FILKRYPTO
TUTUS - DIGITAL GATEKEEPERS

Security Target

Filkrypto, Release 1.0.2

Document Version: 1.3.1

Sponsor: Tutus Data AB

Title	Security Target Filkrypto		
Author(s):	Melanie Wahl	Status:	Released
Version:	1.3.1	Classification:	Public
File name:	ST-Filkrypto.pdf	Date:	01/10/07

Tutus Data AB, Svärdvägen 11, S-182 33 Danderyd, Sweden, Tel: +46 (0)8 551 102 30, Fax: +46 (0)8 551 102 39, E-mail: info@tutus.se, Web: http://www.tutus.se

Document History

Version	Date	Author	Changes to Previous Version
0.5	2006-09-06	Melanie Wahl	Version sent for application
1.0	2006-09-19	Melanie Wahl	Released Version
1.0.1	2006-09-20	Melanie Wahl	Minor updates to 1.0 regarding the comments of Per Holmer
1.0.2	2006-10-16	Melanie Wahl	Updates regarding the comments of the CB (TOR, Application Filkrypto 1.0, issued 2006-10-02)
1.0.3	2006-12-19	Staffan Persson	Updates regarding the SER of ST Filkrypto 1.0.2, version 1.0 provided by ITSEF Combitech AB, issued on 2006-11-14 (Ref. No L1-06:0191)
1.0.4	2007-01-17	Staffan Persson	Updates based on comments from developer and external reviews.
1.0.5	2007-02-13	Sebastian Mayer	Updates based on comments from evaluator.
1.0.6	2007-02-27	Staffan Persson	Updated after comments from the developer and the certifier.
1.1	2007-03-01	Staffan Persson	Updated version after internal review
1.2	2007-08-10	Staffan Persson	Updated application notes for RFCs.
1.3	2007-09-30	Staffan Persson	Updated for publication.
1.3.1	2007-10-01	Per Holmer	Layout changes and spelling

Table of Contents

Index of Tables

Illustration Index

This page is intentionally left blank

1 Introduction

1.1 ST Identification

ST Title:	Filkrypto Security Target
Product Name:	Filkrypto
Product Version:	Release 1.0.2
Assurance level:	EAL3
CC Version:	2.3 as of August 2005
ST Author:	Melanie Wahl, Staffan Persson
ST publication date:	01/10/07
ST Version:	1.3.1
Keywords:	File encryption / decryption

1.2 ST Overview

This document is the Security Target (ST) for the Filkrypto application, release 1.0.2, developed by Tutus AB.

Filkrypto is an application for file encryption on Microsoft Windows platforms. The program is using a Swedish government owned and approved cryptographic library to implement all cryptographic related functions.

Filkrypto is a file encryption application intended to be protect sensitive information in files when transmitting or storing them in unprotected environments. Filkrypto is mainly intended for government use..

The ST contains a description of the security objectives and the requirements, as well as the necessary functional and assurance measures provided by the TOE. The ST provides the basis for the evaluation of the TOE according to the Common Criteria for Information Technology Security Evaluations (CC).

1.3 CC Conformance Claim

This ST is CC Part 2 conform and CC Part 3 conform, with the assurance level of EAL3. No augmentation is performed for security requirements concerning the TOE.

The Security Target is following the structure given in part 1 of the Common Criteria, using the guidance from ISO/IEC JTC 1/SC 27 N 2449 "Information technology – Security techniques – Guide for the production of protection profiles and security targets" ([GPPS]).

This ST does not claim conformance to any existing Protection Profile (PP).

1.4 Strength of Function Claim

The TOE contains of one function realized by a probabilistic mechanism, for which a SOF claim is provided in this ST. This is the integrity check at file decryption. For this function the minimum strength of function claimed is SOF-high.

No claims are made about the strength of function for any cryptographic functions

based on cryptographic mechanisms. Further no claims are made about the functions of key generation and key derivation from password which are based on probabilistic and permutational mechanisms. The cryptographic verification as well as the SOF analyse of random number generation and key derivation mechanism is performed by the government agency TSA.

1.5 ST Content and Organisation

The ST has been structured in accordance with [CC] Part 1 and [GPPS]. The main sections of the ST are the TOE description, TOE security environment, security objectives, IT security requirements, rationale and annexes.

The TOE description provides general information about the TOE, serves as an aid to understanding the nature of the TOE and its security functionality, and provides context for the ST's evaluation.

The TOE security environment describes security aspects of the environment in which the TOE is to be used and the manner in which it is to be employed. The TOE security environment includes:

a) assumptions regarding the TOE's intended usage and environment of use

b) threats relevant to secure TOE operation

c) organisational security policies with which the TOE must comply

The security objectives reflect the stated intent of the TOE. They pertain to how the TOE will counter identified threats and how it will cover identified organisational security policies and assumptions.

Each security objective is categorised as being for the TOE or for the environment.

The security requirements section provides detailed IT security requirements for the TOE in separate subsections.

The IT security requirements are subdivided as follows:

a) TOE Security Functional Requirements

b) TOE Security Assurance Requirements

Security requirements for the IT environment are not defined.

The TOE summary specification addresses the security functions that are represented by the TOE to answer the security requirements.

The rationale presents evidence that the ST is a complete and cohesive set of requirements and that the TOE provides an effective set of IT security countermeasures within the security environment. The rationale is in three main parts. First, a security objectives rationale demonstrates that the stated security objectives are traceable to all of the aspects identified in the TOE security environment and are suitable to cover them. Then, a security requirements rationale demonstrates that the security requirements for the TOE are traceable to the security objectives and are suitable to meet them. Finally the TOE summery specification demonstrates, that the TOE security functions and assurance measures are suitable to meet the security requirements for the TOE.

The appendix (annex) contains a list of abbreviations as well as a glossary for this ST.

1.6 Related Standards and Documents

[BNetzA]	Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen, Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen), 2. Januar 2006
[CC]	Common Criteria (CC) for Information Technology Security Evaluation, August 2005, Version 2.3. » Part 1: Introduction and general model. August 2005. Version 2.3. CCMB-2005-08-001 » Part 2: Security functional requirements. August 2005. Version 2.3. CCMB-2005-08-002 » Part 3: Security Assurance Requirements. August 2005. Version 2.3. CCMB-2005-08-003
[CEM]	Common Methodology for Information Technology Security Evaluation. Evaluation Methodology. August 2005. Version 2.3. CCMB-2005-01-004
[GPPS]	ISO/IEC TR 15446:2004(E), Guide for the production of Protection Profiles and Security Targets, First edition 2004-07-01
[Lenstra]	Lenstra, A.K.; Verheul, E.R.: Selecting Cryptographic Key Sizes, November 1999
[RFC 2104]	Request for Comments: 2104, HMAC: Keyed-Hashing for Message Authentication, Februar 1997
[RFC 2406]	Request for Comments: 2406, IP Encapsulating Security Payload (ESP), November 1998
[RFC 2898]	Request for Comments: 2898, PKCS #5: Password-Based Cryptography Specification, Version 2.0, September 2000

2 TOE Description

This section describes the Target of Evaluation (TOE) in terms of the class of product, the operational environment, and the provided security functionality. This chapter provides a general description of the product without focusing on the evaluated configuration.

2.1 Introduction

Filkrypto is a software only product for file encryption in Microsoft Windows environments, running on a single user PC. The program could be used entirely without any further infrastructure and therefore also contains functions for key generation and distribution.

Only symmetrical algorithms are used to implement the cryptographic operations of encryption and decryption by Filkrypto. The associated keys are either generated within the application as "standard keys" or are generated elsewhere and imported into Filkrypto as "form keys". The form keys are imported into Filkrypto by entering the key from a form received.

Filkrypto stores the keys within password encrypted files called keystores or keyfiles. A certain password policy is implemented for the passwords used. A default keystore also stores the serial number of the application. Further keystores could be created by the user which can be stored on removable media e.g., to exchange keys.

Besides encryption of files ensuring confidentiality of information included, Filkrypto uses another cryptographic mechanism to detect loss of integrity; keyed hash functions are used over every cipher generated by Filkrypto. This applies to data files as well as to key files. Before decryption, these hash values are validated.

Furthermore, Filkrypto removes files by attempting to overwrite them with random data[1]. This is triggered by user actions. For example, in case of emergency the default keystore could be deleted by pushing the emergency erase button.

The program uses a TSA developed cryptographic library for all cryptographic and probabilistic functions. The functions used are described in detail in the following chapters.

2.2 TOE Definition Scope

The Target of Evaluation is limited to the software application Filkrypto, version 1.0.2, developed by Tutus AB. Filkrypto consists of four parts: the graphical user interface (GUI), the cryptographic library (cryptolib), the application supervisor and the XML parser (Expat). The GUI handles all user interaction and the Cryptolib all cryptographic operations, while the application supervisor handles initializations, starting, stopping, and cleanup in the application. All security critical operations are handled by the Cryptolib and the application supervisor..

Cryptolib depends on FMSSL and Expat. FMSSL is a version of the OpenSSL crypto library (http://www.openssl.org) in which all underlying cryptographic and random algorithms have been substituted by compatible algorithms developed and approved by TSA. Expat is an XML-parser (http://expat.sourceforge.net) used to parse the encrypted file format.

The cryptographic mechanisms in the Cryptolib are provided and approved by the

1 Depending on the stotage technologie this might not work in any case.

Swedish NCSA (TSA). The cryptographic mechanisms are part of the TOE, but their cryptographic properties are not being part of the CC evaluation.

The GUI's only job is to identify for the underlying Cryptolib what operation to perform on which file. Therefore the GUI is not security critical.

In the following picture, the architecture of TOE Filkrypto and its boundaries are shown.

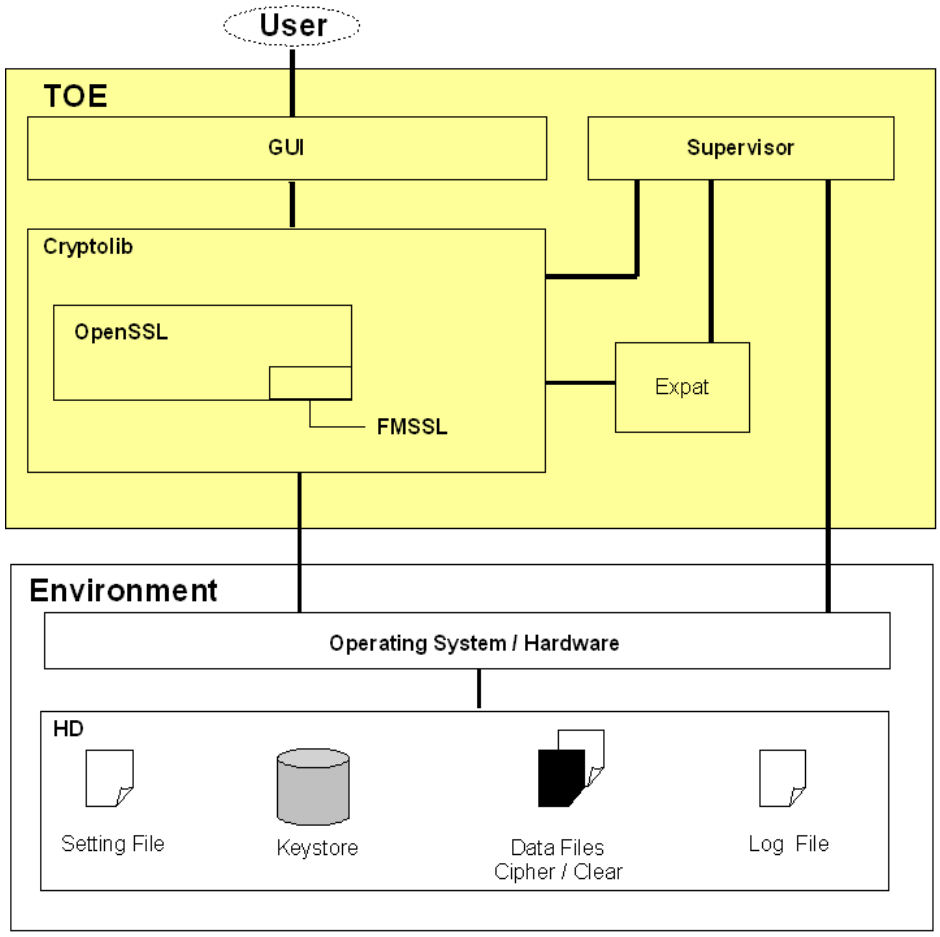

Illustration 1: TOE architecture and boundaries

2.3 Supported Platforms and Environment

The underlying platform for the evaluation is limited to the MS Windows XP Client operating system. No additional special equipment or infrastructure is needed.

2.4 Installation

Filkrypto is easy to install by the user himself. The first time the user runs "Filkrypto.exe", he is called to select an application password (password to access the default application keystore) and enter the serial number of the application. Furthermore the user can choose the language of the application and whether the data should be saved on the hard drive or whether the whole application should be kept in a USB Data storage device, running everything from there.

Any software updates that are required during the life cycle of Filkrypto have to be installed by the user.

2.5 Configurations

The application cannot be configured. The algorithms used, key length, etc. are static parameters of the application, which can only be changed during application development.

The user can only set up in which mode he wants the application to run and what he wants to be displayed in detail. These settings have nothing to do with the security configuration of the application.

2.6 TOE Operation and Use

2.6.1 Intended Use

With Filkrypto, two or more individuals can exchange electronic documents securely over unprotected communication paths, e.g. networks, without risking any unauthorized persons reading the documents. To achieve this, the documents are encrypted before they are sent between the two parties, and are thus made unreadable for anyone who does not have access to the encryption key required for decrypting the documents.

The user could select to run the Filkrypto application in "Simple" or "Advanced mode". Encryption and file exchange are done in nine steps when the application runs in Simple mode.

First, keys are created, distributed, and made accessible using these steps:

1. The two parties agree on one or more key names that should be used when exchanging files, and which password(s) should be used for protecting the key(s).
2. One of the parties creates the key(s) that have been agreed upon.
3. The encryption key(s) are exported from the Filkrypto application with the selected password in the password-encrypted transport key container.
4. The key(s) are distributed to the concerned parties. This should be done off-line, e.g., via a floppy disc, a USB-Data Storage Device, or similar means, to prevent any unauthorized person from accessing the key.
5. The other parties import the key(s), or add shortcuts to the key(s) if they are located in external media (such as a floppy disc or a USB Data Storage Device), and enter the relevant password, so that all the parties have access to the key(s) within the transport key container that has been agreed upon.

Now the parties can start encrypting files for secure file exchange using these steps:

6. Drag the file(s) that you want to encrypt into the Filkrypto application, and select which key you wish to use.
7. Select the folder in which the file(s) should be saved and enter the desired file name.

Selected file(s) are now available in encrypted format in the directory chosen.

8. Send the file(s) to the concerned parties.
9. The receiving parties drag the file(s) into the Filkrypto application, and drop them there. The file(s) are automatically decrypted if the required keys are available in the receiving parties' Filkrypto applications.

The mode of operation to be selected by the user differs only in the provided level

of guidance, no additional security functionality is given in "Advanced Mode" in contrast to "Simple Mode". Further the way cryptographic operations are performed is the same in both modes. For example in Advanced mode the encrypted and decrypted files are shown inside the application and a user can select if he wants to display files separately or if he only wants to view encrypted / decrypted files. Further, the language can be selected, as well as whether the Filkrypto application is displayed always on top.

2.6.2 Security Roles

The TOE is not aware of any user roles or even the concept of users, so any user with access to the TOE or the TOE environment is able to perform any operation.

2.6.3 Security Functionality

The following security features are provided by the TOE:

- **Encryption of files[1]:** Files are encrypted using the symmetric algorithm AES in CBC[2] mode with a 256-bit key to ensure confidentially of information while stored and/or during transmission. The ciphers are created out of files using keys selected by the user out of the actual key list presented or using the default application key. A file encrypted by Filkrypto results in an xml file containing additional information on which key(id) was used to encrypt the file, the time when the file was encrypted, the original file's file name, and the encrypted data. The format of the encrypted data is based on a modified IPsec ESP [RFC 2406] packet format.
- **Integrity protection of files:** To detect loss of integrity of data files or keystores, a keyed hash functions is used over every cipher generated by Filkrypto. The algorithm used is HMAC-SHA256 – a message authentication function using a 256-bit key and a SHA-256 hash function. The key used to calculate the HMAC is a 256-bit key only used for this purpose. Therefore this key has to be shared between the communication partners, in addition to sharing the respective encryption/decryption key.
- **Decryption of files:** Files are decrypted by Filkrypto, ensuring that only those users who possesses the right key or respectively know the associated password can read the encrypted file. Therefore confidentiality of information can be implemented while storing, as well as during transmission.
- **Integrity check of files:** Within this feature, potential loss of integrity of data files and keystores can be detected by validating the keyed hash values provided before decryption. The algorithm used is HMAC-SHA256 – a message authentication function using a 256-bit key and a SHA-256 hash function. The key used to calculate the HMAC is a 256-bit key generated for this purpose. Therefore this key has to be shared between the communication partners, in addition to sharing the respective encryption / decryption key.
- **Key management:** Key management is a process to manage the whole life-cycle of cryptographic keys from generation through distribution to archiving and destruction. The following security relevant key management features are implemented by Filkrypto:

1 Files are the data files which have to be encrypted and exchanged with other users as well as keystores (default keyfiles and keyfiles for exchanging)

2 Cipher Block Chaining

- **Key generation**:
  - **Key derivation:** The keys used to protect the encrypted keystores are derived from passwords entered by the users. For this purpose, the algorithm PBKDF2 described in [PKCS#5] (RFC 2898) is used.
  - **Data file encryption keys:** Only symmetric AES keys with a key length of 256 bits are generated by the key generator of Filkrypto. The key generator is provided and approved by TSA.
- **Import / Export keys (standard keys):** To exchange the keys between different users (users with different application instances running), the keys can be exported within password protected, encrypted keyfiles created by a user of Filkrypto. On the receivers side, these keys can be imported in the application only by knowing the password applied to the encrypted keystore.
- **Import of form keys**: "Form keys" are generated elsewhere and then imported into Filkrypto by entering the key manually from a form received.
- **Key storage** : All keys are stored in password-protected encrypted key storages.
- **Safe erasure** is implemented by overwriting the files with random data generated by the computer. In the case of an emergency, all encryption keys in the default keystore have to be deleted immediately. Further, files and individual encryption keys can be deleted upon user request, as well.

Features provided by the TOE, but not considered security enforcing functions:

» **Password quality:** Filkrypto uses a simple password quality algorithm. The password gets an original rating based on its length, which must be at least seven characters. Then the password is scanned for repeated character patterns; every repeated pattern reduces the original rating. Then the password gets additional rating if the password contains variations of characters, for example: escape- , numeric-, upper-case and lower-case characters. This algorithm is present only to help the user to select a good password. It can easily be circumvented by an evil user, and therefore no security properties for it are claimed.

» **Audit:** The audit function is for convenience and serves as a help to the user. It shows the different actions (encryption/ decryption) performed, the path and names of the files, the key used, and the key's ID.

» **Key validity:** While generating a key, the user is called to enter the validity time of the key. When trying to us an expired key Filkrypto warns the user about it, but the user is allowed to continue.

» **Parsing XML-files:** Only syntax checks are made; therefore, this feature is not security relevant.

» **Form keys cannot be exported:** Form keys cannot be exported from default keystores. However, the default keystore containing form keys, is an ordinary file and can, of course, be copied using standard Windows utilities. Therefore this feature is only provided to make sure that the user does not export form keys by mistake. Due to this, no security properties for the feature are claimed.

- **Integrity check of form keys:** Before importing a form key, the application validates the integrity of the key entered by the user. Therefore 64 bit of the SHA-1 hash value of the key are concatenated at the end of the key string by the party who generates the key. This value is compared to the corresponding value calculated over the given form key. This feature only ensures that the user does not make failures while entering the "form key" from the form, assumed to be received in a secure manner out of band. This addresses only availability aspects and therefore not regarded as a security enforcing function.
- **Expanding given form keys:** Due to integrity checks of files encrypted with a form key the application needs as well an HMAC key. Since the manual entered length of the string for the form key could only be used for one 256 bit key, Filkrypto uses an algorithm provided and owned by TSA to expand this key to the double length of 512 bit used for both keys. For Filkrypto only the key import is security relevant not the implementation of key expansion and therefore no security properties for it are claimed.

2.7 TOE Environment and Physical Protection

The TOE is expected to be operated as a single user machine in a physically secure and well managed environment without a direct connection to an untrusted network.

3 TOE Security Environment

3.1 Secure Usage Assumptions

The following conditions are assumed to exist in the TOE operational environment. These assumptions include essential environmental constraints on the secure use of the TOE. Assumptions about the intended usage of the TOE are not made.

A.SINGLE The TOE runs on a single user machine with access protected by the TOE environment; i.e., only authorised users of the TOE environment may access the TOE. This includes access control provided by the operating system or equivalent and protection against malware.

A.KEYDIS It is assumed that keys used for encryption/decryption and as well as the associated keys used for integrity checks are of high quality and are not disclosed to unauthorized users. The keys are assumed to be distributed only to those parties who are authorized to use them in order to encrypt and decrypt files.

A.FORMKEYDIS Form keys are assumed to be distributed out of band from the generating party in a secure manner, therefore they are assumed to be not disclosed and tampered during distribution. Otherwise the same assumptions apply to form keys as to keys generated in Filkrypto as described in A.KEYDIS. They are of high quality and not disclosed to unauthorized users.

A.PHYSICAL The TOE is operated in a physically secure and well managed environment.

A.USER The TOE user is trustworthy and trained to manage and perform encryption of classified information in accordance with any existing security policies and information classification policies. This means especially that he knows how to classify information and how to deal with, e.g., encrypting all files containing sensitive information with the appropriate key before exporting the file out of the TOE and/or its TOE environment.

A.CONNECT The single user PC on which the TOE is running is not connected directly to an untrusted network. This means that the PC is either assumed not to be connected to any networks or it is connected to a trusted network which is protected against attacks, so that no undocumented security critical side effects on the security functions of the TOE, which are resided in the PC, are assumed coming from this network.

3.2 Threats

The threats described in this chapter are addressed by the TOE.

3.2.1 Assets and Agents

The assets and user agents used for the definition of threats are defined in the following tables.

Asset	Description	Type of Data
Data files (primary asset)	Filkrypto files that contain the information to be protected.	User data

Table 1: Assets

Agent	Description
Attacker	An attacker who has access to any communication channel over which the integrity protected and encrypted Filkrypto files are transferred, e.g., networks or other paths of transmission where communication media like CDs, DVDs including the encrypted files could be shared.

Table 2: Agents

3.2.2 Threats addressed by the TOE

The threats below must be countered by the TOE.

Threat: T.DISCLOSE – loss of confidentiality	
Attack	An attacker of one of the communication paths over which the Filkrypto file is transferred succeeds in accessing the content of the file, i.e. the attacker violates the confidentiality of the information included in the file. The attack is achieved by passive attacks recording encrypted data during the transfer (e.g. eavesdropping of network communication, interception of dispatch services) and decoding the encrypted data. In general the attacker has no access to the right key and has to perform cryptanalysis to reveal the underlying plain text of the encrypted file.
Asset	Data files
Agent	Attacker

Threat: T.TEMPER – loss of integrity	
Attack	An attacker of one of the communication paths over which the Filkrypto file is transferred tampers with the file, i.e. replacing or modifying the content of the file in a way that is not detected. The attack is achieved by interrupting the transfer due to possess the file to accomplish an active attack violating the integrity of the information included in the file before sending it to the receiver. Therefore the attacker has either to break the integrity protection of the file, modifying the content of the file and reconstructing the protection again. Or the attacker replaces the whole file and constructs the integrity protection. Afterwards the file is sent to the intended destination. In both cases the attacker has to possess either the right key used for integrity protection or he has to perform cryptanalysis to reveal the right key. For possibilities to get the right key see T.DISCLOSE.
Asset	Data files
Agent	Attacker

Table 3: Threats addressed by the TOE

In both threats described above the primary subject of the attacks is the information included in the data files transferred over an unprotected communication path.

The attackers specified as threat agents in both threats above are assumed to possess very limited opportunity of attacks, characterized as follows:

- **Expertise:** It is assumed that the key material has not been leaked (A.KEYDIS, A.FORMKEYDIS) and the implementation is not flowed (A. PHYSICAL). The attackers know IP and related networking protocol basics and are trying to find vulnerabilities publicly known about cryptographic algorithms (systematic weaknesses). The attacker must be familiar with the "alternative" distribution channels over which the encrypted data will be sent. Therefore a high level of expertise is required to successfully gain the plain text from encrypted data.
- **Resources:** The resource requirements to mount an attack of the types described above are high – a very large amount of computing power, either distributed or within one unit would be required to break the encryption in an appropriate time scale, expected to do not exhaust the range of at maximum some man days. In contrast to the attack within T.DISCLOSE the attack within T.TEMPER must be launched e.g. nearly on the fly, to ensure that the attack could not be detected. Network attack tools, especially network sniffers, available on the Internet are considered to be available, too. Further the attacker has the possibility to buy the product and perform cryptanalysis on the algorithms used or disassembling and reverse engineering the TOE. Therefore it is very easy for the attacker to get information about how the TOE operates. But attackers have no access (neither physical nor over the network – A. PHYSICAL, A.CONNECT) to the TOE where the information is encrypted or decrypted.
- **Motivation:** The TOE aims to protect sensitive information during the transfer over any communication paths. So, the attackers are assumed to be motivated by high-value assets and e.g. by the fact to "hack" sensitive information.

As described above it is very easy for an attacker to get information about how the TOE is operating in general – therefore an attacker will reveal easily that he has to combine the attacks described in T.DISCLOSE and T.TEMPER to be successful in

violating the integrity and/or confidentiality of the file's content. Because the transferred file is first encrypted and afterwards the encrypted file is integrity protected.

Attacks which modify the content of the transferred file without breaking the integrity protection are as well conceivable. May by an attacker completely intercepts the communication so that the file does not reach it's destination. This attacks have the same effects as errors during communication have. Preliminary the availability of the information transferred is violated. The receiver fails e.g. in validating the integrity of the file, the file will not be decrypted. This attacks will not be regarded here deliberately, because it will be detected anyway.

If vulnerabilities were present in the TOE's encryption algorithm, cryptographic functions used for integrity protection, key generating algorithm or in there implementation, this may be exploited to decrease the level of expertise or resource required for success.

The opportunity to mount all attacks depends on the fact that the transferred Filkrypto file is in general available for an attacker.

3.3 Organisational Security Policy

P.ERASURE Individual encryption keys shall be deleted upon the request of the authorized user.

P.EMERGENCY All encryption keys contained in the default keystore shall be deleted in case of emergency.

P.ALGORITHM The TOE shall only allow the use of approved encryption algorithms and key lengths, i.e. AES 256 bit.

4 Security Objectives

The security objectives provide a concise statement of the intended response to the security problem. This section describes which security needs will be addressed by the TOE and which will be addressed by the TOE environment, in the form of a statement of security objectives.

4.1 Security Objectives for the TOE

The following are the IT security objectives to be met by the TOE.

O.DISCLOSE The TOE must provide mechanisms that protect the information of a transmitted Filkrypto file such that its content is confidentiality-protected and only accessible for authorized users.

O.TAMPER The TOE must provide mechanisms that detect if an attacker has tampered with a transmitted Filkrypto file (i.e. replacing or modifying the content of the file); mechanisms must be provided to detect loss of integrity of the information in the file.

O.ERASURE Individual encryption keys must be deleted upon the request of the authorized user.

O.EMERGENCY All encryption keys contained in the default keystore must be deleted in case of emergency.

O.ALGORITHM The TOE must only allow the use of approved encryption algorithms and key lengths, i.e. AES and 256 bit.

4.2 Security Objectives for the IT and non-IT Environment

The following are the security objectives that are to be satisfied without imposing technical requirements on the TOE. That is, they do not require the implementation of functions in the TOE hardware and/or software. These security objectives are assumed to be in place in the TOE environment. They are included as necessary to support the TOE security objectives in addressing the security problem defined in the TOE security environment.

Thus, the following environmental objectives may partly be IT specific and partly related to administrative methods and/or procedural measures.

OE.KEYDIS Keys used for encryption and decryption as well as the keys used for integrity checks must be of high quality and must not be disclosed to unauthorized users. They must be distributed only to those parties who are authorized to use them in order to encrypt and decrypt files.

OE.FORMKEYDIS Form keys must be distributed in a secure manner from the generating party, therefore they must not be disclosed and tampered during distribution. Otherwise the same requirements must be ensured to form keys as to keys generated in Filkrypto as described in OE.KEYDIS. They must be of high quality and must not disclosed to unauthorized users.

OE.SINGLE	The TOE must be run on a single user machine with access to the TOE protected by the TOE environment; i.e., only authorised users of the TOE environment have access to the TOE. This includes access control provided by the operating system or equivalent and protection against malware.
OE.PHYSICAL	The TOE must be operated in a physically secure and well managed environment.
OE.USER	The TOE User is trustworthy and trained to perform all actions in accordance with any existing security policies and information classification policies.
OE.CONNECT	The single user PC on which the TOE is running must not be connected directly to an untrusted network. This means that the PC must either not be connected to any networks or it must be connected to a trusted network, which is protected against attacks, so that no undocumented security critical side effects on the security functions of the TOE are coming from this network.

5 IT Security Requirements

The following table gives an overview of the functional components from the Common Criteria Part 2 that are relevant for this TOE.

Component	Component Name
FCS_CKM.1	Cryptographic key generation
FCS_CKM.2	Cryptographic key distribution
FCS_COP.1	Cryptographic operation
FCS_CKM.4	Cryptographic key destruction
FDP_ACC.1	Subset access control
FDP_ACF.1	Security attribute based access control
FDP_ETC.2	Export of user data with security attributes
FDP_ITC.2	Import of user data with security attributes
FMT_MSA.1	Management of security attributes
FMT_MSA.2	Secure security attributes
FMT_MSA.3	Static attribute initialisation
FMT_SMF.1	Specification of Management Functions

Table 4: Functional Requirements on the TOE

The TOE will implement only one Security Function Policy (SFP) called **keystore access control SFP.** The policy' s name indicates that it is a policy regulating the access to the keystore. The policy consist of two parts, one creating the keystore, setting the stage for accessing the keys which will be stored in the keystore.

The SFP regulates that the password for accessing the keystore is chosen by the user and assigned to the keystore first.

This is enforced:

- when starting the application the first time. The user has to choose the password for the default key store. The password is assigned to the default keystore and has to be entered each time the user wants to access the default keystore while starting the application.
- when the user wants to export keys out of the application. Here the user is as well ask to choose a password. This password has to be entered each time when a user wants to access the keystore, due to import the keys into his default keystore.

Further the SFP regulates that the access to the keystore is granted only to users providing the right password which has been assigned before.

5.1 TOE Security Functional Requirements

5.1.1 Class FCS - Cryptographic Support

5.1.1.1 FCS_CKM.1(a) - Cryptographic key generation (standard key)

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance

with a specified cryptographic key generation algorithm ***provided by TSA*** and specified cryptographic key sizes ***256 bits*** that meet the following: ***conform to the TSA requirements***.

Application Note: For cryptographic key generation an algorithm owned and approved by TSA is used. A claim about the strength of function of the underlying random number generator can not be provided in this ST. The SOF analyse is done by TSA.

5.1.1.2 FCS_CKM.1(b) - Cryptographic key generation (derivation from password)

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic ***key derivation algorithm in accordance with a*** specified cryptographic algorithm ***PBKDF2*** and cryptographic key sizes ***512 bits*** that meet the following: **conform *to PKCS#5 and RFC 2898***.

Application Note: Not the full conformance to RFC 2898 is required. Only the key derivation part using PBKDF2 is relevant. The key derived is of the length of 512 bits. The first 256 bit are used for the encryption key and the last 256 bit for the associated HMAC-Key. No SOF is claimed for the key derivation from password in this ST.

5.1.1.3 FCS_CKM.2 - Cryptographic key distribution

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in a password protected encrypted keystore.

Application Note: The keys are distributed in a password protected encrypted keystore. The SFR describes the format used for the encapsulation of the keys as a part of the the keystore. The distribution method is not relevant for the TOE security and therefore not defined. The format is also used for the default keystore file which is not to be used to exchange between users but stored on the hard disk.

5.1.1.4 FCS_CKM.4 - Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method ***overwriting with random data in case of keys stored on files, with zeros when keys are stored in memory*** that meets the following: ***conform to the TSA requirements***.

Application Note: The method used overwriting files with random data is owned and approved by TSA..

5.1.1.5 FCS_COP.1(a) - Cryptographic operation (file encryption)

FCS_COP.1.1 The TSF shall perform ***encryption of files*** in accordance with a specified cryptographic algorithm ***AES*** and cryptographic key sizes ***256 bits*** that meet the following: ***conform to RFC 2406***.

Application Note: Not full conformance to RFC 2406 is required. Only compliance to the payload format as specified by this RFC is required.

5.1.1.6 FCS_COP.1(b) - Cryptographic operation (file decryption)

FCS_COP.1.1 The TSF shall perform ***decryption of files*** in accordance with a specified cryptographic algorithm ***AES*** and cryptographic key sizes ***256 bits*** that meet the following: ***conform to RFC 2406.***

Application Note: Not full conformance to RFC 2406 is required. Only compliance to the payload format as specified by this RFC is required.

5.1.1.7 FCS_COP.1(c) - Cryptographic operation (keyed checksum generation)

FCS_COP.1.1 The TSF shall perform ***generation of keyed checksums*** in accordance with a specified cryptographic algorithm ***HMAC with SHA-256*** and cryptographic key sizes ***256 bits*** that meet the following: ***conform to RFC 2104***.

Application Note: Not full conformance to RFC 2104 is required. Only the format and method for keyed hashing is implemented as in the standard.

5.1.1.8 FCS_COP.1(d) - Cryptographic operation (keyed checksum validation)

FCS_COP.1.1 The TSF shall perform ***validation of keyed checksums*** in accordance with a specified cryptographic algorithm ***HMAC with SHA-256*** and cryptographic key sizes ***256 bits*** that meet the following: ***conform to RFC 2104***.

Application Note: Not full conformance to RFC 2104 is required. Only compliance to the format and method for keyed hashing is implemented as in the standard. For the checksum validation during file decryption (data file encryption as well as key file decryption) SOF high is claimed.

5.1.2 CLASS FDP - User Data Protection

5.1.2.1 FDP_ACC.1 - Subset access control (keystore access)

FDP_ACC.1.1 The TSF shall enforce the ***keystore access control SFP*** on ***all users, the keystore, the creation of the keystore and the access of the keystore.***.

5.1.2.2 FDP_ACF.1 - Security attribute based access control (keystore access)

FDP_ACF.1.1 The TSF shall enforce the ***keystore access control SFP*** to objects based on the following: ***users and the password.***

FDP_ACF.1.2 The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- ***the user must choose a password with appropriate quality which is assigned to the keystore when creating the keystore out of the keyfile,***
- ***the user must enter the correct password to access the keystore.***

FDP_ACF.1.3 The TSF shall explicitly authorise access of subjects to

objects based on the following additional rules: ***none***.

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the *rules:* ***none.***

5.1.2.3 FDP_ETC.2 - Export of user data with security attributes

FDP_ETC.2.1 The TSF shall enforce the ***keystore access control SFP*** when exporting user data, controlled under the SFP(s), outside of the TSC.

FDP_ETC.2.2 The TSF shall export the ***keys*** with the user data's associated security attributes.

FDP_ETC.2.3 The TSF shall ensure that the security attributes, when exported outside the TSC, are unambiguously associated with the exported user data.

FDP_ETC.2.4 The TSF shall enforce the following rules when user data is exported from the TSC: ***the keys must be wrapped in a password encrypted keyfile. Therefore***

- ***the keyfile must be encrypted using the encryption key derived from the assigned password (key unwrap password),***
- ***the keyfile must be integrity protected using the HMAC key derived from the assigned password over the encrypted key file.***

Application Note: This requirement regulates that only password encrypted and integrity protected key files are exported out of the TOE.

The right format of the keystore (binary file) and keyfile (XML-file) is useful regarding availability but not regarded as security relevant and therefore not described in the requirement. The security attribute is therefore only the password.

5.1.2.4 FDP_ITC.2 - Import of user data with security attributes

FDP_ITC.2.1 The TSF shall enforce the ***keystore access control SFP*** on ***keyfiles*** when importing user data, controlled under the SFP, from outside of the TSC.

FDP_ITC.2.2 The TSF shall use the security attributes associated with the imported user data.

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:

- ***the keystore must be provided in a binary format with the right attributes, equal to the format used by Filkrypto to export keys,***

- ***the keys must correctly be unwrapped out of then encrypted keystore,***
- ***the keyfiles must be correctly encrypted and integrity protected by Filkrypto using the right attributes and providing the right format,***
- ***the keys itself must be provided in XML-format, that is expected for keys generated by Filkrypto with the right attributes provided.***

Application Note: The attributes are algorithms, key lengths, number of bytes used for salt and the number of iteration.

The attributes associated with standard keys are, e.g., the associated algorithms used for encryption and integrity protection and the keyID which is an 128-bit key identifier.

Form keys are imported into the application manually, the integrity check while importing the form keys is not regarded as security relevant. Therefore no requirement is provided. Nevertheless form keys use as attributes only the keyID which is a 5 character identification code. These ASCII values are decoded in the first bytes of the keyID the rest is set to zero.

5.1.3 Class FMT - Security Management

5.1.3.1 FMT_MSA.1 - Management of security attributes

FMT_MSA.1.1 The TSF shall enforce the ***keystore access control SFP*** to restrict the ability to ***modify*** the security attributes ***keystore password*** to ***any user who knows the actual password***.

Application Note: The TOE is not aware of any user roles but controls the access to the keystore via a password. The TOE is assumed to operate on a single user machine with only one user having access to the TOE.

5.1.3.2 FMT_MSA.2 - Secure security attributes

FMT_MSA.2.1 The TSF shall ensure that only secure values are accepted for security attributes.

Application Note: Only keys with a key length of 256 bits and with algorithm attributes of cryptographic algorithms supported by the TOE are generated and used by the TOE.

5.1.3.3 FMT_MSA.3 - Static attribute initialisation

FMT_MSA.3.1 The TSF shall enforce the ***keystore access control SFP*** to provide ***no*** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow ***any user who knows the keystore password*** to specify alternative initial values to override the default values when an object or information is created.

Application Note: There are no default values for keystore passwords; the user has to choose a password himself when creating a keystore and/or starting the application the first time.

5.1.3.4 FMT_SMF.1 - Specification of Management Functions

FMT_SMF.1.1 The TSF shall be capable of performing the following security management functions:

- ***generate key (only standard keys)***
- ***delete key***
- ***delete keystore***
- ***change keystore password***
- ***export keys (only standard keys)***
- ***import keys***

Application Note: All management functions except delete keystore requires the user to present the password of the default keystore.

5.2 TOE Security Assurance Requirements

The target assurance components for this TOE are those for EAL3 as specified in Part 3 of the CC. The following table provides an overview of the assurance components that form the assurance level for the TOE.

Assurance class	Assurance components
Configuration management	ACM_CAP.3 Authorisation controls
	ACM_SCP.1 TOE CM coverage
Delivery and operation	ADO_DEL.1 Delivery procedures
	ADO_IGS.1 Installation, generation, and start-up procedures
Development	ADV_FSP.1 Informal functional specification
	ADV_HLD.2 Security enforcing high-level design
	ADV_RCR.1 Informal correspondence demonstration
Guidance and Documentation	AGD_ADM.1 Administrator guidance
	AGD_USR.1 User guidance
Life cycle	ALC_DVS.1 Identification of security measures
Tests	ATE_COV.2 Analysis of coverage
	ATE_DPT.1 Testing: high-level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
Vulnerability assessment	AVA_MSU.1 Examination of guidance
	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.1 Developer vulnerability analysis

Table 5: Security Assurance Components

6 TOE Summary Specification

The TOE summary specification provides a complete high-level definition of the security functions and assurance measures of the TOE and their relationship to the security functional and assurance requirements of this ST.

The TOE summary specification identifies the security functions that the TOE implements to meet the requirements defined in chapter 5 of the security target.

A SOF claims is made, for the SF.CHECK_INTEGRITY. For this security function SOF-high is claimed.

6.1 TOE Security Functions

This chapter describes the IT security functions of the TOE and their relation to the security functional requirements which they are supposed to meet.

A mapping of security functions against requirements is provided in clause 8.3 of the rationale part.

6.1.1 SF.KEYGEN - Key generation

The Keys used for encryption and decryption of data files are generated. These are symmetric AES keys of the length 256 bits. The keys are generated by the random number generator (PRNG) of the TSAlib (via FMSSL). Further keys for the keyed hash function to detect loss of integrity of the respective encryption keys - the HMAC-SHA256 keys - are generated with the same secret PRNG in a separate run. The seeding for the PRNG is done by a mechanism provided by Tutus but approved by TSA.

All information regarding the encryption key and it's associated HMAC key is present in a XML key file which contains in addition to the key data, e.g. as well a key identifier and the encryption format used. In the key data field both keys, the encryption key and the associated HMAC key are presented. The keys are BASE64 encoded.

6.1.2 SF.KEYDER - Key derivation

The Keys used for encryption and decryption of key files as well as the respective HMAC-keys are derived from passwords. For this purpose PBKDF2 in PKCS#5 as described in [RFC 2898] is used with 16 bytes salt and currently 8192 iterations. The key derivation function applies to a pseudo random number function corresponding to the HMAC message authentication code based on the SHA-1 hash function (HMAC-SHA1). The key derived is of the length of 512 bits. The first 256 bit are used for the encryption key and the last 256 bit for the associated HMAC-Key.

6.1.3 SF.FILE_CRYPT - File encryption / decryption

Data files as well as the key files are encrypted and decrypted using the symmetric AES algorithm in CBC mode with a 256-bit key.

All information regarding the encrypted data file is present in a XML file which contains in addition to the encrypted data, e.g. as well the key identifier of the key used for encryption for selecting the right key from the keystore when decrypting the file, the time for encryption, and the original file name.

In contrast to the data files, the encrypted key files are stored as binary files. They are

called as well encrypted keystores.

The format of the encrypted data in both cases is based on a modified IPsec ESP [RFC 2406] packet format. First 16 bytes random date followed by the "Payload Date Field" (encrypted data) of variable length padded up to a block length of 128 bit. Concatenated with the integrity check value of 32 bytes presented in the "Authentication Data Field". This integrity check value is calculated as described in SF.CREATE_HMAC over all former fields.

In the case of data files the encrypted data is in addition BASE64 encoded before included in the XML file. This applies not to the encrypted data in the case of key files.

The keys for encryption and decryption are either taken from the default keystore in the case of data files or derived from a given password (SF.KEYDER) in the case of key files. See as well SF.CHECK_INTEGRITY.

6.1.4 SF.CREATE_HMAC

For purpose of integrity checks, an HMAC with SHA-256 (HMAC-SHA256) is computed according to [RFC 2104] over the 16 bytes random data and the padded encryption data of the ESP packet. The file format is described in SF.FILE_CRYPT above. Therefore the Hash-function uses the 256 bit key provided in the respective last 256 bit of the respective key file data (BASE64 decoded), which is used for encryption and generates a 256 bit hash value which is concatenated to the ESP packet fields described above in the "Authentication Data field".

6.1.5 SF.CHECK_INTEGRITY

The first step is to validate that the file is encrypted with Filkrypto and that the required keys are available.

In the case of data files the XML-structure of the given file is parsed and it is checked if the key referenced in the keyID-field of the XML file is available in the default keystore. In the case of key files (default key files as well as user generated key files) SF.KEYDER is performed with the password given by the user (key unwrap password) to derive the required key.

Afterwards the integrity check is implemented. The HMAC is validated. Therefore the HMAC-SHA256 is calculated over the ESP packet minus the "Authentication Data field" (last 32 bytes) and compared with the value provided as Integrity Check Value in the "Authentication Data field". In both cases the last 256-bit are taken as HMAC-key either from the referenced key in the keystore or from the derived key.

Only if SF.CHECK_INTEGRITY is successful, decryption according to SF.FILE_CRYPT is performed with the first 256-bits of the corresponding key.

For this security function SOF-high is claimed for the mechanism implementing the verification of keyed check sums. This is done in accordance with the strength of function claim for the corresponding security functional requirement.

6.1.6 SF.DIST_KEYFILE

Before standard keys, stored in a XML file, described in SF.KEYGEN, are exported out of the application they are encapsulated in an encrypted keyfile using the modified IPsec ESP format which is described in SF.FILE_CRYPT. This includes as well the calculation of an value to perform integrity checks as described in

SF.CREATE_HMAC. The key to encrypt the keyfile is derived as described in SF.KEYDER from a password which has to be chosen by the user (key unwrap password) first.

The encrypted data is stored in a binary file called the keystore/keyfile of a special format together with additional information concerning the algorithm used for key derivation and its parameters, e.g. the number of bytes used for salt and the number of iterations. This binary file is an ordinary file which can be e.g. stored on removable media or send via e-mail, due to exchange it with another Filkrypto user, who has a as well a instance of the application running.

Before a Filkrypto user who receives an encrypted keyfile is able to import the keys into his default keystore he has to unwrap the keys out of the encrypted keystore by using the correct key unwrap password which is assigned to the keystore. If the decryption with the derived key from the password is successful (thus includes as well the integrity check as described in SF.CHECK_INTEGRITY before decryption) and the format of the presented keyfile complies to the Filkrypto format (see SF.KEYGEN), the keys are stored in the default keyfile for further use. See as well SF.MANAGE.

6.1.7 SF.MANAGE

The TOE allows the user to perform the following management functions:

- generate key (only standard keys)
- delete key
- delete keystore
- change keystore password
- export keys (only standard keys)
- import keys

All management functions except the function deleting the keystore requires the user to present the correct password of the default keystore. The default keystore can be deleted before the password is given and SF.CHECK_INTEGRITY, SF.KEYDER and SF.FILE_CRYPT (keystore decryption) is performed.

For export of keys, the keys are placed in a new keystore and a key unwrap password must be assigned by the user. For import of keys, the key unwrap password of the imported keystore must be entered by the user. See SF.KEYDIS. For default keystores the user is called to enter a password running Filkrypto the first time.

For changing the keystore password (default keystore) the user is called to enter the current password first.

In contrast to standard keys which are generated by the TOE, “form keys” are generated elsewhere and then imported into the application by entering a code manually from a form received. The code is a key label and an ASCII coded bit string representing 256 bit key data and concatenated with an integrity check value – the first 64 bit of the SHA-1 value calculated over the key label and 256 bit key. When importing the form key the application checks the integrity of the form key by calculating the SHA-1 over the given key label and 256 bit key data for the form key and compares the first 64 bit of the calculated value with the given integrity check value. “Form keys” are only imported when the integrity check is valid. This “form key check” is not considered to be a security function because it is only provided to

detect failures while entering the "form key" from the form, assumed to be received in a secure manner out of band. This addresses only availability aspects.

6.1.8 SF.CLEAR

Keys can be properly destroyed when stored in memory and on files. In memory, they are overwritten with zeroes and on files, they are overwritten with random data using Bulk PRNG which is based on an algorithm owned and approved by TSA.

6.2 TOE Assurance Measures

This chapter gives information about the measures the developer has taken to achieve the desired EAL3 assurance level. Because the TOE security assurance requirements are exclusively based on the [CC] assurance components, we only provide a reference to the documents that show that the assurance requirements are met (see the [CEM] application note to ASE_TSS.1-1).

SAR	Assurance Measure
ACM_CAP.3	AM.CAP: The configuration management tool CVS is used to manage the configuration items of the TOE. The manual of the CVS tool and the procedures for using it are documented in separate documents. The TOE is referenced by unique version numbers and is labeled with its reference. Documentation is provided as part of the Filkrypto configuration management documentation. The CM tool is used to provide automated support for generating the TOE from its implementation representation as well as measures for authorized changes to configuration items. It provides unique identification of each configuration item.
ACM_SCP.1	AM.SCP: The CM system, as documented above, tracks the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation and CM documentation. Therefore all evaluation evidence is under the control of the CM system.
ADO_DEL.1	AM.DEL: The procedures for delivery of the TOE to the user can be found in the document Filkrypto delivery procedures, providing details how packaging and delivery is performed and how the integrity of the TOE can be maintained when delivered to the customer.
ADO_IGS.1	AM.USR: Since this is a product that can be installed by the end user, necessary steps for the secure installation, generation and start-up of the TOE are documented in the user documentation.
ADV_FSP.1	AM.FSP: The developer provides the functional specification together with a security enforcing high-level design in the Filkrypto design documentation, covering both the functional specification and the high-level design descriptions. This documentation will also describe the design from a security point of view in external visible security interfaces.
ADV_HLD.2	AM.HLD: The security enforcing high-level design will be provided in the Filkrypto design documentation, as described above in AM.FSP. This documentation will also describe the design from a security point of view in terms of subsystems.

SAR	Assurance Measure
ADV_RCR.1	AM.RCR: An informal correspondence analysis between the security target TOE summary specification, the functional specification and high-level design is given in the separate design document that specifically addresses the correspondence between the different levels of design descriptions.
AGD_ADM.1	AM.USR: The developer provides the administrator guidance together with the user guidance in the Filkrypto users guide, due to the fact that an explicit administrator role is not existing. This document will among other describe the security features of the product and how to use them in secure way and any assumptions for using them in a secure way.
AGD_USR.1	AM.USR: The user guidance is provided in the document Filkrypto user guide as identified above.
ALC_DVS.1	AM.DVS: Development security documentation can be found documented in the description of the Filkrypto development environment". It documents the security aspects in the development environment along with the development processes for the life-cycle definition/model, and the documentation of the development tools.
ATE_COV.2	AM.TST: An analysis of the test coverage and depth of testing is provided together with the test documentation in the test documentation that is describing the test plans, procedures including a documentation of the performed vulnerability analysis.
ATE_DPT.1	AM.TST: see above
ATE_FUN.1	AM.TST: see above Testing will be performed on the platforms as defined by the ST. Test results are documented such that the testing can be repeated.
ATE_IND.2	AM.IND: Independent testing will be performed by the evaluation facility. The TOE and an equivalent set of resources are provided to the evaluation facility in a manner suitable for testing.
AVA_MSU.1	AM.VLA: The misuse analysis, checking the guidance documentation, is documented as part of the Filkrypto vulnerability analysis.
AVA_SOF.1	AM.VLA: The TOE includes one mechanism having a strength of TOE security function claim. For this mechanism, a strength of TOE security function rationale is provided in chapter 8.2.4 of this Security Target. The strength of function analysis is documented as part of the Filkrypto vulnerability analysis.
AVA_VLA.1	AM.VLA: The vulnerability analysis done by the developer is documented in the Filkrypto vulnerability analysis.

Table 6: TOE Assurance Measures

7 PP Claims

This Security Target does not claim conformance with any Protection Profile.

8 Rationale

The rationale section demonstrates how the security objectives of the TOE are met and how objectives, threats and security functions relate to each other. The rationale section will identify which security functions contribute to which objectives and which threats are countered by the individual security functions.

8.1 Security Objectives Rationale

8.1.1 Security Objective Coverage

The following tables provide a mapping of security objectives to the environment defined by the threats, policies and assumptions, illustrating that each security objective covers at least one threat and that each threat is countered by at least one objective, assumption or policy.

	T.TAMPER	T.DISCLOSE	A.KEYDIS	A.FORMKEYDIS	A.SINGLE	A.PHYSICAL	A.USER	A.CONNECT	P.ERASURE	P.EMERGENCY	P.ALGORITHM
O.TAMPER	X										
O.DISCLOSE		X									
O.ERASURE									X		
O.EMERGENCY										X	
O.ALGORITHM											X
OE.KEYDIS	X	X	X								
OE.FORMKEYDIS	X	X		X							
OE.SINGLE					X						
OE.PHYSICAL						X					
OE.USER	X	X					X				
OE.CONNECT								X			

Table 7: Objectives related to threats, assumptions and policies

8.1.2 Security Objectives Sufficiency

The following rationale provides justification that the security objectives are suitable to counter each individual threat and that each security objective tracing back to a threat, when achieved, actually contributes to the removal, diminishing or mitigation of that threat:

Threat	Is addressed by
T.DISCLOSE	O.DISCLOSE (existence of mechanism to protect confidentiality itself) requires the TOE to provide mechanism of high quality to protect the confidentiality of the file's content while transferring it over any unprotected communication channel. OE.KEYDIS and OE.FORMKEY requires in addition that the TOE only uses keys for encryption / decryption of high quality (e.g. exclusion of weak keys and providing a sufficient key length to protect against successful brute-force key search or against attacks together with methods of cryptanalysis). T.DISCLOSE is diminished by reducing the likelihood of a launched attack being successful; greater expertise and greater resources are needed from the attacker to perform attacks based on cryptanalysis. Further OE.KEYDIS and OE.FORMKEY are requiring that the keys used for encryption / decryption are distributed and managed in a way that only authorized parties receive the keys used for encryption / decryption. T.DISCLOSE is diminished since restricting potential attackers in opportunities to decrypt the keystore. Also OE.KEYDIS and OE.FORMKEY are requiring the keys not to be disclosed to unauthorized users. Thus and OE.USER requiring that users to be trustworthy and well trained restricts the opportunity of unauthorized users possessing the right keys as well. T.DISCLOSE is diminished by O.DISCLOSE together with OE.KEYDIS or OE.FORMKEY and OE.USER.
T.TAMPER	O.TEMPER (existence of mechanism to detect integrity violations itself) requires the TOE to provide mechanism of high quality to detect integrity violations of the file's content while transferring it over any unprotected communication channel. OE.KEYDIS and OE.FORMKEY requires in addition that the TOE only uses keys for integrity checks of high quality. T.TEMPER is diminished by reducing the likelihood of a launched attack being successful; greater expertise and greater resources are needed from the attacker to perform attacks based on cryptanalysis. Further OE.KEYDIS and OE.FORMKEY are requiring the key used for integrity protection distributed and managed in a way that only authorized parties receive the keys. T.TEMPER is diminished since restricting potential attackers in opportunities to decrypt the keystore and get access to the key used for validating the integrity of transmitted files. Also OE.KEYDIS and OE.FORMKEY are requiring the key used for integrity checks not to be disclosed to unauthorized users. Thus and OE.USER requiring users to be trustworthy and well trained restricts the opportunity of unauthorized users possessing the right key. T.TEMPER is diminished by O.TEMPER together with OE.KEYDIS or OE.FORMKEY and OE.USER.

Table 8: Sufficiency of objectives countering threats

The following rationale provides justification that the security objectives for the environment are suitable to cover each individual assumption, that each security objective for the environment that traces back to an assumption about the environment of use of the TOE, when achieved, actually contributes to the environment achieving consistency with the assumption.

Assumption	Is fulfilled by
A.KEYDIS: “It is assumed that keys used for encryption/decryption and as well as the associated keys used for integrity checks are of high quality and are not disclosed to unauthorized users. The keys are assumed to be distributed only to those parties who are authorized to use them in order to encrypt and decrypt files.”	OE.KEYDIS require hat keys used for encryption/decryption as well as the keys used for integrity checks must be of high quality and must not be disclosed to unauthorized users. OE.KEYDIS also requires that the keys must be distributed only to those parties who are authorized to use them in order to encrypt and decrypt files. Therefore OE.KEYDIS is only a restatement of A.KEYDIS i.e. OE.KEYDIS fulfils exactly the assumption A.KEYDIS.
A.FORMKEYDIS: “Form keys are assumed to be distributed out of band from the generating party in a secure manner, therefore they are assumed to be not disclosed and tampered during distribution. Otherwise the same assumptions apply to form keys as to keys generated in Filkrypto as described in A.KEYDIS. They are of high quality and not disclosed to unauthorized users.”	OE.FORMKEY requires that the form keys must be distributed from the generating party in a secure manner. Therefore it is required that they must not be disclosed and tampered during distribution. Otherwise it is required by OE.FORMKEY that the same requirements must be ensured to form keys as to keys generated in Filkrypto as described in OE.KEYDIS, i.e. they must be of high quality and must not disclosed to unauthorized users. Therefore OE.FORMKEY is only a restatement of A.FORMKEY; i.e. OE.FORMKEY fulfils exactly the assumption A.FORMKEY.
A.SINGLE: “The TOE runs on a single user machine with access protected by the TOE environment; i.e. only authorised users of the TOE environment may access the TOE. This includes access control provided by the operating system or equivalent and protection against malware.”	OE.SINGLE requires that the TOE must be run on a single user machine with access to the TOE protected by the TOE environment; i.e., only authorised users of the TOE environment have access to the TOE. This includes access control provided by the operating system or equivalent and protection against malware. Therefore OE.SINGLE is only a restatement of A.SINGLE; i.e. OE.SINGLE fulfils exactly the assumption A.SINGLE.
A.PHYSICAL: “The TOE is operated in a physically secure and well managed environment.”	OE.PHYSICAL requires that the TOE must be run and therefore operated n a physically secure and well managed environment. Therefore OE.PHYSICAL is merely a restatement of A.PHYSICAL; i.e. OE.PHYSICAL fulfils the assumption A.PHYSICAL.
A.USER: “The TOE user is trustworthy and trained to manage and perform encryption of classified information in accordance with any existing security policies and information classification policies. This means especially that he knows how to classify information and how to deal with, e.g., encrypting all files containing sensitive information with the appropriate key before exporting the file out of the TOE and/or its TOE environment.”	OE.USER requires that the TOE User is trustworthy and trained to perform all actions in accordance with any existing security policies and information classification policies. OE.USER is merely a restatement of A.USER where the explanation of performing actions in accordance with any existing security policies and information classification policies is not given again because this has to be be clear to the reader. Therefore OE.USER fulfils the assumption A.USER.

Assumption	Is fulfilled by
A.CONNECT: "The single user PC on which the TOE is running is not connected directly to an untrusted network. This means that the PC is either assumed not to be connected to any networks or it is connected to a trusted network which is protected against attacks, so that no undocumented security critical side effects on the security functions of the TOE, which is resided in the PC, are assumed coming from this network."	OE.CONNECT requires that the single user PC on which the TOE is running must not be connected directly to an untrusted network. This means that the PC must either not be connected to any networks or it must be connected to a trusted network, which is protected against attacks, so that no undocumented security critical side effects on the security functions of the TOE are coming from this network. Therefore OE.CONNECT is merely a restatement of A.CONNECT; i.e. OE.CONNECT fulfils the assumption A.CONNECT.

Table 9: Sufficiency of objectives meeting assumptions

The following rationale provides justification hat the security objectives are suitable to cover each individual organizational security policy, that each security objective that traces back to an OSP, when achieved, actually contributes to the implementation of the OSP, and that if all security objectives that trace back to an OSP are achieved, the OSP is implemented:

OSP	Is addressed by
P.ERASURE: "Individual encryption keys shall be deleted upon the request of the authorized user."	O.ERASURE requires that individual encryption keys must be deleted upon the request of the authorized user. Therefore O.ERASURE implements exactly the policy P.ERASURE.
P.EMERGENCY: "All encryption keys contained in the default keystore shall be deleted in case of emergency. "	O.EMERGENCY requires that all encryption keys contained in the default keystore must be deleted in case of emergency. Therefore O.EMERGENCY implements exactly he policy P.EMERGENCY.
P.ALGORITHM: "The TOE shall only allow the use of approved encryption algorithms and key lengths."	O.ALGORITHM requires that only approved encryption algorithms and key lengths must be used. Therefore O.ALGORITHM implements exactly the policy P.ALGORITHM.

Table 10: Sufficiency of objectives meeting OSPs

8.2 Security Requirements Rationale

8.2.1 Security Requirements Coverage

The following tables provide a mapping of the relationships of security functional requirements to objectives, illustrating that each security requirement covers at least one objective and that each objective is covered by at least one security requirement.

	O.TAMPER	O.DISCLOSE	O.ERASURE	O.EMERGENCY	O.ALGORITHM
FCS_CKM.1(a)	X	X			
FCS_CKM.1(b)	X	(X)			
FCS_CKM.2	X	X			
FCS_CKM.4			X	X	
FCS_COP.1(a)		X			X
FCS_COP.1(b)		(X)			X
FCS_COP.1(c)	X				X
FCS_COP.1(d)	X				X
FDP_ACC.1	X	X	X		
FDP_ACF.1	X	X	X		
FDP_ETC.2	X	X			X
FDP_ITC.2	X	(X)			X
FMT_MSA.1	(X)	(X)	(X)		
FMT_MSA.2	X	X			X
FMT_MSA.3					X
FMT_SMF.1	X	X	X	X	

Table 11: TOE Security objectives meeting SFRs

The crosses in brackets apply to availability aspects. See the rational in 8.2.2.

8.2.2 Functional Security Requirements Sufficiency

Objective	Is fulfilled by the SFRs
O.TAMPER	The mechanisms to detect loss of integrity of the information included in a transmitted file is achieved by the cryptographic operations FCS_COP.1(d) together with FCS_COP.1(c). On the sender side the checksum is calculated (FCS_COP.1(c)) first, before it could be validated on the receiver side (FCS_COP.1(d)) where the properly detection of potential integrity violation takes place. The mechanisms are based on keyed hash functions, therefore some supporting requirements regarding keys are needed. Hash keys are generated (FCS_CKM.1(a)) on the sender side, exported out of the TOE (FDP_ETC.2 together with FDP_ACC.1 and FDP_ACF.1) and imported on the receiver side (FDP_ITC.2 together with FDP_ACC.1 and FDP_ACF.1) in a secure way into receivers TOE. The hash key used for keystore checksum validation on the receiver side is derived from a password which has to be assigned first to the keystore during creation on the sender side FCS_CKM.1(b) is needed. With FMT_MSA.1 only users how know the password could change it (availability of keys for integrity checks). For Form keys imported manually no requirement is provided. For keystores the same cryptographic operations as for data files are used. The distribution itself is supported by FCS_CKM.2 demanding keystores of a special format. FMT_MSA.2 ensures that only keyed checksum functions and key length are generated and used that are supported by the TOE. FMT_SMF.1 provides the specific management functions for key generation.

Objective	Is fulfilled by the SFRs
O.DISCLOSE	The mechanism to protect files during transmission against confidentiality violation is achieved by the cryptographic operation FCS_COP.1(a) ensuring, that the files encrypted such that the content is confidentiality protected and that only parties who know the right encryption key could decrypt the file. The cryptographic operation FCS_COP.1(b) providing the file decryption on the receivers site is strictly spoken not needed to achieve integrity protection but implemented to provide access to the encrypted information to those who are authorized (availability). These mechanisms are using keys, therefore supporting requirements regarding keys are needed. Encryption keys are generated (FCS_CKM.1(a)) on the sender side, exported out of the TOE (FDP_ETC.2 together with FDP_ACC.1 and FDP_ACF.1) and imported on the receiver side (availability: FDP_ITC.2 together with FDP_ACC.1 and FDP_ACF.1) in a secure way into receivers TOE. The encryption key used for keystore decryption on the receiver side is derived from a password which has to be assigned first to the keystore during creation on the sender side. FCS_CKM.1(b) is needed. With FMT_MSA.1 only users how know the password could change it (availability of keys). For Form keys imported manually no requirement is provided. For keystores the same cryptographic operations as for data files are used. The distribution itself is supported by FCS_CKM.2 demanding keystores of a special format. FMT_MSA.2 ensure that only the approved algorithms for encryption/decryption and key length are generated and used that are supported by the TOE. FMT_SMF.1 provides the specific management functions for key generation.
O.ERASURE	FDP_ACC.1 and FDP_ACF.1 ensures that keys are only available to those users possessing the right password for the keystore. FCS_CKM.4 ensures that individual encryption keys are deleted, keys in memory are overwritten with zeroes, and stored keyfiles are overwritten with random data. With FMT_MSA.1 only users how know the password could change it (availability of keys to erase). FMT_SMF.1 provides the specific management functions for key erasure.
O.EMERGENCY	FCS_CKM.4 ensures that the default keystore is deleted, keys in memory are overwritten with zeroes, and the stored default keyfile is overwritten with random data. FMT_SMF.1 provides the specific management functions for erasing all keys.

Objective	Is fulfilled by the SFRs
O.ALGORITHM	The key generation and derivation requirement achieves that only approved key generation and derivation algorithms with a specified key size are allowed (FCS_CKM.1(a) / FCS_CKM.1(b)). Further only approved algorithms with specified key sizes are allowed for cryptographic operations in the cryptographic operation requirements (FCS_COP.1(a) – FCS_COP.1(d)). These algorithms could not be managed by the user (see FMT_SMF.1 – no management function exists) they are fixed. Further only keystores could be deleted by authorized users knowing the password (FMT_MSA.1) there exists no function to change the attributes of the keys and the keys itself. FDP_ETC.2 ensures that keys exported are only for the approved encryption algorithms and key length. FDP_ITC.2 ensures that imported keys are for the approved encryption algorithms and key length. FMT_MSA..2 ensures that only the approved algorithms for encryption and decryption as well as key length are accepted by the TOE. FMT_MSA.3 ensures that no insecure algorithms and keys are in the keystore unless they have been generated or exported. Thus leads to the fact that only approved encryption algorithms and key length are allowed and used by the TOE.

Table 12: TOE Security Objectives and the Rationale for Mapping to the SFRs

As stated in the tables above, every objective is addressed by at least one security functional requirement and every SFR is necessitated to cover at least one objective. By showing that the stated security objectives are met, we are able to demonstrate the suitability and sufficiency of the chosen SFRs.

8.2.3 Rationale of Selected Assurance Level

The assurance level EAL3 has been chosen as appropriate for an application that is encrypting files in a secure and well managed environment. The attacker is also assumed only to attack the data exported or imported into the TOE and not the TOE itself, thereby limiting the opportunity of an attacker. For these reasons EAL3 is considered a sufficient level of assurance.

8.2.4 Rationale of SOF

This Security Target claims an overall SOF rating of SOF-high. This claim is made for FCS_COP.1 (d); the HMAC with SHA-256 used to validate keyed hash sums for detection of loss of integrity and authenticity of origin is claimed to be SOF-high. This claim of SOF-high is consistent with the security objectives and the assumption of the intended use.

8.2.5 Security Requirements Dependency Analysis

Following the Common Criteria and choosing security requirements to be met by a TOE, certain dependencies on other security requirements may arise. The following section shows whether these dependencies are resolved and, in case they are not, gives reasons for that.

8.2.5.1 Security Functional Requirements Dependency Analysis

If there are alternative requirements to resolve a dependency the valid ones are put in **bold** letters. Unresolved dependencies are put in ***italic bold*** letters.

Component	Dependencies/comment	Resolved
FCS_CKM.1(a)	[**FCS_CKM.2 Cryptographic key distribution**, or **FCS_COP.1 Cryptographic operation**] FCS_CKM.4 Cryptographic key destruction FMT_MSA.2 Secure security attributes	Yes Yes: FCS_COP.1(a),(b),(c),(d) Yes Yes
FCS_CKM.1(b)	[**FCS_CKM.2 Cryptographic key distribution**, or **FCS_COP.1 Cryptographic operation**] FCS_CKM.4 Cryptographic key destruction FMT_MSA.2 Secure security attributes	Yes Yes: FCS_COP.1(a), (b) Yes Yes
FCS_CKM.2	[FDP_ITC.1 Import of user data without security attributes, or **FDP_ITC.2 Import of user data with security attributes**, or **FCS_CKM.1 Cryptographic key generation**] FCS_CKM.4 Cryptographic key destruction FMT_MSA.2 Secure security attributes	-- Yes *Note: form keys cannot be exported; only standard keys can be distributed, they can as well be imported before* FCS_CKM.1(a) Yes Yes
FCS_CKM.4	[FDP_ITC.1 Import of user data without security attributes, or **FDP_ITC.2 Import of user data with security attributes**, or **FCS_CKM.1 Cryptographic key generation**] FMT_MSA.2 Secure security attributes	Yes Yes Yes: FCS_CKM.1(a), (b) Yes
FCS_COP.1(a)	[FDP_ITC.1 Import of user data without security attributes, or **FDP_ITC.2 Import of user data with security attributes**, or **FCS_CKM.1 Cryptographic key generation**] FCS_CKM.4 Cryptographic key destruction FMT_MSA.2 Secure security attributes	-- Yes *FDP_ITC.2 and FCS_CKM.1 apply as keys may be self-generated or imported* Yes: FCS_CKM.1(a), (b) Yes Yes
FCS_COP.1(b)	[FDP_ITC.1 Import of user data without security attributes, or **FDP_ITC.2 Import of user data with security attributes**, or **FCS_CKM.1 Cryptographic key generation**] FCS_CKM.4 Cryptographic key destruction FMT_MSA.2 Secure security attributes	-- Yes Yes: FCS_CKM.1(a), (b) Yes Yes
FCS_COP.1(c)	[FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or **FCS_CKM.1 Cryptographic key generation**] FCS_CKM.4 Cryptographic key destruction FMT_MSA.2 Secure security attributes	-- -- Yes: FCS_CKM.1(a),(b) Yes Yes
FCS_COP.1(d)	[FDP_ITC.1 Import of user data without security attributes, or **FDP_ITC.2 Import of user data with security attributes**, or **FCS_CKM.1 Cryptographic key generation]** FCS_CKM.4 Cryptographic key destruction FMT_MSA.2 Secure security attributes	-- Yes Yes FCS_CKM.1(a),(b) in addition in the case of key files

Component	Dependencies/comment	Resolved
		Yes Yes
FDP_ACC.1	FDP_ACF.1 Security attribute based access control	Yes FDP_ACF.1
FDP_ACF.1	FDP_ACC.1 Subset access control FMT_MSA.3 Static attribute initialisation	Yes FDP_ACC.1 Yes
FDP_ETC.2	**[FDP_ACC.1 Subset access control,** or FDP_IFC.1 Subset information flow control]	Yes FDP_ACC.1 --
FDP_ITC.2	[**FDP_ACC.1 Subset access control,** or FDP_IFC.1 Subset information flow control] [***FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path]***	Yes FDP_ACC.1 -- No: Either a trusted communication channel between the TSF and another trusted IT products or a trusted communication path between users and the TSF are required, i.e. secure communication ensuring confidentiality, integrity and authenticity protection, is required. The dependencies are not satisfied, however confidentiality, integrity and authenticity is for the security attributes are satisfied. Confidentiality is satisfied for standard keys by FCS_COP.1(a) and integrity protection as well as authenticity protection by FCS_COP.1(c) and (d). In the case of form keys no such protection is needed by the TOE, as they are no subject to any disclosure or tampering.
	FPT_TDC.1 Inter-TSF basic TSF data consistency	No: this dependency is not satisfied by FPT_TDC.1. The confidentiality, integrity and authenticity of keys and key attributes imported is satisfied by the protection used for user data as described above. For user date the integrity and therefore the consistency is protected by requiring FCS_COP.1(c) and (d).
FMT_MSA.1	[**FDP_ACC.1 Subset access control**, or FDP_IFC.1 Subset information flow control] ***FMT_SMR.1 Security roles***	Yes -- No: The TOE does not know about security roles; it relies on the TOE environment for user access control, but there are no specific requirements on the TOE environment to maintain separate roles.
	FMT_SMF.1 Specification of Management Functions	Yes
FMT_MSA.2	***ADV_SPM.1 Informal TOE security policy model***	No: ADV_SPM.1 would apply to the TOE environment as the user handling and user access control is done there (see also A.SINGLE).
	[FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control]	Yes --
	FMT_MSA.1 Management of security attributes ***FMT_SMR.1 Security roles***	Yes No: The TOE does not know about

Component	Dependencies/comment	Resolved
		security roles; it relies on the TOE environment for user access control, but there are no specific requirements on the TOE environment to maintain separate roles.
FMT_MSA.3	FMT_MSA.1Management of security attributes ***FMT_SMR.1 Security roles***	Yes No: The TOE does not know about security roles; it relies on the TOE environment for user access control, but there are no specific requirements on the TOE environment to maintain separate roles.
FMT_SMF.1	No dependencies	Yes

Table 13: Security Functional Requirements Dependencies for the TOE

8.2.5.2 Security Assurance Dependencies Analysis

The assurance level selected within this TOE is EAL3. Since the dependency analysis for EAL3 has been performed by the authors of the CC and as all dependent assurance components have been included, all dependencies of the assurance components within this Security Target are resolved.

8.2.5.3 Rationale of unresolved dependencies

See table 11 for the rationale on unresolved dependencies.

8.2.6 Internal Consistency and Mutual Support of SFRs

Section 8.3.2 has already demonstrated how the IT security requirements work together to implement the individual objectives for the TOE and the IT environment. This section will elaborate on the internal consistency and mutual support of the IT security requirements.

The TOE's purpose is to enable users to exchange electronic documents securely over unprotected communication paths by ensuring confidentiality with encryption and the detection of loss of integrity by using keyed Hash-functions.

Therefore cryptographic keys have to be generated first (FCS_CKM.1(a)). They are stored on the hard disk within a password protected default keyfile (keystore).

When using Filkrypto for the first time an initial password for the keystore has to be set by the user. From this password the key for encryption/decryption of the keystore is derived as well as the HMAC key (FCS_CKM.1(b)). In main memory a keylist is created which is initially empty. Every time the user updates that list by adding or deleting keys the keyfile on the hard disk is actualized. Therefore the keylist is encrypted with the derived encryption key (FCS_COP.1(a)) and stored in the same format (modified IPsec ESP format) used for distribution (FCS_CKM.2) of keyfiles. In the last 32 bytes of the ESP the HMAC-SHA256 is stored which is computed over the 16 bytes random data and the padded encryption data of the ESP packet by using the 256 bit HMAC key derived from the password and the SHA-256 calculating a 256 bit hash value (FCS_COP.1 (c)) for detection of loss of integrity and authenticity of origin.

Every further time the user starts Filkrypto he has to decrypt the default keystore first. Therefore he has to enter the actual password of the keystore, the keys are

derived (FCS_CKM.1(b)) and the checksum is validated (FCS_COP.1(d) by calculating the HMAC-SHA256 over the ESP packet without the "Authentication Data field" (last 32 bytes) and comparing this with the value provided as Integrity Check Value in the "Authentication Data field". Only if this succeeds the keyfile is decrypted (FCS_COP.1(a)) and stored in main memory. If this fails either the password is incorrect or the integrity of the encrypted keyfile is violated in both cases the user could not access the keyfile. If the keylist is available in main memory the user could perform some management functions (FMT_SMF.1.). He can generate new keys FCS_CKM.1(a)), delete keys and keystores (FCS_CKM.4). In case of an emergency the default keystore could be erased. Thus could be done before the password is given and the keylist is present in main memory or when the default keystore is already open. The keystore passwords could be changed.

Self generated standard keys (FCS_CKM.1) could be exported out of Filkrypto in order to distribute them to other users. This is done in a password encrypted keystore. The creation as well as the access to the keystore is regulated by a access control SFP called keystore access control policy (FDP_ACC.1 and FDP_ACF.1). This policy is enforced while exporting keys respectively keyfiles (FDP_ETC.2) as well as during importing them (FDP_ITC.2).

After the user assigns a password for the keyfile (FDP_ACF.1), he wants to export, the respective key is derived by using sing FCS_CKM.1(b) for the key derivation from password. FDP_ETC regulates then that the keys are only exported in a special format encrypted keyfile format. Therefore they must be encrypted and integrity protected first. The format is the same as for the default keystore for encapsulation of keys (FCS_CKM.2). The different fields of the packet are computed in the same way as described above for the default keyfile using FCS_COP.1(a) for encryption and FCS_COP.1(c) for checksum generation.

After exporting the keyfile it can be sent over untrusted communication channels to users who are authorized to use the keys (FCS_CKM.2). After exchanging the password with this user in a secure manner - not part of the application - the user could import the keyfile. Therefore the keystore access control policy must be enforced when importing the keys (FDP_ITC.2). Thus allows only users having the right password to access the keystore. FDP_ITC.2 regulates in addition that only with Filkrypto encrypted and integrity protected keyfiles of the special Filkrypto format with the expected attributes are imported and added to the keyfile of the default keystore. For accessing the keys in the keystore the same SFRs have to be carried out as for accessing the default keystore but with the given exchanged password.

Form keys could not be added to keyfiles for distribution. They could only be imported manually from a form given in clear (FDP_ITC.2) and ASCII-coded with a checksum concatenated for input failure detection which is not regarded as security relevant as described in section 2.

Data Files could be encrypted and decrypted with all keys available in the keylist using FCS_COP.1(a) and FCS_COP.1(b).

As shown above there exist no conflicts between different requirements. Further they are consistent in defining a proper set of demands on the functionality the TOE is supposed to offer.

8.3 TOE Summary Specification Rationale

The TOE IT security functions work together to satisfy the security functional

requirements. Below a justification is presented for each SFR, how the related security functions meet the requirements, and as well for the sum of SARs.

By examining the TOE summary specification and this justification carefully, it becomes clear that the security functions are a well defined set combined to build a sound application for file encryption and therefore to meet the requirements defined in this ST.

The following tables provide a mapping between security functions and security functional requirements, as well as assurance measures and security assurance requirements.

	SF.KEYGEN	SF.KEYDER	SF._FILE_CRYPT	SP.CREATE_HMAC	SF.CHECK_INTEGRITY	SF.DIST_KEYFILE	SF.MANAGE	SF.CLEAR
FCS_CKM.1(a)	X							
FCS_CKM.1(b)		X						
FCS_CKM.2						X		
FCS_CKM.4							X	X
FCS_COP.1(a)			X					
FCS_COP.1(b)			X					
FCS_COP.1(c)				X				
FCS_COP.1(d)					X			
FDP_ACC.1							X	
FDP_ACF.1							X	
FDP_ETC.2						X	X	
FDP_ITC.2						X	X	
FMT_MSA.1							X	
FMT_MSA.2	X		X	X	X	X		
FMT_MSA.3							X	
FMT_SMF.1							X	

Table 14: TOE Security Functions meeting SFRs and Vice Versa

8.3.1 Security Functions Justification

The following table shows that the IT security functions (SF) as specified in the TOE Summary Specification meet all the security functional requirements (SFR) for the TOE and work together to satisfy the TOE security functional requirements.

SFR	Security Functions (TOE Summary Specification)
FCS_CKM.1(a)	The requirement for key generation (standard key) is satisfied by the security function SF.KEYGEN, which will generate all encryption keys AES-keys and in a special run of the same function the HMAC-Keys, that are not imported into the TOE as form keys.
FCS_CKM.1(b)	The requirement for key generation (derivation from password) is satisfied by the security function SF.KEYDER, which will derive a 512 bit key from a given password implemented as described in [PKCS#5] and [RFC2898], where the first 256 bit are used for the encryption key (AES-key) and the last 256 bit are used for the corresponding HMAC-key.
FCS_CKM.2	The requirement for key distribution is satisfied by the security function SF.DIST_KEYFILE, defining the file format for encapsulating (wrapping) keys before exporting them out of the TOE.

SFR	Security Functions (TOE Summary Specification)
FCS_CKM.4	The requirement for key destruction is satisfied by the key destruction function SF.CLEAR in combination with the management function SF.MANAGE. SF.CLEAR is responsible for proper destroying the keys either in memory by overwriting with zeros or on files by overwriting with random data using a TSA owned algorithm. The fact that in the case of an emergency, all encryption keys in the default keystore are deleted immediately by implementing SF.CLEAR overwriting the default key file with random data without having access to the keystore (password not needed) is defined in SF.MANAGE.
FCS_COP.1(a)	The requirement for file encryption is satisfied by the security function SF.FILE_CRYPT, which specifies that AES in CBC mode with a 256 bit key is used for encryption, conform to [RFC 2406].
FCS_COP.1(b)	The requirement for file decryption is satisfied as well by the security function SF.FILE_CRYPT, which specifies that AES in CBC mode with a 256 bit key is used for decryption, conform to [RFC 2406].
FCS_COP.1(c)	The requirement for generation of keyed checksums is satisfied by the security function SF.CREAT_HMAC, which calculate a HMAC-SHA256 (HMAC using a 256 bit SHA key) according to [RFC 2104].
FCS_COP.1(d)	The requirement for validation of keyed checksums is satisfied by the security function SF.CHECK_INTEGRITY, which calculates HMAC-SHA256 (HMAC using a 256 bit SHA key) according to [RFC 2104] over the payload and compares this value with the value provided by SF.CREAT_HMAC.
FDP_ACC.1	The requirement for access control is satisfied by the management function SF.MANAGE, implementing the access control SFP for all users accessing the keystore.
FDP_ACF.1	The requirement for access control rules is satisfied by the management function SF.MANAGE, implementing that users must present the correct default keystore password before performing the management functions, except the function deleting the default keystore, this can be performed without entering password as a emergency erase functionality.
FDP_ETC.2	The requirement for export of keys is satisfied by SF.DIST_KEYFILE in combination with the management function SF.MANAGE. SF.DIST_KEYFILE controls the export of keys, i.e. only keys can be exported when they are encapsulated in an encrypted key file ESP. SF.MANAGE together with SF.DIST_KEYFILE define how the export is implemented in detail – in a password encrypted keystore where the user is ask to assign a password first..
FDP_ITC.2	The requirement for import of keys is satisfied as well by SF.DIST_KEYFILE in combination with the management function SF.MANAGE. SF.DIST_KEYFILE controls the import of standard keys, i.e. only keys can be imported when they are encapsulated in an encrypted key file using IPsec ESP. SF.MANAGE together with SF.DISTKEYFILE define how the import is implemented in detail – in a password encrypted keystore , where the user is ask to enter the correct password to unwrap the keys. Further it is defined in SF_MANAGE how the import of form keys is implemented as required as well in FDP_ITC.2.
FMT_MSA.1	The requirement for authorization of password changes for the keystore is implemented in SF.MANAGE, allowing password changes only to those users who know the current password of the keystore.

SFR	Security Functions (TOE Summary Specification)
FMT_MSA.2	The requirement for secure security attributes is satisfied by SF.KEYGEN together with SF.CREATE_HMAC and SF.FILECRYPT as well with SF.DIST_KEYFILE. SF.KEYGEN ensures that only secure values are generated and SF.DIST_KEYFILE ensures that these secure values are encapsulated in a sec ESP using SF.FILCRYPT and SF.CREATE_HMAC and distributed in this format to the receiver. SF.CHECK_INTEGRITY and SF.FILECRYPT would fail on the receivers side if insecure attributes are used. Therefore only secure attributes could be used within Filkrypto.
FMT_MSA.3	The requirement for static attribute installation is fulfilled by SF.MANAGE, by not providing any default attributes that could be insecure.
FMT_SMF.1	The requirement for the TSF to provide management functions is satisfied by SF.MANAGE implementing exact the same management functions.

Table 15: Security Function Rationale

8.3.2 Mutual Support of Security Functions

The IT security functions provided by the TOE work together to satisfy the TOE security functional requirements defined in this Security Target. The tight relationship between the defined requirements and the fulfilment of these requirements by security functions, as illustrated above in section 8.3.1, provides no room for the introduction of potential security weaknesses not identified in this document.

8.3.3 Assurance Measures Rationale

The TOE summary specification in section 6.2 includes a justification that the TOE security assurance requirements are met by the assurance measures.

8.3.4 Minimum Strength of Function Rationale

For the security function SF_CHECK_INTEGRITY, SOF-high is claimed for the mechanism implementing the verification of the keyed checksum HMAC SHA-256. This is done in accordance with the strength of function claim for the corresponding security functional requirement for checksum validation FCS_COP.1 (d).

No claims are made about the strength of function for any cryptographic algorithms. This is covered by the cryptographic verification performed by the government agency TSA.

8.4 PP Claims Rationale

No claims to any Protection Profile are made.

9 Appendix

A.1 Abbreviations

CCMB	Common Criteria Maintenance Board
EAL	Evaluation Assurance Level
ESP	Encapsulating Security Payload
FMSSL	Försvarsmaktens SSL
GUI	Graphical User Interface
IT	Information Technology
PP	Protection Profile
RFC	Request for comments
SF	Security Function
SFP	Security Function Policy
SOF	Strength of Function
ST	Security Target
TOE	Target of Evaluation
TSA	Totalförsvarets signalskyddssamordning
TSC	TSF Scope of Control
TSF	TOE Security Functions
TSFI	TSF Interface
TSP	TOE Security Policy

A.2 Glossary

Assets	Information or resources to be protected by the countermeasures of a TOE.
Assignment	The specification of an identified parameter in a component.
Assurance	Grounds for confidence that an entity meets its security objectives.
Attack potential	The perceived potential for success of an attack, should an attack be launched, expressed in terms of an attacker's expertise, resources and motivation.
Augmentation	The addition of one or more assurance component(s) from Part3 to an EAL or assurance package.
Authentication data	Information used to verify the claimed identity of a user.
Authorised user	A user who may, in accordance with the TSP, perform an operation.
Class	A grouping of families that share a common focus.

Component	The smallest selectable set of elements that may be included in a PP, an ST, or a package.
Connectivity	The property of the TOE which allows interaction with IT entities external to the TOE. This includes exchange of data by wire or by wireless means, over any distance, in any environment or configuration.
Dependency	A relationship between requirements such that the requirement that is depended upon must normally be satisfied for the other requirements to be able to meet their objectives.
Element	An indivisible security requirement.
Evaluation	Assessment of a PP, an ST or a TOE, against defined criteria.
Evaluation Assurance Level (EAL)	A package consisting of assurance components from Part 3 that represents a point on the CC predefined assurance scale.
Evaluation authority	A body that implements the CC for a specific community by means of an evaluation scheme and thereby sets the standards and monitors the quality of evaluations conducted by bodies within that community.
Evaluation scheme	The administrative and regulatory framework under which the CC is applied by an evaluation authority within a specific community.
Extension	The addition to an ST or PP of functional requirements not contained in Part2 and/ or assurance requirements not contained in Part 3 of the CC.
External IT entity	Any IT product or system, untrusted or trusted, outside of the TOE that interacts with the TOE.
Family	A grouping of components that share security objectives but may differ in emphasis or rigour.
Formal	Expressed in a restricted syntax language with defined semantics based on well-established mathematical concepts.
Human user	Any person who interacts with the TOE.
Identity	A representation (e.g., a string) uniquely identifying an authorised user, which can either be the full or abbreviated name of that user or a pseudonym.
Informal	Expressed in natural language.
Internal communication channel	A communication channel between separated parts of TOE.
Internal TOE transfer	Communicating data between separated parts of the TOE.
Inter-TSF transfers	Communicating data between the TOE and the

	security functions of other trusted IT products.
Iteration	The use of a component more than once with varying operations.
Object	An entity within the TSC that contains or receives information and upon which subjects perform operations.
Organisational security policies	One or more security rules, procedures, practices, or guidelines imposed by an organisation upon its operations.
Package	A reusable set of either functional or assurance components (e.g., an EAL), combined together to satisfy a set of identified security objectives.
Product	A package of IT software, firmware and/or hardware, providing functionality designed for use or incorporation within a multiplicity of systems.
Protection Profile (PP)	An implementation-independent set of security requirements for a category of TOEs that meet specific consumer needs.
Reference monitor	The concept of an abstract machine that enforces TOE access control policies.
Reference validation mechanism	An implementation of the reference monitor concept that possesses the following properties: it is tamperproof, always invoked, and simple enough to be subjected to thorough analysis and testing.
Refinement	The addition of details to a component.
Role	A predefined set of rules establishing the allowed interactions between a user and the TOE.
Secret	Information that must be known only to authorised users and/or the TSF in order to enforce a specific SFP.
Security attribute	Information associated with subjects, users and/or objects that is used for the enforcement of the TSP.
Security Function (SF)	A part or parts of the TOE that have to be relied upon for enforcing a closely related subset of the rules from the TSP.
Security Function Policy (SFP)	The security policy enforced by an SF.
Security objective	A statement of intent to counter identified threats and/or satisfy identified organisation security policies and assumptions.
Security Target (ST)	A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.
Selection	The specification of one or more items from a list in a component.

Semiformal	Expressed in a restricted syntax language with defined semantics.
Strength of Function (SOF)	A qualification of a TOE security function expressing the minimum efforts assumed necessary to defeat its expected security behaviour by directly attacking its underlying security mechanisms.
SOF-basic	A level of the TOE strength of function where analysis shows that the function provides adequate protection against casual breach of TOE security by attackers possessing a low attack potential.
SOF-medium	A level of the TOE strength of function where analysis shows that the function provides adequate protection against straightforward or intentional breach of TOE security by attackers possessing a moderate attack potential.
SOF-high	A level of the TOE strength of function where analysis shows that the function provides adequate protection against deliberately planned or organised breach of TOE security by attackers possessing a high attack potential.
Subject	An entity within the TSC that causes operations to be performed.
System	A specific IT installation, with a particular purpose and operational environment.
Target of Evaluation (TOE)	An IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation.
TOE resource	Anything usable or consumable in the TOE.
TOE Security Functions (TSF)	A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the TSP.
TOE Security Functions Interface (TSFI)	A set of interfaces, whether interactive (man-machine interface) or programmatic (application programming interface), through which TOE resources are accessed, mediated by the TSF, or information is obtained from the TSF.
TOE Security Policy (TSP)	A set of rules that regulate how assets are managed, protected and distributed within a TOE.
TOE security policy model	A structured representation of the security policy to be enforced by the TOE.
Transfers outside TSF control	Communicating data to entities not under control of the TSF.
Trusted channel	A means by which a TSF and a remote trusted IT product can communicate with necessary confidence to support the TSP.

Trusted path	A means by which a user and a TSF can communicate with necessary confidence to support the TSP.
TSF data	Data created by and for the TOE, that might affect the operation of the TOE.
TSF Scope of Control (TSC)	The set of interactions that can occur with or within a TOE and are subject to the rules of the TSP.
User	Any entity (human user or external IT entity) outside the TOE that interacts with the TOE.
User data	Data created by and for the user, that does not affect the operation of the TSF.