Sign Live! CC 3.2.3

certification ID: BSI-DSZ-CC-00397

Security Target

Version 1.12

intarsys consulting GmbH
Bahnhofplatz 8
76137 Karlsruhe
Germany

History

version	date	author	comments
0.5	26.07.2006	jst	initial
0.6	31.07.2006	jst	https protocol for checker added
1.0	11.12.2006	jst	delivered for first evaluation
1.1	16.02.2007	jst	delivered to Open Limit
1.2	26.06.2007	jst	T-Systems working list worked off
1.3	12.07.2007	jst	delivered for workshop on different operational environments
1.4	27.07.2007	jst	assumptions for operational environment extended, delivered to BSI
1.5	01.08.2007	jst	delivered for evaluation - path to Installation Verifier delegated to delivery guide - rationale on mutual support added
1.6	15.08.2007	jst	minor changes in regard to the delivery of the TOE with and without JVM
1.7	13.09.2007	jst	T-Systems todo list (04.09.2007) worked off ADO_DEL.2 augmented MD5 removed SHA-224 added SF.SignatureCreation: All certificates stored on a smartcard are shown. table for SSCDs and CTs devided
1.8	05.10.2007	jst	SF.SelectedDocumentFalsificationPrevention: only SHA-1 is used checking of padding information made conform to implementation SF.TOEIntegrityChecking: A Step 5 added bibliography updated
1.9	06.11.2007	jst	references in definitions for A.SSS and OE.SSS updated mapping of FCS_COP.1 (Signing-RSA) to OT.DOC_OPEN.MAN removed
1.10	28.01.2008	jst, wad	JVM version for browser updated SSCDs and card terminals updated added signature encoding ISO 9796-2 figure made consistent with FSP, HLD references to Bundesnetzanzeiger updated

			signature methods Signtrust and timeproof marked as not accredited
1.11	05.03.2008	jst	assumptions and objectives for signature methods Signtrust and timeproof removed
1.12	25.04.2008	kgo	Version updated (3.2.3)

Contents

1 Introduction

The ST introduction contains document management and overview information.

1.1 Identification

1.1.1 ST IDENTIFICATION

Title	*Sign Live! CC* Security Target
Author(s)	intarsys consulting GmbH, Germany
Version	1.12
Status	Release

1.1.2 TOE IDENTIFICATION

Name	*Sign Live! CC (TOE)*
Version	3.2.3

1.2 Preliminary

1.2.1 ACRONYMS

AIS	Application Notes and Interpretation of the Scheme
CRL	Certificate Revocation List
CT	Card Terminal
JVM	Java Virtual Machine
OCSP	Online Certificate Status Protocol
PDF	Portable Document Format
TSS	timeproof Signature Server
SAC	Signature Application Component (Signaturanwendungskomponente)
SCA	Signature Creation Application
SFP	Security Function Policy
SPE	Secure Pin Entry
SSCD	Secure Signature Creation Device
SSS	Signtrust Signature Server
ST	Security Target
TIFF	Tagged Image File Format
TOE	Target of Evaluation
TSC	TSF Scope of Control

TSF TOE's Security Functions

1.3 ST Overview

Sign Live! CC is a product to work with documents in different formats – PDF amongst others - with focus on electronic signature handling in form of single and batch processing.

It is a signature creation application (SCA, according to [CEN]), which offers the following functions:

- creation of an advanced or qualified electronic signature for a single or a set of documents. For qualified signature creation a SSCD is needed.
- validation of electronic signatures for a single or a set of documents.
- legal binding displaying of documents in PDF, TEXT and TIFF format and functions to examine the content of the document(s) to be signed/validated.

To achieve the different throughput levels *Sign Live! CC* connects SSCDs via different devices:

- for usage in a standard workplace environment with card terminal (CT) via *Sign Live! CC's trusted smart card adapters*.
- for high throughput with timeproof GmbH's signature server *QSS 400* (TSS)[1].
- for remote service via intra- or internet with Deutsche Post Com GmbH's *Signtrust Signature Server* (SSS)[2]

Sign Live! CC provides a command line interface, which offers the user the option to call the mentioned functions via operating system calls.

Sign Live! CC itself is a signature application component ("Signaturanwendungs-komponente") according to the German Electronic Signature Law and Ordinance on Electronic Signatures. This product is to be evaluated and certified according to the Common Criteria 2.3 using assurance level EAL 3+.

Sign Live! CC is designed as a Java Application with a set of necessary and optional plugins called "instruments". The user may check the consistency of a *Sign Live! CC* installation with the Java Applet *Sign Live! CC Installation Verifier* which is available on the intarsys homepage.

The components run on Microsoft Windows operating systems since Windows XP. Dependent on the user's scenario *Sign Live! CC* either requires

- a card terminal with secure pin entry mode as well as a smart card or

[1] As the certifying process for this device has not been finished at *Sign Live! CC's* release date this signature method is not part of the security functions to be certified with this version of the TOE.

[2] As the certifying process for this device has not been finished at *Sign Live! CC's* release date this signature method is not part of the security functions to be certified with this version of the TOE.

- *timeproof Signature Server* available via intra- or internet as well as at least one smart card or
- *Signtrust Signature Server* available via intra- or internet as well as at least one smart card

to run the required cryptographic operations in the process of electronic signature creation.

Sign Live! CC ‘s user interface is presented according to the OS settings in English or German language.

1.4 CC Conformance Claim

Base of this description is [CC2.3], released August 2005

The TOE is compliant to

- CC Part 2 extended
- CC Part 3 conformant.

The assurance requirements are compliant to EAL 3 augmented. The augments are AVA_MSU.3, AVA_VLA.4, ADO_DEL.2, ADV_IMP.1, ADV_LLD.1 and ALC_TAT.1.

2 TOE Description

This part of the ST shall describe the TOE as an aid to the understanding of its security requirements and shall address the product or system type.

The TOE description provides context for the evaluation.

2.1 TOE Abstract

This section is a copy of section 1.3 ST Overview (p. 2).

2.2 TOE Intended Usage

The intended usage of the TOE is defined by the following use cases. Actor is always the user. The operational environment is a PC as defined in 2.6.1.

2.2.1 USE CASE: USER SIGNS A SINGLE DOCUMENT

User opens a document, which corresponds to one of the following types: PDF, TEXT or TIFF. User starts the signing process by activating the signature wizard and provides parameters for the signature process.

TOE shows the selected document in an unambiguous way and informs the user that the signing process is starting on the selected document currently shown in the *Trusted Viewer* and asks him to check the document for its obvious and hidden content.

User may stop the process with the consequence that no data will be signed. Otherwise user confirms the content and requests the electronic signature.

To request a signature via

- smart card terminal: user has to enter a pin on the smart card terminal.
- SSS: user has to enter a password to open a trusted channel to the SSCD
- TSS: user may enter the path of a SSL key and a password to open a trusted channel to the SSCD

TOE calculates the signature in cooperation with a smart card which is connected via smart card terminal, TSS or SSS. TOE appends the signature to the document or creates a separate signature file and stores the result(s) on disk.

2.2.2 USE CASE: USER SIGNS A SET OF DOCUMENTS (BATCH SIGNATURE)

User selects a set of documents, which correspond to one of the following types: PDF, TEXT and TIFF. User has the option to analyse every document individually in a *Trusted Viewer*, which shows the document in an unambiguous way. In the Trusted Viewer user may check the selected document for obvious and hidden contents.

User may stop the whole process with the consequence that no document will be signed.

Otherwise user starts the signing by activating the signature wizard and provides parameters for the signing process.

TOE informs the user that signing process is starting on the selected documents and requests electronic signatures for the selected documents as described in section “Use Case: User Signs a Single Document”.

TOE creates the requested signatures as described in section “Use Case: User Signs a Single Document”.

2.2.3 USE CASE: USER VALIDATES A SINGLE DOCUMENT

User opens a document which corresponds to one of the following types: PDF, TEXT, TIFF, PKCS#7. Depending on the TOE’s configuration the TOE starts the validation automatically or the user starts/restarts the validating process via the GUI.

TOE checks the signatures contained in the document according to the configuration and presents the result in an unambiguous way to the user. TOE checks:

- the integrity of the document (Is the document unchanged?)
- the authenticity of the document (Was the certificate contained in the document valid at signing time?)

If the signature contains an attribute certificate, TOE checks its authenticity and presents the result and the attributes of the certificate to the user.

2.2.4 USE CASE: USER VALIDATES A SET OF DOCUMENTS (BATCH VALIDATION)

The validation is done equivalent to the section “Use Case: User Validates a Single Document”. The differences are:

- user selects a set of documents
- the validation result is presented in a compressed way (valid/not valid) in the GUI and as a protocol.

2.2.5 USE CASE: USER USES THE COMMAND LINE INTERFACE (NO GUI)

User may perform the use cases 2.2.1-2.2.4 via command line call without any additional GUI interaction.

TOE presents results of the signing process - optionally – via the storing place of the signed/not signed files.

TOE presents results of the validation process via protocol and - optionally - via the storing place of a validated file.

Before the user starts the process of signature creation/validation, she has to control the extent and content of the document(s) to be signed/validated. The operational environment ensures that the documents and results cannot be changed by unauthorized access during the process. Due to the restrictions of the Bundes Netz Agentur according to the usage of batch signatures all documents to be signed should have the same type, e.g. invoices.

2.2.6 USE CASE: USER CHECKS THE TOE’S INTEGRITY

User downloads the *Sign Live! CC Installation Verifier* via internet from the intarsys homepage. User starts the *Sign Live! CC Installation Verifier* and optionally

provides the installation path of the *Sign Live! CC* installation to be checked. The *Sign Live! CC Installation Verifier* informs the user that the TOE is consistent or lists the problems which were encountered.

2.3 TOE Scope and Boundaries

Sign Live! CC is a Java Application basing on CABAReT software components and additional *Sign Live! CC* components with focus on electronic signature.

Sign Live! CC Installation Verifier is a signed Java Applet to check the consistency of a product installation.

The logical and physical structure of the components and its boundaries are shown in the following figure:

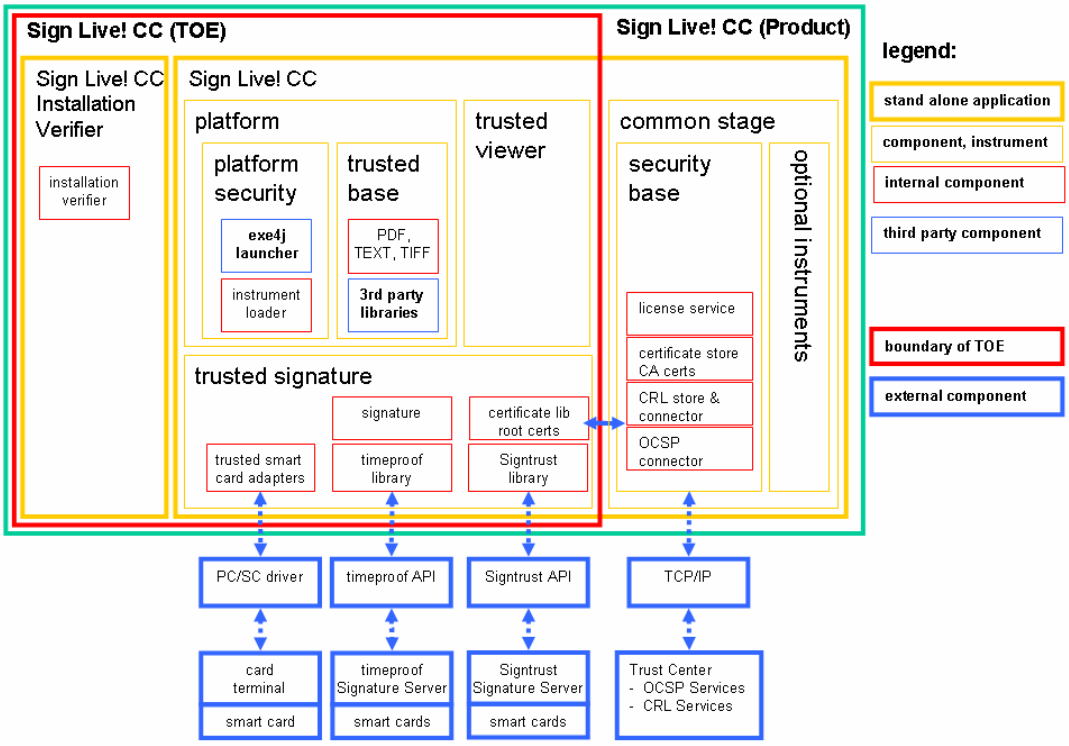

Figure 1 TOE's Scope and Boundaries

The two stand alone applications are framed with a thick yellow line. The external components (framed with thick blue line) are described in more detail in chapter 2.5. To facilitate the installation, the client components of the external components are delivered with *Sign Live! CC* (see Table 2). The TOE – *Sign Live! CC (TOE)* - is framed with a thick red line. Interfaces to external res. non TOE components are represented by dotted blue lines.

Following table provides the TOE's software components and their respective version for the evaluation:

Component	Version
Sign Live! CC	3.2.3
Sign Live! CC Installation Verifier	3.2.3

Table 1 – Software Components of the Evaluated Configuration

Following table provides *Sign Live! CC's* scope of delivery:

Component	Version
Sign Live! CC[3]	3.2.3
Sign Live! CC Installation Verifier	3.2.3
Signtrust clientAPI.dll	see operational environment

Table 2 – Sign Live! CC - Scope of Delivery

2.4 TOE Components

2.4.1 SIGN LIVE! CC

Sign Live! CC is a Java Application basing on a configuration of CABAReT software components. Whilst the CABAReT software components offer basic application infrastructure and functions for document handling, *Sign Live! CC* is the certified version of some CABAReT components plus additional components with focus on electronic signature handling in form of single and batch processing.

The application is installed by a signed setup routine which is available via CD or internet. The setup routine is signed with the intarsys code signing certificate issued by a trusted CA (Thawte).

2.4.1.1 platform security

platform security contains components to control the extend of running code.

exe4j launcher is a generated executable on base of the software package exe4j [EXE4J]. It starts a Java application with Windows look & feel and offers mechanisms to control the used Java VM and options of the Java VM – especially the class path.

instrument loader provides the security function to load instruments in a defined way.

2.4.1.2 trusted base

trusted base contains necessary base libraries to read, parse, store and show PDF, TEXT and TIFF files.

Some of the components are widely used **3rd party libraries**.

2.4.1.3 trusted viewer

trusted viewer presents a selected document in an unambiguous way to the user, checks the document for insecure content and offers functions to examine the document's obvious and hidden contents. The component is used during signature creation and validation process to ensure a consistent view on a document.

[3] To minimize the download size Sign Live! CC is delivered in two formats: with and without included JVM.

2.4.1.4 trusted signature

signature provides all functions to create and validate signatures of a single or a set of documents electronically according to the following table:

Document Type	Signature Type
PDF	PKCS#1 embedded (intern) PKCS#7 detached (intern) PKCS#7 embedded PKCS#7 detached
TEXT (plain text)	PKCS#7 embedded PKCS#7 detached
TIFF	PKCS#7 embedded PKCS#7 detached

Table 3 - Document Types and Signature Types

For signature creation the component delegates the task of encrypting the document's hash value to a SSCD, which is connected via a trusted smart card adapter, timeproof library or Signtrust library.

For validation the signer's certificate and the corresponding certificate chain are checked using the chain model or RFC 3280. *Sign Live! CC* validates each certificate either by CRLs or OCSP requests. The validation state is calculated and presented to the user.

Sign Live! CC supports the validation of attributed certificates and timestamps.

CA certificates, OCSP requests and CRLs are achieved via security base components (not part of the TOE). The results are checked by signature validation.

trusted smart card adapters implement the handling of SSCDs and collect all SSCD specific adapters to communicate to the smart card/terminal combinations defined in section 2.6.2.1.

timeproof library connects TSS for signature creation via SSL.

Signtrust library adapts certified *Signtrust's clientAPI.dll* for signature creation on Signtrust Signature Server.

certificate lib provides functions to access certificate stores in a defined way. For validation purposes it contains the actual set of national root certificates.

2.4.2 SIGN LIVE! CC INSTALLATION VERIFIER

Sign Live! CC Installation Verifier is a signed Java Applet to verify the installation's integrity statically. It can be downloaded via SSL from the intarsys homepage (e.g. http://www.intarsys.de/webapp/verifier, the exact path is defined in the delivery guide, s. 5.2). The applet is signed with the intarsys Code Signing Certificate issued by a trusted CA (Thawte). The applet contains the intarsys Code Signing Certificate and the hash values of the certified products.

Sign Live! CC Installation Verifier checks an installation by validating the installation's directory signature. A directory signature is a signature on a file which describes the content of a directory (directory description file) and - if necessary – by recursive use the content of subdirectories. This signature was created with the

intarsys Code Signing Certificate during building process. Detected inconsistencies between the product's directory signature and the hash values published in the *Sign Live! CC Installation Verifier* are reported to the user.

2.5 Non TOE Components

Sign Live! CC also contains the following non TOE components which are not security relevant. They are mentioned to ease the understanding of the product Sign Live! CC.

2.5.1.1 security base

security base provides the following basic functions necessary for dealing with signatures:

- access to certificate stores
- access to OCSP services
- access to CRL stores
- access to the CABAReT license service to determine which function is licensed by the installation

The checking of the license is not security relevant. The results of the other functions are signed and therefore checkable by the TOE.

2.5.1.2 optional instruments

optional instruments stands for all components which offer functions to handle documents. These functions are not security relevant.

2.6 Operational Environment

2.6.1 GENERAL PLATTFORM

2.6.1.1 Hardware

Processor
Intel Pentium 1 GHz or CPU with equivalent processing speed

Available RAM
at least 256 MB, recommended 512 MB

Available Non-Volatile Storage Space
for Sign Live! CC: at least 60 MB, additionally 60 MB for unpacking of installation data
for JRE: at least 65 MB, additionally 65 MB for unpacking of installation data

Monitor
resolution at least 800 x 600, recommended 1024 x 768

2.6.1.2 Operating System

The system on which the TOE is running has to provide one of the following Operating Systems:

Operating System	Versions
Microsoft Windows	XP Home XP Professional XP TabletPC Edition

Table 4 - Required Operating Systems for the TOE

2.6.1.3 Java Virtual Machine

The TOE requires JVM 1.5.0 to be installed. The TOE is optionally delivered and installed together with the Java Virtual Machine (JVM), which was part of the testing. The JVM will be installed stand alone.

Otherwise the user may also use an already installed JVM of the same type.

Operating System	JVM
Microsoft Windows	SUN 1.5.0

Table 5 - Required Java Virtual Machines for the TOE

To run the Java Applet a browser is needed, which runs a JVM 1.5.0.

2.6.2 ADDITIONAL PLATTFORM

Sign Live! CC connects SSCDs to use their signature creation services in the following ways.

2.6.2.1 Connection to SSCD via Card Terminal

TOE connects the listed SSCDs via the listed card terminals with secure pin entry capabilities.

Following table lists the tested SSCDs:

SSCD	allowed modes
Giesecke & Devrient: STARCOS 3.0 with Electronic Signature Application V3.0 with Initializing Tables Signtrust Card 3.0 und Signtrust MCard 3.0	single + batch
Gemplus ZKA-Signaturkarte, Version 5.11	single
Gemplus ZKA-Signaturkarte, Version 5.11 M	batch

Table 6 – SSCDs tested for the TOE

‘single mode’ indicates that this SSCD is evaluated to use one pin request per signature. ‘batch mode’ indicates that this SSCD is evaluated to use one pin request to sign a finite set of documents.

Following table lists the tested CTs:

CT
KOBIL KAAN Advanced Firmware Version 1.02, Hardware Version K104R3
KOBIL SecOVID Reader III (not security accredited at creation time of this document)
OMNIKEY CardMan Trust CM3821, Firmware-Version 6.00
Reiner SCT cyberJack e-com, Version 2.0

Table 7 – CTs tested for the TOE

2.6.2.2 Connection to SSCD via Signtrust Signature Server 4.1

As the certifying process for this device has not been finished at *Sign Live! CC's* release date this method is not part of the security functions to be certified with this version of the TOE.

The TOE connects the SSCD via inter- or intranet by the security certified Signtrust Signature Server. The connectable smart cards are defined in the *Signtrust Signature Server* security certificate [SSSSC].

For signature creation Sign Live! CC calculates the document's hash value and transfers it via native call to the *clientAPI.dll* which is responsible for the communication to the Signature Server. Sign Live! CC stores the received PKCS#7 structure separately as file or integrates it into the PDF.

The trusted path to the Signature Server is built by the security certified third party component *clientAPI.dll*, which is integrated into *Sign Live! CC* as an external component. The correct installation of the component can be verified by *Sign Live! CC Installation Verifier.*

The *Signtrust Signature Server* security certificate [SSSSC] defines the requirements for the operational environment of the *clientAPI.dll*.

2.6.2.3 Connection to SSCD via timeproof Signature Server Version 4.00

As the certifying process for this device has not been finished at *Sign Live! CC's* release date this method is not part of the security functions to be certified with this version of the TOE.

The TOE connects the SSCD via intranet by the security certified TSS. The connectable smart cards are defined in the *TSS* security certificate [TSSSC].

For signature creation *Sign Live! CC* calculates the document's hash value. The *Sign Live! CC* component *timeproof library* transfers it via network to the security certified timeproof *Protokollserver*. *Sign Live! CC* stores the received PKCS#7 structure separately as file or integrates it into the PDF.

The *timeproof Signature Server* security certificate [TSSSC] defines the requirements for the operational environment of the *TSS*.

The trusted path between the *Sign Live! CC* component *timeproof library* and the certified timeproof *Protokollserver* is protected by adequate operational environment:

- The *Sign Live! CC* workstation is physically only accessible for authorized personal.
- TOE and TSS have to be installed and communicate only in the same physical environment.

Optionally the communication may be encrypted by a SSL-certificate, whose private key is protected by password which is available only to authorized personnel.

2.7 SigG/SigV Conformance

The producer of the TOE claims, that it is in conformance with

- German Signature Law [SigG] §17 (2) and
- German Ordinance on Electronic Signatures [SigV] §15 (2) and (4).

The following tables show the conformance of law/ordinance and assumptions/security functions, which are defined in chapter 3.1 Assumptions and in chapter 6.1 TOE Security Functions.

2.7.1 SIGNATURE LAW

SigG §17 (2)	**Assumptions/Security Function**
The presentation of data to be signed requires signature application components that will first clearly indicate the production of a qualified electronic signature and enable the data to which the signature refers to be identified.	SF.SelectedDocumentFalsificationPrevention SF.SignatureCreation
To check signed data, signature application components are needed that will show	-
1. To which data the signature refers	SF.OpenedDocumentFalsificationPrevention SF.SignatureValidation SF.DocumentPresentation
2. Whether the signed data are unchanged	SF.SignatureValidation SF.DocumentPresentation
3. To which signature code owner the signature is to be assigned	SF.SignatureValidation
4. The contents of the qualified certificate on which the signature is based and	SF.SignatureValidation
5. The results of the subsequent	SF.SignatureValidation

check of certificates under §5 (1) sentence 2.	
Signature application components shall, if necessary, also make the contents of the data to be signed or already signed sufficiently evident. The signature code owners should use these signature application components or take other suitable steps to secure qualified electronic signatures.	SF.DocumentPresentation

2.7.2 ORDINANCE ON ELECTRONIC SIGNATURES

SigV §15 (2)	**Security Function**
Signature application components pursuant to §17 (2) of the Signature Act must ensure that 1. when producing a qualified electronic signature a) the identification data are not disclosed and are stored only on the relevant SSCD, b) a signature is provided only at the initiation by the authorized signing person, c) the production of a signature is clearly indicated in advance […]	This requirements are basically fulfilled by the use of a smart card connected to the TOE via one of the following security certificated devices: - smart card terminal with secure pin entry mode (A.CT) - SSS (A.SSS) - TSS (A.TSS) SF.SelectedDocumentFalsificationPrevention SF.SignatureCreation
2. when verifying a qualified electronic signature a) the correctness of a signature is reliably verified and appropriately displayed and b) it can be clearly determined whether the verified qualified certificates were present in the relevant register of certificates at the given time and were not revoked.	SF.SignatureValidation

SigV §15 (4)	**Security Function**
Security-relevant changes in technical components pursuant to sections (1) to (3) must be apparent for the user.	Assumptions on the TOE security environment SF.TOEIntegrityChecking

3 TOE security environment

The statement of TOE security environment shall describe the security aspects of the environment in which the TOE is intended to be used and the manner in which it is expected to be employed.

3.1 Assumptions

3.1.1 GENERAL ASSUMPTIONS

A.PLATFORM	The used hard- and software components have to meet the requirements defined in chapter 2.6.1.
A.SSCD	Independent from the connected device at least one corresponding SSCD shall be available. The requirements concerning the environment for the used SSCD have to be respected and the corresponding guidance has to be followed.
A.NETWORK	The system on which the TOE is installed may have internet access. In this case a firewall and a virus scanner must be used to prevent compromising by internet attacks and to detect malicious programs installed on the system.
A.ACCESS	User has full control about inserted storage devices of the system, on which the TOE is installed. The TOE is protected in such a way that it is not possible to access parts of the TOE or its working directories[4] through existing network connections.
A.PERSONAL	Users and administrators are trustworthy and follow all user guidance. Especially the user verifies the TOE's integrity as described in the user guide.
A.NOGUI	In the case the TOE is operated without GUI, user should take sufficient precautions to protect the TOE's working directories against unauthorized manipulation, to be sure that the documents selected for the signature process won't be manipulated in the time between optional viewing the documents and starting the signature process.

[4] The TOE uses working directories to store documents to be signed and resulting documents.

3.1.2 ASSUMPTIONS FOR ADDITIONAL COMPONENTS

The TOE can be connected to a smart card via several additional devices. These are

- Card Terminal
- Signtrust Signature Server
- timeproof Signature Server

Depending on these connections different assumptions for the environment arise (in case more than one device is connected, the stronger requirements have to be fulfilled):

A.CT	At least one of the desired smart card terminals as defined in 2.6.2.1 shall be connected to the workstation.
A.SSS[5]	The desired SCA Signtrust Signature Server as defined in 2.6.2.2 is available via inter- or intranet. Following additional requirements for the environment have to be fulfilled as defined by the environment requirements of the Signature Server's *clientAPI.dll* as formulated in the [SSSSC] chapter 3.2.4 (2). The person who starts the signing process – the signature initiator - has authorization information in form of a password which only she knows. The SCA SSS is configured in the way that the signature initiator giving her password has access only to the SSCD(s), which sign in her name.
A.TSS[6]	The desired SCA timeproof Signature Server as defined in 2.6.2.3 is connected via secure network connection. Following additional requirements for the environment have to be fulfilled as defined by the environment requirements of the timeproof Signature Server as formulated in the [TSSSC] 3.2. These are the same as required by the German Signature Law for Certification Authorities. TOE and TSS have to be installed and communicate only in the same physical environment. The SCA TSS assures that the signature initiator has access only to SSCD(s), which sign in her name.

[5] As signature creation via Signtrust Signature Server is not part of the security functions to be certified with this but a following version of the TOE the corresponding assumption for the environment is still listed.

[6] As signature creation via timeproof Signature Server is not part of the security functions to be certified with this but a following version of the TOE the corresponding assumption for the environment is still listed.

3.2 Threats

User Data to be protected (assets) are

DOC:	document, not known to the TOE
DOC_SEL:	document selected by the TOE (*selected document*). The TOE has identified the document in the file system.
DOC_OPEN:	document opened by the TOE (*opened document*)
DOC_SIG:	signed document
TOE:	TOE itself

All threats assume an attacker with high attack potential.

Threats are defined according to the different lifecycle periods of the document:

T.DOC. AMBIGUOUS_CONT	Attacker manipulates a document in the way that it is shown ambiguously or contains insecure or hidden content that may show the document differently to different users.
T.DOC.MAN	Attacker manipulates a document before it is selected by the user for signing purposes and the manipulation is not detected.
T.DOC_SEL.MAN	Attacker manipulates a selected document or its representations and the manipulation is not detected.
T.DOC_OPEN.MAN	Attacker manipulates a document or its representations (e.g. as a hash value) during the document is opened by the TOE and the manipulation is not detected.
T.DOC_SIG.MAN	Attacker manipulates a signed document and the manipulation is not detected.
T.TOE.MAN	Attacker manipulates the TOE by changing parts of the TOE installed on the computer and the manipulation is not detected.

4 Security Objectives

The statement of security objectives shall define the security objectives for the TOE and its environment. The security objectives shall address all identified security environment aspects. The security objectives shall reflect the stated intent and shall be suitable to counter all identified threats and cover all identified organisational security policies and assumptions.

4.1 Security Objectives for the TOE

The security objectives for the TOE are derived from the threats identified for the TOE.

OT.DOC.DISPLAY	TOE must offer functions to show a document in an unambiguous way to each user and to detect insecure or hidden content that may show the document differently to different users.
OT.DOC_SEL.MAN	TOE must offer functions to identify manipulations of a selected document on base of hash value calculation.
OT.DOC_OPEN.MAN	TOE must offer functions either to work on an opened document in a way that the document's representations can only accessed by the TOE or to identify manipulations of an opened document or its representation, e. g. as a hash value.
OT.DOC_SIG.MAN	TOE must offer functions to identify manipulations of a signed document.
OT.TOE.MAN	TOE must offer functions to identify manipulations of the TOE or parts of it.

4.2 Security Objectives for the Environment

The security objectives for the environment are derived from the assumptions on a secure usage of the TOE.

4.2.1 GENERAL SECURITY OBJECTIVES

OE.PLATFORM	The used hard- and software components have to meet the requirements defined in chapter 2.6.1.
OE.SSCD	Independent from the connected device at least one corresponding SSCD shall be available to create an electronic signature from a hash value. The requirements concerning the environment for the used SSCD have to be respected and the corresponding guidance has to be followed.
OE.NETWORK	The system on which the TOE is installed may have internet access. In this case a firewall and a virus scanner must be used to prevent compromising by internet attacks and to detect malicious programs installed on the system.
OE.ACCESS	User must have full control about inserted storage devices of the system, on which the TOE is installed. The TOE has to be protected in such a way that it is not possible to access parts of the TOE or its working directories[7] through existing network connections.
OE.PERSONAL	Users and administrators have to be trustworthy and must follow all user guidance. Especially the user has to verify the TOE's integrity as described in the user guide.
OE.NOGUI	In case the TOE is operated without GUI, user must take sufficient precautions to protect the TOE's working directories against unauthorized manipulation, to be sure that the documents selected for the signature process won't be manipulated in the time between optional viewing the documents and starting the signature process.

[7] The TOE uses working directories to store documents to be signed and resulting documents.

4.2.2 SECURITY OBJECTIVES FOR ADDITIONAL COMPONENTS

The TOE can be connected to a smart card via several additional devices. These are

- Card Terminal
- Signtrust Signature Server
- timeproof Signature Server

Depending on these devices different security objectives for the environment arise (in case more than one device is connected, the stronger objectives have to be fulfilled):

OE.CT	At least one of the desired smart card terminals as defined in 2.6.2.1 shall be connected to the workstation.
OE.SSS[8]	The desired SCA SSS as defined in 2.6.2.2 is available via inter- or intranet. Following additional requirements for the environment have to be fulfilled as defined by the environment requirements of the Signature Server's *clientAPI.dll* as formulated in the [SSSSC] chapter 3.2.4 (2). The person who starts the signing process – the signature initiator – must have authorization information in form of a password which only she knows. The SCA SSS is configured in the way that the signature initiator giving her password has access only to the SSCD(s), which sign in her name.
OE.TSS[9]	The desired SCA TSS as defined in 2.6.2.3 is connected via secure network connection. Following additional requirements for the environment have to be fulfilled as defined by the environment requirements of the timeproof Signature Server as formulated in the [TSSSC] 3.2. These are the same as required by the German Signature Law for Certification Authorities. TOE and TSS have to be installed and communicate only in the same physical environment. The SCA TSS assures that the signature initiator has access only to SSCD(s), which sign in her name.

[8] As signature creation via Signtrust Signature Server is not part of the security functions to be certified with this but a following version of the TOE the corresponding security objective is still listed.

[9] As signature creation via timeproof Signature Server is not part of the security functions to be certified with this but a following version of the TOE the corresponding security objective is still listed.

5 IT Security Requirements

This part of the ST defines the detailed IT security requirements that shall be satisfied by the TOE or its environment.

5.1 TOE Security Functional Requirements

With the exception of FDP_TVR – Trusted Viewer all components are drawn from Part 2 of the CC. The new functional family is defined in chapter 9 Definition of Functional Family FDP_TVR.

5.1.1 FAMILY FCS_COP - CRYPTOGRAPHIC OPERATION

5.1.1.1 FCS_COP.1 (SHA-1)

FCS_COP.1.1

The TSF shall perform [assignment: ***computation of hash values***] in accordance with a specified cryptographic algorithm [assignment: ***SHA-1***] and cryptographic key sizes [assignment: none] that meet the following: [assignment: ***[FIPS 180-2]***].

5.1.1.2 FCS_COP.1 (SHA-224)

FCS_COP.1.1

The TSF shall perform [assignment: ***computation of hash values***] in accordance with a specified cryptographic algorithm [assignment: ***SHA-224***] and cryptographic key sizes [assignment: none] that meet the following: [assignment: ***[FIPS 180-2]***].

5.1.1.3 FCS_COP.1 (SHA-256)

FCS_COP.1.1

The TSF shall perform [assignment: ***computation of hash values***] in accordance with a specified cryptographic algorithm [assignment: ***SHA-256***] and cryptographic key sizes [assignment: none] that meet the following: [assignment: ***[FIPS 180-2]***].

5.1.1.4 FCS_COP.1 (SHA-384)

FCS_COP.1.1

The TSF shall perform [assignment: ***computation of hash values***] in accordance with a specified cryptographic algorithm [assignment: ***SHA-384***] and cryptographic key sizes [assignment: none] that meet the following: [assignment: ***[FIPS 180-2]***].

5.1.1.5 FCS_COP.1 (SHA-512)

FCS_COP.1.1

The TSF shall perform [assignment: ***computation of hash values***] in accordance with a specified cryptographic algorithm [assignment: ***SHA-512***] and cryptographic key sizes [assignment: none] that meet the following: [assignment: ***[FIPS 180-2]***].

5.1.1.6 FCS_COP.1 (RIPEMD-160)

FCS_COP.1.1

The TSF shall perform [assignment: ***computation of hash values***] in accordance with a specified cryptographic algorithm [assignment: ***RIPEMD-160***] and cryptographic key sizes [assignment: none] that meet the following: [assignment: ***[RIPEMD-160]***].

5.1.1.7 FCS_COP.1 (Validation1-RSA)

FCS_COP.1.1

The TSF shall perform [assignment: ***validation of electronic signatures***] in accordance with a specified cryptographic algorithm [assignment: ***RSA***] and cryptographic key sizes [assignment: ***1024-2048 bit***] that meet the following: [assignment: ***PKCS#1, ISO 9796-2***].

Application Note

Used to validate the consistency of the document signed by the TOE.

5.1.1.8 FCS_COP.1 (Validation2-RSA)

FCS_COP.1.1

The TSF shall perform [assignment: ***validation of electronic signatures***] in accordance with a specified cryptographic algorithm [assignment: ***RSA***] and cryptographic key sizes [assignment: ***1024-2048 bit***] that meet the following: [assignment: ***PKCS#1, ISO 9796-2***].

Refinement 1:

If the validation model of the CA is based on the chain model, TSF must verify the certificate chain using the chain model according to [ISISMTT].

Refinement 2:

If the validation model of the CA is not based on the chain model, TSF must verify the certificate chain using standard [RFC3280].

Application Note

Used to validate the consistency and authenticity of a signed document.

5.1.2 FAMILY FDP_DAU – DATA AUTHENTICATION

5.1.2.1 FDP_DAU.2 Data Authentication with Identity of Guarantor

FDP_DAU.2.1

The TSF shall provide a capability to generate evidence that can be used as a guarantee of the validity of [assignment: ***a file chosen by the user***].

FDP_DAU.2.2

The TSF shall provide [assignment: ***the user***] with the ability to verify evidence of the validity of the indicated information and the identity of the user that generated the evidence.

5.1.3 FAMILY FDP_ITC – IMPORT FROM OUTSIDE TSF CONTROL

5.1.3.1 FDP_ITC.1 (Signature Creation) – Import User Data Without Security Attributes

FDP_ITC.1.1

The TSF shall enforce the [assignment: none] when importing user data, controlled under the SFP, from outside of the TSC.

FDP_ITC.1.2

The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC.

FDP_ITC.1.3

The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: [assignment: ***Usage of Certificates After Signature Creation***].

Rule: Usage of certificates after signature creation

TSF must check the created electronic signature. TSF decrypts the electronic signature with the public key of the contained certificate and compares the result with the hash value used to create the electronic signature.

5.1.3.2 FDP_ITC.1 (Signature Validation) – Import User Data Without Security Attributes

FDP_ITC.1.1

The TSF shall enforce the [assignment: none] when importing user data, controlled under the SFP, from outside of the TSC.

FDP_ITC.1.2

The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC.

FDP_ITC.1.3

The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: [assignment: ***Usage of Network Certificates***].

Rule: Usage of Network Certificates

For signature validation TSF must consider available revocation lists to verify the certificate's validity at signature's creation time.

5.1.3.3 FDP_ITC.2 (OCSP Responses) – Import User Data With Security Attributes

FDP_ITC.2.1

The TSF shall enforce the [assignment: none] when importing user data, controlled under the SFP, from outside of the TSC.

FDP_ITC.2.2

The TSF shall use the security attributes associated with the user data when imported from outside the TSC.

FDP_ITC 2.3

The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC.2.4

The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5

The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: [assignment: ***Importing of OCSP Responses***].

Rule: Importing of OCSP responses

Before importing an OCSP response TSF must verify the validity of its signature.

5.1.4 FAMILY FDP_TVR – TRUSTED VIEWER

5.1.4.1 FDP_TVR.1 - Trusted Viewer

FDP_TVR.1.1

The TSF shall ensure, that the displayed content of a document is unambiguous according to [assignment: ***PDF Subset as defined in Sign Live! CC User Guidance [SLUG], TIFF as defined in [TIFF], Text***].

FDP_TVR.1.2

The TSF shall ensure, that the document is free of active and hidden content.

FDP_TVR.1.3

The TSF shall ensure, that user is informed about content that cannot be displayed due to the capabilities of the TSF.

5.1.5 FAMILY FTP_ITC – INTER-TSF TRUSTED CHANNEL

5.1.5.1 FTP_ITC.1 (Smart Card) – Import User Data Without Security Attributes

FTP_ITC.1.1

The TSF shall provide a communication channel between itself and a remote trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

FTP_ITC.1.2

The TSF shall permit [selection: ***the TSF***] to initiate communication via the trusted channel.

FTP_ITC.1.3

The TSF shall initiate communication via the trusted channel for [assignment: ***creation of an electronic signature***].

5.2 TOE Security Assurance Requirements

The assurance requirements are defined according to EAL3 augmented (Common Criteria part 3). All components are drawn from Common Criteria Part 3.

The following table shows how the assurance requirements are met by documents delivered for evaluation:

Class	**Family**	**Document**
Configuration Management	ACM_CAP.3	Process Guide
	ACM_SCP.1	
Delivery and Operation	ADO_DEL.2	Delivery Guide
	ADO_IGS.1	TOE's Online Documentation
Development	ADV_FSP.1	Functional Specification
	ADV_HLD.2	High-Level Design
	ADV_IMP.1	relevant source code is available for evaluation
	ADV_LLD.1	Low-Level Design
	ADV_RCR.1	handled in Specification and Design documents
Guidance Documents	AGD_ADM.1	TOE's Online Documentation
	AGD_USR.1	

Class	Family	Document
Life Cycle Support	ALC_DVS.1	Security Guide (Sicherheitshandbuch)
	ALC_TAT.1	Process Guide
Tests	ATE_COV.2	Testing Guide
	ATE_DPT.1	
	ATE_FUN.1	
	ATE_IND.2	evaluator's task
Vulnerability Assessment	AVA_MSU.3	the documentation is internally reviewed for clearness, consistence and reasonability
	AVA_VLA.4	analysis is done and documented in the vulnerability assessment
	AVA_SOF.1	analysis is done and documented in the vulnerability assessment

Table 8 - Security Assurance Requirements and corresponding documents

5.3 Security Requirements for the IT Environment

5.3.1 FAMILY FCS_COP - CRYPTOGRAPHIC OPERATION

5.3.1.1 FCS_COP.1 (Signing – RSA)

FCS_COP.1.1

The TSF shall perform [assignment: ***computation of electronic signatures***] in accordance with a specified cryptographic algorithm [assignment: ***RSA***] and cryptographic key sizes [assignment: ***1024-2048***] that meet the following: [assignment: ***PKCS#1, ISO 9796-2***]

6 TOE Summary Specification

The TOE summary specification shall define the instantiation of the security requirements for the TOE. This specification shall provide a description of the security functions and assurance measures of the TOE that meet the TOE security requirements.

6.1 TOE Security Functions

6.1.1 SF.OpenedDocumentFalsificationPrevention

When user opens a document (declares it as an *opened document*), the whole document is loaded into memory. All document accesses and changes are performed in memory. If the available memory is not sufficient, TOE informs the user.

When user starts a signing process

1. TOE calculates the document's *signature* hash value according to (FCS_COP.1 (SHA-1), FCS_COP.1(SHA-224), FCS_COP.1 (SHA-256), FCS_COP.1 (SHA-384), FCS_COP.1 (SHA-512), FCS_COP.1 (RIPEMD-160)) on base of the document in memory.
2. TOE transfers the *signature* hash value to the SSCD to generate the electronic signature (FTP_ITC.1 (Smart Card)).
3. TOE validates the signature received from the SSCD. If the signature is not correct, either the document was corrupted during signing process or the signature's *signature* hash value was corrupted during transmitting process (FTP_ITC.1 (Smart Card), FDP_ITC.1 (SignatureCreation), FCS_COP.1 (Validation1 RSA)).

Strength of Function

The cryptographic nature of the used mechanism entails that no comment about their strength can be given. The used cryptographic mechanisms are identified as usable according to [SAlg].

6.1.2 SF.SelectedDocumentFalsificationPrevention

When user selects a document for batch processing (declares it as a *selected document*), TOE calculates the corresponding *integrity* hash value according to (FCS_COP.1 (SHA-1)) and stores it in memory as long as the document is selected.

When TOE opens the document, TOE recalculates the *integrity* hash value and compares it with the stored *integrity* hash value. If the two hash values are not identical, the document was manipulated and the TOE informs the user.

Otherwise TOE uses the opened document for further processes as described in 6.1.1 SF.OpenedDocumentFalsificationPrevention.

Strength of Function

The cryptographic nature of the used mechanism entails that no comment about their strength can be given. The used cryptographic mechanisms are identified as usable according to [SAlg].

6.1.3 SF.SIGNATURECREATION

Single Signature

User opens one document, chooses a signing method and asks the TOE to start the signing process. Hereby the document is declared as *document selected to be signed*. The hash value of the *document selected to be signed* – the original hash value – is calculated using one of the following algorithms (FCS_COP.1 (SHA-1), FCS_COP.1(SHA-224), FCS_COP.1 (SHA-256), FCS_COP.1 (SHA-384), FCS_COP.1 (SHA-512), FCS_COP.1 (RIPEMD-160)).

TOE informs the user unambiguously that an electronic signing process is beginning and – if desired by the user – presents the document in a Trusted Viewer, so that it is clear to which data the signature will belong. User may verify the document for hidden and not displayable content. Finally user requests the signature.

TOE transmits the original hash value via a SSCD connection library to initiate the electronic signature creation process on the smart card. The electronic signature is calculated outside the TOE on the SSCD using the RSA algorithm (FCS_COP.1 (Signing – RSA)).

TOE fetches the signer's certificate data from the smart card and adds it to the signature.

Selecting the Certificate for Signing

The selection of the certificate used for signing depends on the connected device:

CT: In case more than one CT is connected, TOE offers the option to select the CT for signing. The TOE shows all certificates stored on the SSCD connected to the selected CT to the user. The TOE indicates clearly qualified certificates. The user chooses one certificate for signing.

Batch Signature

Optionally user selects a set of documents for signing and may inspect each document individually in the Trusted Viewer.

Afterwards the user chooses the signing method and asks the TOE to start the signing process. TOE informs the user unambiguously that an electronic signing process is starting now. TOE interacts with a connected SSCD by requesting one signature for each selected document as described for a single signature.

The TOE calculates the number of documents to be signed and requests exactly this number of signatures.

If the SSCD is connected via CT and this combination is evaluated to be used in batch mode, the TOE initiates an exclusive connection to the SSCD, requests the SSCD pin only once and closes this connection after receiving the last signature or when user stops the process. The pin must be entered on the CT.

Process stops by user interaction or when the last document is processed.

Call via Operating System

Optionally user may start the signing process for one or a set of documents via operating system call without any further interaction (in case the SSCD is connected via CT, the pin has to be entered on the CT).

Strength of Function

The cryptographic nature of the used mechanism entails that no comment about their strength can be given. The used cryptographic mechanisms are identified as usable according to [SAlg].

6.1.4 SF.SIGNATUREVALIDATION

The TOE verifies electronically signed documents, i.e. it validates the document's signature, constructs a corresponding certificate chain and validates the certificate chain's signature(s). Additionally all certificates are checked for revocation information.

The validation algorithm handles OCSP responses and timestamps which are eventually contained in the signature.

Validation of Document's Signature

For this the TOE

1. extracts and decrypts the hash value from the document's signature using the RSA algorithm (FCS_COP.1 (Validation2 RSA)) and the public key of the given signer certificate.
2. calculates the hash value of the signed document using the same algorithm which was used for the hash value contained in the signature (FCS_COP.1 (SHA-1), FCS_COP.1(SHA-224), FCS_COP.1 (SHA-256), FCS_COP.1 (SHA-384), FCS_COP.1 (SHA-512), FCS_COP.1 (RIPEMD-160)).
3. compares the two hash values[10]

If both hash values are identical it is unambiguous that the signature is correct and the document is not manipulated. The TOE informs the user with an unambiguous message (FDP_DAU.2, FDP_DAU.2.2). User may view the selected document unambiguously in the Trusted Viewer.

Validation of Certificate Chain's Signatures

TOE uses the same process as for the document's signature to check each certificate of the certificate chain according to the chain model or RFC 3280 (FCS_COP.1 (Validation2 RSA)).

If all pairs of hash values are identical it is unambiguous that the certificates are correct. To identify the user who created the signature the TOE can display the signer's certificate in an unambiguous way. So the requirements FDP_DAU.2, FDP_DAU.2.2 are fulfilled.

[10] In this process the padding is checked according to EMSA-PKCS1-v1_5 [PKCS#1] or [ISO 9796-2].

CRL

If available, TOE checks the status of a certificate by a certificate revocation list (CRL) of the corresponding certificate issuer. TOE checks whether an entry for the signer exists and whether this entry existed already at creation time of the signature and displays an appropriate message (FDP_ITC.1 (SignatureValidation)).

OCSP

If available, TOE checks the status of a certificate by a given OCSP response. OCSP response may be included in the signed document or delivered by an OCSP handler. For details see security function SF.OCSPProcessing. TOE informs the user by an appropriate message.

Timestamp

If a timestamp is contained in the signature to validate, TOE checks its status. For details see security function SF.TimestampValidation. TOE informs the user by an appropriate message.

Determining of Certificate Creation Time

As signature creation time TOE uses information as available in the following order:

1. the time contained in the signature's timestamp
2. time of signature creation as documented in the document respectively in the signature
3. current system time

Batch Validation

Optionally user can select a set of documents to be validated. TOE informs the user unambiguously about the comprised result for each document. TOE documents the comprised result in the GUI by an icon per document and – optionally – by the storing place of the validated document ('good' or 'bad' directory). Optionally TOE presents the comprised result by a protocol per document or per batch. Additionally user may view the detailed results for each document individually via GUI if desired.

Call via Operating System

Optionally user may start the validating process for one or a set of documents via operating system call without any further interaction. In this case the result is only available via storing place and protocols.

Strength of Function

The cryptographic nature of the used mechanism entails that no comment about their strength can be given. The used cryptographic mechanisms are identified as usable according to [SAlg].

6.1.5 SF.OCSPPROCESSING

To validate a *certificate* TOE is able to interpret the information contained in an OCSP response. TOE may receive the OCSP response corresponding to the *certificate to be validated* via the following ways:

- OCSP response is contained in a signature
- OCSP response is delivered by an OCSP handler

OCSP response is always delivered in a signed way, i.e. consists of data, signature and certificate. TOE ensures the correctness of the response by validating the OCSP response's signature and certificate to fulfil (FDP_ITC.2 (OCSP Response)).

Validation of the OCSP Response's Signature

For this the TOE

1. extracts and decrypts the hash value from the OCSP response's signature using the RSA algorithm (FCS_COP.1 (Validation1 RSA)) and the public key of the given signer certificate.
2. calculates the hash value of the signed OCSP response using the same algorithm which was used for the hash value contained in the signature (FCS_COP.1 (SHA-1), FCS_COP.1(SHA-224), FCS_COP.1 (SHA-256), FCS_COP.1 (SHA-384), FCS_COP.1 (SHA-512), FCS_COP.1 (RIPEMD-160)).
3. compares the two hash values[11].

If both hash values are identical it is unambiguous that the OCSP response is correct and was not manipulated.

Validation of the OCSP Response's Certificate

TOE uses the same process as for the OCSP response's signature to check the OCSP response's certificate and the corresponding certificate chain according to the chain model or RFC 3280 (FCS_COP.1 (Validation2 RSA)).

Additionally all certificates are checked for revocation information that is available by appropriate CRLs (FDP_ITC.1 (Signature Validation)).

Usage of the OCSP Response

TOE is able to display unambiguously the following information of an OCSP response:

- the information that the OCSP response is valid
- the validity information contained in the OCSP response
- certificate information of the OCSP response's signature (FDP_DAU.2 with focus on FDP_DAU.2.2).

The validity information may be used in the process of signature validation.

[11] In this process the padding is checked according to EMSA-PKCS1-v1_5 [PKCS#1].

The OCSP response may be used in the process of signature creation by being added to the signature.

Strength of Function

The cryptographic nature of the used mechanism entails that no comment about their strength can be given. The used cryptographic mechanisms are identified as usable according to [SAlg].

6.1.6 SF.TIMESTAMPVALIDATION

TOE is able to verify timestamps given in a signature. For this the timestamp creator's certificate must be known to the TOE either by a certificate store the TOE has access to or because it was delivered together with the timestamp. Otherwise the timestamp will not be validated.

To verify a timestamp TOE verifies the signature contained in the timestamp and the certificate chain corresponding to the timestamp creator's certificate.

For this the TOE

1. calculates the hash value of the timestamped data using one of the following algorithms: (FCS_COP.1 (SHA-1), FCS_COP.1(SHA-224), FCS_COP.1 (SHA-256), FCS_COP.1 (SHA-384), FCS_COP.1 (SHA-512), FCS_COP.1 (RIPEMD-160)).
2. extracts the hash value from the timestamp's signature using the RSA algorithm (FCS_COP.1 (Validation2 RSA)) and the public key of the timestamp service's certificate.
3. compares the two hash values[12].

Afterwards TOE uses the same process to check the certificate chain corresponding to the timestamp signer's certificate using the chain model or RFC 3280. All certificates in the chain are checked for revocations (FDP_ITC.1 (Signature Validation)).

If all pairs of hash values are identical it is unambiguous that the timestamp is reliable.

TOE displays the following information to the user:

1. the date/time indication itself
2. the information whether the timestamp is correct or not
3. the certificate information of the timestamp's signer (FDP_DAU.2 with focus on FDP_DAU.2.2).

Strength of Function

The cryptographic nature of the used mechanism entails that no comment about their strength can be given. The used cryptographic mechanisms are identified as usable according to [SAlg].

[12] In this process the padding is checked according to EMSA-PKCS1-v1_5 [PKCS#1] or [ISO 9796-2].

6.1.7 SF.DOCUMENTPRESENTATION

The TOE ensures the unambiguous presentation of a document to the user (FDP_TVR.1.1).

TOE will inform the user by an unambiguous message, if one of the following conditions is met:

- TOE detects an unknown document type
- TOE detects unsupported content (FDP_TVR.1.3)

TOE offers the possibility to report hidden, active content (FDP_TVR.1.2).

Documents that can be presented must satisfy one of the following formats:

- PDF as defined in the user guidance
- TEXT as defined in the user guidance
- TIFF as defined in the user guidance

Strength of Function

No permutational or probabilistic mechanism is contained in this SF. Therefore no strength of function is postulated.

6.1.8 SF.TOEINTEGRITYCHECKING

Integrity of the TOE means that the correct components are active and their function is not manipulated by other components/configurations.

TOE assures its integrity by the following mechanisms:

Identification of the TOE

User may view component information which indicates the TOE's name, release, certification and confirmation ID.

Verification of the TOE's configuration (*Trusted Mode*)

The active Trusted Mode indicates that the TOE is correctly configured to create and validate qualified signatures. Following configurations are checked by the Trusted Mode:

- configuration for validation is either SigG or EU
- Java Script Live Connect Feature (Java API available in JavaScript) is deactivated for document scripting

TOE indicates the Trusted Mode by an icon in the status bar and in the application log.

Verification of the TOE's Directory Signature (Installation Verifier)

The TOE installed on the computer, is signed by a hierarchical structure of directory signatures in the way that one hash value identifies all components of one product. This structure was created during the build process with the intarsys Code Signing Private Key issued by trusted CA (Thawte).

Sign Live! CC Installation Verifier is a TOE component to verify the directory signatures. For this purpose it contains the intarsys Code Signing Certificate and hash values of valid products. It is available on the intarsys homepage. *Sign Live! CC Installation Verifier* is realized as a Java Applet, also signed with the above mentioned intarsys Code Signing Certificate, in the way that standard browsers can verify the authenticity and consistency of the applet.

One directory signature consists of the following files:

- **directory description** *(component.desc)* – the definition of the content allowed in the described directory as concrete file/directory reference and corresponding hash value or as rule
- **directory hash values** *(component.sf)* – the hash values of the elements contained in the directory description
- **directory signature** *(component sf.p7s)* – the signature of *component.sf* created with the intarsys Code Signing Certificate

To verify the consistency of the installed TOE, *Sign Live! CC Installation Verifier* performs the following steps for each directory signature:

A. Validation of the directory signature

1. It validates the signer certificate by validating its certificate path with the certificates delivered with the *Sign Live! CC Installation Verifier*. The certificate used for signing must be delivered with *Sign Live! CC Installation Verifier.*
2. It extracts and decrypts the hash value from the *directory signature* using the RSA algorithm (FCS_COP.1 (Validation1 RSA)) and the public key of the given signer certificate.
3. It calculates the hash value of the signed directory hash values using the same algorithm which was used for the hash value contained in the signature (FCS_COP.1 (SHA-1), FCS_COP.1(SHA-224), FCS_COP.1 (SHA-256), FCS_COP.1 (SHA-384), FCS_COP.1 (SHA-512), FCS_COP.1 (RIPEMD-160)).
4. compares the two hash values[13].
5. If the top level directory signature is examined, the calculated hash value is additionally compared with the hash values of valid products, which are delivered with the *Sign Live! CC Installation Verifier*.

B. Validation of the directory hash values

For each hash value entry in directory hash values

1. It calculates the hash value of the corresponding directory description element (FCS_COP.1 (SHA-1), FCS_COP.1(SHA-224), FCS_COP.1 (SHA-256), FCS_COP.1 (SHA-384), FCS_COP.1 (SHA-512), FCS_COP.1 (RIPEMD-160))
2. and compares the two hash values[14]

[13] In this process the padding is checked according to EMSA-PKCS1-v1_5 [PKCS#1] or [ISO 9796-2].

[14] In this process the padding is checked according to EMSA-PKCS1-v1_5 [PKCS#1] or [ISO 9796-2].

C. Validation of the directory description

For each description entry in directory description

1. TOE interprets the directory description rule,
2. calculates the hash value of a file/directory description (FCS_COP.1 (SHA-1), FCS_COP.1(SHA-224), FCS_COP.1 (SHA-256), FCS_COP.1 (SHA-384), FCS_COP.1 (SHA-512), FCS_COP.1 (RIPEMD-160)) and
3. compares it with the corresponding hash value given in the rule[15].

TOE informs the user, if one of the following conditions is met:

- the certificate is not known to the Verifiers certificate store
- a hash value comparison fails
- a rule interpretation fails

Otherwise the TOE is correct and was not manipulated.

Strength of Function

The cryptographic nature of the used mechanism entails that no comment about their strength can be given. The used cryptographic mechanisms are identified as usable according to [SAlg].

6.2 Assurance Measures

Table Security Assurance Requirements and corresponding documents p. 27 lists all documents which describe the assurance measures.

[15] In this process the padding is checked according to EMSA-PKCS1-v1_5 [PKCS#1] or [ISO 9796-2].

7 PP Claims

There are no Protection Profile Claims.

8 Rationale

This part of the ST presents the evidence used in the ST evaluation. This evidence supports the claims that the ST is a complete and cohesive set of requirements, that a conformant TOE would provide an effective set of IT security countermeasures within the security environment, and that the TOE summary specification addresses the requirements.

8.1 Security Objectives Rationale

The purpose of this rationale is to demonstrate that the identified security objectives are suitable and necessary.

8.1.1 SECURITY OBJECTIVES COVERAGE

Following table shows that all threats and assumptions are addressed and that no objective is redundant.

Objectives Threads/ Assumptions	OT.DOC.DISPLAY	OT.DOC_SEL.MAN	OT.DOC_OPEN.MAN	OT.DOC_SIG. MAN	OT.TOE. MAN	OE.PLATFORM	OE.SSCD	OE.NETWORK	OE.ACCESS	OE.PERSONAL	OE.NOGUI	OE.CT
T.DOC.AMBI-GUOUS_CONT	x											
T.DOC.MAN	x										x	
T.DOC_SEL. MAN		x										
T.DOC_OPEN. MAN			x									
T.DOC_SIG. MAN				x								
T.TOE.MAN					x							
A.PLATFORM						x						
A.SSCD							x					
A.NETWORK								x				
A.ACCESS									x			
A.PERSONAL										x		
A.NOGUI											x	
A.CT												x

Table 9 – Threats and Assumptions vs. Objectives

8.1.2 SECURITY OBJECTIVES SUFFICIENCY

The purpose of this rationale is to demonstrate that all identified threats are countered and all assumptions are properly addressed.

T.DOC.AMBIGUOUS_CONT addresses the threat, that a document may be manipulated in a way that it will be presented ambiguously or contains insecure or hidden content, which may show the document differently to different users. OT.DOC.DISPLAY ensures that the TOE provides functions that will display the document in the same unambiguous way to each user and detects hidden or insecure contents. Therefore the security objective counters the threat completely.

T.DOC.MAN addresses the threat that a document, that may eventually be selected for signature processes, may be manipulated. OT.DOC.DISPLAY ensures that the TOE provides functions that will display the document in the same unambiguous way to each user and detect insecure or hidden content before it will be signed. Therefore the security objective counters the threat completely, if the TOE is operated with GUI.
In case the TOE is operated without GUI, the user should assure herself that the selected documents stored in the TOE's working directory are unambiguous and do not contain any hidden content. The security objectives for the TOE's environment assure that the documents proved in that way will not be altered by unauthorized user (OE.NOGUI).

T.DOC_SEL.MAN addresses the threat that a selected document or its representations may be manipulated in the time the document is identified by the TOE. OT.DOC.SELMAN ensures that manipulation of the selected document can be uniquely identified on base of hash value calculation. In this manner the security objectives counter the threat completely.

T.DOC_OPEN.MAN addresses the threat that an opened document or its representations may be manipulated in the time the document is under control of the TOE. OT.DOC_OPEN.MAN ensures either that the opened document can only be accessed by the TOE in the time it is under control of the TOE or that the TOE offers functions to detect manipulations of an opened document. In this manner the security objectives counter the threat completely.

T.DOC_SIG.MAN addresses the threat that a signed document is manipulated and the manipulation is not detected. OT.DOC_SIG.MAN ensures that the TOE will provide functions to detect manipulations of a signed file. Therefore the security objective counters the threat completely.

T.TOE.MAN addresses the threat that a TOE might be manipulated and the manipulation is not detected. OT.TOE.MAN ensures that the TOE will provide functions to detect changes of the installation. Therefore the security objective counters the threat completely.

A.PLATFORM is completely addressed by objective OE.PLATFORM because OE.PLATFORM mandates what A.PLATFORM specifies.

A.SSCD is completely addressed by objective OE.SSCD because OE.SSCD mandates what A.SSCD specifies.

A.NETWORK is completely addressed by objective OE. NETWORK because OE.NETWORK mandates what A.NETWORK specifies.

A.ACCESS is completely addressed by objective OE.ACCESS because OE.ACCESS mandates what A.ACCESS specifies.

A.PERSONAL is completely addressed by objective OE.PERSONAL because OE.PERSONAL mandates what A.PERSONAL specifies.

A.NOGUI is completely addressed by objective OE.NOGUI because OE.NOGUI mandates what A.NOGUI specifies.

A.CT is completely addressed by objective OE.CT because OE.CT mandates what A.CT specifies.

8.2 Security Requirements Rationale

The purpose of this rationale is to demonstrate that the identified security requirements are suitable and necessary to meet the security objectives.

8.2.1 TRACEABILITY OF FUNCTIONAL REQUIREMENTS

Following table shows that all objectives are addressed and that no security objectives are redundant.

Objectives / Functional Security Requirements	OT.DOC.DISPLAY	OT.DOC_SEL.MAN	OT.DOC_OPEN.MAN	OT.DOC_SIG. MAN	OT.TOE. MAN	OE.CT
FCS_COP.1 (SHA-1)		x	x	x	x	
FCS_COP.1 (SHA-224)			x	x	x	
FCS_COP.1 (SHA-256)			x	x	x	
FCS_COP.1 (SHA-384)			x	x	x	
FCS_COP.1 (SHA-512)			x	x	x	
FCS_COP.1 (RIPEMD-160)			x	x	x	
FCS_COP.1 (Validation1 RSA)			x		x	
FCS_COP.1 (Validation2 RSA)				x		
FCS_COP.1 (Signing – RSA)						x
FDP_DAU.2				x		
FDP_ITC.1 (Signature Creation)			x			
FDP_ITC.1 (Signature Validation)				x		
FDP_ITC.2 (OCSP Responses)				x		
FDP_TVR.1	x					
FTP_ITC.1 (Smart Card)			x			

Table 10 – Functional Security Requirements vs. Objectives

Security requirement FCS_COP.1 (Signing – RSA) for the environment traces back to OE.CT. It is a requirement which has to be resolved by the IT environment of the TOE.

8.2.2 FUNCTIONAL REQUIREMENTS SUFFICIENCY

The purpose of this rationale is to demonstrate that the requirements are adequate to meet all security objectives.

OT.DOC.DISPLAY ensures that the TOE offers functions to the user to guarantee an unambiguous presentation of a document. This objective is fulfilled by FDP_TVR.1.

OT.DOC_SEL.MAN ensures that the user has the possibility to detect manipulation of a selected document. This objective is fulfilled by FCS_COP.1(SHA-1).

OT.DOC_OPEN.MAN ensures either that the opened document can only be accessed by the TOE in the time it is under control of the TOE or that the TOE offers functions to detect manipulations of an opened document. This objective is fulfilled by FTP_ITC.1 (Smart Card). FCS_COP.1(SHA-1), FCS_COP.1(SHA-224), FCS_COP.1(SHA-256), FCS_COP.1(SHA-384), FCS_COP.1(SHA-512), FCS_COP.1(RIPEMD-160), FCS_COP.1 (Validation1 RSA), FDP_ITC.1 (SIGNATURE CREATION) define the requirements of the additional cryptographic operations.

OT.DOC_SIG.MAN ensures that the user has the possibility to detect manipulation of a signed document. This objective is fulfilled by FDP_DAU.2, especially FDP_DAU.2.2. FCS_COP.1(SHA-1), FCS_COP.1(SHA-224), FCS_COP.1(SHA-256), FCS_COP.1(SHA-384), FCS_COP.1(SHA-512), FCS_COP.1(RIPEMD-160), FCS_COP.1 (Validation2 RSA), FDP_ITC.1 (Signature Validation), FDP_ITC.2 (OCSP Responses) define the requirements of the additional cryptographic operations.

OT.TOE.MAN ensures that the user has the possibility to detect manipulation of the TOE installed on the computer. This objective is fulfilled by FCS_COP.1(SHA-1), FCS_COP.1(SHA-224), FCS_COP.1(SHA-256), FCS_COP.1(SHA-384), FCS_COP.1(SHA-512) , FCS_COP.1(RIPEMD-160) and FCS_COP.1 (Validation1 RSA).

OE.CT ensure by providing a SSCD that the environment offers functions to calculate a signature. This objective is partly fulfilled by FCS_COP.1(Signing – RSA). Due to the fact that other objectives are not expressed in a functional way further appropriate functional requirements cannot be defined.

8.2.3 RATIONALE FOR ASSURANCE REQUIREMENTS

A signature application component according to §17 paragraph 2 of the German Signature Law (SigG) requires for an evaluation with Common Criteria the evaluation assurance level EAL 3 augmented. The Ordinance on the Signature Law (SigV) requires AVA_VLA.4 and AVA_MSU.3. Strength of function claim is high. AIS 27 requires ADO_DEL.2.

These requirements motivated the decision to choose EAL 3 augmented.

8.2.4 MUTUALLY SUPPORTIVE SECURITY REQUIREMENTS

Purpose of this rationale is to demonstrate that all dependencies are satisfied, or why specific requirements are not relevant.

The assurance requirements for the TOE were defined using EAL 3 with the augmentations AVA_MSU.3, AVA_VLA.4, ADO_DEL.2, ADV_IMP.1, ADV_LLD.1 and ALC_TAT.1.

Following table shows, how the dependencies are resolved:

requirement	**dependency**	**resolved**
AVA_MSU.3	ADO_IGS.1 ADV_FSP.1 AGD_ADM.1 AGD_USR.1	**EAL3** **EAL3** **EAL3** **EAL3**
AVA_VLA.4	ADV_FSP.1 ADV_HLD.2 ADV_IMP.1 ADV_LLD.1 AGD_ADM.1 AGD_USR.1	**EAL3** **EAL3** **by augmentation** **by augmentation** **EAL3** **EAL3**
ADO_DEL.2	ACM_CAP.3	**EAL3**
ADV_IMP.1	ALC_TAT.1 ADV_LLD.1 ADV_RCR.1	**by augmentation** **by augmentation** **EAL3**
ADV_LLD.1	ADV_HLD.2 ADV_RCR.1	**EAL3** **EAL3**
ALC_TAT.1	ADV_IMP.1	**by augmentation**
FCS_COP.1 (SHA-1)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4 FMT_MSA.2	**No**, see note 1.
FCS_COP.1 (SHA-224)	ditto	**No**, see note 1.
FCS_COP.1 (SHA-256)	ditto	**No**, see note 1.
FCS_COP.1 (SHA-384)	ditto	**No**, see note 1.
FCS_COP.1 (SHA-512)	ditto	**No**, see note 1.
FCS_COP.1 (RIPEMD-160)	ditto	**No**, see note 1.
FCS_COP.1 (Validation1 RSA)	ditto	**Partially**, see note 2.
FCS_COP.1 (Validation2 RSA)	ditto	**Partially**, see note 2.
FCS_COP.1 (Signing – RSA)	ditto	**Partially**, see note 3.
FDP_DAU.2	FIA_UID.1	**No**, see note 4.
FDP_ITC.1 (Signature Creation)	[FDP_ACC.1 or FDP_IFC.1] FMT_MSA.3	**No**, see note 5.

requirement	dependency	resolved
FDP_ITC.1 (Signature Validation)	ditto	**No**, see note 5.
FDP_ITC.2 (OCSP Responses)	[FDP_ACC.1 or FDP_IFC.1] [FTP_ITC.1 or FTP_TRP.1] FPT_TDC.1	**No**, see note 6.
FDP_TVR.1	-	**Yes**, implicitly.
FTP_ITC.1 (Smart Card)	-	**Yes**, implicitly.

Table 11 – Resolving of Dependencies

NOTE 1

The listed requirements refer to cryptographic operations of keyless hash algorithms. Therefore FCS_CKM.1, FCS_CKM.4, FMT_MSA.2 are not applicable.

NOTE 2

The listed requirements refer to cryptographic operations with imported public keys. The dependency FDP_ITC.1 is resolved by FDP_ITC.1 (Signature Creation) and FDP_ITC.1 (Signature Validation). FCS_CKM.4, FMT_MSA.2 are not applicable.

NOTE 3

The listed requirements refer to cryptographic operations performed in the IT-environment. The implementation of the dependencies is specified in the corresponding documents.

NOTE 4

The listed requirements refer to data authentication. The user's identity is recognized by appropriate components (see A.PLATFORM) on base of the certificate. The component FIA_UID.1 is not applicable in the context of the TOE.

NOTE 5

The listed requirements refer to the use of public keys in context of electronic signature validation (FCS_COP.1 (Validation1 RSA) and FCS_COP.1 (Validation2 RSA)). Because the public keys need not to be protected by the TOE, FDP_ACC.1 and FDP_IFC.1 are not required. Import is done without any attributes and no attributes will be initialized (FMT_MSA.3).

NOTE 6

The listed requirements refer to the import of an OCSP response. Because the integrity of the OCSP response is protected by hash value and an electronic signature, that offer the required level of protection, FDP_ACC.1 and FDP_IFC.1 are not required. Because the root certificate is a trusted certificate of the TOE and the TOE validates the complete certificate chain on base of CRLs before using the

OCSP information, FTP_ITC.1 and FTP_TRP.1 are not required. Because the TOE does not exchange these TSF data with another trusted IT product, FPT_TDC.1 is not applicable.

8.2.5 RATIONALE ON MUTUAL SUPPORT

Signature Creation

FCS_COP.1(SHA-1), FCS_COP.1(SHA-224), FCS_COP.1(SHA-256), FCS_COP.1(SHA-384), FCS_COP.1(SHA-512), FCS_COP.1(RIPEMD-160) and FCS_COP.1 (Signing – RSA) support each other in the process of signature creation.

Those requirements support the fulfilment of FDP_DAU.2.

FCS_COP.1(SHA-1), FCS_COP.1(SHA-224), FCS_COP.1(SHA-256), FCS_COP.1(SHA-384), FCS_COP.1(SHA-512), FCS_COP.1(RIPEMD-160) do the hash value computation. FDP_TVR.1 ensures the unambiguous presentation of the document to be signed. FCS_COP.1 (Signing – RSA) performs the encryption of hash value with the RSA algorithm. FTP_ITC.1 (Smart Card) ensures the correctness of the hash value after transport to the SSCD.

Signature Validation without Certificate Chain

FCS_COP.1(SHA-1), FCS_COP.1(SHA-224), FCS_COP.1(SHA-256), FCS_COP.1(SHA-384), FCS_COP.1(SHA-512), FCS_COP.1(RIPEMD-160) and FCS_COP.1 (Validation1 RSA) support each other in the process of signature validation.

Those requirements support the fulfilment of FDP_DAU.2, especially FDP_DAU.2.2.

FCS_COP.1(SHA-1), FCS_COP.1(SHA-224), FCS_COP.1(SHA-256), FCS_COP.1(SHA-384), FCS_COP.1(SHA-512), FCS_COP.1(RIPEMD-160) do the hash value computation. FCS_COP.1 (Validation1 RSA) decrypts the hash value for the hash value comparison. The key is provided by FDP_ITC.1 (Signature Creation). The document which refers to the created signature, was displayed by FDP_TVR.1.

Signature Validation with Certificate Chain

FCS_COP.1(SHA-1), FCS_COP.1(SHA-224), FCS_COP.1(SHA-256), FCS_COP.1(SHA-384), FCS_COP.1(SHA-512), FCS_COP.1(RIPEMD-160) and FCS_COP.1 (Validation2 RSA) support each other in the process of signature validation.

Those requirements support the fulfilment of FDP_DAU.2, especially FDP_DAU.2.2.

FCS_COP.1(SHA-1), FCS_COP.1(SHA-224), FCS_COP.1(SHA-256), FCS_COP.1(SHA-384), FCS_COP.1(SHA-512), FCS_COP.1(RIPEMD-160) do the hash value computation. FCS_COP.1 (Validation2 RSA) decrypts the hash value from the original electronic signature for the hash value comparison. The key is provided by FDP_ITC.1 (Signature Validation). An OCSP response may be provided by FDP_ITC.2 (OCSP Responses).

8.3 TOE Summary Specification Rationale

This section provides a mapping between TOE security functions and security functional requirements for the TOE and a mapping between TOE security measures and security assurance requirements for the TOE.

8.3.1 MAPPING BETWEEN TOE SECURITY FUNCTIONS AND SFRS

The specification of the TOE security functions (chapter 6.1) refers directly to the TOE security requirements. Following table shows that all security requirements are addressed.

Security Function / Functional Security Requirements	SF.OpenedDocument FalsificationPrevention	SF.SelectedDocument FalsificationPrevention	SF.SignatureCreation	SF.SignatureValidation	SF.OCSPProcessing	SF.TimestampValidation	SF.DocumentPresentation	SF.TOEIntegrityChecking
FCS_COP.1 (SHA-1)	x	x	x	x	x	x		x
FCS_COP.1 (SHA-224)	x		x	x	x	x		x
FCS_COP.1 (SHA-256)	x		x	x	x	x		x
FCS_COP.1 (SHA-384)	x		x	x	x	x		x
FCS_COP.1 (SHA-512)	x		x	x	x	x		x
FCS_COP.1 (RIPEMD-160)	x		x	x	x	x		x
FCS_COP.1 (Validation1 RSA)	x				x			x
FCS_COP.1 (Validation2 RSA)				x	x	x		
FDP_DAU.2				x	x	x		
FDP_ITC.1 (Signature Creation)	x							
FDP_ITC.1 (Signature Validation)				x	x	x		
FDP_ITC.2 (OCSP Responses)					x			
FDP_TVR.1							x	
FTP_ITC.1 (Smart Card)	x							

Table 12 – Security Requirements vs. Security Functions

FCS_COP.1 (Signing – RSA) is not listed here, because it is part of the TOE's IT-Environment.

9 Definition of Functional Family FDP_TVR

9.1 Introduction

The signature law requires a legal binding viewer component, which cannot be modelled by components provided by the Common Criteria framework. Therefore it is necessary to define the new functional family FDP_TVR (Trusted Viewer) of class FDP (user data protection).

9.2 Definition: Trusted Viewer (FDP_TVR)

Family Behavior

This family defines functional requirements to a Trusted Viewer component for electronic signature applications. Electronic signature applications require a viewer component, which ensures, that the displayed data is unambiguous. The user must be informed about content, that may not be displayed but the electronic signature will refer to.

Component Levelling

FDP_TVR.1 Trusted Viewer requires the TSF to be able to display the document's content in an unambiguous way, which is free of hidden content. Additionally the user has to be informed about content, which cannot be displayed.

Management : FDP_TVR.1

There are no management activities foreseen.

Audit : FDP_TVR.1

There are no auditable events foreseen.

9.2.1 FDP_TVR.1 TRUSTED VIEWER

Hierarchical to: No other components

Dependencies: No dependencies.

FDP_TVR.1.1	The TSF shall ensure, that the displayed content of a document is unambiguous according to [assignment: norms for displaying content].
FDP_TVR.1.2	The TSF shall ensure that the displayed content of a document is free of active and hidden content. The TSF shall ensure, that the user is informed about

	active and hidden content.
FDP_TVR.1.3	The TSF shall ensure, that the user is informed about content, which cannot be displayed due to the capabilities of the TSF.

10 Bibliography

[CC2.3]	Common Criteria v2.3 announced August 2005 http://www.commoncriteriaportal.org/public/expert/index.php?menu=2
[CEN]	CEN Workshop Agreement CWA 14170 announced May 2004 http://www.cenorm.be/cenorm/businessdomains/businessdomains/isss/cwa/electronic+signatures.asp
[EXE4J]	exe4j – Java Exe Maker announced by ej technologies http://www.ej-technologies.com/products/exe4j/overview.html
[FIPS-180-2]	Secure Hash Standard Federal Information Processing Standards Publication 180-2 German Information Security Agency and Katholieke Universiteit Leuven announced 1 August 2002
[ISISMTT]	ISIS-MTT Specification Optional Profile SigG-Profile Version 1.1 T7 & TeleTrust announced 16 March 2004 http://www.isis-mtt.t7-isis.org/uploads/media/ISIS-MTT_Optional_Profiles_v1.1_02.pdf
[ISO 9796-2]	ISO/IEC 9796-2:2002: Digital signature schemes giving message recovery -- Part 2: Integer factorization based mechanisms International Organization for Standardization announced in 2002
[PKCS#1]	Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography, Specifications Version 2.1 (RFC3447) Internet Society announced in February 2003
[PKCS#7]	PKCS #7: Cryptographic Message Syntax (CMS) (RFC3815) Internet Society

	announced in July 2004
[RFC2560]	Internet X.509 Public Key Infrastructure Online Certificate Status Protocol - OCSP (RFC2560) Internet Society announced in June 1999
[RFC3280]	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (RFC3280) Internet Society announced in April 2002
[RIPEMD-160]	A strengthened version of RIPEMD Hans Dobbertin, Antoon Bosselaers, Bart Preneel German Information Security Agency and Katholieke Universiteit Leuven announced 18 April 1996 http://homes.esat.kuleuven.be/~bosselae/ripemd160.html
[SAlg]	Table of Suitable Algorithms announced 17 December2007 http://www.bundesnetzagentur.de/media/archive/12198.pdf
[SigG]	Signaturgesetz announced 16 May 2001 http://www.bundesnetzagentur.de/media/archive/2247.pdf
[SigV]	Verordnung zur elektronischen Signatur announced 16 November 2001 http://www.bundesnetzagentur.de/media/archive/5264.pdf
[SLUG]	Sign Live! CC - User Guidance v3.2.3, announced January 2008
[SSSSC]	Signtrust Signaturserver Version 4.1 Sicherheitsbestätigung
[TIFF]	TIFF Revision 6.0 final draft Adobe Developers Association announced 3 June 1992 http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf
[TSSSC]	timeproof QSS 400 Version 4.00 Sicherheitsbestätigung http://timeproof.de/de/index.html