Security Target

Electronic Signature Application

S-TRUST Sign-it base components 2.5

Version 2.5.1.1

Certification ID: BSI-DSZ-CC-0585

OPENLiMiT SignCubes AG

Zuger Str. 76 B

6411 Baar

Switzerland

Version	Date	Author	Comments
1.0	2008-11-11	A. Lunkeit	Initial version
1.1	2008-12-04	A. Lunkeit	Refinement
1.2	2008-12-17	A. Lunkeit	Final version

Contents

Glossary and Abbreviations

BSI	Bundesamt für Sicherheit in der Informationstechnik (German Federal Office for Information Security)
CC	Common Criteria
CRL	Certificate Revocation List
EAL	Evaluation Assurance Level
IT	Information Technology
OCSP	Online Certificate Status Protocol
PP	Protection Profile
SigG	German Signature Law
SigV	Ordinance to the German Signature Law
SOF	Strength of Function
SSCD	Secure Signature Creation Device
SSEM	S-TRUST Sign-it Security Environment Manager
ST	Security Target
TOE	Target of Evaluation
TSC	TSF Scope of Control
TSF	TOE Security Functions

Tables

Figures

1 Introduction

The following sections contain general information about the TOE[1] and other general information.

1.1 Identification

Title:	Security Target for S-TRUST Sign-it base components 2.5, Version 2.5.1.1
Author:	A. Lunkeit, OPENLiMiT SignCubes GmbH, Berlin, Germany
CC-Version:	Version 2.3
General Status:	Final
TOE:	S-TRUST Sign-it base components 2.5, Version 2.5.1.1

The German name of the TOE is S-TRUST Sign-it Basiskomponenten 2.5, Version 2.5.1.1.

AIS-Version[2]: 02.07.2001
Publisher: Bundesamt für Informationstechnik in der Bundesrepublik (BSI)

1.2 Security Target Overview

S-TRUST Sign-it base components 2.5 is an electronic signature application compliant to the German electronic signature law and ordinance on electronic signatures. The application itself is a set of executables and programming libraries. This means that S-TRUST Sign-it base components 2.5 may be used as a single application but also may be integrated into third party products.

The S-TRUST Sign-it base components 2.5 have been developed for the use on the operating systems from Microsoft since Microsoft Windows NT 4 SP 6. The IT-security environment of the TOE requires a smartcard and a card terminal with secure PIN entry mode to run the required cryptographic operations during the process of signature creation.

[1] TOE – Target of Evaluation

[2] Allgemeines Interpretationsschema – General Interpretation Scheme

The product does provide additional cryptographic functionality like data encryption based on symmetric encryption algorithms. These product capabilities are not part of the Common Criteria evaluation of this product.

The TOE itself is limited to the creation of hash values, using the SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and RIPE-MD 160 algorithms and is therefore able to check and ensure the integrity as well as the trustworthiness of signed data based on the components responsible for CRL-processing, OCSP-processing, timestamp processing and PDF processing.

The TOE provides a legal binding displaying unit (S-TRUST Sign-it Viewer) for the Text, TIFF and PDF format. The displaying unit of the TOE allows the examination of the files content in order to ensure that the user is assured about the content to be signed or the content of the signed file.

This document contains the Security Target for the product S-TRUST Sign-it base components 2.5. The product is intended to be evaluated and certified using assurance level EAL 4+ in line with Common Criteria 2.3 in order to achieve a CC- based security certificate and a SigG-confirmation from the German Federal Office for Information Security (BSI).

The name of the TOE is S-TRUST Sign-it base components 2.5, Version 2.5.1.1, the short form of that name will be used in that document: S-TRUST Sign-it base components 2.5. This abbreviation does always refer to the TOE, respective its full name.

The TOE is a re-evaluated version of the evaluated TOE OPENLiMiT SignCubes base components 2.5, Version 2.5.0.1.

1.3 CC Conformance Claim

The TOE is compliant to:

- Part 2 extended of Common Criteria 2.3, released August 2005

In order to provide a complete description of the functional requirements addressed by the TOE, functional components of part 2 of the Common Criteria framework were used. But also additions to the Common Criteria part 2 were defined, to fulfill the requirement of a complete and consistent TOE description.

- Part 3 conformant to Common Criteria 2.3, released August 2005

For the description of the requirements due to the trustworthiness of the TOE, only security assurance requirements of CC part 3 were used. The assurance requirements are compliant to EAL 4 augmented. The augments are AVA_MSU.3 and AVA_VLA.4.

1.4 SigG and SigV Conformance Claim

The vendor of the product S-TRUST Sign-it base components 2.5 claims that the product is compliant to the signature law (SigG) §17 paragraph 2. In addition to that the vendor claims that the product is also compliant to the ordinance on electronic signatures (SigV) §15 paragraph 2 and paragraph 4.

The following listing demonstrates the compliance of the TOE with the signature law and ordinance on electronic signatures.

§17 SigG, paragraph 2 defines:

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. […]

The product implements a security function, namely SF.1, which is intended to meet the requirements, defined in this sentence.

[…] To check signed data, signature-application components are needed that will show

1. To which data the signature refers (included by SF.1, SF.2, SF.4 and SF.5)
2. Whether the signed data are unchanged (included by SF.2 and SF.4)
3. To which signature-code owner the signature is to be assigned (included by SF.1 and SF.2)
4. The contents of the qualified certificate on which the signature is based, and of the appropriate qualified attribute certificates, and (included by SF.1 and SF.2)
5. The results of the subsequent check of certificates under Section 5(1) Sentence 2. (implicitly included through SF.1 and SF.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.

These last requirements are fulfilled through the combination of the TOE's security functions. The definition of the security functions is given in 6.1 of this document.

§15 SigV, paragraph 2 defines:

Signature application components pursuant to Section 17 (2) of the Signatures 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 secure signature creation device,
b) a signature is provided only at the initiation of the authorized signing person,
c) the production of a signature is clearly indicated in advance […]

The product meets the requirements through the combination of security functions and the requirements defined for the IT-security environment. Namely security function SF.1 ensures, that the production of a signature is clearly indicated. The use of smart cards together with smart-card terminals, which support secure-pin entry, conforms to the requirements defined in 1a) and 1b). Through the use of a smart-card (SSCD) and through the security functions of those smart cards it is ensured that a signature is only provided at the initiation of the authorized signing person.

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.

The Security function SF.2 conforms to the requirements defined in 2a) and 2b).

§15 SigV, paragraph 4 defines:

Security-relevant changes in technical components pursuant to subsections (1) to (3) must be apparent for the user.

Through the combination of several security functions, the IT-security environment assumptions and the secure usage assumptions, the product ensures the compliance the requirements defined in §15 SigV, paragraph 4. Moreover, the product contains mechanisms for manipulation detection, which ensures, that security-relevant changes in the technical components are apparent to the user. Especially SF.3, SF.5 and SF.6 are intended to ensure the correctness of the application and of all its relevant components.

Remarks to the use of cryptographic algorithms

The TOE supports hash algorithms and RSA keys, which might possibly no longer useable for qualified electronic signatures. Therefore the user is required to check, if the algorithms, which have been used for the generation of the qualified electronic signature, are suitable for the requirements of qualified electronic signatures.

2 TOE Description

The TOE is a set of executables and programming libraries. As stated before, the application may be used as stand-alone signature client but may also be integrated into third party products.

In a logic model, the application consists of two main components: the S-TRUST Sign-it Security Environment Manager and the S-TRUST Sign-it Viewer that is responsible for a legal binding presentation of the data to be signed. Both together are the base application that is installed on the user computer.

The S-TRUST Sign-it Security Environment Manager does also export an API (Application Programming Interface) that may be used to interact with the TOE using a programming language[3]. Moreover the S-TRUST Sign-it base components 2.5 do provide a graphical user interface that allows the interaction with product using interface devices (e.g. monitors, keyboards and a computer-mouse).

For partners of the company, OPENLiMiT SignCubes provides a development kit which consists of the files that are required to use the programming interface of the product as well as a complete description of the public accessible functionality, required parameters and return values.[4]

For the creation of electronic signatures, a secure pin-entry device and a smart card are required. In addition to that the TOE must be installed in a configuration that contains a valid set of smartcard terminal, smartcard operating system and smartcard profile modules. The S-TRUST Sign-it base components include the capabilities for OCSP, CRL and timestamp processing. The modules that implement the functionality of how to retrieve an OCSP response, how to download a CRL and how to retrieve a timestamp are not part of TOE but the TOE implements mechanisms that ensure that only modules may be used that have been developed by OPENLiMiT SignCubes. This approach is for example intended to allow

[3] This API is called “S-TRUST Sign-it Job Interface”

[4] This depends on commercial purposes.

the change of authentication mechanisms against the providers of such information without a re-evaluation, re-certification and re-confirmation of the product.

The evaluated product supports the German language.

The target of evaluation (TOE) is the product S-TRUST Sign-it base components 2.5 with the S-TRUST Sign-it Security Environment Manager (SSEM) and the S-TRUST Sign-it Viewer. In addition the S-TRUST Sign-it Job Interface should be certified together with the provided API documentation so that third party solutions could integrate the provided functionality of that API to extend their own set of functionality based on the OPENLiMiT SignCubes development kit. The use of the S-TRUST Sign-it Job Interface should also be confirmed so that it becomes possible to build applications that are allowed to create and verify qualified electronic signatures by using the S-TRUST Sign-it Job Interface in a certified and confirmed way.

The TOE implements a mechanism that allows the generation of a verification protocol during the process of signature validation. Therefore a report library is required that is responsible to generate the report files. This library is not part of the TOE. Equally to OCSP, timestamp and CRL download modules the verification report library is not part of the TOE but it also implements mechanisms that ensure that only modules may be used that have been developed by OPENLiMiT SignCubes. This approach is for example intended to allow the change of authentication mechanisms against the providers of such information without a re-evaluation, re-certification and re-confirmation of the product.

The TOE includes a dynamic library that is used by the IBM Lotus Forms Viewer application for working with XFDL Forms to start the process of generating or validating an electronic signature via the TOE. This plugin library is an optional library and is not always part of the TOE deliverables. In order to get detailed information about the installed product configuration, the user has the possibility to use the S-TRUST Sign-it Integrity Tool.

The term product will be used in the further document as a synonym for the S-TRUST Sign-it base components used as a stand-alone application for electronic signatures.

The version of the TOE binary files is 2.5.1.1.

In addition to the main components, OPENLiMiT SignCubes provides the S-TRUST Sign-it Integrity Tool that must be used to ensure the integrity of the products installation. The S-TRUST Sign-it Integrity Tool is a separate application implemented as a Java Applet. So the TOE consists of the S-TRUST Sign-it Security Environment Manager, the S-TRUST Sign-it Viewer and the S-TRUST Sign-it Integrity Tool.

The TOE supports the smartcards and smartcard terminals listed in section 3.1. The installed configuration does not necessarily contain all these modules. Configurations of the TOE that do only contain a minimum set of card terminal modules as well as a minimum set of smartcard modules are also allowed.

This means the following: smartcard terminal, smartcard operating system and smartcard profile modules are optional modules. This means, that the TOE can be installed without any smartcard and smartcard terminal support up to the complete set of these modules.

2.1 Architectural Overview

The following graphic depicts the architecture of the product with its main components.

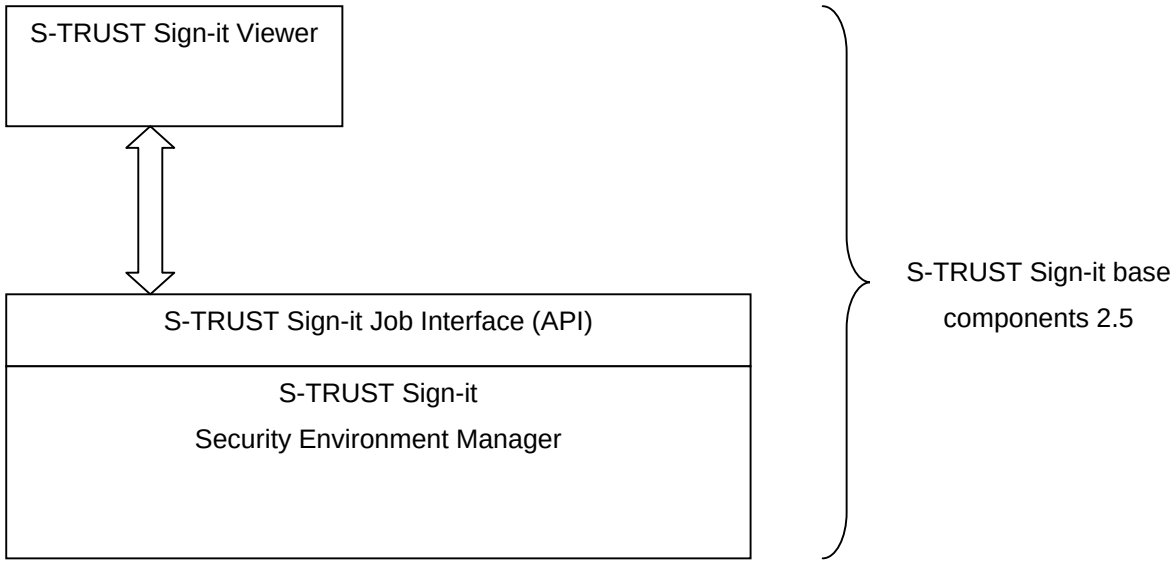


Figure 1 Architectural Overview

The S-TRUST Sign-it Security Environment Manager is the main component of the whole application and provides all security functionality except the capabilities of the legal binding viewer component.

The S-TRUST Sign-it Viewer uses the S-TRUST Sign-it Job Interface for any security relevant operation and does not implement any security functionality except the one that is required for legal binding viewing of PDF, TIFF and Text files.

As the graphic depicts, it is not possible to interact with the S-TRUST Sign-it Security Environment Manager directly using a programming interface, the S-TRUST Sign-it Job Interface must always be used to access any functionality of the SSEM in a programmed way.

Underneath the S-TRUST Sign-it Security Environment Manager the required security hardware is located. Required security hardware means a card reader with secure pin entry mode and a secure signature creation device (SSCD).

OPENLiMiT SignCubes provides the S-TRUST Sign-it Integrity Tool as a separate application to ensure the integrity of the products installation on the computer of the user. The description for that tool can be found in the section “S-TRUST Sign-it Integrity Tool”.

2.2 S-TRUST Sign-it Viewer

The S-TRUST Sign-it Viewer component is a software component for displaying signed data or data to be signed according to the signature law §17 paragraph 2.

In accordance to the German signature law §17, paragraph 2, in the process of signature creation the S-TRUST Sign-it Viewer component is required to ensure that the data, to which the signature will refer to, is unambiguously displayed.

In the process of signature verification the S-TRUST Sign-it Viewer ensures, that it is evident, to which data the electronic signature refers.

The following functionalities are identified for the S-TRUST Sign-it Viewer:

- Functionality to obtain information about the document to be displayed
- Functionality for displaying PDF, TIFF and Text documents[5]

The S-TRUST Sign-it Viewer does not provide any API's that may be used be third party components. This component makes use of the S-TRUST Sign-it Job Interface to provide signature processing functionality on its graphical user interface. This means that the S-TRUST Sign-it Viewer provides the functionality to start the creation of an electronic signature on the currently displayed document as well as the start of the verification of an electronic signature that has been found for a document.

The S-TRUST Sign-it Viewer is able to display TIFF documents following the Adobe TIFF specification, PDF documents that follow the PDF 1.7 document format as well as documents that contain ASCII characters. The TIFF document can also be a multipage TIFF.

[5] The displayed formats do depend on the license file of the user. The license file may specify that a file format, e.g. PDF, cannot be displayed. In this case a message is displayed to the user that informs him that he has not the required license to display that data format.

This component implements a parser that checks the correctness of the document to be displayed. This means, that the TIFF and/or PDF tags and elements are checked and the viewer ensures that the document does not contain information that could not be displayed to the user. If the S-TRUST Sign-it Viewer detects an error in the documents structure, a warning message is displayed to the user. If the component detects that an ASCII file should be displayed, the Viewer ensures that the document fits with the rules for ASCII code and displays a warning message if the documents content does not behave the specified way[6].

The file formats that can be opened with the S-TRUST Sign-it Viewer are PDF, TIFF, Text and PKCS#7 encoded data corresponding to one of the listed document formats. The PKCS#7 encoded data may contain one or more electronic signatures that may be verified using the S-TRUST Sign-it Viewer as an indirect interface to this functionality of the S-TRUST Sign-it Security Environment Manager.

The S-TRUST Sign-it Viewer offers the possibility to validate a document with a detached XML signature. This means that the OPENLiMiT Viewer is able to recognize detached XML signatures and to start the validation process of such signatures by the use of the OPENLiMiT SDK interface.

If the user decides to sign the document that is currently displayed with the S-TRUST Sign-it Viewer, he can start the process of electronic signature creation using the S-TRUST Sign-it Viewer as an indirect interface to that functionality provided by the S-TRUST Sign-it Security Environment Manager.

With the choice of these file formats the user cannot be the target of malicious code or active code that may be included in the document, because the document formats do not support active or hidden code or content or the active code is ignored[7] by the application.

[6] The rules are documented in the TOE's user guidance.

[7] [26] does not allow active code elements, e.g. JavaScript. If JavaScript is included in the document, a warning message is generated and the code is not executed because the TOE does not contain any mechanisms to interpret such content. [26] does allow active elements, e.g. embedded multi media content, but these types of content are ignored by the S-TRUST Sign-it Viewer.

2.3 S-TRUST Sign-it Security Environment Manager with Job API

The S-TRUST Sign-it Security Environment Manager provides the following functionality that may be accessed in parts or completely through the use of the S-TRUST Sign-it Job API:

- Computation of hash values using the SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and RIPE-MD 160 algorithms.
- Creation of electronic signatures using a smart card and a secure pin entry device.
- Timestamp processing during the process of electronic signature creation.
- Support for attribute certificates in the process of electronic signature creation.
- Support for OCSP processing during the electronic signature creation.
- Electronic signature verification including OCSP and CRL processing as well as timestamp processing. The use of attribute certificates is supported.
- API's for applications/product parts that want to use the provided functionality.
- Ensuring the integrity and correctness of the SignCubes base components installed on the users computer.
- Providing graphical interfaces in the process of signature creation, verification and product configuration.
- Optional verification report creation in the process of signature verification[8].

In the process of signature verification the S-TRUST Sign-it Security Environment Manager is required to provide information about qualified attribute certificates corresponding to a qualified certificate that was used for the creation of a qualified electronic signature. The S-TRUST Sign-it Security Environment Manager includes the capabilities to detect, if an attribute certificate belongs to a presented certificate and, if the attribute certificate is accessible, to display the information encoded in the attribute certificate.

The S-TRUST Sign-it base components do work with PKCS#7 or XMLDSIG encoded data[9]. This means, that the application is compatible with any application that processes data in the same format. PKCS#7 (or CMS) is a data format, which is based on ASN.1 and can contain

[8] The verification report library itself is not part of the TOE.

[9] The TOE's security functionality is limited to decrypted data and does not cover encryption or decryption capabilities.

the originally signed data and the signature as well. XMLDSIG encoded means: The TOE is able to create and validate electronic signatures in the XMLDSIG format but does not process XML documents or streams. In other words: The TOE provides the capability to create an XML signature and to validate an XML signature for any binary format but does not provide the capability to extract an XML signature from an XML encoded document. This has to be performed by a third component, e.g. when integrating this functionality into a third system.

Depending on the driver implementation for each smart-card terminal, the S-TRUST Sign-it base components can use the PC/SC or the CT-API for the communication with the smart card and the smart-card terminal. Some manufacturers deliver separate program libraries for starting the secure pin entry mode of their smart-card terminals. The S-TRUST Sign-it base components implement the special requirements of the terminals, if there are any. The list of supported smart card readers can be found in the chapter “Secure Usage Assumptions”. The list of supported smart cards can also be found in the chapter “Secure Usage Assumptions”.

The S-TRUST Sign-it base components do contain a module that supports the secure PIN entry mode via generic CCID-conformant commands. This means that all card readers, which support this mechanism, can be used by the S-TRUST Sign-it base components. The evaluated configuration is limited to the card terminals listed in this document.

Only devices, which have a SigG confirmation, are allowed for the use with the S-TRUST Sign-it base components. The German Federal Network Agency for Electricity, Gas, Telecommunications, Post and Railway (Federal Network Agency, formerly known as RegTP, http://www.bundesnetzagentur.de/) provides a list of these products.

The S-TRUST Sign-it Security Environment Manager can be used by any application. This facilitates the integration into software products of third parties. Therefore the S-TRUST Sign-it Job Interface is implemented and appropriate documentation material is delivered.

The S-TRUST Sign-it Security Environment Manager provides two signature request dialogs (a default dialog and an advanced dialog which includes a textual preview). The dialog usage is controlled by the job interface.

The S-TRUST Sign-it Job Interface offers a job that allows the creation of electronic signatures for more than one document. This job is not offered by the graphical user interface of the TOE. If an external application utilizes this job, the signature creation dialog is displayed to the user. This signature creation dialog provides a list of the files to be signed and allows the inspection of each file by the S-TRUST Sign-it Viewer that is part of the TOE. The creation of electronic signatures is started when the user confirms the signature creation process by pressing a button that is part of the signature creation dialog.

During the creation of the electronic signatures by the secure signature creation system, which consists of a smart card and a card terminal with secure pin entry mode, the user is required to enter the PIN for each document separately, if a smart card is utilized, which does not allow to leave the PIN open. If the user utilizes a smart card that allows leaving the card open, the PIN is only requested one time[10]. In any case, the PIN is closed by the S-TRUST Sign-it base components 2.5 when the job has finished.

The TOE provides a settings dialog that enables the user to specify the limits for the generation of electronic signatures. The limit can be set by the time that the card is left open or by the number of signatures to be created. These options can be utilized to define breakpoints in the generation of electronic signatures to enter the PIN again. The first criterion that has been reached during the execution of the job closes the PIN and the user is required to enter the PIN once again.

The user has the possibility to cancel the process of signature creation. In this case, all signature files that have been created by this job are removed from the directory.

The job cannot be utilized to implement a signature service that automatically signs a document. Through the presentation of the signature request dialog and the list of documents to be signed it is ensured, that the user has the possibility to view all documents using the S-TRUST Sign-it Viewer before the signature process is started. Furthermore the

[10] Therefore the TOE has to be configured via a settings dialog as described below. If the TOE is not configured to leave the card open, the PIN is requested for each signature operation.

user must acknowledge the creation of the signatures by pressing a button in the request dialog.

Consequently the automating of signature creation of different types of documents is excluded. The user is required to define a limit for the creation of electronic signatures using the settings dialog for time and number of electronic signatures that are allowed to be created without requesting the PIN again. In each case the PIN is closed by the S-TRUST Sign-it base components 2.5 when the job has been executed, so the maximum count of electronic signatures that can be created without requesting the PIN again is defined by the amount of documents to be signed by the job.

The job can only be accessed by OPENLiMiT applications and is not permitted for third party applications because the calling application needs a special signature that identifies the application as a permitted OPENLiMiT application. If an application, which is not allowed to utilize this job, tries to execute this job, the S-TRUST Sign-it Security Environment Manager is deactivated due to suspected manipulation. This means, that no security related functionality (especially the initiation of creation of electronic signatures), can be accessed through the S-TRUST Sign-it Job Interface or through the graphical interface of the TOE. This state is signaled to the user with an error message and a special icon in the system tray.

The S-TRUST Sign-it base components 2.5.1.1 ensure that the PIN is closed after the execution of the job. This ensures that the user is required to enter the PIN again, if the creation of an electronic signature is initiated the next time.

Qualified certificates that have been issued by a German shown or accredited certification authority are always checked using the chain model. All other electronic signatures are checked using the shell model. When validating an electronic signature, the TOE does always validate the certificate chain from the end user certificate down to the root CA in combination with CRL processing.

For PKI models, which are not compliant to the X.509 standard, a configuration file is required[11]. After checking the revocation list, an unambiguous status value is set on the programmable interface. In addition, an unambiguous text message can be displayed[12]. This depends on several configuration issues. If it is impossible to load the revocation list or another error occurs, an unambiguous status code is set. The TOE owns a configuration file that specifies the hash algorithms and signature key length that are accepted when validating qualified electronic signatures. In case that a hash algorithms or key size is detected that should not be used for qualified electronic signatures, the TOE displays an appropriate message to the user[13].

In addition to CRL processing, the S-TRUST Sign-it base components offer the possibility to use the online certificate status protocol (OCSP) to verify the validity of a given certificate. This can only be done, if enough information about the OCSP responder is available. This means that the certificate must contain a field identifying an OCSP responder for this certificate or a special configuration file exists, which identifies the OCSP responder for the CA that issued the certificate under examination. After processing the OCSP response, an unambiguous status code is set on the programmable interface. If the product is configured to show dialogs, an unambiguous message will be shown to the user that informs about the validity of the certificate that has been requested on the OCSP responder.

The scope OCSP, CRL and timestamp processing in this certification does not include the way of how to obtain that information. The way of how to obtain that information is explicitly not part of the evaluation. The product implements policies that allow using several modules

[11] The file is digitally signed with same private key that is used for the signatures on the TOE's binary files and part of the TOE.

[12] The verification of an electronic signature can be done in the modes "verbose" or "silent". The S-TRUST Sign-it base components, e.g. the S-TRUST Sign-it Viewer, always use the "verbose" mode so that the user is always informed about the validity of an electronic signature that he checks. The use of these modes is configured by the use of the S-TRUST Sign-it Job Interface and is unique for each operation on the S-TRUST Sign-it Job Interface.

[13] This mechanism is mainly required because the TOE can validate electronic signatures with a key size starting with 1024 bits. This key size is not accepted especially in Germany, where a minimum key size of 1280 bits is required for the generation of qualified electronic signatures.

to retrieve that requested information. The modules itself must be signed by OPENLiMiT SignCubes. With this approach OPENLiMiT SignCubes ensures that no malicious modules may be installed on the computer of the user and may be used by the application without detection of this state. Even if an attacker does reverse engineering and identifies the internal API's that are used to retrieve such data he would not be able to manipulate the application using that approach. The addition of a new module underlies the update procedure of the product that is described in the section "Delivery".

Another capability of the S-TRUST Sign-it Security Environment Manager is the processing of timestamps. A Timestamp can be requested during the process of signature creation and that timestamp can be used during the signature verification.

The use of attribute certificates is also supported. The S-TRUST Sign-it Security Environment Manager includes the capabilities to display attribute certificates and to process their content. Any information about the attribute certificate under examination may also be accessed using the S-TRUST Sign-it Job Interface.

The S-TRUST Sign-it Security Environment Manager implements all required capabilities for the management of the program modules. The S-TRUST Sign-it Security Environment Manager also manages messages, which may be displayed to the user[14]. The default configuration of the S-TRUST Sign-it Security Environment Manager enforces displaying any dialogs to the user. The capabilities to display messages and information to the user as well as the correctness of the status codes that are set on the user interface must be evaluated.

The capabilities of the S-TRUST Sign-it base components to process PDF documents depend on the license code of the user. If the user has no license code, the product disables all capabilities to generate signatures but is still usable for the purpose of signature verification. The electronic signature initiation and PDF operating modes that depend on the license are defined as following:

- No signature creation capabilities of the TOE, if the user does not have a license code. The displaying of PDF documents is not allowed.
- No PDF displaying capabilities of the S-TRUST Sign-it Viewer. No possibilities to create any kind of signature on PDF documents. The S-TRUST Sign-it Viewer can be utilized to display Text and TIFF documents and to create electronic signatures on these document types.
- PDF displaying but no capabilities to create signatures on PDF documents using the S-TRUST Sign-it Viewer. Signatures, including embedded PDF signatures on PDF documents, can be verified. The S-TRUST Sign-it Viewer can be utilized to display Text and TIFF documents and to create electronic signatures on these document types.
- PDF displaying and the capability to create attached and detached electronic signatures as well as the verification of these signatures using the S-TRUST Sign-it Viewer[15]. The S-TRUST Sign-it Viewer can be utilized to display Text and TIFF documents and to create electronic signatures on these document types. The capability to create embedded PDF signatures is not enabled.

[14] Only, if the S-TRUST Sign-it Security Environment Manager is configured to display messages (this is the default behaviour). Otherwise, the S-TRUST Sign-it Security Environment Manager does only provide appropriate status codes on the programmable interface.

[15] Embedded PDF signatures can also be verified.

- PDF displaying is enabled; the creation of all supported kinds of electronic signatures[16] is enabled as well as the verification of these signature types. The S-TRUST Sign-it Viewer can be utilized to display Text and TIFF documents and to create electronic signatures on these document types.

The license influences the TOE's behaviour on the graphical interfaces, namely the S-TRUST Sign-it Viewer. If the user utilizes a version that does not allow the creation of electronic signatures, the TOE cannot be used for the generation of electronic signatures except the creation of an electronic signature during the licensing process. All other licenses do allow the generation of PKCS#7 encoded signatures on PDF documents using the S-TRUST Sign-it Job Interface. The generation of PDF signatures using the S-TRUST Sign-it Job Interface is not supported for any kind of license.

The S-TRUST Sign-it Job Interface is allowed to utilize the capabilities of the TOE in order to display PDF documents and to verify signatures on PDF documents, even if these capabilities are not part of the user's license.

The TOE supports the creation of a verification protocol in case that a suitable report library is available. Therefore the TOE utilizes an interface that is part of the certification. The report library itself is not part of the certification and might be part of updates that do not change the certified configuration.

The TOE provides the possibility to update the current license without reinstallation of the product. The configuration on the user's computer is managed by the license code that is provided to the user together with the setup routine for the TOE.

[16] This means PKCS#7 signatures, XML-signatures and embedded PDF-signatures.

2.4 S-TRUST Sign-it Integrity Tool

The S-TRUST Sign-it Integrity Tool is a Java Applet that is used to allow the user to check the integrity of the installed product. This Java Applet knows the SHA-256 hash values of each file that is part of the TOE and checks, if the file's hash value is identical with the known hash value.

The Java Applet is also digitally signed; the Java Virtual Machine verifies the signature of the Applet. To ensure the correct behavior of this tool, the Java Virtual Machine v1.4 or higher is required. The fingerprint of the signing certificate is published on the OPENLiMiT Website (www.openlimit.com).

If the Java Applet detects that the hash value of the file under test does not fit with the hash value that is known to the Applet for this file, an unambiguous message is displayed to the user. In this case, the user is required to reinstall the product once again and to check the integrity of the current installation once again.

Because not all smartcard terminal and smartcard modules are required to be installed, the Integrity Tool provides information about the installed smartcard terminal, smartcard operating system and smartcard profile modules. If one of the modules has not been installed, the Integrity Tool signals a not installed file that does not lead to an Integrity fault. If one of these modules is available and has been changed, the Integrity Tool signals a manipulation of the installed TOE.

The Integrity Tool offers the possibility to store an Integrity Check protocol on the local file system. The Check Protocol informs the user about the list of verified files, supported card terminals, smartcard operating systems and smartcard profile modules.

The S-TRUST Sign-it Integrity Tool does not check the settings that the user has applied to the TOE because the TOE has a wide range of configuration items. The requirements for a secure product configuration are listed in the user guide for the product and must be checked manually by the user. After the installation of the product, the S-TRUST Sign-it base components 2.5.1.1 have a secure configuration that is recommended by the manufacturer.

The S-TRUST Sign-it Integrity Tool itself generates all messages that are displayed to the user. The text displaying functionality of the S-TRUST Sign-it Security Environment Manager is not used because the SSEM itself can be manipulated.

2.5 TOE limitations

The TOE cannot assure the correctness of the following functions:

- Private Key material. The secure signature creation device must assure the correctness and integrity of the private key material.
- Assurance of the operating system integrity. The TOE does not contain any capabilities for ensuring the integrity of the operating system and its environment. The user must assure, that sufficient actions are undertaken to avoid, that the operating system may be compromised.
- Strength and security of cryptographic operations. The TOE uses libraries for hash value creation and the RSA algorithm for signature validation. Therefore the TOE can only assure the compliance to given standardization documentation and test vectors but must not make any statement about the strength of the cryptographic operations.

The capability characteristics of the TOE are limited to the computation of hash values and the usage of secure signature creation devices for electronic signature creation and the usage of the RSA algorithm for signature verification. Manipulations on the IT-security environment cannot be recognized or even prevented by the TOE.

2.6 Delivery

The DSV and its partners are responsible for the delivery of the TOE. The following delivery procedures are foreseen:

- Delivery on Read Only Storage Devices
- Online Delivery Procedures

In both cases, the S-TRUST Sign-it Integrity Tool is part of the delivery procedure. The JAR Archive that contains the S-TRUST Sign-it Integrity Tool is electronically signed (not with a qualified signature). If the user executes the S-TRUST Sign-it Integrity Tool, the Java Virtual Machine verifies the electronic signature of the JAR archive automatically.

After the installation of the TOE, the user is required to execute the S-TRUST Sign-it Integrity Tool to verify the integrity of the installed TOE. If the integrity check was successful, the customer received the TOE that was intended to be delivered.

Another possibility to access the Integrity Tool is the use of the following link:

https://www.s-trust.de/sign-it/sicherheit

This URL leads the customer to the S-TRUST Sign-it Integrity Tool that is located on the DSV Website.

The installation routine, which is delivered to the customer, can vary between the maximum available configuration and the minimal available configuration. The minimum available configuration contains no smartcard terminal, smartcard operating system and smartcard profile modules. The maximum TOE configuration contains all of these “optional” files. In each case, the user is required to use the Integrity Tool to verify the installed configuration. The Integrity Tool displays a list of verified files and shows a list of supported smartcard terminals, smartcard operating systems and smartcard profile modules.

The customer may use add-on setups to install optional files that were not part of the initial installation. After the execution of an add-on setup the customer must utilize the Integrity Tool to check whether the required additional modules have successfully been installed.

The main setup routine does always install a version of the S-TRUST Sign-it base components 2.5, version 2.5.1.1, which contains in minimum all necessarily required files.

The mechanism of a basic setup that only installs a basic version of the TOE and additional add-on setups is intended to minimize the download size of the program files when delivering the TOE via download procedures. In some cases the user may only require a basic installation for verifying electronic signatures, in other cases the user might require the full set of smartcard and smartcard terminal support.

If the user wishes to utilize the TOE capabilities, he is required to utilize a license code[17].The license code is provided by the DSV or its resellers to the customer.

If the TOE is received by download, the OPENLiMiT SignCubes AG may require the DSV and its resellers to implement a registration procedure that generates evidence for the license codes that have been delivered to customers.

The OPENLiMiT AG or its resellers provide the license code to the customer. For usage in a Microsoft Windows Terminal Server environment the vendor or its resellers provide an electronically signed package that contains a user specific license file as well as information how to install the license file in a Terminal Server environment.

[17] see 2.3

3 TOE Security Environment

3.1 Secure Usage Assumptions

The following assumptions about the TOE's security environment are devised.

A.Platform

The user utilizes an Intel 586 compatible computer as hardware platform, which contains at least 128 MB of RAM and 120 MB of free disk space.

On the computer is one of the following operating systems installed:

- Windows NT 4 SP 6
- Windows 2000 SP 2
- Windows 2003
- Windows 2003 64 Bit Edition
- Windows XP Home
- Windows XP Professional
- Windows XP Tablet PC Edition
- Windows XP 64 Bit Edition
- Windows Vista
- Windows Vista 64 Bit Edition
- Windows 2008
- Windows 2008 x64 Edition

Additionally the TOE supports a terminal server environment under Windows operating systems Windows 2000 with Citrix Metaframe, Windows 2003 with and without Citrix Metaframe and Windows 2008 without Citrix Metaframe.

In a terminal server environment the communication between server and client/s is conducted via an encrypted channel and is therefore to be considered trustworthy.

In order to use the 128 Bit encryption, the installation of the Windows 2000 High Encryption pack is compulsory for each Windows 2000 based computer that is in communication with the server. With Windows Terminal Server 2003 and 2008 in conjunction with the Operating System Windows XP and Windows Vista on the client side, the 128 Bit encryption is already standard and is to be used.

In addition to these requirements, the Internet Explorer version 5.01 or higher is installed. Moreover, the Microsoft smart card base components are installed on the computer[18]. In addition to that, a Java Virtual Machine (JVM) is installed on the computer, which complies at least with the Java Runtime Environment v1.4.

The user ensures that all components of the operating system are correct. The user ensures that no malicious or harmful program is installed on the system.

The user utilizes a secure signature creation system, which consists of a smart-card terminal with secure pin entry capabilities together with a smart card. The user utilizes one of the following security evaluated smart cards:

- ZKA Banking signature card, v6.2b NP and 6.2f NP, Type 3 from Giesecke & Devrient
- ZKA Banking signature card, v6.2 NP, Type 3 from Giesecke & Devrient
- ZKA Banking signature card v6.31 NP, Type 3 from Giesecke & Devrient
- ZKA Banking signature card v6.32, Type 3 from Giesecke & Devrient
- ZKA Banking Signature Card, Version 6.4 from Giesecke & Devrient
- ZKA Banking Signature Card, Version 6.51 from Giesecke & Devrient
- ZKA Banking Signature Card, v6.6 from Giesecke & Devrient
- ZKA signature card, version 5.02 from Gemplus-mids GmbH
- ZKA signature card, ZKA 680 V5A from Gemplus-mids GmbH
- ZKA signature card, version 5.11 from Gemplus-mids GmbH
- ZKA signature card, version 5.11 M from Gemplus GmbH (Gemalto)

[18] The manual installation of the Microsoft SmartCard base components is required for Microsoft Windows NT 4.0

- ZKA SECCOS Sig v1.5.3 from Sagem Orga GmbH
- ZKA Banking Signature Card, Version 7.1.2 from Giesecke & Devrient
- ZKA Banking Signature Card, Version 7.2.1 from Giesecke & Devrient
- ZKA signature card, ZKA 680 V6A from Gemplus GmbH (Gemalto)

In addition to the listed smart cards, the user utilizes any smart card that provides a PKCS#15 interface or a SigG-application for qualified electronic signatures.

The user utilizes one of the following smart-card terminals:

- Cherry ST-2000
- Fujitsu Siemens S26381-K329-V2xx HOS:01
- Kobil KAAN Advanced
- Kobil Systems B1 Pro USB
- Kobil TriB@nk
- Omnikey Cardman 3621
- Omnikey Cardman 3821
- Reiner SCT cyberJack e-com v2.0
- Reiner SCT cyberJack e-com v3.0
- Reiner SCT cyberJack e-com plus v3.0
- Reiner SCT cyberJack pinpad v2.0
- Reiner SCT cyberJack pinpad v3.0
- Reiner SCT cyberJack secoder
- SCM Microsystems SPRx32

The signature law approval according to the German signature law can be obtained from the Federal Agency (www.bundesnetzagentur.de). The signature law approval for the TOE contains the list of approved components that can be used with the TOE.

A.Personnel

The user, the administrator and the maintenance staff are trustworthy and follow the user guide of the TOE. Especially the user verifies the integrity of the TOE as described in the user documentation.

A.Network

The computer, where the TOE is installed, may have Internet access. In this case a firewall is used to ensure, that no system services or components are compromised through Internet attacks. In addition to this, the user utilizes a virus scanner, which is able to detect virus programs as well as backdoor programs and root kits. At least the virus scanner is able to inform the user about attacks or detected malicious programs.

A.Access

The computer, on which the TOE is installed, is located in an environment, where the user has full control about inserted storage devices and shared network storage places. The TOE is protected in such way, that it is not possible to access parts of the TOE or the TOE as a whole through existing network connections.

3.2 Threats to Security

The analysis of security threats to the TOE and to objects, which should be protected by the TOE were supplemented using the document “Maßnahmenkatalog für technische Komponenten nach dem Signaturgesetz”. It must be said, that this catalogue refers to the “old” signature law (from 22.07.1997). The usage of this catalogue is tolerable, because the arrangements can be implied to the current signature law.

In addition to that the CEN Workshop Agreement CWA 14170 “Security Requirements for Signature Creation Applications” has been used for the analysis of security threats.

All security threats assume an adversary with high attack potential. For the protection against the threats identified in this section the kind of exploited vulnerability is irrelevant because the identified threats should be prevented in general.

Objects that must be protected by the TOE are: the document that the user wants to sign (“user file”), a signed file and the TOE with its own data and files.

User File

A user file is a file that user decided to sign with the TOE and that is currently processed by the TOE. Currently processed means that the TOE holds a kind of reference to that file expressed by the directory and the name of the file.

Signed File

A Signed File is a file with an electronic signature.

TOE’s data and files

This term means all binary and configuration files of the TOE. The term data refers to files that are required to operate the TOE correctly, e.g. certificates.

T.DAT

Manipulation of a user file

The adversary manipulates a user file using any instrument and the manipulation is not detected.

This security threat is very general, because several scenarios are covered through this. The term *user file* has been defined in the section above. The adversary may manipulate the file using an appropriate file editor, a network tool or any other applicable mechanism. A manipulation covers random manipulations as well as systematic changes to the file.

T.SIG_DAT

Manipulation of a signed file

The adversary manipulates a signed file using any instrument and the manipulation is not detected.

This security threat is very general, because several scenarios are covered through this. The definition of the term *signed file* has been given in the section above. The adversary may manipulate the file using an appropriate file editor, a network tool or any other applicable mechanism. A manipulation covers random manipulations as well as systematic changes to the file.

T.TOE

Manipulation of the TOE and of its files

The adversary manipulates or replaces parts (modules) or data of the TOE on the computer and the manipulation is not detected.

The manipulation of TOE files or modules is a direct attack against the product. The adversary manipulates or changes parts of the TOE with the scope, to change some of the security functionality or even to deactivate this functionality of the TOE.

T.PRE_SIG

Manipulation of a file before the users decision to sign the file

The adversary changes the file using any mechanisms, before the user decides to sign the file and the manipulation are not detected.

The file is a file that the user wants to sign. The security threat implies an adversary who is able to change files in the time between the selection of the file[19] and the beginning of the signing process.

T.POST_SIG

Creation of a falsified electronic signature

The adversary manipulates the computed hash value of the document before the hash value is transmitted to the secure signature creation device and the manipulation is not detected.

The adversary is able to manipulate the hash value of the data to be signed in the time slice between the start of the signature process triggered by the user and the transmission to the smart card. It may be possible, to change the hash value of the data during transmission to the secure signature creation device.

T.LIC

Downgrading of TOE's manageable capabilities

[19] This is a user file as defined in the section above. The TOE holds a reference to file but the file is still not signed.

The adversary utilizes a license code or license file that disables parts of the TOE's manageable capabilities and this downgrading is not detected.

The adversary owns a license code or a license file for the TOE that enables a smaller set of capabilities of the TOE that are managed through the license code and enters this license code. The TOE would not provide the full set of functionality the user has licensed.

3.3 Organizational Security Policies

There are no organizational security policies defined for the TOE.

4 Security Objectives

4.1 Security Objectives for the TOE

OT.DAT

Protection of a user file

The TOE must offer the possibility of manipulation protection through the computation of hash values over the data of a file.

OT.SIG_DAT

Protection of a signed file

The TOE must offer the possibility to determinate, that it is unambiguous, whether a signed file has been manipulated or not.

OT.TOE

Protection of the TOE

The TOE must offer the possibility, to identify manipulations of TOE components or of TOE data.

OT.PRE_SIG

Protection of a file before the users decision, to sign the file

The TOE must offer the possibility to display the file and the data to be signed in an unambiguous way.

OT.POST_SIG

Protection against hash value falsification

The TOE must offer the possibility to identify, if the hash value of the data to be signed was manipulated after signature creation.

OT.LIC

Prevention of Downgrading of manageable TOE capabilities

The TOE must protect the integrity of the license through hash value computation and electronic signatures. The TOE must prevent a downgrade of the users license.

4.2 Security Objectives for the TOE Environment

The security objectives for the TOE environment are based on the secure usage assumptions and from security threat T.DAT.

OE.Platform

The user must utilize an Intel 586 compatible computer as hardware platform, which has at least 128 MB of RAM and 120 MB of free disk space.

On the computer one of the following operating systems has to be installed:

- Windows NT 4 SP 6
- Windows 2000 SP 2
- Windows 2003
- Windows 2003 64 Bit Edition
- Windows XP Home
- Windows XP Professional
- Windows XP Tablet PC Edition
- Windows XP 64 Bit Edition
- Windows Vista
- Windows Vista 64 Bit Edition
- Windows 2008
- Windows 2008 x64 Edition

Additionally the TOE supports a terminal server environment under Windows operating systems Windows 2000 with Citrix Metaframe, Windows 2003 with and without Citrix Metaframe and Windows 2008 without Citrix Metaframe.

In a terminal server environment the communication between server and client/s is conducted via an encrypted channel and is therefore to be considered trustworthy.

In order to use the 128 Bit encryption, the installation of the Windows 2000 High Encryption pack is compulsory for each Windows 2000 based computer that is in communication with the server. With Windows Terminal Server 2003 and 2008 in conjunction with the Operating System Windows XP and Windows Vista on the client side, the 128 Bit encryption is already standard and is to be used.

In addition to these requirements, the Internet Explorer version 5.01 or higher must be installed. Moreover, the Microsoft smart-card base components must be installed on the computer. In addition to that, a Java Virtual Machine (JVM) must be installed on the computer, which complies at least with the Java Runtime Environment v1.4.

The user must ensure that all components of the operating system are correct. The user must ensure that no harmful or malicious software is installed on the system.

The user must utilize a secure signature creation system, which consists of a smart-card terminal with secure pin entry capabilities together with a smart card. The user must utilize one of the following security evaluated smart cards:

- ZKA Banking signature card, v6.2b NP and 6.2f NP, Type 3 from Giesecke & Devrient
- ZKA Banking signature card, v6.2 NP, Type 3 from Giesecke & Devrient
- ZKA Banking signature card v6.31 NP, Type 3 from Giesecke & Devrient
- ZKA Banking signature card v6.32, Type 3 from Giesecke & Devrient
- ZKA Banking Signature Card, Version 6.4 from Giesecke & Devrient
- ZKA Banking Signature Card, Version 6.51 from Giesecke & Devrient
- ZKA Banking Signature Card, v6.6 from Giesecke & Devrient
- ZKA signature card, version 5.02 from Gemplus-mids GmbH
- ZKA signature card, ZKA 680 V5A from Gemplus-mids GmbH
- ZKA signature card, version 5.11 from Gemplus-mids GmbH
- ZKA signature card, version 5.11 M from Gemplus GmbH (Gemalto)

- ZKA SECCOS Sig v1.5.3 from Sagem Orga GmbH
- ZKA Banking Signature Card, Version 7.1.2 from Giesecke & Devrient
- ZKA Banking Signature Card, Version 7.2.1 from Giesecke & Devrient
- ZKA signature card, ZKA 680 V6A from Gemplus GmbH (Gemalto)

In addition to the listed smart cards, the user can utilize any smart card that provides a PKCS#15 interface or a SigG-application for qualified electronic signatures.

The user must utilize one of the following smart-card terminals:

- Cherry ST-2000
- Fujitsu Siemens S26381-K329-V2xx HOS:01
- Kobil KAAN Advanced
- Kobil Systems B1 Pro USB
- Kobil TriB@nk
- Omnikey Cardman 3621
- Omnikey Cardman 3821
- Reiner SCT cyberJack e-com v2.0
- Reiner SCT cyberJack e-com v3.0
- Reiner SCT cyberJack e-com plus v3.0
- Reiner SCT cyberJack pinpad v2.0
- Reiner SCT cyberJack pinpad v3.0
- Reiner SCT cyberJack secoder
- SCM Microsystems SPRx32

The signature law approval according to the German signature law can be obtained from the Federal Agency (www.bundesnetzagentur.de). The signature law approval for the TOE contains the list of approved components that can be used with the TOE.

OE.Personnel

The user, the administrator and the maintenance staff must be trustworthy and must follow the user guide of the TOE. Especially the user must verify the integrity of the TOE as described in the user documentation.

OE.Network

The computer, where the TOE is installed, may have Internet access. In this case a firewall must be used to ensure, that no system services or components are compromised through Internet attacks. In addition to this, the user must utilize a virus scanner, which is able to detect virus programs as well as backdoor programs and root kits. At least the virus scanner must be able to inform the user about attacks or detected malicious programs.

OE.Access

The computer, on which the TOE is installed, must be located in an environment, where the user has full control of inserted storage devices and shared network storage places. The TOE must be protected in such way, that it is not possible to access parts of the TOE or the TOE as a whole through existing network connections.

OE.SIG_DAT

Through the use of appropriate organizational measurements it must be ensured that the IT-environment offers the functionality to compute an electronic signature from a hash value.

5 IT-Security Requirements

5.1 TOE Security Functional Requirements

The specified functional requirements are compliant with Common Criteria v2.3 part 2 and are corresponding with the given functional components except the component FDP_SVR.1, which is defined in this security target.

FCS_COP.1 (SHA-1)[20]
Cryptographic Operation
FCS_COP.1.1

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

FCS_COP.1 (SHA-224)[21]
Cryptographic Operation
FCS_COP.1.1

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

FCS_COP.1 (SHA-256)[22]
Cryptographic Operation
FCS_COP.1.1

[20] Iteration

[21] Iteration

[22] Iteration

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

FCS_COP.1 (SHA-384)[23]
Cryptographic Operation
FCS_COP.1.1

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

FCS_COP.1 (SHA-512)[24]
Cryptographic Operation
FCS_COP.1.1

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

FCS_COP.1 (160)[25]
Cryptographic Operation
FCS_COP.1.1

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

[23] Iteration

[24] Iteration

[25] Iteration

FCS_COP.1 (RSA1-1024 to 2048)[26]

Cryptographic Operation

FCS_COP.1.1

The TSF shall perform [assignment: *verification of electronic signatures*] in accordance with specified cryptographic algorithm [assignment: *RSA*] and cryptographic key sizes [assignment: *1024 to 2048 bit*] that meet the following: [assignment: *standard PKCS#1, standard ISO9796-2*].

FCS_COP.1 (RSA2-1024 to 2048)[27]

Cryptographic Operation

FCS_COP.1.1

The TSF shall perform [assignment: *verification of electronic signatures*] in accordance with specified cryptographic algorithm [assignment: *RSA*] and cryptographic key sizes [assignment: *1024 to 2048 bit*] that meet the following: [assignment: *standard PKCS#1, standard ISO9796-2*].

Refinement for FCS_COP.1 (RSA2-1024 to 2048)

Part 1:

The TSF must validate the certificate chain using the chain model, if the validation model for the CA is based on the chain model. Therefore the standard ISIS-MTT Common ISIS MTT Mailtrust Specifications for Interoperable PKI Applications Version 1.02 in conjunction with the ISIS-MTT Optional Profile: SigG-Profile must be used.

Part 2:

[26] Iteration

[27] Iteration

The TSF must validate the certificate chain using the standard RFC 3280, if the validation model for the CA is not based on the chain model.

FCS_COP.1(ECC1) [28]

Cryptographic Operation

FCS_COP.1.1

The TSF shall perform [assignment: *verification of electronic signatures*] in accordance with specified cryptographic algorithm [assignment: *ECDSA*] and cryptographic key sizes [assignment: *p=192*[29]*, m=191*[30] *and q≥180*] that meet the following: [assignment: *standard X9.62, Technical Guideline TR-03111, Geeignete Algorithmen [17]*].

FCS_COP.1 (ECC2)[31]

Cryptographic Operation

FCS_COP.1.1

The TSF shall perform [assignment: *verification of electronic signatures*] in accordance with specified cryptographic algorithm [assignment: *ECDSA*] and cryptographic key sizes [assignment: *p=192*[32]*, m=191*[33] *and q≥180*] that meet the following: [assignment: *standard X9.62, Technical Guideline TR-03111, Geeignete Algorithmen [17]*].

Refinement for FCS_COP.1 (ECC2)

Part 1:

The TSF must validate the certificate chain by the use of the chain model if the Root or CA certificate is registered to be a certificate of an accredited or shown certificate service provider under the terms of the German digital signature law. Therefore the standard ISIS

[28] Iteration

[29] with

[30] with

[31] Iteration

[32] $E\left(F_p\right)$ with $ord(P) = q$

[33] $E\left(F_{2^m}\right)$ with $ord(P) = q$

MTT Mailtrust Specifications for Interoperable PKI Applications Version 1.02 in conjunction with the ISIS-MTT Optional Profile: SigG-Profile must be used.

Part 2:

The TSF must validate the certificate chain using the standard RFC 3280 if the root or CA certificate is not registered to be an accredited or shown certificate service provider in the sense of the German digital signature law.

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.

FDP_ITC.1(1)[34]
Import of 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

[34] Iteration

The TSF shall ignore any security attributes associated with 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: *Rules for the usage of a certificate for verification after signature creation*].

Rules for the usage of a certificate for verification after signature creation

After the creation of an electronic signature, the TSF must decrypt the electronic signature using the public key of the user certificate and compare the operation result with the cryptographic checksum (hash value), which was used as the initial value for signature creation (comparison for mathematical correctness of the electronic signature).

FDP_ITC.1 (2)

Import of 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 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: *Rules for the usage of actual certificates from the network*].

Rules for the usage of actual certificates from the network

During the process of signature verification the TSF must use the available certificate revocation lists to verify the validity of a given certificate at the time of signature creation.

FDP_ITC.2 (1)

Import of 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 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: *Rules for the Import of OCSP responses*].

Rules for the Import of OCSP responses

The TSF must verify the validity of the signature on the OCSP response mathematically before importing the OCSP response.

FDP_ITC.2 (2)

Import of 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 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: *Rules for the Import of timestamps*].

Rules for the Import of timestamps

The TSF must verify the validity of the signature on the timestamp mathematically before importing the timestamp.

FTP_ITC.1

Inter-TSF trusted channel

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*].

FMT_MOF.1(1)

Management of Security Functions Behaviour

FMT_MOF.1.1

The TSF shall restrict the ability to [selection: *modify the behaviour of*] the functions [assignment: *PDF displaying capabilities, signature creation capabilities*] to [assignment: *none*].

FMT_SMF.1(1)

Specification of Management Functions

FMT_SMF.1.1

The TSF shall be capable of performing the following security management functions [assignment: *enabling of initiation of electronic signature creation, enabling of PDF displaying, enabling PDF PKCS#7 formatted signing, enabling of PDF signing using PDF signatures*].

FMT_MOF.1(2)

Management of Security Functions Behaviour

FMT_MOF.1.1

The TSF shall restrict the ability to [selection: *modify the behaviour of*] the functions [assignment: *changing the TOE's licensed capabilities*] to [assignment: *user*].

FMT_SMF.1(2)

Specification of Management Functions

FMT_SMF.1.1

The TSF shall be capable of performing the following security management functions [assignment: *accept only equal or upgraded licensed functionality*].

FDP_SVR.1

Secure Viewer

FDP_SVR.1.1

The TSF shall ensure, that the displayed content of a document is unambiguous according to [assignment: *TIFF Revision 6.0 final draft June 3, 1992, Adobe Developers Association, User Guidance S-TRUST Sign-it base components 2.5, subset of PDF reference sixth edition (Adobe Portable Document Format Version 1.7)]*.

FDP_SVR.1.2

The TSF shall ensure, that the displayed content of a document is free of hidden and active content. The TSF shall ensure, that the user is informed about hidden or active content.

FDP_SVR.1.3

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

5.2 TOE Security Assurance Requirements

The following table offers an overview of the documents, which describe, how the requirements are fulfilled. The assurance requirements are defined according to EAL4 augmented (Common Criteria part 3). All assurance requirements were taken from Common Criteria part 3.

Class	Family	Document
Configuration Management	ACM_CAP.4	A configuration management system is used to manage all versions of all TOE parts. In addition to that task, the configuration management system is used to ensure, that no unauthorized modifications on the TOE take place. The configuration control system is used to ensure, that each version of the implementation, design, test and documentation is logged. Details are described in the document “Configuration Management”, which is part of the manufacturer documentation for the evaluation.
	ACM_SCP.2	
	ACM_AUT.1	
Delivery and Operation	ADO_DEL.2	This aspect is handled in the document “Delivery Procedures”.
	ADO_IGS.1	These aspects are documented in the user guide.

Class	Family	Document
Development	ADV_FSP.2	This aspect is handled in the document “Informal functional specification” provided by the manufacturer for the evaluation.
	ADV_HLD.2	This aspect is handled in the document “High Level Design” provided by the manufacturer for the evaluation.
	ADV_IMP.1	The relevant source code is available for the evaluation.
	ADV_LLD.1	This aspect is handled in the document “Low Level Design” provided by the manufacturer for the evaluation.
	ADV_RCR.1	
	ADV_SPM.1	This aspect is handled in the document “Security Policy Model”
Guidance documents	AGD_ADM.1	The user and developer guide for the product is very detailed and provides all required information for secure installation, management and use.
	AGD_USR.1	
Life Cycle Support	ALC_DVS.1	The security of the development environment is ensured through physical and personnel activities.
	ALC_LCD.1	This aspect is handled in the document “Life Cycle Support”
	ALC_TAT.1	Well-defined tools (compilers etc.) are used for the development of the TOE. Details are documented in “Configuration Management”, which is part of the manufacturer documentation for evaluation.

Class	Family	Document
Tests	ATE_COV.2	Well defined test procedures are used: • Tests according to the functional specification • Tests on subsystem level • Tests of all security functions
	ATE_DPT.1	
	ATE_FUN.1	
	ATE_IND.2	This is the task of the evaluator.
Vulnerability assessment	AVA_MSU.3	An internal review process ensures, that the documentation is clear, consistent and reasonable.
	AVA_VLA.4	For every mechanism, that has an SOF postulation, an appropriate analysis is done and documented.
	AVA_SOF.1	An analysis with the scope to identify vulnerabilities is done and documented.

Table 1 Documentation Overview

5.3 Security Requirements for the IT-Environment

FCS_COP.1 (ES-1024 to 2048)[35]

Cryptographic Operation

FCS_COP.1.1

The TSF shall perform [assignment: *computation of electronic signatures*] in accordance with specified cryptographic algorithm [assignment: *RSA*] and cryptographic key sizes [assignment: *1024 to 2048 bits*] that meet the following: [assignment: *standard PKCS#1, standard ISO9796-2*].

FCS_COP.1 (ES ECC)[36]

Cryptographic Operation

FCS_COP.1.1

The TSF shall perform [assignment: *computation of electronic signatures*] in accordance with specified cryptographic algorithm [assignment: *ECDSA*] and cryptographic key sizes [assignment: *p=192*[37]*, m=191*[38] *and q≥180*] that meet the following: [assignment: *standard X9.62, Technical Guideline TR-03111, Geeignete Algorithmen [17]*].

[35] Iteration

[36] Iteration

[37] $E\left(F_p\right)$ with $ord(P) = q$

[38] $E\left(F_{2^m}\right)$ with $ord(P) = q$

6 TOE Summary Specification

6.1 TOE Security Functions

SF.1

Hash value computation and initiation of the electronic signature creation process using certificates, smart-card terminals and secure signature creation devices.

The TOE computes hash values of any file or data buffer using the SHA algorithm family[39] (as required 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(160), standards are listed in SFR's). After the hash value computation, the TOE uses the PC/SC or CT API or a card terminal vendor specific module to initiate the electronic signature creation by a secure signature creation system, which consists of a smart-card terminal and a smart card. The electronic signature is computed using the RSA or ECDSA algorithm, which is implemented as a part of the SSCD's functionality[40]. The TOE adds the signer certificate to the resulting documents. Through the electronic signature, the data authentication is ensured. Through the addition of the signer certificate, the possibility of data verification is offered.

Before the computation of the hash value starts, the TOE displays an unambiguous message[41], that electronic signatures should be created. It is unambiguous, to which data each electronic signature refers[42]. Through the combination of a secure pin entry device and

[39] The concrete SHA algorithm to be used can be set using the OPENLiMiT SignCubes Job Interface API. If no algorithm identifier is set, SF.1 uses an appropriate hash algorithm according to the latest crypto catalogue [17] as the default algorithm.

[40] The encryption of the hash value is part of the security requirements for the IT-environment. Algorithm and key size is specified in FCS_COP.1(ES-1024 to 2048) and FCS_COP.1(ES-ECC).

[41] A default signature request dialog or an advanced dialog with textual preview is used (controlled by the performed job of the S-TRUST Sign-it job interface). In case of signature request dialog with textual preview showing signed input data are expected which are verified cryptographically as well as against a special trust list.

[42] In case of default signature request dialog showing the user has the possibility to get signature creation details through pressing an appropriate button.

a smart card it is guaranteed that authorized persons only perform the electronic signature creation and identification data is not abandoned.

The user has the possibility to include an OCSP response for the user certificate that is used for the creation of the electronic signature into the resulting PKCS#7 or XML encoded signature file or data buffer.

The TOE offers the possibility to add a timestamp into the PKCS#7 or XML encoded file or data buffer that is the output of this TSF.

The TOE offers the capability to create more than one electronic signature by the execution of a special job via the S-TRUST Sign-it Job Interface. This capability is not accessible via the graphical interface of the TOE.

If the user configures the TOE to leave the card open, he is only one time requested to enter PIN for the execution of that job. The configuration of the TOE is performed via a configuration dialog offered by the S-TRUST Sign-it Security Environment Manager. This dialog allows the differentiation between the amount of time that the PIN is left open and the number of electronic signatures to be created. The first criterion that is reached closes the PIN and the user is required to enter the PIN again. The S-TRUST Sign-it Security Environment Manager ensures that exactly the files presented by the signature creation dialog are signed. If the user cancels the creation of signatures, all signature files created by this job are deleted. In any case, it is ensured by the S-TRUST Sign-it base components 2.5 that the PIN is closed when the job has been executed.

This security function requires a TOE configuration that contains a smartcard terminal module, a smartcard operating system module and a smartcard profile module. Otherwise the operation fails with an unambiguous error message.

All used mechanisms are of cryptographic nature. Therefore, no comment about the strength of the mechanisms can be given. The used cryptographic mechanisms are identified as usable according to §17 paragraph 1 to 3 SigG 22.05.2001 in conjunction with annex 1, paragraph 1 no. 2 SigV 22.11.2001[43].

[43] The applicability of the cryptographic mechanisms according to the German signature law and the ordinance on electronic signatures is subject to change.

SF.2

Verification of hash values and electronic signatures using certificate revocation lists, OCSP responses (optional) and timestamps (optional)

The TOE is able to verify electronic signatures that are based on a SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 or RIPE-MD 160 hash value. In this process it is unambiguous, to which data the electronic signature refers. For the purpose of electronic signature verification, the hash value of the signed data is computed, using the SHA algorithm family or the RIPE-MD 160 algorithm (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) and FCS_COP.1(160)) and the original hash value extracted from the signature using the RSA algorithm[44] (FCS_COP.1 (RSA2-1024 to 2048)) or the ECDSA algorithm (FCS_COP.1(ECC2)) and the public key of the given signer certificate. In addition to this operation, the certificate chain is checked, using the chain model or RFC 3280 (iteration of FCS_COP.1(ECC2) and iteration of FCS_COP.1 (RSA2-1024 to 2048))[45].

The TOE displays an unambiguous message, whether the hash values were identical or not. (FDP_DAU.2 with focus on FDP_DAU.2.2). Because of this, it is unambiguous, whether the original data has changed or not. The correctness of the electronic signature is reliably checked and displayed. Through the use of the S-TRUST Sign-it Viewer component it is ensured, that the content of the signed data is unambiguously displayed.

The TOE offers the possibility to identify the user that has generated the electronic signature based on the certificate that has been used. The user, who verifies the validity of the electronic signature by using the TOE, has the possibility to view the certificate that has been used for the generation of the electronic signature in an unambiguous way. This covers the requirements defined by FDP_DAU.2, especially FDP_DAU.2.2.

[44] In this process, the whole PKCS#1 block v.1.5 with padding type 01 is checked. Alternatively the padding can be ISO 9796-2 conform.

[45] When validating qualified certificates issued by a shown or accredited German certification authority. All other signature types are validated by the use of the shell model.

During the verification process, the issuer of the signers certificate is determined and a corresponding certificate revocation list is loaded. This revocation list is checked, if a revocation entry for the signers certificate exists. If this is the case, this information is taken (FDP_ITC.1 (2)). If the certificate was already revoked during the signature creation, this information is displayed to the user.

In addition to the checking of the revocation list the user has the possibility to use an OCSP response to verify the validity of a certificate under examination[46]. The OCSP response may be encoded in the signature data under examination or is requested from an OCSP responder using software modules that are not part of the evaluation.

Also the user has the possibility to use a given time stamp that is encoded in the signature data as the point of time where the signature has been created. The time stamp may be part of the PKCS#7 encoded data under examination or is presented to the TOE as a separate file[47]. The timestamp can be used by the TOE to provide validity information for the signature under examination at the point of time that is specified by the timestamp.

The basic validity checking is always done using the time that is encoded in the signature block. This time is normally the system time when the signature has been created. If no time of signature creation is available, the current system time is used as the point of time of signature creation.

All used mechanisms are of cryptographic nature. Therefore, no comment about the strength of the mechanisms can be given. The used cryptographic mechanisms are identified as usable according to §17 paragraph 1 to 3 SigG 22.05.2001 in alliance with annex 1, paragraph 1 no. 2 SigV 22.11.2001[48].

[46] SF.7 provides more information.

[47] SF.8 provides more information.

[48] The usability of the cryptographic mechanisms according to the German signature law and the ordinance on electronic signatures is subject to change.

SF.3

Program module manipulation detection

The TOE is delivered with electronic signed libraries, files and executables. In order to implement the required functionality of manipulation detection, a separate program module is implemented as dynamic linked library (dll). All subsystems of the product know the SHA-512 hash value of this check module. The check module knows the public key, whose counterpart (the private key) was used to sign the program libraries, files and executables.

If the S-TRUST Sign-it base components are started, the S-TRUST Sign-it Security Environment Manager checks its environment using the check module. In step 1, the Security Environment Managers validates the hash value of the check module (FCS_COP.1 (SHA-512)). In step 2, the check module verifies the application, which loads the check module, by verifying the electronic signature of the loading application mathematically. (FCS_COP.1 (SHA-512) and FCS_COP.1(RSA1-1024 to 2048)).

If a dynamic module should be loaded by the application, the check module is always used to verify the integrity of the module to be loaded. Therefore the check module computes the hash value of the module to be loaded and verifies the electronic signature of the module to be loaded mathematically (FCS_COP.1 (SHA-512) and FCS_COP.1(RSA1-1024 to 2048)). If the verification fails, the check module sends a signal to all program modules, which are now deactivated. Using this mechanism, no security related operation could be performed using the product.

All modules of the application know the hash value of the check module. If the hash value of the check module cannot be validated (FCS_COP.1(SHA-512)), the modules can not longer be used.

All used mechanisms are of cryptographic nature. Therefore, no comment about the strength of the mechanisms can be given. The used cryptographic mechanisms are identified as usable according to §17 paragraph 1 to 3 SigG 22.05.2001 in alliance with annex 1, paragraph 1 no. 2 SigV 22.11.2001.

SF.4

Unambiguous presentation of the data to be signed

The TOE ensures that the content of displayed document is unambiguous as required by FDP_SVR.1.1. In addition to this, the user is informed, if the data contains hidden or active content or content that cannot be displayed. (FDP_SVR.1.2 and FDP_SVR.1.3).

The file that should be signed must be a Text, Tiff or PDF[49] formatted file. An appropriate parser explores the type of the data. If the parser is unable to determine the type of the data an appropriate error message is displayed that contains a hint that the data could not be displayed as required by FDP_SVR.1. After this first operation the data is checked for hidden and active content. If the file contains unknown tags, elements or control characters the user is informed that the file may contain unintended content as required by FDP_SVR.1.2. If the parser detects active or hidden content an appropriate message is displayed to the user that informs him about this state as required by FDP_SVR.1.2 and FDP_SVR.1.3. If the user wants to display the file and the file contains hidden content or content, that cannot be displayed, a warning message is generated and displayed to the user.

Security function SF.4 does not contain permutational or probalistic mechanisms. Therefore no strength of function is postulated.

[49] The TOE does implement a subset of the PDF reference fifth edition. Not all requirements of this reference have been implemented in order to fulfill the requirements of FDP_SVR.1. More information is provided by the user guidance.

SF.5

Protection against hash value manipulation

Before the process of electronic signature creation starts, the hash value using the SHA algorithm family is used[50] (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)). Alternatively the RIPE-MD 160 algorithm may be used (FCS_COP.1(160)). After the electronic signature creation process, the TOE verifies the electronic signature using the public key of the given signer certificate (FDP_ITC.1(1) and FCS_COP.1(RSA1-1024 to 2048)[51] or FCS_COP.1(ECC1)). If the original hash value and the hash value encoded in the electronic signature are not identical, the hash value was corrupted during the transmission to the secure signature creation device. After this operation, an unambiguous message is displayed, if the correct data has been signed.

By comparing the hash value that was meant to be used for the creation of the electronic signature and the hash value that was used by the SSCD for the generation of the electronic signature the requirements of FTP_ITC.1 are fulfilled.

All used mechanisms are of cryptographic nature. Therefore, no comment about the strength of the mechanisms can be given. The used cryptographic mechanisms are identified as usable according to §17 paragraph 1 to 3 SigG 22.05.2001 in alliance with annex 1, paragraph 1 no. 2 SigV 22.11.2001[52].

[50] The hash algorithm to be used can be set using the S-TRUST Sign-it Job Interface API. If no hash algorithm is set, an appropriate hash algorithm according to the latest crypto catalogue [17] will be used by default.

[51] The key size depends on the key size of the certificate that has been used for the creation of the electronic signature.

[52] The usability of the cryptographic mechanisms according to the German signature law and the ordinance on electronic signatures is subject to change.

SF.6

Assurance of the TOE's integrity

The integrity of the S-TRUST Sign-it base components 2.5 can be ensured by the user using the check utility, which could be accessed online. This check utility is a Java Applet, which ensures the integrity of the application by checking and comparing the SHA-256 hash values of the program modules (FCS_COP.1 (SHA-256)). Therefore the hash values are known to the Applet.

The Java Applet is a signed Java Archive (with the extension JAR), the integrity of the Applet itself is ensured by the mechanisms of the Java Virtual Machine (JVM). Therefore the user is required to install a Java Virtual Machine. The JVM must be at least compatible with the Java Runtime Environment 1.4 from Sun.

The user must use a dialog to point to the current installation path of the product or uses the installation directory detected by the S-TRUST Sign-it Integrity Tool from the registry. The integrity check does not verify any registry entries.

If the computed hash values and the expected hash values of the program modules are not identical, an error message is generated and displayed to user in an unambiguous way. If no differences in the configuration were detected, the check utility displays an unambiguous dialog with an appropriate summary.

The Integrity Tool does always check a complete list[53] of the TOE binary files. If files are missing[54], the user is informed by an appropriate message. The card terminal modules as well as the smart card modules are not necessarily required items of a valid TOE. Therefore the Integrity Tool informs the user about all installed modules. The Integrity Tool checks whether the IBM Lotus Forms Viewer plugin is installed or not. If this plugin is installed the user is informed by an appropriate message.

[53] This list represents a maximum set of verifiable files.

[54] The Integrity Tool knows which files are optional respectively could be missed to still ensure the integrity of the TOE. The user will be informed about the TOE integrity state in any case.

The Integrity Tool checks the configuration of the supported secure signature creation devices and displays the configuration information to the user[55].

The possibility to get a valid integrity in case files are missing has no misuse capabilities. It is not possible to add modules the integrity tool doesn't know.

All used mechanisms are of cryptographic nature. Therefore, no comment about the strength of the mechanisms can be given. The used cryptographic mechanisms are identified as usable according to §17 paragraph 1 to 3 SigG 22.05.2001 in alliance with annex 1, paragraph 1 no. 2 SigV 22.11.2001.

[55] Especially the used hash algorithms are checked. If the SSEE is configured for the use of SHA-1, the Integrity Tool generates a hint that the user is required to check the website of the Bundesnetzagentur to check, if this signature creation mode is still allowed.

SF.7

Processing of OCSP information for certificate validation

The TOE offers the possibility to process OCSP information in order to verify the validity of a certificate. This certificate is called ‘certificate under examination’. In order to use the validity information that is provided for the certificate under examination through the OCSP response, the TOE is required to verify the electronic signature of that response.

The OCSP response might be part of a signature, a plain file or is received through an appropriate OCSP handler. In the process of OCSP response import the TOE verifies the validity of the OCSP response through the mathematical verification of the electronic signature that belongs to the OCSP response[56] (FDP_ITC.2 (1)). Mathematical verification means that the OCSP response must contain a signature from the certificate that is included in the OCSP response as the OCSP response signing certificate and the signature must be valid. During the import the TOE does not check the complete certificate chain. The signature might be based on the same algorithms as specified in the next section (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) and FCS_COP.1(160)),(FCS_COP.1 (RSA1-1024 to 2048)) and (FCS_COP.1 (ECC1)). If the TOE is not able to verify the signature, the OCSP response is not imported and an appropriate message is displayed to the user. After the import process the TOE is also able to store the OCSP response in a PKCS#7 encoded file[57] or in a separate file.

[56] The SFR’s that define the algorithms and key sizes are defined in the section below.

[57] During signature creation

When the TOE uses the OCSP information for certificate validation a complete verification of the OCSP signature based on certificate revocation lists is required. The signature might be based on a SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 or RIPE-MD 160 hash value. For the purpose of electronic signature verification, the hash value of the signed data is computed, using the SHA algorithm family or the RIPE-MD 160 algorithm (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) and FCS_COP.1(160)) and the original hash value extracted from the signature using the RSA algorithm[58] (FCS_COP.1 (RSA2-1024 to 2048)) or the ECDSA algorithm (FCS_COP.1 (ECC2) and the public key of the OCSP response signing certificate. In addition to this operation, the certificate chain for that OCSP signing certificate is checked, using the chain model or RFC 3280 ((iteration of FCS_COP.1 (RSA2-1024 to 2048)) and (iteration of FCS_COP.1 (ECC2)). All certificates in the chain are checked for revocation information that is provided by appropriate CRL's (FDP_ITC.1(2)).

After the validation of the electronic signature the TOE displays an unambiguous dialog to the user that informs him about the validity of the OCSP response and the validity information for the certificate under examination extracted from that OCSP response. In addition to that the TOE provides a second dialog that provides detailed information about the content of the OCSP response, including the certificate that has signed the response (FDP_DAU.2 with focus on FDP_DAU.2.2).

The validity information that is based on the OCSP response for the certificate under examination might be used in the process of signature verification (SF.2) to determine the validity of a certificate under examination.

All used mechanisms are of cryptographic nature. Therefore, no comment about the strength of the mechanisms can be given. The used cryptographic mechanisms are identified as usable according to §17 paragraph 1 to 3 SigG 22.05.2001 in conjunction with annex 1, paragraph 1 no. 2 SigV 22.11.2001.[59]

[58] In this process, the whole PKCS#1 block v.1.5 with padding type 01 and padding considering to ISO 9796-2 is checked

[59] The usability of the cryptographic mechanisms according to the German signature law and the ordinance on electronic signatures is subject to change.

SF.8

Application of Timestamps

The TOE offers the possibility to apply timestamps to files. Therefore an external timestamp handler is used in order to receive the timestamp, the correctness of the timestamp itself is ensured by the TOE.

In the process of timestamp import the TOE verifies the electronic signature of the timestamp using the public key of the certificate that has signed the timestamp[60] (FDP_ITC.2(2)). The certificate must be known to the TOE in order to verify the signature, otherwise the TOE would not import the timestamp[61].

The signature might be based on a SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 or RIPE-MD 160 hash value. For the purpose of the mathematical validation, the hash value of the signed data is computed, using the SHA algorithm family or the RIPE-MD 160 algorithm (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) and FCS_COP.1(160)) and compared with the original hash value extracted from the signature using the RSA algorithm[62] (FCS_COP.1 (RSA1-1024 to 2048)) or the ECDSA algorithm (FCS_COP.1 (ECC1)) and the public key of the timestamp signing certificate.

After importing the timestamp, the timestamp is applied[63] to the file that has been chosen by the user. Application means the encoding of the timestamp into an existing signature file or the storage of the timestamp in a separate file.

All used mechanisms are of cryptographic nature. Therefore, no comment about the strength of the mechanisms can be given. The used cryptographic mechanisms are identified as

[60] Mathematical verification of the signature.

[61] The certificate must be part of the PKD files that are located in the installation directory of the TOE or must be located in the Operating Systems Certificate Store.

[62] See footnote 58

[63] The timestamp can then be part of a PKCS#7 signature block or be a separate file that contains the timestamp that has been received.

usable according to §17 paragraph 1 to 3 SigG 22.05.2001 in conjunction with annex 1, paragraph 1 no. 2 SigV 22.11.2001.[64]

SF.9

Validation of Timestamps

In the process of timestamp validation the TOE verifies the electronic signature of the timestamp with the public key that has signed the timestamp. The certificate must be known to the TOE in order to verify the signature; otherwise the TOE is unable to validate the timestamp[65]. The signature might be based on a SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 or RIPE-MD 160 hash value. For the purpose of the mathematical validation, the hash value of the signed data is computed, using the SHA algorithm family or the RIPE-MD 160 algorithm (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) and FCS_COP.1(160)) and compared with the original hash value extracted from the signature using the RSA algorithm[66] (FCS_COP.1 (RSA2-1024 to 2048)) or the ECDSA algorithm (FCS_COP.1(ECC2)) and the public key of the timestamp signing certificate. In addition to this operation, the certificate chain for that timestamp signing certificate is checked, using the chain model or RFC 3280 ((iteration of FCS_COP.1 (RSA2-1024 to 2048)) and (iteration of FCS_COP.1(ECC2)). All certificates in the certificate chain are checked for revocation information provided by appropriate CRL's (FDP_ITC.1(2)).

The result of the timestamp validation is displayed to user through a dialog that is provided by the TOE. This dialog provides the information that is encoded in the timestamp and does also provide the capability to display the certificate that signed the timestamp (FDP_DAU.2 with focus on FDP_DAU.2.2).

All used mechanisms are of cryptographic nature. Therefore, no comment about the strength of the mechanisms can be given. The used cryptographic mechanisms are identified as

[64] The usability of the cryptographic mechanisms according to the German signature law and the ordinance on electronic signatures is subject to change.

[65] The validation operation would be not being executed.

[66] See footnote 58

usable according to §17 paragraph 1 to 3 SigG 22.05.2001 in conjunction with annex 1, paragraph 1 no. 2 SigV 22.11.2001.

SF.10

Management of Security Functions depending on licenses

The TOE implements a licensing mechanism that allows the management of security functionality in the product. (FMT_SMF.1(1))

Each time the TOE is started it validates its license. If no license file is found, the TOE displays a dialog and requests the user to enter a license number that he received together with the TOE's setup routine. The license number encodes information about the product capabilities under the aspect of PDF displaying and the ability to initiate the computation of an electronic signature. The electronic signature initiation and PDF operating modes[67] that depend on the license are defined as following:

- **Mode 1:** No signature creation capabilities of the TOE, if the user does not have a license code. The displaying of PDF documents is not allowed.
- **Mode 2:** No PDF displaying capabilities of the S-TRUST Sign-it Viewer. No possibilities to create any kind of signature on PDF documents. The S-TRUST Sign-it Viewer can be utilized to display Text and TIFF documents and to create signatures on these document types.
- **Mode 3:** PDF displaying but no capabilities to create signatures on PDF documents using the S-TRUST Sign-it Viewer. All kinds of supported signature formats and embedded PDF signatures on PDF documents can be verified. The S-TRUST Sign-it Viewer can be utilized to display Text and TIFF documents and to create signatures on these document types.
- **Mode 4:** PDF displaying and the capability to create signatures[68] as well as the verification of these signatures using the S-TRUST Sign-it Viewer[69]. The S-TRUST Sign-it Viewer can be utilized to display Text and TIFF documents and to create

[67] The verification of a PDF signature is also an PDF operating mode.

[68] With the exception of embedded PDF signatures

[69] Embedded PDF signatures can also be verified

signatures on these document types. The capability to create PDF signatures is not enabled.

- **Mode 5:** PDF displaying is enabled, the creation signatures for PDF documents is enabled as well as the verification of these signature types. The S-TRUST Sign-it Viewer can be utilized to display Text and TIFF documents and to create PKCS#7 signatures on these document types.

The following table provides a clear overview for the different license modes.

Mode	**Electronic Signature Initiation via S-TRUST Sign-it Viewer**	**Displayed Document Format**	**Possible Signature Format to be verified**	**Electronic Signature Initiation via OPENLiMiT SignCubes Job Interface**
1	No	TIFF, Text	PKCS#7, XMLDSIG	No
2	Yes (PKCS#7 encoded only)	TIFF, Text	PKCS#7, XMLDSIG	Yes
3	Yes (only PKCS#7 encoded for TIFF and Text)	TIFF, Text, PDF	PKCS#7, XMLDSIG and PDF	Yes
4	Yes (PKCS#7 encoded only)	TIFF, Text, PDF	PKCS#7, XMLDSIG and PDF	Yes
5	Yes, PDF and PKCS#7 signatures supported	TIFF, Text, PDF	PKCS#7, XMLDSIG and PDF	Yes

Table 2 License Modes in S-TRUST Sign-it base components 2.5

The process of generating a license file is handled as following: If the user has entered the license code, the TOE uses unambiguous hardware information[70] and generates an SHA-1 hash value of this data (FCS_COP.1 (SHA-1)). After this operation the license code is taken and encoded into an XML encoded file together with the hash value that has been computed. This step is the generation of the license file. After the generation of the license file the TOE

[70] The hard disks serial number connected to a logical drive.

displays an unambiguous dialog that informs the user that he is now required to create an electronic signature on the license file. In the case of an already existing license file, the TOE will only accept the license information that is equal or upgraded functionality (FMT_MOF.1(2), FMT_SMF.1(2)).

If the user has not entered a license code, the TOE extracts a license file that enables Mode 1. In this case the generation of an electronic signature is not required. This license file is signed by OPENLiMiT SignCubes and uses no hardware binding, because for Mode 1 this binding is not required[71].

After the hash value computation over the resulting license file, the TOE uses the PC/SC or CT API or a card terminal vendor specific module to initiate the electronic signature creation by a secure signature creation system, which consists of a smart-card terminal and a smart card. The electronic signature is computed using the RSA algorithm, which is implemented as a part of the SSCD's functionality[72]. The TOE adds the signer certificate to the resulting document. Through the electronic signature, the data authentication is ensured. Through the addition of the signer certificate, the possibility of data verification is offered and the evidence of the user who has signed the license file is added (FDP_DAU.2 with focus on FDP_DAU.2.2).

When the TOE is started the next time it verifies the licensing information together with the binding on the hardware. Therefore the TOE verifies the electronic signature on the license file with the public key that has been used for the generation of the electronic 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(160), FCS_COP.1(RSA1-1024 to 2048) and FCS_COP.1 (ECC1)). In conjunction with this operation the TOE sets up the certificate chain

[71] Otherwise the TOE would be required to generate a license file using private key material in a PKCS#12 file.

[72] The encryption of the hash value is part of the security requirements for the IT-environment. Algorithm and key size is specified in FCS_COP.1(ES-1024 to 2048) respectively FCS_COP.1 (ECC1).

up to a trusted certificate that must be located in a certificate trust list file[73] that contains all root and CA certificates that are trusted as certificates for issuing user certificates to be accepted by the TOE. (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(160), FCS_COP.1(RSA1-1024 to 2048) and FCS_COP.1 (ECC1)).

After this operation the found licensing information is used to enable the capabilities that have been defined to be manageable in the TOE (FMT_MOF.1(1)).

The TOE offers an appropriate dialog that informs the user about:

- The licensed functionality
- The serial number
- The certificate that has been used to enable the license on the users computer

The same dialog provides the functionality to update the license by repeating the licensing procedure of the TOE. In the case that the user has utilized license mode 1 and the user repeats the licensing process or generates the license for the first time, the TOE allows the creation of an electronic signature for that purpose.

The certificate that has been used for the creation of the electronic signature provides the evidence of the user that has enabled the product features (FDP_DAU.2 with focus on FDP_DAU.2.2).

The OPENLiMiT SignCubes Job Interface is allowed to utilize the TSF of the TOE, even if these capabilities are not part of the users license. Therefore the TOE contains a certificate in its resources[74] that is used in this case to verify the electronic signature of that external

[73] This certificate trust list file is also electronically signed with a key that is known to the TOE. This key is an OPENLiMiT internal key and is only used for that purpose.

[74] The certificate does only contain the public part. The private key material is located on an SSCD. Through the electronic signature on the TOE's binaries that also protects the TOE's resource files it is not possible to change this certificate without detection by the TOE.

module (FCS_COP.1(SHA-256) in conjunction with FCS_COP.1(RSA1-1024 to 2048)). This external module is not signed with the same key as the TOE.

The functionality of enabling a license code requires a TOE configuration to be installed that contains in minimum a smartcard terminal module, a smartcard operating system and a smartcard profile module. Otherwise the mechanism of upgrading the license is not applicable.

Additional Information for Microsoft Windows Terminal Server envirnoment

Differently from the license file described above the license file for a Microsoft Windows Terminal Server environment contains user specific domain information of his Terminal Server environment. The license file equals structurally to a hard disk bonded license file.

All used mechanisms are of cryptographic nature. Therefore, no comment about the strength of the mechanisms can be given. The used cryptographic mechanisms are identified as usable according to §17 paragraph 1 to 3 SigG 22.05.2001 in conjunction with annex 1, paragraph 1 no. 2 SigV 22.11.2001[75].

6.2 Assurance Measures

The chapter 5.2 contains a table, which lists all documents that describe the assurance measures.

7 PP Claims

No PP Claim is given for the TOE.

7.1 PP Reference

N/A

[75] The usability of the cryptographic mechanisms according to the German signature law and the ordinance on electronic signatures is subject to change.

7.2 PP Refinements

N/A

7.3 PP Additions

N/A

8 Rationale

8.1 Security Objectives Rationale

Threats and Assumptions vs. Security Objectives	OT.DAT	OT.SIG_DAT	OT.TOE	OT.PRE_SIG	OT.POST_SIG	OT.LIC	OE.Platform	OE.Personnel	OE.Network	OE.Access	OE.SIG_DAT
T.DAT	x	x									x
T.SIG_DAT		x									
T.TOE			x								
T.PRE_SIG				x							
T.POST_SIG					x						
T.LIC						x					
A.Platform							x				
A.Personnel								x			
A.Network									x		
A.Access										x	

Table 3 Threats and Assumptions vs. Objectives

A.Personnel ensures that the user and administration staff is reliable and trustworthy. In addition, this assumption ensures that the user validates the TOE's integrity. The security objective OE.Personnel defines the same requirements. Therefore it accomplishes the requirement.

A.Platform ensures that the operating system meets the requirements and all components and installed software is trustworthy. The security objective OE.Platform defines the same requirements. Therefore it accomplishes the requirement.

A.Network ensures that no system services or components are compromised through access from the Internet. This ensures the trustworthiness of the environment. OE.Network defines the same requirements. Therefore it accomplishes the requirement.

A.Access ensures that the user has full control about data carriers and shared network places. OE.Access defines the same requirements. Therefore it accomplishes the requirements.

T.DAT defines the security threat, that an adversary manipulates a user file using any mechanisms and the manipulation keeps undetected. OT.DAT defines as a security objective, the hash value computation corresponding to a given file or data buffer. OE.SIG_DAT ensures, that an electronic signature is computed on the basis of the previously computed hash value. OT.SIG_DAT defines the security objective, that the TOE shall offer the possibility, to detect the manipulation of the content of a file or data buffer. All mechanisms together ensure that user is able to detect the manipulation of the content of a file or data buffer. The combination of this three security objectives repel the security threat completely.

T.SIG_DAT defines the security threat, that an adversary might manipulate a signed file and manipulation might keep undetected. OT.SIG_DAT defines the security objective, that the TOE shall detect the manipulation of a signed file. The security threat is repelled.

T.TOE defines the security threat, that an adversary might manipulate parts of the TOE and the manipulations might keep undetected. OT.SIG_DAT defines the security objective, that the TOE shall offer the possibility, to inform the user about corrupted TOE modules or data. The security threat is repelled.

T.PRE_SIG defines the security threat, that an adversary might manipulate the content of a file or data buffer, before the user decides to sign the file. OT.PRE_SIG ensures, that the file or data buffer is displayed in a manner, that the user can identify the content of the given data in an unambiguous way. The security threat is repelled.

T.POST_SIG defines the security threat, that an adversary might manipulate the hash value after the users decision to sign the file. OT.POST_SIG ensures that the user has the possibility, to detect the manipulation of the hash value of a file or data buffer to be signed after his decision to sign. The security threat is repelled.

T.LIC defines the security threat that an adversary might manipulate the currently used license and this manipulation is not detected. OT.LIC ensures the integrity of the license file through hash values and electronic signatures. OT.LIC does also ensure that the TOE's capabilities can not be downgraded through the insertion of a license code that offers fewer capabilities than the license code already entered by the user.

8.2 Security Requirements Rationale

Security Objectives vs. Security Requirements	OT.DAT	OT.SIG_DAT	OT.TOE	OT.PRE_SIG	OT.POST_SIG	OT.LIC	OE.SIG_DAT
FCS_COP.1 (SHA-1)	x	x			x	x	
FCS_COP.1 (SHA-224)	x	x			x	x	
FCS_COP.1 (SHA-256)	x	x			x	x	
FCS_COP.1 (SHA-384)	x	x			x	x	
FCS_COP.1 (SHA-512)	x	x	x		x	x	
FCS_COP.1 (160)		x				x	
FCS_COP.1 (RSA1-1024 to 2048)			x		x	x	
FCS_COP.1 (RSA2-1024 to 2048)		x					
FCS_COP.1(ECC1)					x	x	
FCS_COP.1(ECC2)		x					
FDP_DAU.2	x	x				x	
FDP_ITC.1(1)					x		
FDP_ITC.1(2)		x					
FDP_ITC.2(1)		x					
FDP_ITC.2(2)		x					
FDP_SVR.1				x			
FTP_ITC.1					x		
FCS_COP.1 (ES-1024 to 2048)						x	x
FCS_COP.1 (ES-ECC)						x	x
FMT_MOF.1(1)						x	
FMT_SMF.1(1)						x	
FMT_MOF.1(2)						x	
FMT_SMF.1(2)						x	

Table 4 Security Objectives vs. Security Requirements

OE.SIG_DAT ensures that the IT-environment provides functionalities for the encryption of cryptographic check sums (hash values). The required encryption algorithm and key size is specified by FCS_COP.1(ES-1024 to 2048) and FCS_COP.1(ES-ECC).

OT.DAT ensures, that the TOE provides functionalities to protect a user file against manipulation through hash value computation. FDP_DAU.2, FCS_COP.1(SHA-1), FCS_COP.1(SHA-224), FCS_COP.1(SHA-256), FCS_COP.1(SHA-384) and FCS_COP.1(SHA-512) define the required protection and the requirement of additional cryptographic functions: The file protection is implemented through hash value computation.

OT.SIG_DAT ensures, that the user has the possibility to detect the manipulation of a signed file. This is defined through security requirement 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(160), FCS_COP.1(RSA2-1024 to 2048), FCS_COP.1(ECC2), FDP_ITC.1(2), FDP_ITC.2(1) and FDP_ITC.2(2) define the requirement of additional cryptographic functions: Implementation through hash value computation and decryption with imported key.

OT.TOE ensures, that the user has the possibility to detect manipulations of TOE components or data. This is fulfilled by the security requirements FCS_COP.1(SHA-512) and FCS_COP.1(RSA1-1024 to 2048).

OT.PRE_SIG ensures the unambiguous identification of the data to be signed by the user. This requirement is fulfilled through FDP_SVR.1.

OT.POST_SIG ensures, that user has the possibility to identify, whether the hash value of the data to be signed was manipulated or not after the electronic signature creation. This is ensured by the security requirement FTP_ITC.1(1). 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(RSA1-1024 to 2048), FCS_COP.1(ECC1) and FDP_ITC.1(1) define the requirements of the additional cryptographic operations.

OT.LIC ensures protection of the license files integrity through the use of has values and electronic signatures (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(160) in conjunction with FCS_COP.1(RSA1-1024 to 2048), FCS_COP.1(ECC1),FCS_COP.1(ES-1024 to 2048) and FCS_COP.1(ES_ECC) provide the cryptographic support that is required to generate the signed license file. 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(160) in conjunction with FCS_COP.1(RSA1-1024 to 2048) and FCS_COP.1(ECC1) are required to set up the certificate chain in the signature verification process of the license file and are used to verify the integrity of the license file before the license information is used (FMT_MOF.1(1) and FMT_SMF.1(1)). FCS_COP.1(SHA-512) in conjunction with FCS_COP.1(RSA1-1024 to 2048) and FCS_COP.1(ECC1) are used to ensure the integrity of the resource file that contains the trusted root and CA certificates to be used for setting up the certificate chain. FDP_DAU.2 provides the capability to inspect the certificate that has been used to generate the signature on the license file. If a license file does already exist and the process of license file generation is repeated, the TOE does only accept license code that enable capabilities equal or upgraded to the current licensed capabilities (FMT_MOF.1(2) and FMT_SMF.1(2)).

8.3 Dependency Rationale

The assurance requirements for the TOE were defined using EAL 4 augmented. The augmentations are AVA_MSU.3 and AVA_VLA.4.

The dependencies of AVA_MSU.3 are resolved by ADO_IGS.1, ADV_FSP.1, AGD_ADM.1 and AGD_USR.1. The dependencies of AVA_VLA.4 are resolved by ADV_FSP.1, ADV_HLD.2, ADV_IMP.1, ADV_LLD.1 and AGD_ADM.1. The dependencies of ADV_IMP.1 are resolved by ALC_TAT.1.

The dependencies of the security requirements cannot be resolved completely. The following table gives an overview about the dependencies.

Requirement	Dependency	Fulfilled
FCS_COP.1 (SHA-1)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	No, see explanation below
FCS_COP.1 (SHA-224)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	No, see explanation below
FCS_COP.1 (SHA-256)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	No, see explanation below
FCS_COP.1 (SHA-384)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	No, see explanation below
FCS_COP.1 (SHA-512)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	No, see explanation below
FCS_COP.1(160)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	No, see explanation below
FCS_COP.1(RSA1-1024 to 2048)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	Yes, see explanation below
FCS_COP.1(RSA2-1024 to 2048)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	Yes, see explanation below
FCS_COP.1(ECC1)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	Yes, see explanation below
FCS_COP.1(ECC2-2)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	Yes, see explanation below
FCS_COP.1(ES-1024 to 2048)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	No, see explanation below
FCS_COP.1(ES-ECC)	[FDP_ITC.1 or FCS_CKM.1], FCS_CKM.4 and FMT_MSA.2	No, see explanation below
FDP_DAU.2	FIA_UID.1	No, see explanation below
FDP_ITC.1(1)	[FDP_ACC.1 or FDP_IFC.1], FMT_MSA.3	No, see explanation below
FDP_ITC.1(2)	[FDP_ACC.1 or FDP_IFC.1], FMT_MSA.3	No, see explanation below
FDP_ITC.2(1)	[FDP_ACC.1 or FDP_IFC.1], [FTP_ITC.1 or FTP_TRP.1],	No, see explanation below

Requirement	Dependency	Fulfilled
	FPT_TDC.1	
FDP_ITC.2(2)	[FDP_ACC.1 or FDP_IFC.1], [FTP_ITC.1 or FTP_TRP.1], FPT_TDC.1	No, see explanation below
FDP_SVR.1	None	Yes, implicit
FTP_ITC.1	None	Yes, implicit
FMT_MOF.1(1)	FMT_SMR.1 and FMT_SMF.1	In parts, not full

Requirement	Dependency	Fulfilled
FMT_SMF.1(1)	No dependencies	Yes, because no dependencies are defined for that SFR.
FMT_MOF.1(2)	FMT_SMR.1 and FMT_SMF.1	In parts, not full
FMT_SMF.1(2)	No dependencies	Yes, because no dependencies are defined for that SFR.

Table 5 Dependencies of security requirements

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) and FCS_COP.1 (160) require cryptographic operation of keyless hash algorithms. Therefore FCS_CKM.1, FCS_CKM.4 and FMT_MSA.2 are not applicable.

FCS_COP.1(RSA1-1024 to 2048) and FCS_COP.1(RSA2-1024 to 2048) refer to cryptographic operation with imported public keys. The dependency to import (FDP_ITC.1) is resolved by FDP_ITC.1(1) and FDP_ITC.1(2). FCS_CKM.4 and FDP_MSA.2 are in the context of the TOE not applicable.

FCS_COP.1(ECC1) and FCS_COP.1(ECC2) refer to cryptographic operation with imported public keys. The dependency to import (FDP_ITC.1) is resolved by FDP_ITC.1(1) and FDP_ITC.1(2). FCS_CKM.4 and FDP_MSA.2 are in the context of the TOE not applicable.

FCS_COP.1 (ES-1024 to 2048) refers to the RSA algorithm that is applied in the IT-environment. The implementation of the dependencies is part of the environment and therefore not specified in this document.

FCS_COP.1 (ES-ECC) refers to the ECDSA algorithm that is applied in the IT-environment. The implementation of the dependencies is part of the environment and therefore not specified in this document.

FDP_DAU.2 refers to data authentication. The identity of the user is recognized on the basis of the certificates content. The certificate is provided by appropriate components (see A.Platform). The component FIA_UID.1 is not applicable in the context of the TOE.

FDP_ITC.1(1) and FDP_ITC.1(2) refer to the use of public keys for electronic signature verification (FCS_COP.1(RSA1-1024 to 2048), FCS_COP.1(RSA2-1024 to 2048), FCS_COP.1(ECC1) and FCS_COP.1(ECC2)). The TOE must not protect the public keys. Therefore FDP_ACC.1 and FDP_IFC.1 are not required to fulfill the TOE security requirements. The import is done without any attributes and after the import no security attributes are initialized (FMT_MSA.3).

FDP_ITC.2(1) refers to the import of an OCSP response to verify the validity of a user certificate under examination. The OCSP response itself is not required to be protected by the TOE, because the integrity of an OCSP response is protected by hash values and electronic signatures that offer the required level of protection. Therefore FDP_ACC.1 and FDP_IFC.1 are not required to fulfill the TOE security requirements. FTP_ITC.1 and FTP_TRP.1 are not required, because the TOE verifies the complete certificate chain using CRL’s before using the OCSP information and the root certificate is a trusted certificate for the TOE. Through this mechanism the manipulation of data’s integrity is prevented. FPT_TDC.1 refers to TSF data exchange in a distributed or composite system environment where the TOE exchanges such data with another trusted IT product. This is not applicable in the context of the TOE.

FDP_ITC.2(2) refers to the import of timestamps. The timestamp itself is not required to be protected by the TOE, because timestamps are protected by hash values and electronic signatures that offer the required level of protection. Therefore FDP_ACC.1 and FDP_IFC.1 are not required to fulfill the TOE security requirements. FTP_ITC.1 and FTP_TRP.1 are not required, because the TOE verifies the complete certificate chain using CRL's before using the timestamp information and the root certificate is a trusted certificate for the TOE. Through this mechanism the manipulation of data's integrity is prevented. FPT_TDC.1 refers to TSF data exchange in a distributed or composite system environment where the TOE exchanges such data with another trusted IT product. This is not applicable in the context of the TOE.

FMT_MOF.1(1) and FMT_MOF.1(2) do refer to the management of security functions behaviour in the TOE. The mechanisms that are implemented by the TOE in order to manage the behaviour do depend on the licensing mechanism of the TOE and are independent from any security roles. The TOE does not assign different roles to users, therefore FMT_SMR.1 is not applicable in the context of the TOE. FMT_SMF.1(1) and FMT_SMF.1(2) are explicit SFR's of the TOE, the dependency is therefore fulfilled.

8.4 Rationale on mutual support

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(160), FCS_COP.1(ES-1024 to 2048) and FCS_COP.1(ES-ECC) support each other during the process of signature creation. FCS_COP.1(SHA-XXX)[76] as well as FCS_COP.1(160) compute the hash value and FCS_COP.1(ES-1024 to 2048) or FCS_COP.1(ES-ECC) applies the required signature algorithm[77]. Those security requirements support the fulfillment of FDP_DAU.2. FDP_SVR.1 is responsible for the unambiguous display of the data. FTP_ITC.1 ensures the correctness of the hash value to be encrypted with the previously computed hash value (FCS_COP.1(SHA-256)).

FCS_COP.1(160), 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) and FCS_COP.1(RSA1-1024 to 2048) support each other in the process of signature verification. Those requirements support the fulfillment of FDP_DAU.2, especially FDP_DAU.2.2. FCS_COP.1(160), FCS_COP.1(SHA-XXX)[78] do the hash value computation and FCS_COP.1(RSA.1-1024 to 2048) or FCS_COP.1(ECC1) decrypts the hash value for the hash value comparison. The key is provided by FDP_ITC.1(1). Additional OCSP information is provided by FDP_ITC.2(1) and a timestamp is provided by FDP_ITC.2(2). The content of the document, which refers to the electronic signature, is displayed by FDP_SVR.1.

[76] FCS_COP.1(SHA-1) or FCS_COP.1(SHA-224) or FCS_COP.1(SHA-256) or FCS_COP.1(SHA-384) or FCS_COP.1(SHA-512)

[77] RSA or ECDSA

[78] FCS_COP.1(SHA-1) or FCS_COP.1(SHA-224) or FCS_COP.1(SHA-256) or FCS_COP.1(SHA-384) or FCS_COP.1(SHA-512)

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(160), FCS_COP.1(RSA2-1024 to 2048) and FCS_COP.1(ECC2) support each other in the process of signature verification. FCS_COP.1(SHA-XXX)[79], alternatively FCS_COP.1(160) computes the hash value and FCS_COP.1(RSA2-1024 to 2048) or FCS_COP.1(ECC2) decrypts the hash value from the original electronic signature for comparison. The key is provided by FDP_ITC.1(2), an OCSP response is provided by FDP_ITC.2(1) and a timestamp is provided by FDP_ITC.2(2).

FCS_COP.1(SHA-1), FDP_ITC.1(1) ,FCS_COP.1(RSA1-1024 to 2048) and FCS_COP.1(ECC1) support each other in the implementation of the licensing mechanism of the TOE and do therefore support FMT_MOF.1(1), FMT_MOF.1(2), FMT_SMF(1) and FMT_SMF.1(2). FCS_COP.1(SHA-1) is used to compute hash values, FCS_COP.1(RSA1-1024 to 2048) and FCS_COP.1(ECC1) are required to verify the mathematical correctness of the electronic signature on the license file. The key is provided by FDP_TC.1(1).

[79] FCS_COP.1(SHA-1) or FCS_COP.1(SHA-224) or FCS_COP.1(SHA-256) or FCS_COP.1(SHA-384) or FCS_COP.1(SHA-512)

8.5 Rationale on assurance requirements, EAL4+ and SOF-high

The discussed application is a signature application component in the sense of §17 paragraph 2 of the German signature law. For the process of evaluation with Common Criteria the evaluation assurance level EAL 3 with augmentations is required for such components. The ordinance on the signature law (SigV) requires AVA_VLA.4, which requires SOF-high.

The vendor of the product S-TRUST Sign-it base components 2.5 has chosen the evaluation assurance level EAL 4 with augmentations. This ensures, that the requirements defined by the signature law and the ordinance on electronic signatures are met by the product.

In addition to those facts, the SigV requires a complete misuse analysis, which explains the choice of AVA_MSU.3.

The manufacturer did not identify any additional dependencies according to the chosen evaluation level.

8.6 Rationale on TOE specification

The specification of the TOE security functions refers directly to the TOE security requirements. The following table displays the correlation between security requirements and security functions.

Security Requirements vs. Security Functions	**SF.1**	**SF.2**	**SF.3**	**SF.4**	**SF.5**	**SF.6**	**SF.7**	**SF.8**	**SF.9**	**SF.10**
FCS_COP.1 (SHA-1)	x	x			x		x	x	x	x
FCS_COP.1 (SHA-224)	x	x			x		x	x	x	x
FCS_COP.1 (SHA-256)	x	x			x	x	x	x	x	x
FCS_COP.1 (SHA-384)	x	x			x		x	x	x	x
FCS_COP.1 (SHA-512)	x	x	x		x		x	x	x	x
FCS_COP.1 (160)	x	x			x		x	x	x	x
FCS_COP.1 (RSA1-1024 to 2048)	x		x		x		x	x		x
FCS_COP.1 (RSA2-1024 to 2048)		x					x		x	
FCS_COP.1 (ECC1)	x				x		x	x		x
FCS_COP.1 (ECC2)		x					x		x	
FDP_DAU.2		x					x		x	x
FDP_ITC.1(1)					x					
FDP_ITC.1(2)		x					x		x	
FDP_ITC.2(1)							x			
FDP_ITC.2(2)								x		
FDP_SVR.1				x						
FTP_ITC.1					x					
FMT_MOF.1(1)										x
FMT_SMF.1(1)										x
FMT_MOF.1(2)										x
FMT_SMF.1(2)										x

Table 6 Security Requirements vs. Security Functions

The security requirement FCS_COP.1(ES-1024 to 2048) and FCS_COP.1(ES-ECC) are part of the IT-Environment of the TOE and therefore not listed here.

9 Definition of functional family FDP_SVR

On order to define the IT-security requirements of the TOE completely, an additional functional family (FDP_SVR) of class FDP (user data protection) is defined. This family describes the functional requirements for a secure viewer component of a signature application component.

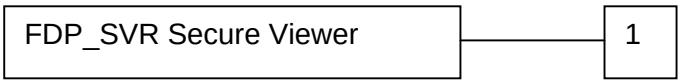
Due to the complexity of a legal binding viewer component as required by the signature law this component could not be modeled from the components that are provided by the Common Criteria framework. Therefore the introduction of a separate functional family is necessary that covers the requirements to describe the TOE consistently as needed for a confirmation that is based on the results of the Common Criteria evaluation.

FDP_SVR Secure Viewer

Family behaviour

This family defines the functional requirements to a secure 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_SVR.1 Secure Viewer requires the TSF to offer the ability to display the documents content in an unambiguous way that is free of hidden content. In addition, the ability to inform the user about hidden content is required.

Management : FDP_SVR.1
For this component no management activities are foreseen.

Audit: FDP_SVR.1
No actions are identified, that should be logged, if FAU_GEN is part of the PP/ST.

FDP_SVR.1 Secure Viewer

Hierarchical to: No other components

FDP_SVR.1.1 The TSF shall ensure, that the displayed content of a document is unambiguous according to [assignment: norms for displaying content].

FDP_SVR.1.2 The TSF shall ensure that the displayed content of a document is free of active or hidden content. The TSF shall ensure that the user is informed about hidden content.

FDP_SVR.1.3 The TSF shall ensure, that the user is informed about content, that cannot be displayed.

Dependencies: No dependencies identified.

The assurance requirements that have been defined by the Common Criteria v2.3 Part 3 are applicable to the functional family FDP_SVR. This functional family has been defined to meet the requirements of the secure viewer component in a signature application component.

Because this component is a software component with a well defined behaviour on its external interfaces, the assurance requirements that have been defined in part 3 of Common Criteria are applicable to this functional family.

Through its nature as a software component the assurance classes ACM, ADO, ADV, AGD, ALC, ATE and AVA are applicable in the evaluation process. It is not required to define a new assurance class or assurance family for a consistent and complete description to cover this SFR. This SFR does not define any behaviour that might require an extension of Part 3 of the Common Criteria Evaluation Framework.

Documents

[1] Acrobat Digital Signature API Reference, Technical Note #5192, Adobe Systems Incorporated

[2] Gesetz über die Rahmenbedingungen für elektronische Signaturen, Fundstelle: BGBI I 2001, 876, 16.05.2001

[3] Verordnung zur elektronischen Signatur, Fundstelle: BGBI 2001, 3074, 16.11.2001

[4] Law Governing Framework Conditions for Electronic Signatures and Amending Other Regulations, unofficial version for industry consultation

[5] Ordinance on Electronic Signatures, unofficial working translation

[6] Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures

[7] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, August 2005, Version 2.3, CCMB-2005-08-001

[8] Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements, August 2005, Version 2.3, CCMB-2005-08-002

[9] Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements, August 2005, Version 2.3, CCMB-2005-08-003

[10] IT Sicherheit auf Basis der Common Criteria – ein Leitfaden, Hrsg: Bundesamt für Sicherheit in der Informationstechnik der Bundesrepublik Deutschland

[11] Secure Hash Standard, Federal Information Processing Standards Publication 180 – 2, announced 2002 August 1 (FIPS PUB 180-2)

[12] RIPEMD-160: A Strengthened Version of RIPMD, published by German Information Security Agency and Katholieke Universiteit Leuven, announced 18 April 1996

[13] Maßnahmenkatalog für technische Komponenten nach dem Signaturgesetz, announced 15.07.1998 by German RegTP

[14] Einheitliche Spezifizierung der Einsatzbedingungen für Signaturanwendungskomponenten Version 1.0, announced 30.01.2002

[15] RFC 3447 – Public Key Cryptography Standards (PKCS) #1, RSA Cryptography Specifications Version 2.1, announced by The Internet Society

[16] User documentation S-TRUST Sign-it base components 2.5, version 2.5.1.1

[17] Geeignete Algorithmen zur Erfüllung der Anforderungen nach §17 Abs.1 bis 3 SigG in Verbindung mit Anlage 1 Abschnitt 1 Nr. 2 SigV, published February 5, 2008

[18] RFC 3161 – Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), announced by The Internet Society

[19] RFC 2560 – X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP), announced by The Internet Society

[20] RFC 3280 – Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, announced by The Internet Society

[21] RFC 2315 – PKCS#7: Cryptographic Message Syntax Version 1.5, announced in March 1998 by The Internet Society

[22] RFC 3778 – The application/pdf media type, announced May 2004 by The Internet Society

[23] Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung, announced by the German Regulierungsbehörde für Telekommunikation und Post, 02.01.2004

[24] CWA 14170:2004 CEN Workshop Agreement, Security requirements for signature creation applications

[25] PKCS#1 v2.1: RSA Cryptography Standard, published by RSA Laboratories June 14, 2002

[26] PDF Reference sixth edition, Adobe Portable Document Format Version 1.7, Adobe Systems Incorporated

[27] International Standard ISO/IEC 9796-2 Second Edition, 2002-10-01, ISO copyright office

[28] RFC 3275 – (Extensible Markup Language) XML-Signature Syntax and Processing, Network Working Group, March 2002

[29] X9.62 – 1998 Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), American National Standards Institute, January 7, 1999

[30] Technical Guideline TR-03111 – Elliptic Curve Cryptography Based on ISO 15946, Federal Office for Information Technology (BSI), Version 1.00, February 14, 2007

[31] RFC 3874 – A 224-bit One-way Hash Function: SHA-224, Network Working Group, September 2004