client elektronisch an die Finanzbehörde übermittelt.

Ein ElsterOnline-Client sollte aus folgenden Komponenten bestehen:

- Benutzerführende Schicht (Web-Applikation, Anwendersoftware, ERP-Software oder ähnliches)
- Zugriffe auf den Software-Schlüssel (Soft-PSE, Softtoken) oder Hardware-PSE (Sicherheitsstick oder Signaturkarte), in einer Anwendungsschnittstelle zusammengefasst.
- Kommunikation mit dem ELSTER Hintergrundsystem (ElsterOnline-Portal oder Clearingstelle)
- Sonderfunktionen, z.B. Nutzung eines Zeitstempeldienstes

Gegenstand dieses Dokuments ist die Beschreibung der Zugriffe auf die Benutzer-PSE.

Für die elektronische Kommunikation mit der Finanzbehörde stehen 2 Schnittstellen zur Verfügung:

- Clearingstelle: über diese Schnittstelle ist nur Dateneinreichung möglich (signierte XML Dokumente gemäß EBA /4/)
- ElsterOnline-Portal: diese Schnittstelle bietet
- den vollständigen Registrierprozess, d.h. die Bindung eines Authentisierungsmediums (Software-Schlüssel, Sicherheitsstick oder Signaturkarte) zu einem Account in ElsterOnline. Dies beinhaltet die Schlüssel- und Zertifikatserzeugung für Software-Schlüssel und Sicherheitsstick und die Registrierung eines Zertifikats im Falle der Signaturkarte
- Aktualisierung des Authentisierungsmediums
- Wechsel des Authentisierungsmediums
- Sperrung des Accounts
- Dateneinreichung (Steuererklärung, Steuermeldung)
- Steuerkontoabfrage

Eine individuelle ElsterOnline-Client-Implementierung wird aktuell nur die Schnittstelle „Clearingstelle“ nutzen können, da die Spezifikation für das ElsterOnline-Portal nicht öffentlich ist. Daher muss der Registrierprozess, also die Erzeugung eines Accounts in ElsterOnline in jedem Fall über die Schnittstelle ElsterOnline-Portal erfolgen. Eine Beschreibung des Registrierprozess ist in /5/ zu finden.

Eine ElsterOnline-Client-Implementierung übernimmt die Kommunikation mit den integrierten Hard- oder Softtoken, um Signaturen oder Entschlüsselungen vorzunehmen. Alle Zugriffe werden ausschließlich über standardisierte Schnittstellen vorgenommen. Auf Hardwaretoken wird über eine in der Regel vom Hersteller des Tokens zur Verfügung gestellte PKCS#11 Library oder, ab 2017 auf die neuen Sicherheitssticks vom

Typ „G&D StarSign CUT S“, direkt per PC/SC zugegriffen, Softtoken werden gemäß der Spezifikation PKCS#12 als sichere Container zur Ablage geheimer Informationen erzeugt und verwaltet.

Für den Sicherheitsstick bzw. die Elster Soft-PSE werden als Managementfunktionen nur die PIN-Änderung bereitgestellt. Alle andern Management Funktionen (Schlüsselgenerierung, Zertifizierung durch Elster-CA) sind im Rahmen des Registrierprozesses nur über das ElsterOnline-Portal möglich.

Eine ElsterOnline-Client-Implementierung kann sowohl als Java- als auch als C/C++ Anwendung implementiert werden.

Im Folgenden ist schematisch das Umfeld skizziert:

ElsterOnline-Client-Implementierung						
PKCS#12		PC/SC	PKCS#11			
Elster Soft-PSE		Sicherheitsstick		externe Karten		
Authentisierungsschlüssel	Verschlüsselungsschlüssel	Authentisierungsschlüssel	Verschlüsselungsschlüssel	Signatur-schlüssel	Verschlüsselungsschlüssel	weitere Schlüssel

Tabelle 2: Systemüberblick ElsterOnline-Client

Die verwendeten Standards sind bewusst Plattform-unabhängig gewählt, um in der Softwarearchitektur nicht an Betriebssystem-abhängige Besonderheiten gebunden zu sein.

Im Folgenden wird für die Kennzeichnung von Schlüsseln und Zertifikaten eine vereinfachte Backus-Naur-Notation verwendet.



6 Spezifikation ELSTER Schlüsselspeicher

6.1 Datencontainer

6.1.1 Private Schlüssel

Es existieren zwei private RSA Schlüssel 2048 Bit (bzw. 1024 Bit bei älteren Schlüsselspeichern), die in geschützten Bereichen des Datencontainers zu speichern sind. Folgende Namen werden in dieser Spezifikation benutzt:

Für Authentisierung (digitale Signatur):

- PK.USR.AUT / SK.USR.AUT

Für Entschlüsselung

- PK.USR.ENC / SK.USR.ENC

Die Schlüssel sind RSA Schlüssel mit einer Schlüssellänge von 2048 Bit.

6.1.2 Zertifikate

Zertifikate beinhalten den öffentlichen Schlüssel, verknüpft mit bestimmten personalisierten Daten. Das Format der Zertifikate entspricht X509.

6.1.2.1 Benutzer-Zertifikate

Es existieren

- C.USR.AUT Authentisierungszertifikat enthält den public key „PK.USR.AUT“
- C.USR.ENC Verschlüsselungszertifikat enthält den public key „PK.USR.ENC“

6.1.2.2 CA Zertifikate

C.CA.DS enthält den public key PK.CA.DS

Das/Die CA-Zertifikat(e) wird/werden für die Signaturprüfung der Benutzer-Zertifikate benötigt.

6.1.2.3 Wurzel CA Zertifikate

C.RCA.DS enthält den public key PK.RCA.DS

Das/Die CA-Zertifikat(e) wird/werden für die Signaturprüfung des/der CA-Zertifikat(e) benötigt.

6.2 Softtoken

Der Softtoken wird als PKCS#12 Datencontainer implementiert.

Dies bietet folgende Vorteile:

- Nur eine Datei zur Ablage von Schlüsseln und Zertifikaten, daher leicht portierbar und einfaches Backup
- PKCS#12 ist ein programmiersprachenübergreifender Standard
- Betriebssystemunabhängigkeit

Nachteile:

- Privater Schlüssel verlässt den Softtoken zur Verwendung in der Software für Signatur- oder Entschlüsselungsfunktionen.
- Benutzer kann die Datei beliebig oft kopieren
- Passwort kann mit Brut-Force-Attacken (Ausprobieren) angegriffen werden
- Benutzer kann ein einfaches Passwort wählen oder auf das Passwort verzichten



6.2.1 Definition der PKCS#12 Parameter

Die Verschlüsselung der geheimen Teile des Containers (private RSA Schlüssel) erfolgt mit pbeWithSHAAnd3-KeyTripleDES-CBC (OID: 1.2.840.113549.1.12.1.3), die Verschlüsselung der öffentlichen Teile des Containers (Zertifikate) erfolgt mit pbeWithSHAAnd40BitRC2-CBC (OID: 1.2.840.113549.1.12.1.6) Bzgl. der Schlüsselsicherheit wurde festgelegt, dass die Schlüsselgenerierung aus dem Passwort mit 1024 Iterationen erfolgt. Es wird nur ein Passwort für alle Schlüssel verwendet.

6.2.1.1 Die enthaltenen SafeBags und das Attribut "friendlyName"

Das Attribut friendlyName (OID: 1.2.840.113549.1.9.20) wird in allen SafeBags (siehe /1/) verwendet:

- Authentisierung
 - KeyBag für den Authentisierungsschlüssel: „SignatureKey“
 - CertBag für das Authentisierungszertifikat: Der "distinguished name" (DN) des Authentisierungszertifikats
(Beispiel 1: CN=tester\,karl-heinz,2.5.4.5=#130b313030303030303030333843 (veraltet)
Beispiel 2: CN=1000123321,SERIALNUMBER=1000123321A (aktuell, oder nach Zertifikatsupdate))
 - CertBag für das Ausstellerzertifikat: Der "distinguished name" (DN) des Ausstellerzertifikats
(Beispiel: CN=ElsterSoftCA,OU=CA,O=Elster,C=DE)
 - CertBag für das Wurzelzertifikat: Der "distinguished name" (DN) des Wurzelzertifikats
(Beispiel: CN=ElsterRootCA,OU=RootCA,O=Elster,C=DE)
- Verschlüsselung
 - KeyBag für den Verschlüsselungsschlüssel: „EncryptionKey“
 - CertBag for encryption certificate: Der "distinguished name" (DN) des Verschlüsselungszertifikats
(Beispiel: CN=tester\,karl-heinz,2.5.4.5=#130b313030303030303030333843 (veraltet)
Beispiel 2: CN=1000123322,SERIALNUMBER=1000123322C (aktuell, oder nach Zertifikatsupdate))
 - CertBag for issuer certificate: Der "distinguished name" (DN) des Ausstellerzertifikats
(Beispiel: CN=ElsterSoftCA,OU=CA,O=Elster,C=DE)
 - CertBag for signature root certificate: Der "distinguished name" (DN) des Wurzelzertifikats
(Beispiel: CN=ElsterRootCA,OU=RootCA,O=Elster,C=DE)

Der friendlyName wird genutzt, um den passenden KeyBag für Verschlüsselung bzw. Authentisierung auszuwählen. Der friendlyName wird durch die Anwendung ELSTER Webclient vergeben (Code Bestandteil). Die Auswahl des passenden CertBag, der das Zertifikat zu diesem Schlüssel enthält erfolgt über die localKeyID.



6.2.1.2 Attribut "localKeyID"

Das Attribut localKeyID (for PKCS #12) (1 2 840 113549 1 9 21) wird in allen KeyBags und in den CertBags für die Zertifikate C.USR.AUT und C.USR.ENC (siehe /1/) verwendet. :

- Authentisierung
 - KeyBag für den Authentisierungsschlüssel: „Time “ + Long wert aus Systemzeit berechnet (Bsp.: „Time 1116412528780“), muss eindeutig für alle KeyBags sein
 - CertBag für das Authentisierungszertifikat: „Time “ + Long wert aus Systemzeit berechnet (Bsp.: „Time 1116412528780“ – muss gleich dem Wert des KeyBag sein)
- Verschlüsselung
 - KeyBag für den Verschlüsselungsschlüssel: „Time “ + Long wert aus Systemzeit berechnet, muss eindeutig für alle KeyBags sein (Bsp.: „Time 1116412528784“)
 - CertBag for encryption certificate: „Time “ + Long wert aus Systemzeit berechnet (Bsp.: „Time 1116412528784“ – muss gleich dem Wert des KeyBag sein)

Die localKeyID wird genutzt, um die Verbindung KeyBag und dazu passenden CertBag herzustellen.

6.2.2 Schlüssellängen, Schlüsseltypen und Hashalgorithmen

Die von ELSTER erzeugten Softtoken enthalten aktuell nur RSA Schlüssel mit Schlüssellängen von 2048 bit.

Der in den gespeicherten Zertifikaten verwendete Signaturalgorithmus ist RSASSA-PSS nach PKCS#1v2.1 mit Parametern Hash-Algorithmus SHA-256 und Mask Generation Function MGF1 ebenfalls mit SHA-256.

Die ElsterClient-Software sollte deshalb mit diesem RSA Verfahren bei dieser Schlüssellänge arbeiten können.

6.2.3 Sicherheitsdiskussion

Bei der Verwendung eines Softtokens sollte aus Sicherheitsgründen der Token nicht geöffnet gehalten werden. Passwörter und extrahierte Schlüssel werden, sobald sie nicht mehr aktuell benötigt werden, in der Software überschrieben. Für diese Funktionalität ist eine Methode „release“ vorzusehen. Der Benutzer muss bei jeder Schlüsselverwendung sein Passwort eingeben (Variante 1).

Sollte diese Benutzerführung nicht gewünscht sein, ist das Passwort oder sind die Schlüssel im Hauptspeicher zu halten, so dass nur bei Programm- oder Applet-Neustart eine wiederholte Passwordeingabe notwendig ist (Variante 2).

Die Risiken sind abzuwägen:

- Variante1:
 - Häufige Passwordeingabe, erleichtert z.B. über Tastaturtracing eine Passwort-Kompromittierung
- Variante2:
 - Passwort oder Schlüssel im Hauptspeicher sind theoretisch von anderen Programmen lesbar.



6.2.4 Alternative Verwendung des Softtoken

Es ist möglich den Softtoken nur als Transportmedium zwischen ElsterOnline-Portal und der ElsterOnline-Client-Implementierung zu verwenden.

Folgende Schritte sind durchzuführen:

- Komplette Registrierung am ElsterOnline-Portal bis zum ersten Login
- Transport der P12 Datei (Soft-PSE)
- Start einer Importroutine innerhalb der ElsterOnline-Client-Implementierung
- Sicherung der P12 Datei (inklusive Passwort) für späteren Zertifikatsupdate

In diesem Fall kann die ElsterOnline-Client-Implementierung eine eigene proprietäre PSE nutzen. Die Importroutine öffnet die Elster Soft-PSE, extrahiert die Schlüssel und Zertifikate und speichert diese Daten dann in der proprietären PSE. Die Schlüsselsicherheit sollte aber mindestens dem PKCS#12 Standard mit den hier beschriebenen Parametern entsprechen.

6.3 Sicherheitsstick

Der Sicherheitsstick ist eine Smartcard basierte All-in-one Lösung bei der Sicherheitschip und ein Kartenleser eine Einheit bilden.

Dies bietet folgende Vorteile:

- Privater Schlüssel verlässt nie den Sicherheitschip, Kryptooperationen für Signatur- oder Entschlüsselungsfunktionen werden auf dem Sicherheitschip durchgeführt.
- Keine Duplikate von Schlüsseln möglich
- Token erfordert zwingend eine bestimmte Passwortlänge
- Token wird nach mehreren Passwort Fehleingaben aus Sicherheitsgründen gesperrt

Nachteile:

- Kein Backup möglich, bei Verlust können alte Daten nicht mehr entschlüsselt werden.
- Einschränkung auf Betriebssysteme, die USB unterstützen
- Erfordert Treiber Installation (gilt nicht für den neuen CUT-S-Sicherheitsstick)
- Einschränkung auf Betriebssysteme für die eine PKCS#11 bzw. PC/SC (für den neuen CUT-S-Sicherheitsstick) Schnittstelle verfügbar ist.

6.3.1 Verwendete Hardware

6.3.1.1 G&D StarSign USB Token für ELSTER

Der Sicherheitsstick mit 2048 bit RSA Kryptographie . Der „G&D StarSign USB Token für ELSTER“ ist als ein StarSign Token der Firma Giesecke&Devrient und besteht aus einem USB Kartenleser mit eingesetztem Chip mit dem Betriebssystem SmartCafé 3.1. Leser und Chip sind eine kompakte Einheit. Der „G&D StarSign USB Token für ELSTER“ ist seit Okt. 2008 im Einsatz.

6.3.1.2 G&D StarSign Crypto USB Token für ELSTER

Der neue Sicherheitsstick mit 2048 bit RSA Kryptographie . Der „G&D StarSign Crypto USB Token für ELSTER“ der Firma Giesecke&Devrient und besteht aus einem USB Kartenleser mit eingesetztem Chip mit dem Betriebssystem SmartCafé 6.0 und AET-Middleware. Leser und Chip sind eine kompakte Einheit. Der „G&D StarSign Crypto USB Token für ELSTER“ ist seit Juni 2013 im Einsatz.

6.3.1.3 G&D StarSign CUT S Token für ELSTER

Der neueste Sicherheitsstick mit 2048 bit RSA Kryptographie . Der „G&D StarSign CUT S Token für ELSTER“ der Firma Giesecke&Devrient und besteht aus einem USB Kartenleser mit eingesetztem Chip mit dem Betriebssystem SmartCafé 7.0 und eigenem Applet der Firma secunet. Leser und Chip sind eine kompakte Einheit. Der „G&D StarSign CUT S Token für ELSTER“ wird ab September 2017 verfügbar sein. Spezifikation des Java-Applets ist beschrieben in siehe Kap.1 Referenzliste 6.

6.3.2 Definition der PKCS#11 Parameter

Die Auflistung erfolgt entsprechend den Objekten die sich auf dem Token befinden. Es werden die verwendeten PKCS#11 Attribute aufgelistet. Die Attribute “CKA_LABEL“ und “CKA_ID“ werden durch die Anwendung ELSTER Webclient vergeben (Code Bestandteil).

6.3.2.1 Privater Authentisierungsschlüssel SK.USR.AUT

- CKA_CLASS: CKO_PRIVATE_KEY
- CKA_TOKEN: CK_TRUE
- CKA_PRIVATE: CK_TRUE
- CKA_LABEL: “ELSTER_SIGN“
- CKA_ID: “ELSTER_SIGN“

6.3.2.2 Öffentlicher Authentisierungsschlüssel PK.USR.AUT

- CKA_CLASS: CKO_PUBLIC_KEY
- CKA_TOKEN: CK_TRUE
- CKA_PRIVATE: CK_FALSE
- CKA_LABEL: “ELSTER_SIGN“
- CKA_ID: “ELSTER_SIGN“

6.3.2.3 Authentisierungszertifikat C.USR.AUT

- CKA_CLASS: CKO_CERTIFICATE
- CKA_TOKEN: CK_TRUE
- CKA_PRIVATE: CK_FALSE

- CKA_LABEL: "ELSTER_SIGN"
- CKA_ID: "ELSTER_SIGN"

6.3.2.4 Privater Verschlüsselungsschlüssel SK.USR.ENC

- CKA_CLASS: CKO_PRIVATE_KEY
- CKA_TOKEN: CK_TRUE
- CKA_PRIVATE: CK_TRUE
- CKA_LABEL: "ELSTER_ENCR"
- CKA_ID: "ELSTER_ENCR"

6.3.2.5 Öffentlicher Verschlüsselungsschlüssel PK.USR.ENC

- CKA_CLASS: CKO_PUBLIC_KEY
- CKA_TOKEN: CK_TRUE
- CKA_PRIVATE: CK_FALSE
- CKA_LABEL: "ELSTER_ENCR"
- CKA_ID: "ELSTER_ENCR"

6.3.2.6 Verschlüsselungszertifikat C.USR.ENC

- CKA_CLASS: CKO_CERTIFICATE
- CKA_TOKEN: CK_TRUE
- CKA_PRIVATE: CK_FALSE
- CKA_LABEL: "ELSTER_ENCR"
- CKA_ID: "ELSTER_ENCR"

6.3.2.7 CA Zertifikat C.CA.DS

- CKA_CLASS: CKO_CERTIFICATE
- CKA_TOKEN: CK_TRUE
- CKA_PRIVATE: CK_FALSE
- CKA_LABEL:
 - nicht vergeben/undefiniert (veraltet, ELSTER-Stick)
 - "ISSUER-1" (StarSign USB Token, ELSTER-Stick nach Zertifikatsupdate)
- CKA_ID:
 - nicht vergeben/undefiniert (veraltet, ELSTER-Stick)
 - "ISSUER-1" (StarSign USB Token, ELSTER-Stick nach Zertifikatsupdate)

6.3.2.8 Wurzel CA Zertifikat C.RCA.DS

- CKA_CLASS: CKO_CERTIFICATE
- CKA_TOKEN: CK_TRUE
- CKA_PRIVATE: CK_FALSE
- CKA_LABEL:
 - nicht vergeben/undefiniert (veraltet, ELSTER-Stick)
 - "ISSUER-2" (StarSign USB Token, ELSTER-Stick nach Zertifikatsupdate)
- CKA_ID:
 - nicht vergeben/undefiniert (veraltet, ELSTER-Stick)
 - "ISSUER-2" (StarSign USB Token, ELSTER-Stick nach Zertifikatsupdate)

6.3.3 Sicherheitsdiskussion

Bei der Verwendung eines Sicherheitssticks über die PKCS#11 Schnittstelle wird eine Session zwischen Sicherheitsstick und Anwendung hergestellt. Zum Zugriff auf private Objekte ist ein Login mit der PIN in dieser Session erforderlich. Dieser Session Mechanismus schützt relativ zuverlässig gegen Zugriffe fremder Programme. Deshalb kann die Session geöffnet gehalten werden. Der Benutzer muss nach dem Login sein Passwort nicht immer wieder neu eingeben.

Ein Risiko ist lediglich die Passwortkompromittierung z.B. durch Tastaturtracing. Eine sichere Passwordeingabe über Class2 - oder Class3 - Leser ist bei der Kombination Smartcardchip / USB-Kartenleser (= Sicherheitsstick) nicht möglich.



7 Schnittstellenbeschreibung

7.1 Verwaltungsfunktionen

Alle Verwaltungsfunktionen, außer der PIN Änderung sind aktuell nur über das ElsterOnline-Portal möglich. Die Schnittstelle „Clearingstelle“ bietet diese Funktionalität nicht.

7.1.1 PIN Änderung

7.1.1.1 Softtoken

Eine PIN Änderung des Softtoken ist im Prinzip eine erneute Abspeicherung der ASN1 Struktur in eine Datei (nachdem die SafeBags mit dem neuen Passwort verschlüsselt wurden).

Folgendes ist dabei zu beachten:

- Die in dieser Spezifikation enthaltenen Festlegungen (siehe 6.2.1) dürfen nicht verändert werden.
- Der Dateiname sollte durch PIN-Änderung nicht geändert werden.
- Die Kompatibilität zu den JDK1.4, JDK5.0, JDK6.0 (Implementierung der Klasse SUN PKCS12 KeyStore: com.sun.net.ssl.internal.ssl.PKCS12KeyStore in jsse.jar) muss gewährleistet bleiben, wenn noch mit dem Portal gearbeitet werden soll (z.B. Zertifikatsupdate).

7.1.1.2 Sicherheitsstick

Eine PIN Änderung des Sicherheitsstick ist jederzeit möglich, nachdem die PKCS#11 Session zum Token hergestellt wurde. Es wird die Funktion „C_SetPIN“ genutzt. Die Funktion hat als Parameter den Session handle, die alte PIN und die neue PIN. Die neue PIN muss der Tokenspezifikation entsprechen. Für den Sicherheitsstick bedeutet das mindestens 4, maximal 8 beliebige Zeichen (Ein Zeichen ist ein 8-bit UTF-8 character).

7.1.2 Schlüsselerzeugung und Erzeugung der Zertifikatsanfrage im PKCS#10 Format

Die Schlüsselerzeugung wird angestoßen vom Applet, das über das ElsterOnline-Portal bereitgestellt wird. Beim Softtoken werden die RSA-Schlüsselpaare innerhalb des Applets erzeugt, beim ElsterStick geschieht die Schlüsselgenerierung innerhalb der Hardware des ElsterStick. Zur Information wird der Ablauf hier kurz beschrieben.

Folgende Schritte werden jeweils für den Verschlüsselungs- und Authentisierungsschlüssel ausgeführt:

- Erzeugung von eines RSA Schlüsselpaars mit Schlüssellänge 2048 bit.
- Erzeugung eines PKCS10 Request (zur Beantragung des Benutzer-Zertifikats bei der ELSTER-CA)
- Erzeugung eines selbstsignierten Zertifikats (zur temporären Speicherung im Softtoken/Sicherheitsstick)

Danach wird einmal ausgeführt:

- Speichern der RSA Schlüsselpaare und selbstsignierten Zertifikate in der vorläufigen PKCS12 Datei oder im Sicherheitsstick
- Senden der PKCS10 Requests an das ElsterOnline-Portal (kryptografisch gesichert mit dem Aktivierungscode, der als PIN Brief versendet wurde).



7.1.3 Zertifikatsimport

Die Benutzerzertifikate werden über das ElsterOnline-Portal in der PKCS12 Datei bzw. im Sicherheitsstick installiert. In diesem Schritt werden die temporären Zertifikate in der PSE des Benutzers durch die von der ELSTER-CA neu ausgestellten Benutzerzertifikate ersetzt.

Ebenfalls abgespeichert werden die notwendigen Ausstellerzertifikate. In der Soft-PSE werden bedingt durch die verwendete Java-Schnittstelle die Ausstellerzertifikate doppelt abgespeichert. (Siehe 6.2.1.1)

7.1.4 Zertifikatsaustausch

Die Benutzerzertifikate können über Anmeldung am ElsterOnline-Portal ausgetauscht werden. Wenn ein Zertifikat abläuft, wird automatisch in einer bestimmten Frist vor dem Ablaufdatum ein neues bereitgestellt. Im Rahmen eines Logins wird das abgelaufene Benutzerzertifikat am Portal automatisch gegen das neue Benutzerzertifikat ausgetauscht. Dies gilt für Softtoken wie auch für den Sicherheitsstick. Ein anderer Weg des Zertifikatsaustausches ist aktuell nicht vorgesehen.

Bei Token-Wechsel Signaturkarte => Sicherheitsstick oder Signaturkarte => Softtoken erfolgt keine Übertragung der Schlüssel und Zertifikate von der Signaturkarte auf den Sicherheitsstick bzw. auf die PKCS12 Datei. Stattdessen wird eine normale Registrierung durchgeführt, wobei aber die Registrierungsdaten mit der Signaturkarte signiert werden.

7.1.5 Schlüsseltausch

7.1.5.1 Softtoken

Es existieren Softtoken mit einer Schlüssellänge von 1024 Bit. Diese Schlüssellänge wird nicht mehr für lange Zeit als sicher angesehen. Aus diesem Grund sollen Softtoken auf größere Schlüssellängen von 2048 Bit umgestellt werden. Dabei bleiben die alten 1024-Bit Schlüssel und Zertifikate im Softtoken erhalten, um bei Bedarf zum Entschlüsseln und zum Signieren der neuen 2048-Bit Zertifikatsrequests verwendet werden zu können. Die Alias- bzw. KeyBag-Namen der alten 1204-Bit Schlüssel werden von „SignatureKey“ und „EncryptionKey“ zu „SignatureKey1024“ und „EncryptionKey1024“ geändert. Die neuen 2048 Bit Schlüssel können dann wieder unter den Default-Alias-Namen „SignatureKey“ und „EncryptionKey“ gefunden werden.

Ein Schlüsseltausch von/zu Schlüsseln gleicher Schlüssellänge innerhalb des Softtoken durch ELSTER ist dagegen nicht vorgesehen. Wenn Schlüssel als nicht mehr sicher angesehen werden (Schlüssel ist anderen Personen bekannt geworden), sollte der alte Schlüssel nie mehr eingesetzt werden, auch nicht für einen Schlüsseltausch.

7.1.5.2 Sicherheitsstick und Signaturkarte

Ein Schlüsseltausch innerhalb des Sicherheitsstick durch ELSTER ist nicht vorgesehen. Wenn Schlüssel als nicht mehr sicher angesehen werden (Schlüssel ist anderen Personen bekannt geworden), sollte der alte Schlüssel nie mehr eingesetzt werden, auch nicht für einen Schlüsseltausch. In einem solchen Fall muss der Sicherheitsstick (falls möglich) gelöscht und mit neuen Schlüsseln versehen werden.

7.1.5.3 Signaturkarte

Ein Schlüsseltausch innerhalb der Signaturkarte ist durch ELSTER nicht möglich, da die Schlüsselerzeugung und –Zertifizierung in der Hand des Kartenherausgebers liegt.



7.2 Operative Funktionen

7.2.1 Signaturerstellung zwecks Authentisierung

Für die Signaturerstellung ist der Zugriff auf den privaten Schlüssel SK.USR.AUT notwendig. Weiterhin erwartet die ELSTER Signaturprüfung das Authentisierungszertifikat (C.USR.AUT). Dieses muss aus dem Zertifikatsspeicher ausgelesen werden.

Für die Einbettung der Signatur in eine XML Struktur siehe /4/

7.2.1.1 Softtoken

Der Schlüssel SK.USR.AUT befindet sich im KeyBag mit dem friendlyName "signaturekey". Aus diesem KeyBag wird das Attribut "localKeyID" ermittelt und der CertBag mit dieser "localKeyID" ermittelt. Dieser CertBag enthält das Zertifikat C.USR.AUT.

Der KeyBag (friendlyName "SignatureKey") wird mit dem Passwort entschlüsselt. Danach liegt der private RSA Schlüssel ASN1 codiert vor (*Chinese Remainder Theorem* (CRT)).

Der CertBag ("localKeyID" identisch mit dem KeyBag) wird mit dem Passwort entschlüsselt. Danach liegt das Zertifikat ASN1 codiert vor.

7.2.1.2 Sicherheitsstick

Der Schlüssel SK.USR.ENC befindet sich in der Kryptohardware und kann diese nicht verlassen.

Es sind deshalb folgende Operationen durchzuführen:

- Auswahl des Schlüssels SK.USR.AUT: Es wird dazu die Funktionsfolge C_FindObjectsInit/ C_FindObjects/ C_FindObjectsFinal genutzt. Um den Schlüssel auf dem Token eindeutig zu ermitteln, empfiehlt sich als Suchparameter die Kombination aus den Attributen CKA_CLASS und CKA_ID
- Signieren und Ausgabe der Signatur: Es wird die Funktionsfolge C_SignInit/ C_Sign genutzt. Default-Signaturmechanismus ist CKM_RSA_PKCS_PSS mit Parametern CKM_SHA256 und CKG_MGF1_SHA256. Ergebnis ist eine Folge von Bytes, die die Signatur darstellen.

7.2.2 Entschlüsselung


Die im ELSTER Umfeld eingesetzte Verschlüsselung ist immer eine Hybridverschlüsselung. Die Daten sind mit einem zufälligen symmetrischen Schlüssel verschlüsselt und dieser Schlüssel wird RSA verschlüsselt mit den Daten übertragen. Für die Entschlüsselung ist deshalb der Zugriff auf den privaten Schlüssel SK.USR.ENC notwendig.

7.2.2.1 Softtoken

Der Schlüssel SK.USR.ENC befindet sich im KeyBag mit dem friendlyName "EncryptionKey". Der KeyBag wird mit dem Passwort entschlüsselt, danach liegt der private RSA Schlüssel ASN1 codiert vor (*Chinese Remainder Theorem* (CRT))

7.2.2.2 Sicherheitsstick

Der Schlüssel SK.USR.ENC befindet sich in der Kryptohardware und kann diese nicht verlassen. Es sind deshalb folgende Operationen durchzuführen:

	Technische Dokumentation (TDoc) Einheitliche Elster-Datenschnittstelle: Spezifikation ELSTER-Token	Stand: 11.07.2017
--	---	-------------------

- Auswahl des Schlüssels SK.USR.ENC: Es wird dazu Funktionsfolge C_FindObjectsInit/ C_FindObjects/ C_FindObjectsFinal genutzt. Um den Schlüssel auf dem Token eindeutig zu ermitteln, empfiehlt sich als Suchparameter die Kombination aus den Attributen CKA_CLASS und CKA_ID.
- Entschlüsseln und Ausgabe des symmetrischen Schlüssels: Es wird die Funktionsfolge C_DecryptInit/ C_Decrypt genutzt. Entschlüsselungsmechanismus ist CKM_RSA_OAEP_PKCS. Ergebnis ist ein Folge von Bytes, die den symmetrischen Schlüssel darstellen.
- Entschlüsseln der Daten in der ELSTER Client Implementierung.
- Entsprechend dem übermittelten Verschlüsselungsalgorithmus (Standard: DESede/CBC/PKCS5Padding, OID: 1.2.840.113549.3.7) ist mit dem symmetrischen Schlüssel die Entschlüsselung durchzuführen.



8 Glossar

Aktivierungscode	Der Aktivierungscode ist ein von der Finanzbehörde generierter, 12-stelliger alphanumerischer Wert, der dem Anwender einmalig zur Aktivierung einer persönlichen Zugangsmöglichkeit zum ElsterOnline-Portal dient. Er kann effektiv nur in Kombination mit einer dazugehörigen Aktivierungs-ID verwendet werden. Der individuelle Aktivierungscode wird dem ElsterOnline-Anwender auf dem Postweg mitgeteilt.
Aktivierungs-ID	Die Aktivierungs-ID eine von der Finanzbehörde generierte, 10-stellige Zahl, welche dem Anwender zur Aktivierung einer persönlichen Zugangsmöglichkeit zum ElsterOnline-Portal dient. Sie kann effektiv nur in Kombination mit einem dazugehörigen Aktivierungscode verwendet werden. Die individuelle Aktivierungs-ID wird dem ElsterOnline Anwender per E-Mail zugestellt.
Asymmetrische Schlüsselpaare	Ein asymmetrisches Schlüsselpaar ist individuell einem Anwender oder System zugeordnet. Es besteht aus einem öffentlichen Schlüssel und einem privaten bzw. geheimen Schlüssel. Der private Schlüssel darf nur dem Inhaber bekannt sein und dient zur individuellen Signatur oder Entschlüsselung elektronischer Informationen mittels asymmetrischen kryptographischen Verfahren. Der öffentliche Schlüssel dient der Allgemeinheit zur Verifizierung einer mit dem privaten Schlüssel durchgeführten Signatur oder zur individuellen Verschlüsselung. Die Mathematik stellt praktisch sicher, dass Verschlüsselung und Entschlüsselung sowie Signatur und Signaturprüfung nur mit dem entsprechenden Schlüsselpaar funktioniert.
Asymmetrische kryptographische Verfahren	Bei asymmetrischen kryptographischen Verfahren bekommt eine Person oder ein System immer zwei Schlüssel bzw. ein asymmetrisches Schlüsselpaare für Signatur- oder Verschlüsselungsfunktionen zugewiesen, und zwar einen öffentlichen und einen privaten Schlüssel. Der öffentliche Schlüssel ist jedem zugänglich, der private Schlüssel nur der jeweiligen Person bzw. dem jeweiligen System. Mit dem privaten Schlüssel können Daten signiert und von jedem mit dem zugehörigen öffentlichen Schlüssel geprüft werden. Außerdem können mit dem öffentlichen Schlüssel Daten von jedem verschlüsselt und nur mit dem zugehörigen privaten Schlüssel entschlüsselt werden. Bekannte asymmetrisch kryptographische Verfahren sind zum Beispiel RSA, DSA, DSS und ECC.
Authentisierung	Authentisierung ist der Vorgang, eine behauptete Identität, beispielsweise eine Person in Bezug zu einer Benutzererkennung nachzuweisen. Meistens erfolgt der Nachweis mittels Angabe der Benutzererkennung und einem Passwort. Stärkere Sicherheit bietet die Authentisierung mit asymmetrischen kryptographischen Verfahren oder biometrischen Merkmalen (z. B. Fingerabdruck).
Authentizität	Unter Authentizität wird allgemein die Echtheit (Unverfälschtheit) und Glaubwürdigkeit von Daten oder einer Partnerinstanz verstanden. Die Authentizität kann durch kryptographische Verfahren gesichert und überprüft werden. Zum Beispiel über asymmetrische kryptographische Verfahren.
Common Criteria	Gemeinsame Kriterien für die Prüfung und Bewertung der Sicherheit von Informationstechnik. Sie sind für die Bewertung der Sicherheitseigenschaften praktisch aller informationstechnischen Produkte und Systeme geeignet. Durch die internationale Standardisierungsorganisation (ISO) sind sie unter der Nummer 15408 als internationaler Standard aufgenommen.



Computerviren	In der Fachsprache ist ein Computervirus eine nichtselbständige Programmroutine, die sich selbst reproduziert, indem sie sich an andere Software oder Bereiche des Betriebssystems z. B. eines Clients anhängt und, einmal gestartet, vom Anwender nicht kontrollierbare Manipulationen an selbigen vornimmt. Die Idee zu Computerviren leitete sich von dem biologischen Vorbild der Viren ab und gab ihnen ihren Namen. Durch Computerviren kommt es auf einem Computer häufig zur Veränderung oder Verlust von Daten und Programmen sowie zu Störungen des regulären Betriebs.
ELSTER-Funktionsbibliothek	Bezeichnet eine Software bzw. eine Datei der Finanzbehörde, welche vordefinierte Funktionen für den technischen Zugang eines Clients auf das ElsterOnline-Portal zur Einbindung in andere Client-Software bereithält.
ELSTER-Policy	Signaturerzeugungskomponenten (z. B. Signaturkarten) und Zertifizierungsdiensteanbieter (z. B. Trustcenter), die im Zusammenhang mit der elektronischen Steuererklärung ElsterOnline und weiteren zugehörigen Diensten eingesetzt werden sollen, müssen sicherheitstechnisch einen gewissen Mindeststandard besitzen und einigen technischen Anforderungen genügen. Der sicherheitstechnische Mindeststandard und die technischen Anforderungen werden in der ELSTER-Policy der Finanzbehörde beschrieben.
Sicherheitsstick	Der Sicherheitsstick ist ein an den USB-Anschluss des PC anschließbares individuell geschütztes Gerät, welches einen Kryptochip beinhaltet und dort die persönlichen Kryptomittel speichert. Das Aussehen ist ähnlich einem USB-Memory-Stick. Die Funktionen des integrierten Kryptochips entsprechen in Hard- und Software identisch denen einer Chipkarte. Der Sicherheitsstick kann käuflich erworben werden.
Fortgeschrittene Signaturkarte	Die auf einer fortgeschrittenen Signaturkarte enthaltenen Schlüsselpaare sind durch anerkannte Trustcenter oder durch Vertragsbeziehungen eindeutig einem Inhaber zugeordnet und gemäß international anerkannten Richtlinien generiert. Sie ermöglichen dem Inhaber Signaturen zu erstellen, welche eindeutig die Identifizierung der Signatur für andere ermöglichen. Die Signaturkarte ist dem Inhaber persönlich zugeordnet und muss unter seiner Kontrolle sein.
HSM	Ein Hardware Security Modul (HSM) ermöglicht bei angemessener Konfiguration einem System Hochsicherheit in der Verarbeitung eigener asymmetrischer Schlüsselpaare sowie in der Anwendung asymmetrischer kryptographischer Verfahren. Im Besonderen wird dabei die sichere Speicherung notwendiger privater Schlüssel im System möglich.
Hybrider kryptographischer Algorithmus	Kryptographisches Verfahren, welches symmetrische und asymmetrische Verfahren kombiniert. Zumeist werden dabei symmetrische Verfahren als Verschlüsselungsmechanismus eingesetzt und die asymmetrischen Verfahren als Schlüsselaustauschmethoden verwendet (Verschlüsselung des symmetrischen Schlüssels mit dem öffentlichen Schlüssel des Empfängers).
Integrität	Integrität stellt sicher, dass Daten korrekt sind und unberechtigte Änderungen verhindert oder zumindest festgestellt werden können.
ITSec	Die Evaluierung nach ITSec beinhaltet die Prüfung und Bewertung der Sicherheitseigenschaften eines informationstechnischen Produkts nach festgelegten Sicherheitskriterien, angeleitet durch ein Evaluationshandbuch. Sie geht über eine einfache Konformitätsprüfung zwischen einem Benutzerhandbuch und tatsächlichem Verhalten des Produkts weit hinaus.
Java-Runtime-Umgebung	Für diverse auf einem Internet-Browser basierte Anwendungen kann eine Plattform unabhängige Java Runtime Umgebung notwendig sein. Die Umgebung ist eine Software und kann im Internet bei der Firma Sun Microsystems herunter geladen und auf einem Client installiert werden. Da ElsterOnline plattformunabhängig mittels Java abläuft, wird die Java-Runtime Umgebung auf dem Client benötigt.



Java-Applet	Ein Java-Applet ist eine in der Programmiersprache Java geschriebene Anwendung, die innerhalb eines Internet-Browsers ausgeführt werden kann. Es dient als eine Möglichkeit der clientseitigen Datenverarbeitung von aktiven Web-Inhalten in einer Internet-basierten Client/Server Umgebung. Das Java-Applet wird dabei nach Erfordernis automatisch vom Server in den Internet-Browser des Clients geladen.
Kryptoboxen	Kryptoboxen dienen in der Regel einer generellen Verschlüsselung von Kommunikationsverbindungen bzw. einer Leitungsver schlüsselung. Alle elektronischen Informationen, die über eine Kommunikationsverbindung gesendet werden, werden auf der einen Seite in einem Knotenpunkt durch eine Kryptobox verschlüsselt und auf der anderen Seite wieder in einem Knotenpunkt durch eine andere Kryptobox entschlüsselt. Im Allgemeinen basiert die Sicherheit der Leitungsver schlüsselung auf starken symmetrischen kryptographischen Verfahren. Alle elektronischen Informationen werden dabei mit den gleichen symmetrischen Schlüssel verschlüsselt, der nur den relevanten Kryptoboxen bekannt ist.
Kryptographie	Originäres Ziel der Kryptografie ist das Unkenntlichmachen von Daten für unberechtigte Dritte durch Anwenden von Verschlüsselungsmethoden, bzw. die Lehre von der Geheimhaltung von Informationen. Die Verschlüsselung gilt als umso stärker, je mehr theoretischer bzw. mathematischer Aufwand betrieben werden müsste, um die Rekonstruktion der Daten durch einen Unbefugten durchzuführen. Zur Kryptographie zählen auch Methoden der Authentisierung, der elektronischen Signatur und des Authentizitätsnachweises.
Mindestschlüssellänge	Eine Beurteilungsmöglichkeit für die Stärke kryptographischer Verschlüsselungs- und Signaturverfahren ist die Länge der innerhalb der Verfahren einsetzbaren kryptographischen Schlüssel in Bits gemessen. Je höher die Anzahl der Bits der eingesetzten Schlüssel, umso stärker wird das kryptographische Verfahren beurteilt. Die Mindestschlüssellänge bezeichnet die minimal einzusetzende Anzahl an Bits eines kryptographischen Schlüssels, damit das entsprechende Verfahren noch als stark anzusehen ist.
Öffentlicher Schlüssel	Das ist der verwendete öffentliche Schlüssel bei asymmetrischen kryptographischen Verfahren. Dieser wird vom Inhaber oder dem ausstellenden Trustcenter öffentlich zugänglich gemacht, z. B. über den Verzeichnisdienst des Trustcenters. Mit Hilfe eines von einem Trustcenter ausgestellten und signierten elektronischen Zertifikats wird der öffentliche Schlüssel offiziell zum Inhaber gehörig beglaubigt. Zu jedem öffentlichen Schlüssel gehört ein privater Schlüssel, der nur dem Inhaber des so genannten asymmetrischen Schlüsselpaars bekannt sein darf.
Personal-Firewall	Eine Personal Firewall ist eine Sicherheitssoftware für den persönlichen Client, um den Zugriffsschutz für Unberechtigte aus dem Internet verstärkt abzusichern. Sie soll den Client vor Angriffen von außen schützen und auch verhindern, dass bestimmte Programme, zum Beispiel Computerviren, Kontakt vom Client zum Internet herstellen. Dazu kontrolliert sie alle Verbindungen in andere Netzwerke und überprüft sowohl die Anfragen ins Internet als auch die Daten, die zum Client kommen.
PKCS#12	Defacto Standard der Firma RSA Security, welcher das Format für die Speicherung und den Transport asymmetrischer Schlüsselpaare, entsprechender Zertifikate und weiterer elektronischer Sicherheitsmittel definiert.



Post-Ident Basic Verfahren	Post-Ident Basic bietet eine sichere Identifikation von Anwendern in den Filialen der deutschen Post. Es genügt nachweisbar den verschärften Kontrollen des Geldwäschegesetzes. Um Post-Ident Basic im Rahmen von ElsterOnline zu nutzen, erhält der Kunde per Post oder Online ein spezielles Formular von der Finanzbehörde. Das ausgefüllte Formular wird dann vom Anwender zwecks Identifikation bei seiner Filiale der Deutschen Post vorgelegt. Der Anwender kommt mit dem Formular und seinem Personalausweis oder Reisepass zu seiner Filiale der Deutschen Post und wird dort identifiziert. Die Überprüfung der Unterschrift des Kunden, die notwendige Dokumentation im Verfahren und die Identifikationsbestätigung erfolgt durch einen Filialmitarbeiter. Ein vollständig ausgefülltes Post-Ident Basic Formular wird an die Finanzbehörde zurückgesendet.
Privater Schlüssel	Dies ist der bei asymmetrischen Verfahren verwendete kryptographische Schlüssel, auf den nur der Inhaber Zugriff haben darf. Der private Schlüssel dient zur Erzeugung von elektronischen Signaturen und zur Entschlüsselung von Daten. Zu jedem privaten Schlüssel gehört ein öffentlicher Schlüssel, der nur dem Inhaber des so genannten asymmetrischen Schlüsselpaars bekannt sein darf.
PSE	Das ist ein persönlicher elektronischer Sicherheitsbereich, in dem persönliche sicherheitsrelevante Daten, wie beispielsweise ein privater Schlüssel, enthalten sind. Zu finden ist ein PSE in der Regel auf einer Chipkarte, kann aber auch als verschlüsselte Datei vorliegen. Der PSE ist durch ein Passwort, eine PIN oder durch biometrische Verfahren (Fingerabdruck, Augenscanner) gesichert. Eine konkrete PSE ist hier z.B. ein Softtoken, der Sicherheitsstick oder eine Signaturkarte.
Qualifizierte Signaturkarten	Qualifizierte Signaturkarten ermöglichen dem Inhaber eine gemäß dem deutschen Signaturgesetz rechtskonforme elektronische Signatur, welche die Verbindlichkeit einer manuellen Unterschrift besitzt. Die auf der qualifizierten Signaturkarte enthaltenen Schlüsselpaare sind durch ein von der Regulierungsbehörde für Post und Telekommunikation akkreditiertes Trustcenter eindeutig einem Inhaber zugeordnet und werden durch Produkte für qualifizierte Signaturen verarbeitet.
RSA-Algorithmus	Der RSA-Algorithmus ist ein asymmetrisches kryptographisches Verfahren und eignet sich zur elektronischen Signatur sowie zum Schlüsselaustausch symmetrischer Schlüssel durch asymmetrische Verschlüsselung. Er wurde von Rivest, Shamir und Adleman 1977 entwickelt. Die Sicherheit dieses Algorithmus beruht vom Prinzip her auf der Schwierigkeit der Faktorisierung großer Zahlen.
Signatur (elektronische)	Signatur bezeichnet im vorliegenden Dokument eine elektronische Signatur bzw. Unterschrift. Mit der elektronischen Signatur kann der Ursprung von Daten eindeutig nachgewiesen werden. Sie kann auf dem elektronischen Weg für eine Willenserklärung oder zur Authentisierung heran gezogen werden. Unter digitalen Signaturen werden solche elektronischen Signaturen verstanden, die mittels asymmetrischen kryptographischen Verfahren erzeugt und überprüft werden können. Die EU-Richtlinie für elektronische Signaturen sowie deren nationale Umsetzung, beispielsweise in Deutschland durch das Signaturgesetz, die Signaturverordnung sowie das Formanpassungsgesetz, ermöglichen die rechtliche Gleichstellung der elektronischen mit der handschriftlichen Signatur. Die realisiert die so genannte qualifizierte elektronische Signatur.
Software-Schlüssel	Der Software-Schlüssel ist eine individuell geschützte Datei, die auf dem PC in einer speziellen Sicherheitsumgebung gespeichert wird und die persönlichen Kryptomittel enthält. Die Datei auf dem PC (d.h. der Festplatte) oder einem externen Speichermedium (z. B. Diskette, ZIP-Laufwerk oder USB-Memory-Stick) gespeichert werden.



Sperrlisten	Eine Sperrliste dient einem Trustcenter zur Veröffentlichung von Zertifikaten, die vor Ablauf ihrer Gültigkeit gesperrt wurden. Alle in einer Sperrliste aufgeführten Zertifikate sind ab dem Zeitpunkt der Publikation ungültig.
SSL	Secure Socket Layer (SSL) ist ein Protokoll für sicheren Datenaustausch zwischen Client und Server über das Internet. Client und Server können sich mit Hilfe asymmetrischer kryptographischer Verfahren gegenseitig authentisieren und die Daten beim Datenaustausch verschlüsseln. Entwickelt wurde dieses Protokoll von der Firma Netscape.
StDÜV	In der Steuerdaten-Übermittlungsverordnung (StDÜV) wird geregelt, in welchen Fällen bei der elektronischen Übertragung von Steuerdaten auf eine qualifizierte elektronische Signatur bzw. Signaturkarte verzichtet, d. h. somit zum Beispiel eine fortgeschrittene Signatur angewendet werden kann (z. B. Sicherheitsstick und fortgeschrittene Signaturkarte).
Symmetrische Verfahren	Bei symmetrischen Verfahren wird ein geheimer Schlüssel zur Verschlüsselung und Entschlüsselung von Daten verwendet. Soll die verschlüsselte Datei weitergegeben werden, muss dem Empfänger der geheime Schlüssel auf einem sicheren Übertragungsweg mitgeteilt werden. Da es neben dem Gespräch unter vier Augen keine wirklich sichere Methode gibt, wird dies zum Problem. Asymmetrische Verfahren lösen das Problem der Mitteilung.
TESTA	Bei TESTA (Trans-European Services for Telematics between Administrations) handelt es sich um ein Overlay-Netz der europäischen Verwaltungen. Das primäre Ziel von TESTA besteht darin, den europäischen Einrichtungen, Agenturen und Verwaltungen ein umfassendes, gut strukturiertes Dienstangebot auf der Basis anerkannter Marktstandards bereitzustellen, das einen einfachen und zuverlässigen Austausch von Daten ermöglicht und optimale Interoperabilität gewährleistet. Ein Teil dieses großen Projektes ist TESTA Deutschland, die Zusammenarbeit des Bundes und der Länder im Sinne eines Zusammenschluss der einzelnen Landesnetze sowie dem direkte Anschluss einzelner Bundesbehörden und des Informationsverbundes Berlin-Bonn (IVBB). Das TESTA-Netz bildet damit das Rückgrat eines "Corporate Network Verwaltung" für die länderübergreifende Kommunikation.
Trojaner	Eine scheinbar nützliche Datei bzw. Software die nicht den vermuteten Inhalt hat. Dadurch besteht die Möglichkeit für die Datei bzw. Software, unbemerkt auf einem Client nicht vorgesehene Funktionen zu installieren. Durch einen solchen Vorgang können beispielsweise Passwörter und andere vertrauliche Daten ausgespäht, verändert, gelöscht oder bei der nächsten Datenübertragung an einen Unberechtigten verschickt werden. Dieser "Datendiebstahl" bleibt ohne dedizierte Sicherheitsmechanismen in der Regel unbemerkt.
Trustcenter	Ein Trustcenter ist eine unabhängige, vertrauenswürdige Instanz, die für die Vergabe und die Verwaltung von elektronischen Zertifikaten zuständig ist. Das Trustcenter signiert die von Ihr ausgestellten Zertifikate digital und garantiert somit für die Echtheit der Daten auf dem Zertifikat. Da asymmetrischen Verfahren alle Teilnehmer dem Trustcenter vertrauen, können sie auf diese Weise auch auf die Gültigkeit der ausgestellten Zertifikate und damit den öffentlichen Schlüsseln anderer Teilnehmer vertrauen.
Verschlüsselung	Unter Verschlüsselung versteht man die Transformation von Daten zu deren sicheren Aufbewahrung oder Übermittlung. Dazu wird mit Hilfe eines kryptographischen Schlüssels der Inhalt z. B. eines Dokuments, einer Datei oder E-Mail für unbefugte Dritte unleserlich gemacht. Nur der richtige Empfänger kann die Daten mit Hilfe eines passenden (Entschlüsselungs-) Schlüssel wieder lesen. Es gibt unterschiedliche Verschlüsselungsverfahren wie symmetrische, asymmetrische und hybride Verschlüsselung.

	Technische Dokumentation (TDoc) Einheitliche Elster-Datenschnittstelle: Spezifikation ELSTER-Token	Stand: 11.07.2017
--	---	-------------------

Verzeichnisdienst	Verzeichnisdienst bezeichnet in diesem Dokument eine nach dem ITU-Standard X.500 hierarchisch (baumförmig) aufgebaute Datenbank, in der von einem geeigneten System aus Informationen abgerufen werden können. Anwendung findet dies z. B. in Adress-, E-Mail- und Zertifikatsverzeichnissen, in denen nach unterschiedlichen Kriterien die gewünschte Information gesucht werden kann. Die Datenbank kann auch über mehrere Server verteilt vorliegen.
Zertifikat	Ein elektronischer Ausweis für eine Person bzw. System, der von einem Trustcenter ausgestellt und durch dessen elektronische Signatur beglaubigt ist und insbesondere die Zuordnung eines öffentlichen Schlüssels zu einer Person bzw. einem System garantiert. In der Regel werden Zertifikate in einem Verzeichnisdienst veröffentlicht.



9 Anhang:

9.1 Beispiel eines PKCS12 Softtoken (ASN1 dump):

```
0 8235: SEQUENCE {
4 1: INTEGER 3
7 8165: SEQUENCE {
11 9: OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
22 8150: [0] {
26 8146: OCTET STRING, encapsulates {
30 8142: SEQUENCE {
34 1603: SEQUENCE {
38 9: OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
49 1588: [0] {
53 1584: OCTET STRING, encapsulates {
57 1580: SEQUENCE {
61 787: SEQUENCE {
65 11: OBJECT IDENTIFIER
: pkcs-12-pkcs-8ShroudedKeyBag (1 2 840 113549 1 12 10 1 2)
78 690: [0] {
82 686: SEQUENCE {
86 40: SEQUENCE {
88 10: OBJECT IDENTIFIER
: pbeWithSHAAnd3-KeyTripleDES-CBC (1 2 840 113549 1 12 1 3)
100 26: SEQUENCE {
102 20: OCTET STRING
: E9 D4 DB 92 C7 C9 14 46 03 A0 11 F7 C3 32 85 37
: 00 A2 B2 CF
124 2: INTEGER 1024
: }
: }
128 640: OCTET STRING
: DB C7 DA 7E 4E F6 07 31 A0 15 42 0C 96 9F EF 7A
: 5F 09 07 75 B3 59 CD 4E 70 84 CD D7 39 DC D1 4C
: 5D 47 C1 59 7F 33 1D 3B CF FF 27 AF 2B 31 70 E8
: 0F 02 76 0F 62 A9 68 41 1B E4 D6 78 22 C4 E6 C1
: 3F A4 16 8E AE 87 FD 1A 12 9E D7 92 3E 0A 95 30
: 18 3A 20 08 22 9A 9A A4 4C 8E D1 C5 DB E3 B0 C8
: 6C AF F7 72 85 56 E9 4D F2 DE E4 C6 83 15 45 CC
```



```
      :          9E F5 46 A4 34 E2 1A EA 1E DA 4A C2 B8 E9 0D 64
      :          [ Another 512 bytes skipped ]
      :          }
      :          }
772 78:          SET {
774 41:          SEQUENCE {
776  9:          OBJECT IDENTIFIER
      :          friendlyName (for PKCS #12) (1 2 840 113549 1 9 20)
787 28:          SET {
789 26:          BMPString 'encryptionkey'
      :          }
      :          }
817 33:          SEQUENCE {
819  9:          OBJECT IDENTIFIER
      :          localKeyID (for PKCS #12) (1 2 840 113549 1 9 21)
830 20:          SET {
832 18:          OCTET STRING 'Time 1116412528840'
      :          }
      :          }
      :          }
852 785:         SEQUENCE {
856 11:         OBJECT IDENTIFIER
      :         pkcs-12-pkcs-8ShroudedKeyBag (1 2 840 113549 1 12 10 1 2)
869 690:         [0] {
873 686:         SEQUENCE {
877 40:         SEQUENCE {
879 10:         OBJECT IDENTIFIER
      :         pbeWithSHAAnd3-KeyTripleDES-CBC (1 2 840 113549 1 12 1 3)
891 26:         SEQUENCE {
893 20:         OCTET STRING
      :         5D 5F 68 2A 24 E1 A0 27 69 F6 D4 F5 9F 39 BA 68
      :         CC 03 D1 29
915  2:         INTEGER 1024
      :         }
      :         }
919 640:         OCTET STRING
      :         8B 0B 88 E0 A8 AF 0D CE 38 B4 9A 8B 1A C2 21 E7
```



```
      :          12 0E 3C 5F 10 C7 22 38 49 31 CC 9D AA DA 86 F1
      :          03 9E D2 B6 C0 5E 62 D8 DC 6B FA 1A 8D 98 F1 CE
      :          F9 42 FB 35 24 2D 47 51 5A D3 D7 D8 39 C9 AB 76
      :          CD 66 1D DC 6E DB E4 39 CC ED B0 41 18 15 3E 7D
      :          7A D2 ED 8C D9 72 B5 95 55 A1 BB E9 79 18 00 E5
      :          24 BE A3 65 E0 7E 9A 6E 54 08 7D FF AD 7B 22 11
      :          5B A6 24 20 DC 08 30 9C 06 BA D1 69 44 25 D3 08
      :          [ Another 512 bytes skipped ]
      :          }
      :          }
1563 76:          SET {
1565 39:          SEQUENCE {
1567 9:          OBJECT IDENTIFIER
      :          friendlyName (for PKCS #12) (1 2 840 113549 1 9 20)
1578 26:          SET {
1580 24:          BMPString 'signaturekey'
      :          }
      :          }
1606 33:          SEQUENCE {
1608 9:          OBJECT IDENTIFIER
      :          localKeyID (for PKCS #12) (1 2 840 113549 1 9 21)
1619 20:          SET {
1621 18:          OCTET STRING 'Time 1116412528780'
      :          }
      :          }
      :          }
      :          }
      :          }
      :          }
      :          }
      :          }
1641 6531:         SEQUENCE {
1645 9:          OBJECT IDENTIFIER encryptedData (1 2 840 113549 1 7 6)
1656 6516:         [0] {
1660 6512:         SEQUENCE {
1664 1:          INTEGER 0
1667 6505:         SEQUENCE {
1671 9:          OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
```



```
1682 40: SEQUENCE {
1684 10:     OBJECT IDENTIFIER
      :     pbeWithSHAAnd40BitRC2-CBC (1 2 840 113549 1 12 1 6)
1696 26: SEQUENCE {
1698 20:     OCTET STRING
      :     6A 8A 56 0C 04 B5 1D 6D 0D 82 0D 52 58 A3 54 50
      :     3E 6B 6D 25
1720 2:     INTEGER 1024
      :     }
      :     }
1724 6448: [0]
      :     22 F6 71 DE C6 0B 10 6E 18 F7 F5 53 95 52 EA 91
      :     DF 0C 05 93 0D B0 9F 11 21 1D 24 37 46 6F B7 1E
      :     24 2C 02 35 2B 31 A6 87 BD F4 D8 4E E6 1E D6 BD
      :     32 71 C2 B0 6F 9B 16 93 D9 17 85 73 1F 04 73 CA
      :     C5 1C CB 9D EE 84 85 D0 94 BD 11 89 8F 35 21 1B
      :     9E 77 77 DE 5D AF AB 42 D3 D7 3D 4C 0B BA AA AB
      :     AC 6D A7 02 DF 42 AB F8 1C 9C 00 46 E6 14 A3 B9
      :     E0 3D D6 DE 1D 15 C6 B0 C4 D2 3F 05 57 E1 A0 BA
      :     [ Another 6320 bytes skipped ]
      :     }
      :     }
      :     }
      :     }
      :     }
      :     }
      :     }
8176 61: SEQUENCE {
8178 33:     SEQUENCE {
8180 9:     SEQUENCE {
8182 5:     OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
8189 0:     NULL
      :     }
8191 20: OCTET STRING
      :     25 F2 C5 86 9F 03 34 0B 93 D4 EE 10 2A 4E 1B 93
      :     43 5D 31 C6
      :     }
```



```
8213 20: OCTET STRING
      : 56 CF 2D 7E 00 1D D4 1E 62 8D 53 5E 5A 59 56 72
      : 21 B7 5A D8
8235 2: INTEGER 1024
      : }
      : }
```

9.2 Beispielcode zur Generierung des PKCS12 Keystore

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;

public class Foo {

    /**
     * @param keyStoreFilename Dateiname der Soft-PSE
     * @param pin PIN zur Verschlüsselung der SafeBags
     * @param keyPriv1 Signaturschlüssel
     * @param keyPriv2 Verschlüsselungsschlüssel
     * @param certChain1 Zertifikatskette zum Signaturschlüssel
     * @param certChain2 Zertifikatskette zum Verschlüsselungsschlüssel
     */
    public static void foo(String keyStoreFilename, String pin,
        PrivateKey keyPriv1, PrivateKey keyPriv2, Certificate[] certChain1,
        Certificate[] certChain2) {
        try {
            KeyStore store = KeyStore.getInstance("PKCS12");
            store.load(null, null);
            store.setKeyEntry("SignatureKey", keyPriv1, pin.toCharArray(),
                certChain1);
            store.setKeyEntry("Encryptionkey", keyPriv1, pin.toCharArray(),
                certChain2);
        }
    }
}
```



```
        FileOutputStream fOut = new FileOutputStream(keyStoreFilename);
        store.store(fOut, pin.toCharArray());

    } catch (KeyStoreException e) {
        e.printStackTrace();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    } catch (CertificateException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}
```

9.3 Beispielcode zur PIN Änderung des PKCS12 Keystore

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.cert.CertificateException;

public class Foo {

    /**
     * @param keyStoreFilename
     *         Dateiname der Soft-PSE
     * @param oldPin
     *         alte PIN zur Entschlüsselung der SafeBags
     * @param newPin
     *         neue PIN zur Verschlüsselung der SafeBags
     */
    public static void foo(String keyStoreFilename, String oldPin, String newPin) {
        try {
            KeyStore store = KeyStore.getInstance("PKCS12");
```




```
store.load(new FileInputStream(keyStoreFilename), oldPin
    .toCharArray());
store.store(new FileOutputStream(keyStoreFilename), newPin
    .toCharArray());
} catch (KeyStoreException e) {
    e.printStackTrace();
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
} catch (CertificateException e) {
    e.printStackTrace();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
}
```

9.4 Beispielcode zur PIN Änderung des PKCS11 Keystore

```
import iaik.pkcs.pkcs11.Session;
import iaik.pkcs.pkcs11.Token;
import iaik.pkcs.pkcs11.TokenException;

public class Foo {

    /**
     * @param token
     *         das PKCS#11 Token Objekt
     * @param oldPin
     *         alte PIN zur Entschlüsselung der SafeBags
     * @param newPin
     *         neue PIN zur Verschlüsselung der SafeBags
     */
    public static void foo(Token token, String keyStoreFilename, String oldPin,
        String newPin) {
        try {
            // open rw session
```



```
    Session session = token.openSession(  
        Token.SessionType.SERIAL_SESSION,  
        Token.SessionReadWriteBehavior.RW_SESSION, null, null);  
  
    // change PIN  
  
    char[] op = oldPin.toCharArray();  
    char[] np = newPin.toCharArray();  
  
    session.setPIN(oldPin.toCharArray(), newPin.toCharArray());  
  
    session.closeSession();  
  
} catch (TokenException e) {  
  
    e.printStackTrace();  
  
}  
  
}
```

9.5 Beispielcode zur Signatur mit PKCS12 Keystore

```
package de.elster.client.demo;  
  
import java.io.FileInputStream;  
import java.security.KeyStore;  
import java.security.PrivateKey;  
import java.security.Signature;  
import java.security.cert.Certificate;  
  
public class Foo {  
  
    /**  
     * @param keyStoreFilename  
     *         Dateiname der Soft-PSE  
     * @param pin  
     *         PIN zur Entschlüsselung der SafeBags  
     * @param dataToSign  
     *         die zu signierenden Daten  
     * @return die digitale Signatur  
     */  
    public static byte[] foo(String keyStoreFilename, String pin, byte[] dataToSign) {  
        byte[] digSig = null;  
        try {
```



```
KeyStore store = KeyStore.getInstance("PKCS12");
store.load(new FileInputStream(keyStoreFilename), pin
    .toCharArray());
PrivateKey key1_priv = (PrivateKey)store.getKey("signaturekey",
    pin.toCharArray());

// das Attribut 'localKeyID' wird innerhalb dieser Bibliotheksfunktion genutzt
Certificate cert1 = store.getCertificateChain("signaturekey")[0];

Signature sigObj = Signature.getInstance("SHA1withRSA");

// Schlüssel brauch hier nicht über die ASN1 Struktur eingebunden zu werden
sigObj.initSign(key1_priv);
sigObj.update(dataToSign);
digSig = sigObj.sign();

} catch (Exception e) {
    e.printStackTrace();
}
return digSig;
}
}
```

9.6 Beispielcode zur Signatur mit PKCS11 Keystore

```
package de.elster.client.demo;

import iaik.pkcs.pkcs11.Mechanism;
import iaik.pkcs.pkcs11.Session;
import iaik.pkcs.pkcs11.Token;
import iaik.pkcs.pkcs11.objects.Object;
import iaik.pkcs.pkcs11.objects.PrivateKey;
import iaik.pkcs.pkcs11.objects.RSAPrivateKey;

public class Foo {

    /**
     * @param token
```



```
*          das PKCS#11 Token Objekt
* @param pin
*          PIN zur Entschlüsselung der SafeBags
* @param dataToSign
*          die zu signierenden Daten
* @return die digitale Signatur
*/
public static byte[] foo(Token token, String pin, byte[] dataToSign) {
    byte[] digSig = null;
    try {
        // oeffne rw session
        Session session = token.openSession(
            Token.SessionType.SERIAL_SESSION,
            Token.SessionReadWriteBehavior.RW_SESSION, null, null);
        // login
        session.login(Session.UserType.USER, pin.toCharArray());

        // erstelle Suchmaske
        RSAPrivateKey privateSignatureKeyTemplate = new RSAPrivateKey();
        privateSignatureKeyTemplate.getSign().setBooleanValue(Boolean.TRUE);
        privateSignatureKeyTemplate.getId().setByteArrayValue("ELSTER_SIGN".getBytes());
        Object[] foundPrivateKeyObjects = null;

        // Suche
        session.findObjectsInit(privateSignatureKeyTemplate);
        foundPrivateKeyObjects = session.findObjects(1); // find first
        session.findObjectsFinal();

        // Im Falle einer erfolgreichen Suche: Signaturerstellung
        if (foundPrivateKeyObjects != null && foundPrivateKeyObjects.length > 0)
        {
            PrivateKey signatureKey =
                (PrivateKey)foundPrivateKeyObjects[0];

            // Initialisierung des Signaturprozesses
            session.signInit(Mechanism.SHA1_RSA_PKCS, signatureKey);
            // Signatur der Daten
            digSig = session.sign(dataToSign);
        }
    }
}
```



```
    }  
    session.logout();  
} catch (Exception e) {  
    e.printStackTrace();  
}  
return digSig;  
}  
}
```