Myndigheten för samhällsskydd och beredskap

Datum	Diarienr	Utgåva
2018-07-04		1.0

Avd. för cybersäkerhet och skydd av samhällsviktig verksamhet
Ronny Harpe
010-2404426
ronny.harpe@msb.se

File Encryption Protection Profile

Table of content

1. Introduction

1.1 PP Reference

Title: File Encryption Protection Profile

Version: 1.0

Status: Released

Date: 2018-07-04

PP Author: Yi Cheng, atsec information security AB

Keywords: file encryption/decryption, integrity protection, non-repudiation, certificate, certification authority

1.2 TOE Type

The Target of Evaluation (TOE) is a file encryption application for protecting sensitive information in files when they are transmitted through unprotected environments.

1.3 TOE Overview

The TOE described in this PP is a software application for file encryption running on COTS operating system and hardware platform.

The TOE supports both pre-shared key based file encryption and public key based file encryption. Please note that in both cases a symmetric key is used to encrypt a file and the resulting ciphertext is integrity protected. In the case of pre-shared key based file encryption, two symmetric keys (one for encryption and the other for integrity protection) are generated and distributed to authorized parties in advance. In the case of public key based file encryption, the symmetric key for encryption is generated on the fly by the TOE, encrypted with the receiver's public key and provided to the receiver together with the encrypted file. The integrity of both the encrypted file and the encrypted symmetric key is provided by using the sender's private key to create a digital signature.

For pre-shared key based file encryption, the TOE supports import and export of pre-shared keys by putting these keys in a protected file. To distinguish such files from normal data files, the term "key file" is used wherever we refer to files that contain pre-shared keys.

The TOE provides the following security functionalities:

- Encryption of files
- Decryption of files
- Non-repudiation of file origin (when public key based file encryption is used)
- Passphrase generation

- Key management
- Certificate validation
- Configuration of trusted CAs
- Testing of crypto modules
- Logging
- Protection of sender/receiver identity

1.4 TOE Description

1.4.1 Introduction and intended use

This Protection Profile provides a high-level set of security requirements for file encryption applications, providing minimal functionalities that are necessary for such a product.

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

The TOE also provides integrity protection for exchanged files. A receiver can detect unauthorized modifications by checking digital signature or keyed hash value.

A user can sign a file before sending it to others. The sender's signature provides, in addition to integrity protection, proof of file origin. In this way non-repudiation can be achieved, i.e., the sender cannot falsely deny having sent the file. There may be situations in which the file content is not secret but the file's integrity and origin authenticity are critical. In these situations the user may only sign the file without encrypting it.

In order to exchange files securely both the sender and receiver need to have the TOE installed. The TOE is intended to be used in a physically secure environment and only one user shall have access to the TOE. The machine on which the TOE is running shall not be connected directly to any untrusted network. As such the operational environment prevents attacks towards the TOE itself. So this PP only considers attacks towards data files and key files that are exchanged between different instances of the TOE.

The TOE can work both online and offline. When working online, the TOE is able to obtain certificate revocation lists (CRLs) from the network and store them locally. When working offline, the TOE uses the locally stored CRLs to check certificate revocation status. The TOE should therefore be run in online mode in a frequency that is sufficient to keep the locally stored CRLs up to date. While online, the TOE may use OCSP (Online Certificate Status Protocol), as an alternative to CRL, to check certificate revocation status.

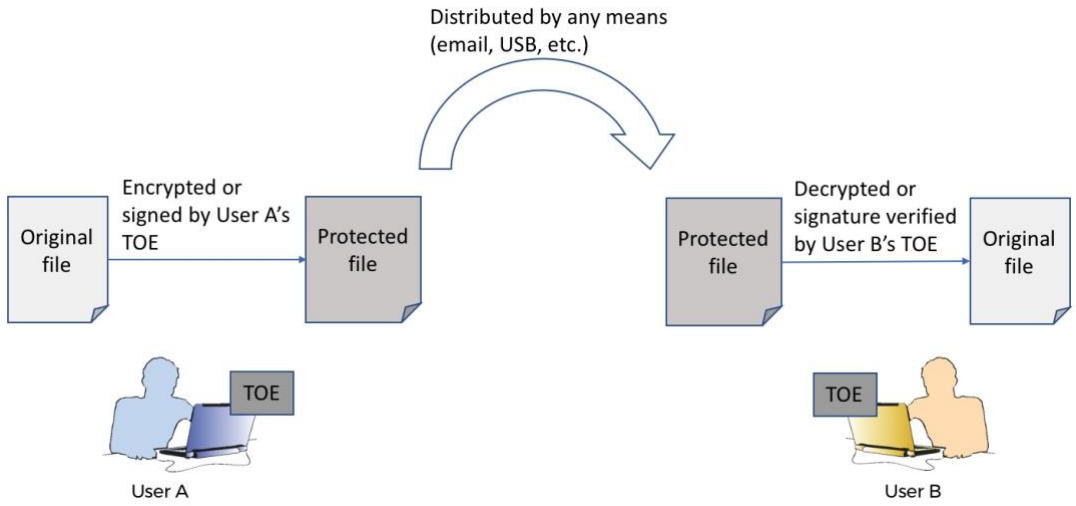


Figure 1 Secure exchange of files

1.4.1.1 Pre-shared key based file encryption

First, the pre-shared keys (PSKs) are created, distributed, and made accessible using the following steps:

1. The two parties agree on the key names for one or more PSKs that should be used when exchanging files.
2. One of the parties uses the TOE to create the PSK(s) and these keys are put in a key file.
3. The TOE generates a passphrase which is then used for protecting the key file. Alternatively, if these parties have asymmetric key pairs, the key pairs can be used to protect the key file.
4. The protected key file is distributed to the other party by any means, e.g. email. If a passphrase has been used to protect the key file, the passphrase is securely distributed to the other party by some out-of-band means (for instance when the parties meet in person).
5. The other party imports the PSK(s) from the key file, using either the passphrase or the asymmetric key pairs, so that both parties have access to the PSK(s).

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

6. Select the file(s) that you want to encrypt, and select which PSK you wish to use.
7. Initiate the encryption.

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

8. Send the file(s) to the concerned parties.
9. The receiving parties use the TOE to decrypt the file(s).

1.4.1.2 Public key based file encryption

In order to encrypt a file, the sender of the file needs to know the public key of the receiver. The sender can get the receiver's public key in the form of digital certificate from a certificate catalogue service. Alternatively, the receiver's certificate can be manually imported into the TOE of the sender.

The sender also has a pair of keys and can use the private key to sign the file to provide integrity protection and proof of file origin.

Now the sender can start to encrypt or sign files for secure file exchange using these steps:

1. Select the file(s) that you want to encrypt or sign, and select the receiver.
2. Initiate the encryption or signing.

Selected file(s) are now available encrypted or signed only.

3. Send the file(s) to the receiver(s).
4. The receiver(s) use the TOE to decrypt the file(s) or verify the sender's signature.

1.4.1.3 Anonymous file encryption

After pre-shared key based or public key based file protection has been applied, the resulting file contains metadata that identifies the key used to encrypt/sign the file (e.g. certificate, unique reference to certificate, or symmetric key identifier), so that the receiver knows which key to use for decryption/signature verification. Such metadata could reveal the identity of the sender and receiver. To prevent this kind of user identity disclosure, an extra layer of encryption can be applied to the encrypted or signed file by encrypting everything including the key identifying metadata. This extra encryption itself uses a symmetric key, but the metadata identifying this key is not included in the final output.

The symmetric key for anonymous file encryption must be made known to the sender and the receiver in advance (using steps 1-5 described in section 1.4.1.1). It could for example be a group key for the organization to which both the sender and receiver are affiliated. In order to decrypt a file protected by anonymous encryption, the receiver needs to know, by some out-of-band means, which anonymous encryption key has been used. Otherwise the receiver may have to try different keys.

The sender performs the following steps:

1. Select the file that you want to encrypt (with either pre-shared key or public key based encryption) or sign, and select the receiver.
2. Select the symmetric key that should be used for anonymous file encryption.
3. Initiate anonymous file encryption.

The selected file is first encrypted or signed. The encrypted or signed file is then encrypted with the selected anonymous encryption key to produce the final output, omitting the metadata identifying this key.

4. Send the final output to the receiver.
5. The receiver uses the TOE to first recover the encrypted or signed file, and then decrypt the file or verify the sender's signature.

1.4.2 The TOE architecture and functions

The TOE is a software application that supports both pre-shared key based file encryption and public key based file encryption. The file is encrypted with a pre-shared key (PSK) and then integrity protected using keyed hash with another PSK in the former case. In the latter case, the file is encrypted with a symmetric file encryption key (FEK) that is generated on the fly and the FEK itself is encrypted with the receiver's public key. A digital signature is then created using the sender's private key to provide integrity protection of both the encrypted file and the encrypted FEK. The digital signature also provides non-repudiation of file origin.

It is also possible to use the TOE to do anonymous file encryption, in which an extra layer of encryption is applied to the encrypted or signed file to prevent the disclosure of sender's/receiver's identity.

User's private key is managed by an external private key unit that stores the private key securely and uses it to perform signing and asymmetric decryption. The private key unit can be a smart card or a soft certificate utility, and it is a part of the TOE environment.

The PSKs (for symmetric encryption, keyed hash or anonymous file encryption) are generated within the TOE. They can be exported from the TOE in an encrypted and integrity protected key file. Key file protections can be performed using a randomly generated passphrase or using the sender's and receiver's asymmetric key pairs. The protected key file is distributed to authorized parties who can then import the PSKs into their instances of the TOE.

The TOE stores the PSKs, both locally generated and imported, in an encrypted and integrity protected key store. Key store protections can be performed using a user-chosen password or using the user's asymmetric key pair. In the former case the password should be chosen according to certain password policy to ensure good quality. In the latter case a key store encryption key (KSEK) is generated by the TOE and used to encrypt the key store. The KSEK is in turn encrypted with the user's public key and stored within the TOE.

Public keys can be obtained in the form of digital certificates from an external server. This server provides a trusted certificate catalogue service for clients to search and retrieve certificates (including CA certificates) and CRLs. The catalogue service is a part of the TOE environment.

The catalogue service also provides a list of Certification Authorities (CAs) that are recommended to the users as trusted CAs. Each individual user can choose to trust all

or only a subset of those CAs for the purpose of file protection (i.e. confidentiality and integrity protection of files). The user can also add other CAs to his/her list of trusted CAs. It is assumed that the user is security conscious and only trusts CA certificates that are known to be trustworthy (e.g. a self-signed root certificate issued by the user's organization). The TOE maintains the list of trusted CAs and stores the certificates of these CAs in the protected key store. Note that different trust levels may be defined and assigned to each trusted CA (e.g. trusted only for receiving files), but this PP does not specify such functionalities.

Public key certificates and CRLs can also be imported into the TOE manually, for example, from a USB-Data Storage Device.

The imported certificates and CRLs are stored in a local database called cert store. At the time point when a certificate is to be used for a file protection operation, the TOE verifies that the certificate is valid, i.e. the certificate is in a valid format, has a valid certificate path that terminates with a trusted CA certificate, has not expired or been revoked. Certificate revocation checking is done by using CRL or OCSP. The latter requires the existence of OCSP responders which are considered part of the operational environment.

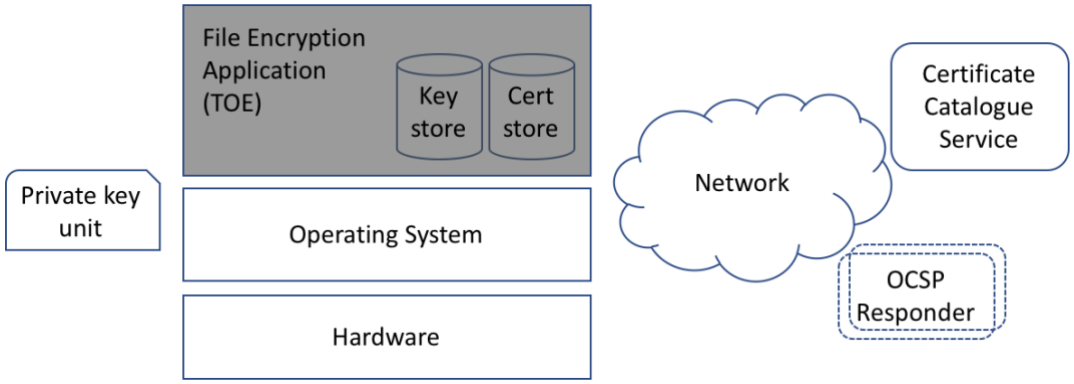


Figure 2 The TOE security architecture

Furthermore, the TOE erases FEKs after usage. The TOE deletes individual PSKs upon user request. In case of emergency, all the keys contained in the key store can be deleted at once, for example when the user pushes an emergency erase button.

The TOE uses only cryptographic algorithms and key lengths that are recommended by the Swedish Certification Body for IT Security (CSEC). The CSEC recommended algorithms and key lengths are given in CSEC Scheme Publication 188: Scheme Crypto Policy [SP-188]. It is assumed that the operational environment provides random numbers with sufficient entropy to the TOE for key generation.

In the following paragraphs the TOE's security functionalities are described in more details.

1.4.3 The TOE security functionality (TSF) summary

Encryption of files

The TOE can encrypt a file with a pre-shared key (PSK) selected by the user, and then protect the integrity of the encrypted file using a keyed hash function (HMAC). The key used for HMAC has to be shared between the communicating parties, in addition to sharing the respective encryption/decryption key.

The TOE can also encrypt a file with a file encryption key (FEK) that is generated on the fly. The FEK itself is encrypted with the receiver's public key. If there are multiple receivers the FEK is encrypted multiple times, one for each receiver. The TOE then calculates a hash value of both the encrypted file and the encrypted FEK, and sends it to the private key unit for signing. The private key unit creates a digital signature by encrypting the hash value with the user's private key and returns the signature to the TOE. The TOE attaches the signature to the encrypted file and the encrypted FEK and sends all of them the receiver.

Decryption of files

The TOE first verifies the integrity of the received file. It detects whether a keyed hash or a digital signature has been used for integrity protection. If it is a keyed hash, the TOE retrieves the HMAC key from the key store and uses it to validate the keyed hash value. If it is a digital signature, the TOE verifies the signature using the sender's public key.

If the HMAC validation or signature verification fails the TOE stops processing the file and informs the user. Otherwise, the TOE detects whether a PSK or a FEK has been used to encrypt the file. If it is a PSK, the TOE retrieves the key from the key store, and uses it to decrypt the file. If it is a FEK, the TOE first sends the encrypted FEK to the private key unit for decryption. The private key unit performs asymmetric decryption with the user's private key to recover the FEK. The FEK is returned to the TOE which then uses it to decrypt the file.

Non-repudiation of file origin

When public key based file encryption is used the TOE enforces non-repudiation of file origin, i.e., a user cannot falsely deny having sent a file.

The TOE calculates a hash value of the file (encrypted or not) and sends it to the private key unit for signing. The private key unit creates a digital signature by encrypting the hash value with the user's private key and returns the signature to the TOE. The TOE attaches the signature to the file and sends both to the receiver.

The receiver verifies the signature on the received file using the sender's public key. If the verification succeeds, the receiver is assured that the file is originated from the sender. The signature can be used as proof of origin so that the sender cannot deny having sent the file, which enforces non-repudiation.

Passphrase generation

The TOE includes a passphrase generator that generates passphrases by randomly selecting words from a word list. The word list shall contain at least 7776 unique words and the generated passphrases shall contain at least 10 words.

Key management

Key management is a process to manage the whole lifecycle of cryptographic keys from generation through distribution to archiving and destruction. The TOE implements the following security relevant key management functions:

Key generation

- PSK/FEK/KSEK generation: symmetric keys with appropriate key lengths are generated by the TOE for file encryption and key store encryption.
- HMAC key generation: Keys with appropriate key lengths are generated by the TOE for integrity protection of data files.
- Key derivation:
    - The keys used to protect the key store can be derived from a password chosen by the user.
    - The keys used to protect a key file can be derived from a passphrase generated by the TOE.

Key storage

All PSKs and trusted CA certificates are stored in a key store and managed by the TOE.

The key store can be protected using a user-chosen password. When starting the TOE for the first time, the user is requested to choose a password for the key store. The TOE derives encryption and HMAC keys from the password and uses these keys to encrypt and integrity protect the key store. This password has to be entered each time the user wants to access the key store when starting the TOE.

Alternatively, the key store can be protected with the user's asymmetric key pair. In this case, the TOE generates a key store encryption key (KSEK) and uses it to encrypt the key store. The KSEK is in turn encrypted with the user's public key and stored within the TOE. The TOE also calculates a hash value of the key store and sends it to the user's private key unit for signing. Each time the user wants to access the key store when starting the TOE, the TOE verifies the signature with the user's public key, sends the encrypted KSEK to the private key unit for decryption and then uses the KSEK to decrypt the key store. Note that the user has to unlock the private key unit by entering the correct PIN/password, but that is enforced by the TOE environment.

The TOE closes the key store when the user terminates the application. The application may close the key store after a period of user inactivity and require the user to re-enter the password or PIN before allowing him/her to access the key store, but this PP does not mandate such functionality.

FEK distribution

FEKs are encrypted with receiver's public key and distributed to the receiver together with the encrypted data file.

PSK import/export

To exchange PSKs between different users, the keys can be exported by one user (the sender) and imported by another user (the receiver) within a key file.

The key file can be protected using a randomly generated passphrase. The sender's TOE generates the passphrase and derives encryption and HMAC keys from it. The key file is encrypted and integrity protected with the derived keys and then sent to the

receiver by any means. The passphrase needs to be securely distributed to the receiver by some out-of-band means. When importing PSKs on the receiver's side, the user is requested by the TOE to enter the correct passphrase. The receiver's TOE verifies the keyed hash and decrypts the key file to recover the PSKs.

Alternatively, the key file can be protected using the sender's and the receiver's asymmetric key pairs. In this case, the key file is encrypted with the receiver's public key and signed with the sender's private key (the signing operation is performed within the sender's private key unit). When importing PSKs on the receiver's side, the receiver's TOE verifies the sender's signature and sends the encrypted key file to the private key unit for decryption.

Key erasure

- Individual PSKs can be deleted from the key store upon user request.
- Emergency erasure: all PSKs contained in the key store are deleted immediately in case of emergency.
- All secret keys and keying material that temporarily exist in the memory (FEKs, PSKs, KSEK, passwords, passphrases, and encryption and HMAC keys derived from passwords and passphrases) are erased after usage.

Certificate validation

The TOE verifies the validity of a certificate by checking the following:

- The certificate is in a valid format with required fields;
- The certificate path is valid;
- The certificate path terminates with a trusted CA certificate;
- The certificate has not expired;
- The certificate is not revoked (using CRL or OCSP)

The TOE verifies certificate validity at the point of time the certificate is to be used for a file protection operation (encrypting the FEK for a file to be sent or verifying the signature on a received file). Certificate validation may also be performed before a certificate is imported into the TOE, but this PP does not mandate such functionality.

If the validation of a certificate fails, the TOE logs the failure and alerts the user.

Configuration of trusted CAs

The TOE gets a list of trusted CAs from the certificate catalogue service. A user can use the TOE to add or remove CAs from that list. The TOE maintains this list and stores the certificates of the trusted CAs in the key store.

Testing of crypto modules

The TOE tests the crypto modules within it at initial start-up to verify that they operate correctly.

The TOE also performs tests to verify that the external private key unit performs signing and other asymmetric cryptographic operations correctly.

Logging

The TOE generates a log of performed file protection operations (encryption, decryption, signing, signature verification, etc.). It records the time when the operation was performed, file name, and with which key. It also records failures (encryption failure, decryption failure, integrity verification failure, certificate validation failure, etc.). The logging function requires a reliable time source.

Protection of sender/receiver identity

When desired, anonymous file encryption can be used to protect the identity of the sender/receiver of a data file. In this case, the TOE first encrypts or signs the data file to produce an intermediate output, and then encrypts the intermediate output with a symmetric key to produce the final output. Since the metadata contained in the intermediate output is encrypted, it is not possible to identify the sender/receiver by examining the metadata.

1.4.4 Available non-TOE hardware/software/firmware

The TOE is the file encryption application. The following items are outside the TOE physical boundaries and therefore considered part of the TOE operational environment:

- The operating system on which the TOE is installed.
- The hardware platform on which the TOE is running.
- The private key unit (either smart card or soft certificate utility) that stores user private key and performs cryptographic operations.
- In case of smart card, the reader and associated driver.
- The external certificate catalogue service which provides certificates, CRLs and a list of trusted CAs on request.
- A reliable time source (which may be provided by the operating system).
- A high-quality entropy source (which may be provided by the operating system).
- OCSP responders if OCSP is used by the TOE for certificate revocation checking.

2. Conformance Claims

2.1 CC Conformance Claim

This PP is CC Part 2 extended and CC Part 3 conformant. This PP claims conformance to CC version 3.1 Revision 5.

This PP claims no conformance to any Protection Profile. This PP claims conformance to the EAL3 package of security assurance requirements, augmented with ALC_FLR.2.

2.2 Conformance Statement

This PP requires demonstrable conformance by any ST or PP claiming conformance to this PP.

3. Security Problem Definition

The security problem definition describes the security problem that is to be addressed by the TOE and its operational environment.

3.1 Threat Environment

This section describes the threat model for the TOE in term of threat agents, assets to be protected, and the actual threats addressed by the TOE.

3.1.1 Assets

The assets to be protected by the TOE are:

- Data files that contain information to be protected.
- Secret keys that are used to protect data files.

3.1.2 Threat agents

Threat agents are:

- Attackers who have access to any communication channel over which the integrity protected and confidentiality protected files (data files or key files) are transferred, e.g., networks or other paths of transmission where communication media like CDs, DVDs including the encrypted files could be shared.
- Attackers or users who use invalid certificates (including certificates issued by untrusted CAs), the former intentionally while the latter unintentionally.

The motivation of a threat agent is assumed to be commensurate with the assurance level claimed by this PP. Therefore, threat agents are assumed to have basic attack potential.

3.1.3 Threats

The TOE addresses the following threats.

T.DISCLOSE – loss of confidentiality

An attacker of one of the communication paths over which the data 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 can for example be achieved by eavesdropping, recording encrypted data during the transfer and decoding the encrypted data.

T.TAMPER – loss of integrity

An attacker of one of the communication paths over which the data 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 can for example be achieved by interrupting the transfer, modifying the content of the file (including replacing the whole file) and then re-constructing the integrity protection. Afterwards the modified file is sent to the intended destination.

T.KEY – key disclosure or modification

An attacker succeeds in accessing or modifying secret keys when the keys are stored outside the TOE or in distribution.

The attack can for example be achieved by eavesdropping or modifying the content of key files during network transfer, or by accessing and modifying the key files when they are stored outside the TOE (e.g. in a USB-Data Storage Device).

T.BADCERT – bad certificate

An attacker or a user uses invalid certificates (including certificates issued by untrusted CAs) to encrypt or sign files so that the content of the files are disclosed to unauthorized parties or the receiver accepts content from malicious sources.

3.2 Organizational Security Policies

The TOE and/or its operational environment shall comply with the following organizational security policies (OSPs) as security rules, procedures, practices or guidelines imposed by an organization upon its operation.

P.NONREPUDIATION

Non-repudiation of file origin shall be enforced when public key based file encryption is used.

P.PASSPHRASE

The passphrases used to protect key files shall contain at least 10 randomly selected words. The word list used for passphrase generation shall contain at least 7776 unique words.

P.ERASURE

Cryptographic keys shall be deleted when no longer needed by the TOE or upon the request of authorized user.

P.EMERGENCY

All keys contained in the key store shall be deleted in case of emergency.

P.ALGORITHM

Only CSEC recommended cryptographic algorithms and key lengths shall be used.

CSEC recommended algorithms and key lengths are given in CSEC Scheme Publication 188: Scheme Crypto Policy, and the latest version applies.

P.CRYPTO

The correct operation of crypto modules shall be verified. This covers both the crypto modules within the TOE and the external private key unit.

P.LOGGING

File protection operations performed by the TOE shall be logged.

P.KEYSTORE

The key store shall be confidentiality and integrity protected.

P.ANONYMITY

The TOE shall be able to protect the identity of the sender and receiver of a data file.

3.3 Assumptions

This section specifies the assumptions that must be satisfied by the TOE operational environment.

A.PLATFORM

The underlying operating system and hardware platform on which the TOE is installed work correctly and have no undocumented security critical side effects on the security functions of the TOE.

A.SINGLE

Access to the TOE is restricted to a single user. Access control is provided by the operating system or equivalent.

A.CATALOGUE

A trusted certificate catalogue service is available for the TOE to search and retrieve public key certificates and CRLs. The certificate catalogue service also provides a list of trusted CAs to the TOE. The connection between the TOE and the catalogue service is secure.

A.PKUNIT

A trusted private key unit is available that stores the user's private key securely and upon request correctly performs cryptographic operations using the private key. Before it can be used, the private key unit shall be unlocked by user entering a PIN or password. The communication between the private key unit and the TOE is also secure.

A.UPTODATE

The TOE user ensures that the certificates and CRLs in the cert store of the TOE are up-to-date.

A.PHYSICAL

The TOE is operated in a physically secure environment, i.e. no unauthorized persons have physical access to the TOE and its underlying system.

A.USER

The TOE user is trustworthy and trained to manage and perform encryption and decryption of sensitive information (including classified information) in accordance with any existing security policies and information classification policies. This includes that the user trusts only CAs (CA certificates) that are known to be trustworthy and distributes keys only to authorized parties. The user also unlocks the private key unit by entering the correct PIN or password.

A.PASSWORD

The password chosen by the TOE user for protecting the key store is of good quality and it is kept secret.

A.PASSPHRASE

The passphrase generated by the TOE for protecting a key file is distributed securely to authorized parties by out-of-band means.

A.CONNECT

The machine on which the TOE is running is not connected directly to an untrusted network.

A.TIME

The TOE is provided with a reliable time source.

A.ENTROPY

The TOE is provided with random numbers that have sufficient amount of entropy.

A.OCSPRES

OCSP responders, if available, provide correct certificate status to the TOE.

4. Security Objectives

The security objectives provide a concise statement of the intended response to the security problem.

4.1 Security Objectives for the TOE

O.CONFIDENTIALITY

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

O.INTEGRITY

The TOE shall provide mechanisms that detect if an attacker has tampered with a transmitted data file (i.e. replacing or modifying the content of the data file).

O.NONREPUDIATION

The TOE shall provide proof of file origin to enforce non-repudiation when public key based file encryption is used.

O.PASSPHRASE

The TOE shall support passphrases that contain at least 10 randomly selected words. The word list used for passphrase generation shall contain at least 7776 unique words.

O.KEY

The TOE shall provide mechanisms to protect secret keys so that they are not disclosed to or tampered with by unauthorized parties when stored outside the TOE or in distribution.

O.KEYSTORE

The TOE shall provide mechanisms to protect the key store so that its content is not disclosed to or tampered with by unauthorized parties.

O.VALIDATE

The TOE shall verify the validity of a certificate at the point of time it is to be used for a file protection operation.

O.ERASURE

The TOE shall delete cryptographic keys when they are no longer needed by the TOE or upon the request of authorized user.

O.EMERGENCY

All keys contained in the key store must be deleted in case of emergency.

O.ALGORITHM

The TOE shall only allow the use of CSEC recommended cryptographic algorithms and key lengths.

O.CRYPTO

The TOE shall verify the correct operation of crypto modules. This covers both the crypto modules within the TOE and the external private key unit.

O.LOGGING

The TOE shall log performed file protection operations.

O.ANONYMITY

The TOE shall provide mechanisms that protect the identity of the sender and receiver of a data file.

4.2 Security Objectives for the TOE Operational Environment

OE.PLATFORM

The operational environment must ensure that the underlying operating system and hardware platform on which the TOE is installed work correctly and that they have no undocumented security critical side effects on the security functions of the TOE.

OE.SINGLE

Access to the TOE is restricted to a singer user. Access control is provided by the operating system or equivalent.

OE.CATALOGUE

The operational environment must ensure that a trusted certificate catalogue service is available for the TOE to search and retrieve public key certificates and CRLs and that the certificate catalogue service provides a list of trusted CAs to the TOE. The operational environment must also ensure that the connection between the TOE and the catalogue service is secure.

OE.PKUNIT

The operational environment must ensure that a trusted private key unit is available that stores the user's private key securely and upon request correctly performs cryptographic operations using the private key. Before it can be used, the private key unit shall be unlocked by user entering a PIN or password. The operational environment must also ensure that the communication between the private key unit and the TOE is secure.

OE.UPTODATE

The TOE user must ensure that the certificates and CRLs in the cert store are up-to-date.

OE.PHYSICAL

The TOE is operated in a physically secure environment, i.e. no unauthorized persons have physical access to the TOE and its underlying system.

OE.USER

The TOE user is trustworthy and trained to manage and perform encryption and decryption of sensitive information (including classified information) in accordance with any existing security policies and information classification policies. This includes that the user trusts only CAs (CA certificates) that are known to be trustworthy and distributes keys only to authorized parties. The user also unlocks the private key unit by entering the correct PIN or password.

OE.PASSWORD

The password chosen by the TOE user for protecting the key store is of good quality and it is kept secret.

OE.PASSPHRASE

The passphrase generated by the TOE for protecting a key file is distributed securely to authorized parties by out-of-band means.

OE.CONNECT

The single user machine on which the TOE is running must not be connected directly to an untrusted network.

OE.TIME

The operational environment must provide a reliable time source to the TOE.

OE.ENTROPY

The operational environment must provide the TOE with random numbers that have sufficient amount of entropy.

OE.OCSPRES

The operational environment must ensure that OCSP responders, if available, provide correct certificate status to the TOE.

4.3 Security Objectives Rationale

4.3.1 Security objectives coverage

The following table provides a mapping of the security objectives for the TOE to threats and policies, showing that each objective counters or enforces at least one threat or policy, respectively.

Objective for the TOE	Threats / OSPs
O.CONFIDENTIALITY	T.DISCLOSE
O.INTEGRITY	T.TAMPER
O.NONREPUDIATION	P.NONREPUDIATION
O.PASSPHRASE	P.PASSPHRASE
O.KEY	T.KEY, T.DISCLOSE, T.TAMPER
O.KEYSTORE	P.KEYSTORE
O.VALIDATE	T.BADCERT
O.ERASURE	P.ERASURE
O.EMERGENCY	P.EMERGENCY
O.ALGORITHM	P.ALGORITHM
O.CRYPTO	P.CRYPTO
O.LOGGING	P.LOGGING
O.ANONYMITY	P.ANONYMITY

The following table provides a mapping of the security objectives for the operational environment to assumptions, threats and policies, showing that each objective holds, counters or enforces at least one assumption, threat or policy, respectively.

Objective	Assumptions / Threats /OSPs
OE.PLATFORM	A.PLATFORM
OE.SINGLE	A.SINGLE
OE.CATALOGUE	A.CATALOGUE, T.BADCERT
OE.PKUNIT	A.PKUNIT, T.TAMPER, T.KEY, P.NONREPUDIATION, P.CRYPTO
OE.UPTODATE	A.UPTODATE
OE.PHYSICAL	A.PHYSICAL
OE.USER	A.USER, T.TAMPER, T.KEY, T.BADCERT, P.NONREPUDIATION
OE.PASSWORD	A.PASSWORD
OE.PASSPHRASE	A.PASSPHRASE
OE.CONNECT	A.CONNECT
OE.TIME	A.TIME, P.LOGGING
OE.ENTROPY	A.ENTROPY, P.ALGORITHM
OE.OCSPRES	A.OCSPRES, T.BADCERT

4.3.2 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 actually contributes to the mitigation of that threat.

Threat	Rationale for the security objectives
T.DISCLOSE	O.CONFIDENTIALITY requires the TOE to provide mechanisms to protect the confidentiality of data file's content while transferring it over any unprotected communication channel. This diminishes T.DISCLOSE 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. O.KEY requires the TOE to provide mechanisms to protect secret keys when they are stored outside the TOE or during transmission. This diminishes T.DISCLOSE by reducing the likelihood of unauthorized users possessing the right key for data file decryption.
T.TAMPER	O.INTEGRITY requires the TOE to provide mechanisms to detect integrity violations of the file's content while transferring it over any unprotected communication channel. This diminishes T.TAMPER 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. O.KEY requires the TOE to provide mechanisms to protect secret keys when they are stored outside the TOE or during transmission. This diminishes T.TAMPER by reducing the likelihood of unauthorized users possessing the right key for data file modification. OE.PKUNIT requires the private key unit to store user's private key securely and perform signing operations within the unit. OE.USER requires the user to enter correct PIN/password in order to unlock the private key unit. They both diminish T.TAMPER as well.
T.KEY	O.KEY requires the TOE to provide mechanisms to protect secret keys when they are stored outside the TOE or in distribution. This diminishes T.KEY by reducing the likelihood of plaintext keys being disclosed to or tampered with by unauthorized parties. OE.PKUNIT requires the private key unit to store user's private key securely and perform signing and asymmetric decryption operations within the unit. OE.USER requires the user to enter correct PIN/password in order to unlock the private key unit. They both diminish T.KEY as well.

T.BADCERT	O.VALIDATE requires the TOE to verify the validity of a certificate at the point of time it is to be used for a file protection operation. This diminishes T.BADCERT by reducing the likelihood of invalid certificates being used. O.VALIDATE is supported by OE.CATALOGUE which ensures that a list of trusted CAs is provided to the TOE (as recommendation) through a secure connection and by OE.USER which ensures that the user only trusts CAs that are known to be trustworthy. OE.CATALOGUE also ensures that CRLs are provided to the TOE for certificate revocation checking. O.VALIDATE is also supported by OE.OCSPRES which ensures that OCSP responders, if available, provide correct certificate status to the TOE.

The following rationale provides justifiation that the security objective of the TOE is suitable to address each individual OSP and that each security objective tracing back to an OSP actually contributes in addressing the OSP.

OSP	**Rationale for the OSP**
P.NONREPUDIATION	This OSP is addressed by O.NONREPUDIATION which requires the TOE to provide proof of file origin when public key based file encryption is used. O.NONREPUDIATION is supported by OE.PKUNIT which requires the private key unit to store the user's private key securely and perform signing operations within the unit, and by OE.USER which requires the user to enter correct PIN/password in order to unlock the private key unit.
P.PASSPHRASE	This OSP is addressed by O.PASSPHRASE which ensures that the TOE supports passphrases containing at least 10 randomly selected words and that the word list used for passphrase generation contains at least 7776 unique words.
P.ERASURE	This OSP is addressed by O.ERASURE which ensures that cryptographic keys are deleted when they are no longer needed by the TOE or upon the request of authorized user.
P.EMERGENCY	This OSP is addressed by O.EMERGENCY which ensures that all keys in the key store are deleted in case of emergency.
P.ALGORITHM	This OSP is addressed by O.ALGORITHM which ensures that only CSEC recommended cryptographic algorithms and key lengths are used. O.ALGORITHM is supported by OE.ENTROPY which ensures that random numbers with sufficient entropy are provided to the TOE for key generation.
P.CRYPTO	This OSP is addressed by O.CRYPTO which ensures that the correct operation of crypto modules is verified. O.CRYPTO is supported by

	OE.PKUNIT which ensures the availability of the private key unit.
P.LOGGING	This OSP is addressed by O.LOGGING which ensures that file protection operations performed by the TOE are logged. O.LOGGING is supported by OE.TIME which provides a secure time stamp for logged events.
P.KEYSTORE	This OSP is addressed by O.KEYSTORE which requires the TOE to provide mechanisms to protect the content of the keystore.
P.ANONYMITY	This OSP is addressed by O.ANONYMITY which requires the TOE to provide mechanisms to protect the identity of the sender and receiver of a data file.

The following rationale provides justification that the security objectives of the TOE environment are suitable to address each individual assumption and that each security objective tracing back to an assumption actually contributes in addressing the assumption.

Assumption	**Rationale for the assumption**
A.PLATFORM	Addressed by OE.PLATFORM, which is identical to the assumption.
A.SINGLE	Addressed by OE.SINGLE, which is identical to the assumption.
A.CATALOGUE	Addressed by OE.CATALOGUE, which is identical to the assumption.
A.PKUNIT	Addressed by OE.PKUNIT, which is identical to the assumption.
A.UPTODATE	Addressed by OE.UPTODATE, which is identical to the assumption.
A.PHYSICAL	Addressed by OE.PHYSICAL, which is identical to the assumption.
A.USER	Addressed by OE.USER, which is identical to the assumption.
A.PASSWORD	Addressed by OE.PASSWORD, which is identical to the assumption.
A.PASSPHRASE	Addressed by OE.PASSPHRASE, which is identical to the assumption.
A.CONNECT	Addressed by OE.CONNECT, which is identical to the assumption.
A.TIME	Addressed by OE.TIME, which is identical to the assumption.
A.ENTROPY	Addressed by OE.ENTROPY, which is identical to the assumption.
A.OCSPRES	Addressed by OE.OCSPRES, which is identical to the assumption.

5. Extended Components Definition

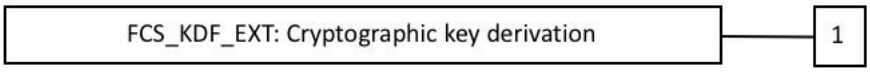
This PP defines three extended components: FCS_KDF_EXT.1, FDP_CRT_EXT.1, and FIA_X509_EXT.1.

5.1 Cryptographic key derivation (FCS_KDF_EXT)

Family behaviour

This family specifies the means by which a cryptographic key is derived from specified keying material. This is a new family defined for the FCS class.

Component levelling



FCS_KDF_EXT.1 Cryptographic key derivation, requires the TSF to derive cryptographic keys from keying material using the specified key derivation method.

Management: FCS_KDF_EXT.1

No specific management functions are identified.

Audit: FCS_KDF_EXT.1

There are no auditable events foreseen.

5.1.1 FCS_KDF_EXT.1 – Cryptographic key derivation

Hierarchical to: No other components.

Dependencies: No dependencies.

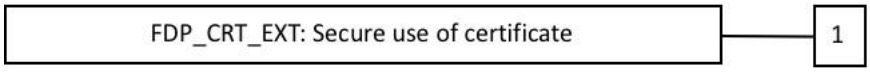
FCS_KDF_EXT.1.1 The TSF shall derive cryptographic keys from [assignment: *input parameters*] in accordance with a specified cryptographic key derivation algorithm [assignment: *key derivation algorithm*] and specified cryptographic key sizes [assignment: *key sizes*] that meet the following: [assignment: *list of standards*].

5.2 Secure use of certificate (FDP_CRT_EXT)

Family behaviour

This family defines the requirements for the TSF to be able to securely use public key certificates for file protection operations. This is a new family defined for the FDP class.

Component levelling



FDP_CRT_EXT.1 Timing of certificate validation, requires the TSF to validate certificates before allowing certain actions.

Management: FDP_CRT_EXT.1

No specific management functions are identified.

Audit: FDP_CRT_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a) Minimal: Unsuccessful certificate validations;

b) Basic: All certificate validations.

5.2.1 FDP_CRT_EXT.1 – Timing of certificate validation

Hierarchical to: No other components.

Dependencies: FIA_X509_EXT.1 X.509 certificate validation

FDP_CRT_EXT.1.1 The TSF shall verify the validity of a certificate before allowing the following actions:

- Use of certificate for [assignment: *purpose*]
- [assignment: *additional actions*]

5.3 X.509 certificate validation (FIA_X509_EXT)

Family behaviour

This family defines the requirements for validation of X.509 public key certificates. This is a new family defined for the FIA class.

Component levelling

FIA_X509_EXT.1 X.509 certificate validation, requires the TSF to check and validate certificates in accordance with the RFCs and rules specified in the component.

Management: FIA_X509_EXT.1

The following actions could be considered for the management functions of FMT:

a) Management of certificate importation.

b) Management of the list of trusted CAs.

Audit: FIA_X509_EXT.1

The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a) Minimal: No specific audit requirements are specified.

5.3.1 FIA_X509_EXT.1 – X.509 certificate validation

Hierarchical to: No other components.

Dependencies: No dependencies.

FIA_X509_EXT.1.1 The TSF shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted CA certificate.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates.
- The TSF shall validate the revocation status of the certificate using [selection: *the Online Certificate Status Protocol (OCSP) as specified in RFC 6960, a Certificate Revocation List (CRL) as specified in RFC 5280*].
- The TSF shall validate the extendedKeyUsage field according to the following rules: [assignment: *rules that govern contents of the extendedKeyUsage field that need to be verified*].

FIA_X509_EXT.1.2 The TSF shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

6. Security Requirements

6.1 Security Functional Requirements

The following table lists all the functional components that are relevant for this PP.

Component	Component name
FAU_GEN.1	Audit data generation
FCS_CKM.1	Cryptographic key generation
FCS_CKM.4	Cryptographic key destruction
FCS_COP.1	Cryptographic operation
FCS_KDF_EXT.1	Cryptographic key derivation
FDP_ACC.1	Subset access control
FDP_ACF.1	Security attribute based access control
FDP_CRT_EXT.1	Timing of certificate validation
FDP_ETC.2	Export of user data with security attributes
FDP_ITC.2	Import of user data with security attributes
FIA_SOS.2	TSF generation of secrets
FIA_X509_EXT.1	X.509 certificate validation
FMT_MSA.1	Management of security attributes
FMT_SMF.1	Specification of management functions
FPT_TEE.1	Testing of external entities
FPT_TST.1	TSF testing

The following convention is used for operations applied to the Security Functional Requirements: Assignment and Selection (indicated in **bold**), Iteration (indicated by appending a letter to the requirement, e.g. FDP_ACC.1a), refinements (indicated by **bold underscore** for additions and by ~~**bold strike through**~~ for deletions). The open operations are taken from Common Criteria Part 2 without any formatting.

6.1.1 Security functional policies implemented by the TOE

The TOE implements the following access control policies.

Key store access control SFP

This policy regulates the access to pre-shared keys contained in a password protected key store. The policy consists of two parts, one creating the key store, the other setting the stage for accessing the keys in the key store.

The SFP regulates that, when starting the TOE the first time, the user has to choose a password for the key store. This password is assigned to the key store and has to be entered each time the user wants to access the key store while starting the TOE.

Application note: This SFP applies only to password protected key store. For a key store that is encrypted with the user's public key, in order to decrypt the key store, the

user has to unlock the private key unit by entering the correct PIN/password, but that is enforced by the TOE environment (OE.USER).

Key file access control SFP

This policy regulates the access to pre-shared keys contained in a passphrase protected key file for PSK import/export. The policy consists of two parts, one the sender creating the key file for PSK export, the other setting the stage for the receiver accessing the keys in the key file for PSK import.

The SFP regulates that, when the sender wants to export pre-shared keys out of the TOE and a key file is created to contain those keys, the sender's TOE generates a passphrase and uses it to encrypt the key file. This passphrase is assigned to the key file and has to be entered by the receiver when the keys are to be imported into the receiver's TOE.

Application note: This SFP applies only to passphrase protected key file. For a key file that is encrypted with the receiver's public key, in order to decrypt the key file, the receiver has to unlock his/her private key unit by entering the correct PIN/password, but that is enforced by the TOE environment.

6.1.2 FAU_GEN.1 – Audit data generation

FAU_GEN.1.1 The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions;
b) All auditable events for the **not specified** level of audit; and
c) The following:
   - **File encryption/decryption**
   - **Generation of pre-shared key**
   - **Encryption/decryption failure**
   - **Integrity verification failure**
   - **Certificate validation failure**
   - **[assignment: *other specifically defined auditable events*]**

FAU_GEN.1.2 The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity (if applicable), **filename (if applicable), key identifier (if applicable),** and the outcome (success or failure) of the event; and
b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [assignment: *other audit relevant information*].

Application note: This SFR specifies the file protection operations and failures that should be logged. The ST author may add additional auditable events.

6.1.3 FCS_CKM.1a – Cryptographic key generation (FEK and KSEK)

FCS_CKM.1.1a The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [assignment: *cryptographic key generation algorithm*] and specified cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

Application note: This SFR applies to FEK and KSEK generation. The ST author is expected to choose algorithms and key sizes that are recommended by CSEC in [SP-188].

6.1.4 FCS_CKM.1b – Cryptographic key generation (PSK for encryption)

FCS_CKM.1.1b The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [assignment: *cryptographic key generation algorithm*] and specified cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

Application note: This SFR applies to the generation of encryption PSK for pre-shared key based file encryption. The ST author is expected to choose algorithms and key sizes that are recommended by CSEC in [SP-188].

6.1.5 FCS_CKM.1c – Cryptographic key generation (PSK for HMAC)

FCS_CKM.1.1c The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [assignment: *cryptographic key generation algorithm*] and specified cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

Application note: This SFR applies to the generation of HMAC PSK for pre-shared key based file encryption. The ST author is expected to choose algorithms and key sizes that are recommended by CSEC in [SP-188].

6.1.6 FCS_CKM.1d – Cryptographic key generation (PSK for anonymous file encryption)

FCS_CKM.1.1d The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [assignment: *cryptographic key generation algorithm*] and specified cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

Application note: This SFR applies to the generation of PSK for anonymous file encryption. The ST author is expected to choose algorithms and key sizes that are recommended by CSEC in [SP-188].

6.1.7 FCS_CKM.4a – Cryptographic key destruction (memory)

FCS_CKM.4.1a The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [assignment: *cryptographic key destruction method*] that meets the following: [assignment: *list of standards*].

Application note: This SFR applies to the deletion of all secret keys and keying material (FEKs, PSKs, KSEK, passwords, passphrases, and encryption and HMAC keys derived from passwords and passphrases) that temporarily exist in the memory.

6.1.8 FCS_CKM.4b – Cryptographic key destruction (storage)

FCS_CKM.4.1b The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [assignment: *cryptographic key destruction method*] that meets the following: [assignment: *list of standards*].

Application note: This SFR applies to the deletion of pre-shared keys from the key store and the deletion of the KSEK from the disk.

6.1.9 FCS_COP.1a – Cryptographic operation (symmetric encryption and decryption)

FCS_COP.1.1a The TSF shall perform **symmetric encryption and decryption** in accordance with a specified cryptographic algorithm [assignment: *cryptographic algorithm and mode*] and cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

Application note: This SFR covers symmetric encryption/decryption of the key store, key files, data files and sender/receiver identifying information. The ST author is expected to choose encryption algorithms, modes and key sizes that are recommended by CSEC in [SP-188].

6.1.10 FCS_COP.1b – Cryptographic operation (digital signature verification)

FCS_COP.1.1b The TSF shall perform **digital signature verification** in accordance with a specified cryptographic algorithm [assignment: *cryptographic algorithm*] and cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

Application note: With public key based file encryption the TOE verifies the sender's digital signature when receiving an encrypted file. If the key store is integrity protected with the TOE user' signature, the TOE verifies the signature before opening the key store. At PSK import, if the key file has been signed by the sender, the TOE also verifies the signature. The ST author is expected to choose digital signature algorithms and key sizes that are recommended by CSEC in [SP-188].

6.1.11 FCS_COP.1c – Cryptographic operation (asymmetric encryption)

FCS_COP.1.1c The TSF shall perform **asymmetric encryption** in accordance with a specified cryptographic algorithm [assignment: *cryptographic algorithm*] and cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

Application note: With public key based file encryption the TOE encrypts the FEK with the receiver's public key. The KSEK which is used to encrypt the key store is encrypted with the TOE user's public key. For PSK export, the key file can also be encrypted with the receiver's public key. The ST author is expected to choose asymmetric algorithms and key sizes that are recommended by CSEC in [SP-188].

6.1.12 FCS_COP.1d – Cryptographic operation (hash)

FCS_COP.1.1d The TSF shall perform **secure hash** in accordance with a specified cryptographic algorithm [assignment: *cryptographic algorithm*] ~~**and cryptographic key sizes [assignment: *cryptographic key sizes*]**~~ that meet the following: [assignment: *list of standards*].

Application note: This SFR applies to integrity protection and non-repudiation of data files with public key based file encryption. It also applies to integrity protection of the key store with the TOE user's signature. The TOE calculates the hash value and sends it to the private key unit for signing at the sender's side. At the receiver's side the TOE calculates the hash value to be checked against the signature. The ST author is expected to choose hash algorithms and hash lengths that are recommended by CSEC in [SP-188].

6.1.13 FCS_COP.1e – Cryptographic operation (keyed hash)

FCS_COP.1.1e The TSF shall perform **keyed hash** in accordance with a specified cryptographic algorithm [assignment: *cryptographic algorithm*] and cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

Application note: This SFR applies to integrity protection of data files with pre-shared key based file encryption. It also applies to integrity protection of key store/key file with password/passphrase. The ST author is expected to choose keyed hash algorithms and hash lengths that are recommended by CSEC in [SP-188].

6.1.14 FCS_COP.1f – Cryptographic operation (anonymous file encryption and decryption)

FCS_COP.1.1f The TSF shall perform **symmetric encryption and decryption** in accordance with a specified cryptographic algorithm [assignment: *cryptographic algorithm and mode*] and cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

Application note: This SFR covers the extra layer of symmetric encryption or decryption in anonymous file encryption. After a data file has been encrypted or signed using other FCS_COP.1 SFRs, the TOE encrypts the whole file, including the metadata. The ST author is expected to choose encryption algorithms, modes and key sizes that are recommended by CSEC in [SP-188].

6.1.15 FCS_KDF_EXT.1 – Cryptographic key derivation

FCS_KDF_EXT.1.1 The TSF shall derive cryptographic keys from **password, passphrase** in accordance with a specified cryptographic key derivation algorithm [assignment: *key derivation algorithm*] and specified cryptographic key sizes [assignment: *key sizes*] that meet the following: [assignment: *list of standards*].

Application note: This SFR covers the derivation of both encryption keys and HMAC keys from password/passphrase. The derived keys are used to protect the key store or key files.

6.1.16 FDP_ACC.1a – Subset access control (key store)

FDP_ACC.1.1a The TSF shall enforce the **Key store access control SFP** on

- **Subjects: user**
- **Objects: PSKs in the key store**
- **Operations:**
  - **creation of the key store**
  - **access to the PSKs in the key store**

6.1.17 FDP_ACF.1a – Security attribute based access control (key store)

FDP_ACF.1.1a The TSF shall enforce the **Key store access control SFP** to objects based on the following:

- **Subjects: user**
- **Objects: PSKs in the key store**
- **Security attributes:**
  - **password**
  - **[assignment: *list of additional security attributes*]**

FDP_ACF.1.2a 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 key store when creating the key store,**
- **The user must enter the correct password to access the PSKs in the key store,**
- **[assignment: *additional rules governing access among controlled subjects and controlled objects using controlled operations on controlled objects*].**

FDP_ACF.1.3a The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: [assignment: *rules, based on security attributes, that explicitly authorise access of subjects to objects*].

FDP_ACF.1.4a The TSF shall explicitly deny access of subjects to objects based on the following additional rules: [assignment: *rules, based on security attributes, that explicitly deny access of subjects to objects*].

Application note: This SFR addresses access control of the pre-shared keys contained in the key store. The user must enter the correct password in order to access the key store. The ST author that claims compliance to this PP may want to add additional rules based on some additional security attributes. In this case the ST author has to specify these rules and the additional security attributes used by these rules through an appropriate instantiation of the assignment operations.

6.1.18 FDP_ACC.1b – Subset access control (key file)

FDP_ACC.1.1b The TSF shall enforce the **Key file access control SFP** on

- **Subjects: user**
- **Objects: PSKs in a key file**
- **Operations:**

- **creation of the key file**
- **access to the PSKs in the key file**

6.1.19 FDP_ACF.1b – Security attribute based access control (key file)

FDP_ACF.1.1b The TSF shall enforce the **Key file access control SFP** to objects based on the following:

- **Subjects: user**
- **Objects: PSKs in a key file**
- **Security attributes:**
  - **passphrase**
  - **[assignment: *list of additional security attributes*]**

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

- **The user initiates the generation of passphrase when creating a key file,**
- **The user must enter the correct passphrase to access the PSKs in the key file.**
- **[assignment: *additional rules governing access among controlled subjects and controlled objects using controlled operations on controlled objects*].**

FDP_ACF.1.3b The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: [assignment: *rules, based on security attributes, that explicitly authorise access of subjects to objects*].

FDP_ACF.1.4b The TSF shall explicitly deny access of subjects to objects based on the following additional rules: [assignment: *rules, based on security attributes, that explicitly deny access of subjects to objects*].

Application note: This SFR addresses access control of the pre-shared keys (for encryption, HMAC or anonymous file encryption) contained in a key file for PSK import/export. The user must enter the correct passphrase in order to access the key file. The ST author that claims compliance to this PP may want to add additional rules based on some additional security attributes. In this case the ST author has to specify these rules and the additional security attributes used by these rules through an appropriate instantiation of the assignment operations.

6.1.20 FDP_CRT_EXT.1 – Timing of certificate validation

FDP_CRT_EXT.1.1 The TSF shall verify the validity of a certificate before allowing the following actions:

- Use of certificate for **file protection**
- [assignment: *additional actions*]

Application note: This SFR mandates certificate validation before a certificate is to be used for a file protection operation (encryption of FEK, verification of digital signature, encryption of key file or key store). Certificate validation is performed in accordance with FIA_X509_EXT.1. The ST author that claims compliance to this PP

may specify additional actions that require certificate validation, for example, import of certificate into the TOE.

6.1.21 FDP_ETC.2 – Export of user data with security attributes

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

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

FDP_ETC.2.3 The TSF shall ensure that the security attributes, when exported outside the TOE, 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 TOE: **the PSKs must be wrapped in an encrypted and integrity protected key file. Therefore**

- **the key file must be encrypted using [selection: *the encryption key derived from a passphrase as specified by FCS_KDF_EXT.1, the receiver's public key*].**
- **the key file must be integrity protected with a [selection: *keyed hash using the HMAC key derived from a passphrase as specified by FCS_KDF_EXT.1, digital signature*].**

Application note: This SFR applies to the exportation of PSKs (PSKs for encryption, HMAC or anonymous file encryption). Only passphrase or public key encrypted and integrity protected key files are exported out of the TOE. The integrity protection can be achieved through HMAC or digital signature. In the case of digital signature, the signature is created by the private key unit (part of the TOE environment) and the TOE only ensures that the key file is signed.

6.1.22 FDP_ITC.2 – Import of user data with security attributes

FDP_ITC.2.1 The TSF shall enforce the **Key file access control SFP** when importing user data, controlled under the SFP(s), from outside of the TOE.

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 TOE:

- **the integrity of the key file must be correctly verified using [selection: *the HMAC key derived from the assigned passphrase as specified by FCS_KDF_EXT.1, sender's public key*].**
- **the key file must be correctly decrypted using [selection: *the encryption key derived from the assigned passphrase as specified***

***by FCS_KDF_EXT.1, user's private key*].**

Application note: This SFR applies to the importation of PSKs (PSKs for encryption, HMAC or anonymous file encryption). Please note that, if the key file is encrypted with the receiver's public key, the decryption of the key file is performed by the private key unit (part of the TOE environment).

6.1.23 FIA_X509_EXT.1 – X.509 certificate validation

FIA_X509_EXT.1.1 The TSF shall validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted CA certificate.
- The TSF shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates.
- The TSF shall validate the revocation status of the certificate using [selection: *the Online Certificate Status Protocol (OCSP) as specified in RFC 6960, a Certificate Revocation List (CRL) as specified in RFC 5280*].
- The TSF shall validate the extendedKeyUsage field according to the following rules: [assignment: *rules that govern contents of the extendedKeyUsage field that need to be verified*].

FIA_X509_EXT.1.2 The TSF shall only treat a certificate as a CA certificate if the basicConstraints extension is present and the CA flag is set to TRUE.

Application note: This SFR lists the rules for certificate validation. The certificate path must end in a trusted root CA certificate or in a trusted intermediate CA certificate (e.g. certificate of a trusted CA in the user's own domain). The ST author may specify rules for validating the extendedKeyUsage field.

6.1.24 FIA_SOS.2 – TSF Generation of secrets

FIA_SOS.2.1 The TSF shall provide a mechanism to generate ~~**secrets**~~**passphrases** that meet

- **Minimum number of at least 10 randomly selected words**
- **Word list used for passphrase generation containing at least 7776 unique words**

FIA_SOS.2.2 The TSF shall be able to enforce the use of TSF generated ~~**secrets**~~**passphrases** for **export and import of pre-shared keys**.

Application note: This SFR specifies the required strength of passphrases that are used to protect the key files for exporting and importing PSKs.

6.1.25 FMT_MSA.1 – Management of security attributes

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

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

6.1.26 FMT_SMF.1 – Specification of management functions

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

- **Generate pre-shared key**
- **Delete individual pre-shared key**
- **Emergency erasure (delete all keys in the key store)**
- **Import certificate**
- **Import pre-shared key**
- **Export pre-shared key**
- **Change key store password**
- **Configure trusted CAs**
- **Enable or disable anonymous file encryption**
- **[assignment: *list of any additional management functions to be provided by the TSF*].**

Application note: In case a specific TOE is also providing additional management functionalities, the ST author has to instantiate the assignment operation to cover those functionalities.

6.1.27 FPT_TEE.1 – Testing of external entities

FPT_TEE.1.1 The TSF shall run a suite of tests [selection: *during initial start-up, periodically during normal operation, at the request of an authorized user, [assignment: other conditions]*] to check the fulfilment of

- **correct signing and decryption with private key**
- **[assignment: *any additional properties of the external entities*]**

FPT_TEE.1.2 If the test fails, the TSF shall

- **display error message and ask the user to react**
- **[assignment: *any additional action(s)*]**

Application note: The TSF shall test the external private key unit to ensure that it functions correctly.

6.1.28 FPT_TST.1 – TSF testing

FPT_TST.1.1 The TSF shall run a suite of self tests:

- **during initial start-up**
- **at the conditions [assignment: *conditions under which self test should occur*]**

to demonstrate the correct operation of

- **the crypto modules**
- **[assignment: *any additional parts of TSF*]**

FPT_TST.1.2 The TSF shall provide authorised users with the capability to verify the integrity of [selection: *[assignment: parts of TSF data], TSF data*].

FPT_TST.1.3 The TSF shall provide authorised users with the capability to verify the integrity of [selection*: [assignment: parts of TSF], TSF*].

Application note: The TSF shall perform self tests at start-up time to ensure that the crypto modules within the TOE that implement supported algorithms function correctly.

6.1.29 FCS_COP.1g – Cryptographic operation (digital signature generation)

FCS_COP.1.1g The TSF shall perform **digital signature generation** in accordance with a specified cryptographic algorithm [assignment: *cryptographic algorithm*] and cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

Application note: This function is performed by the operational environment. When requested by the TOE, the private key unit signs the data received from the TOE with the user's private key and returns the digital signature. The ST author is expected to choose digital signature algorithms and key sizes that are recommended by CSEC in [SP-188].

6.1.30 FCS_COP.1h – Cryptographic operation (asymmetric decryption)

FCS_COP.1.1h The TSF shall perform **asymmetric decryption** in accordance with a specified cryptographic algorithm [assignment: *cryptographic algorithm*] and cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].

Application note: This function is performed by the operational environment. When requested by the TOE, the private key unit decrypts the data received from the TOE with the user's private key and returns the decrypted data. The ST author is expected to choose asymmetric algorithms and key sizes that are recommended by CSEC in [SP-188].

6.2 Security Functional Requirements Rationale

6.2.1 Coverage

The following table provides a mapping of SFR to the security objectives, showing that each security functional requirement addresses at least one security objective.

SFR	Security objectives
FAU_GEN.1	O.LOGGING
FCS_CKM.1a (FEK and KSEK)	O.CONFIDENTIALITY, O.ALGORITHM, O.KEYSTORE
FCS_CKM.1b (PSK for encryption)	O.CONFIDENTIALITY, O.ALGORITHM
FCS_CKM.1c (PSK for HMAC)	O.INTEGRITY, O.ALGORITHM

FCS_CKM.1d (PSK for anonymous file encryption)	O.ANONYMITY, O.ALGORITHM
FCS_CKM.4a (memory)	O.ERASURE
FCS_CKM.4b (storage)	O.ERASURE, O.EMERGENCY
FCS_COP.1a (symmetric encryption and decryption)	O.CONFIDENTIALITY, O.ALGORITHM, O.KEY, O.KEYSTORE
FCS_COP.1b (digital signature verification)	O.INTEGRITY, O.NONREPUDIATION, O.ALGORITHM, O.KEY, O.KEYSTORE
FCS_COP.1c (asymmetric encryption)	O.KEY, O.KEYSTORE, O.ALGORITHM
FCS_COP.1d (hash)	O.INTEGRITY, O.NONREPUDIATION, O.ALGORITHM, O.KEY, O.KEYSTORE
FCS_COP.1e (keyed hash)	O.INTEGRITY, O.ALGORITHM, O.KEY, O.KEYSTORE
FCS_COP.1f (anonymous file encryption and decryption)	O.ANONYMITY, O.ALGORITHM
FCS_KDF_EXT.1 (cryptographic key derivation)	O.KEY, O.KEYSTORE
FDP_ACC.1a (key store)	O.KEYSTORE, O.ERASURE
FDP_ACF.1a (key store)	O.KEYSTORE, O.ERASURE
FDP_ACC.1b (key file)	O.KEY
FDP_ACF.1b (key file)	O.KEY
FDP_CRT_EXT.1	O.VALIDATE
FDP_ETC.2	O.KEY
FDP_ITC.2	O.KEY
FIA_SOS.2	O.PASSPHRASE
FIA_X509_EXT.1	O.VALIDATE
FMT_MSA.1	O.KEYSTORE
FMT_SMF.1	O.KEY, O.KEYSTORE, O.VALIDATE, O.ERASURE, O.EMERGENCY, O.CONFIDENTIALITY, O.INTEGRITY, O.ANONYMITY
FPT_TEE.1	O.CRYPTO
FPT_TST.1	O.CRYPTO
FCS_COP.1g	O.INTEGRITY, O.NONREPUDIATION, O.KEY, O.KEYSTORE, O.ALGORITHM
FCS_COP.1h	O.KEY, O.KEYSTORE, O.ALGORITHM

6.2.2 Sufficiency

The following rationale provides justification for each security objective for the TOE, showing that the security functional requirements are suitable to meet and achieve the security objectives.

Security Objective	Rationale
O.CONFIDENTIALITY	The objective: • The TOE shall provide mechanisms that protect the information of a transmitted data file such that its content is confidentiality-protected and only accessible for authorized users. Is met by: • FCS_COP.1a which provides symmetric encryption for data files. • FCS_CKM.1a and FCS_CKM.1b which specify the generation of keys for data file encryption and decryption. • FMT_SMF.1 which provides the specific management functions for generation of pre-shared keys, import and export of pre-shared keys, and import of certificates.
O.INTEGRITY	The objective: • The TOE shall provide mechanisms that detect if an attacker has tampered with a transmitted data file (i.e. replacing or modifying the content of the data file). Is met by: • FCS_COP.1b which specifies the verification of digital signature to detect integrity violation on data files. • FCS_COP.1d which specifies the hash function for signature generation and verification. • FCS_COP.1e which specifies HMAC for integrity protection of data files. • FCS_CKM.1c which specifies the generation of HMAC keys for integrity protection of data files. • FMT_SMF.1 which provides the specific management functions for generation of pre-shared keys, import and export of pre-shared keys, and import of certificates. • FCS_COP.1g which specifies the signing operation performed by the private key unit (part of the TOE environment).
O.NONREPUDIATION	The objective: • The TOE shall provide proof of file origin to enforce non-repudiation when public key based file encryption is used. Is met by: • FCS_COP.1b which specifies the verification of digital signature to determine file origin. • FCS_COP.1d which specifies the generation of file hash value. The hash value is then encrypted with the sender's private key to create digital signature as proof of origin. Note however the signing operation itself is performed by the private key unit (part of TOE environment) that holds the sender's private key.

	• FCS_COP.1g which specifies the signing operation performed by the private key unit (part of the TOE environment).
O.PASSPHRASE	The objective: • The TOE shall support passphrases that contain at least 10 randomly selected words. The word list used for passphrase generation shall contain at least 7776 unique words. is met by: • FIA_SOS.2 which specifies the generation of passphrases containing at least 10 randomly selected words using a word list that contains at least 7776 unique words.
O.KEY	The objective: • The TOE shall provide mechanisms to protect secret keys so that they are not disclosed to or tampered with by unauthorized parties when stored outside the TOE or in distribution. is met by: • FCS_COP.1a which provides symmetric encryption key files. • FCS_COP.1b which specifies the verification of digital signature on key files. • FCS_COP.1c which provides asymmetric encryption for FEKs and key files. • FCS_COP.1d which specifies the hash function for key file signature generation and verification. • FCS_COP.1e which specifies HMAC for integrity protection of key files. • FCS_KDF_EXT.1 which specifies the derivation of keys from passphrase for key file encryption and decryption. • FDP_ETC.2 and FDP_ITC.2 which specify the export and import of pre-shared keys via protected key file. • FDP_ACC.1b and FDP_ACF.1b which ensure that only users who know the right passphrase can access the pre-shared keys within the key file. • FMT_SMF.1 which provides the specific management functions for pre-shared key import/export. • FCS_COP.1g which specifies the signing operation performed by the private key unit (part of the TOE environment) to protect the integrity of key files. • FCS_COP.1h which specifies the asymmetric operation performed by the private key unit (part of the TOE environment) to decrypt key files.
O.KEYSTORE	The objective: • The TOE shall provide mechanisms to protect the key store so that its content is not disclosed to or tampered with by unauthorized parties. is met by:

	• FCS_COP.1a which provides symmetric encryption for the key store. • FCS_COP.1b which specifies the verification of digital signature on the key store. • FCS_COP.1c which provides asymmetric encryption for the KSEK. • FCS_COP.1d which specifies the hash function for key store signature generation and verification. • FCS_COP.1e which specifies HMAC for integrity protection of the key store. • FCS_CKM.1a which specifies the generation of KSEK for key store encryption and decryption. • FCS_KDF_EXT.1 which specifies the derivation of keys from password for key store encryption and decryption. • FDP_ACC.1a and FDP_ACF.1a which ensure that only users who know the right password can access the pre-shared keys within the key store. • FMT_MSA.1 which ensures that only users who know the password for the key store could change it. • FMT_SMF.1 which provides the specific management functions for certificate import, change of key store password and configuration of trusted CAs. • FCS_COP.1g which specifies the signing operation performed by the private key unit (part of the TOE environment) to protect the integrity of the key store. • FCS_COP.1h which specifies the asymmetric operation performed by the private key unit (part of the TOE environment) to decrypt the KSEK.
O.VALIDATE	The objective: • The TOE shall verify the validity of a certificate at the point of time it is to be used for a file protection operation. is met by: • FDP_CRT_EXT.1 which ensures that certificates are validated before they are to be used for file protection operations. • FIA_X509_EXT.1 which specifies the rules for validating certificates. • FMT_SMF.1 which provides the specific management functions for configuring trusted CAs.
O.ERASURE	The objective: • The TOE shall delete cryptographic keys when they are no longer needed by the TOE or upon the request of authorized user. is met by: • FCS_CKM.4a which ensures that keys and keying material that temporarily exist in the memory are erased when no longer needed.

	• FCS_CKM.4b which ensures that pre-shared keys in the key store and the KSEK are deleted upon user request. • FDP_ACC.1a and FDP_ACF.1a which ensure that keys are only accessible to those users possessing the right password for the key store. • FMT_SMF.1 which provides the specific management functions for key erasure.
O.EMERGENCY	The objective: • All keys contained in the key store must be deleted in case of emergency. is met by: • FCS_CKM.4b which ensures that the keys in the key store are deleted. • FMT_SMF.1 which provides the specific management function for emergency erasure.
O.ALGORITHM	The objective: • The TOE shall only allow the use of CSEC recommended cryptographic algorithms and key lengths. is met by: • FCS_CKM.1a, FCS_CKM.1b, FCS_CKM.1c, FCS_CKM.1d, FCS_COP.1a, FCS_COP.1b, FCS_COP.1c, FCS_COP.1d, FCS_COP.1e, FCS_COP.1f, FCS_COP.1g, and FCS_COP.1h which ensure that only CSEC recommended algorithms and key lengths are used.
O.CRYPTO	The objective: • The TOE shall verify the correct operation of crypto modules. This covers both the crypto modules within the TOE and the external private key unit. is met by: • FPT_TEE.1 which ensures that the external private key unit (e.g. smart card) is tested for correct operation. • FPT_TST.1 which ensures that the crypto modules within the TOE are tested at start-up time.
O.LOGGING	The objective: • The TOE shall log performed file protection operations. is met by: • FAU_GEN.1 which specifies the logging function.
O.ANONYMITY	The objective: • The TOE shall provide mechanisms that protect the identity of the sender and receiver of a data file. Is met by: • FCS_COP.1f which provides an extra layer of symmetric encryption for encrypted or signed data files.

	• FCS_CKM.1d which specifies the generation of keys for anonymous file encryption. • FMT_SMF.1 which provides the specific management functions for generation of pre-shared keys, import and export of pre-shared keys, and enablement/disablement of anonymous file encryption.

6.2.3 Dependency analysis between security functional components

The following table demonstrates the dependencies of SFRs modelled in CC Part 2 and how the SFRs for the TOE resolve those dependencies.

SFR	Dependencies	Resolved?
FAU_GEN.1	FPT_STM.1	No, satisfied by OE.TIME instead
FCS_CKM.1a (FEK and KSEK)	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	Yes, by FCS_COP.1a Yes, FEK destruction is satisfied by FCS_CKM.4a. KSEK destruction is satisfied by FCS_CKM.4a and FCS_CKM.4b.
FCS_CKM.1b (PSK for encryption)	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	Yes, by FCS_COP.1a Yes, by FCS_CKM.4a,b
FCS_CKM.1c (PSK for HMAC)	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	Yes, by FCS_COP.1e Yes, by FCS_CKM.4a,b
FCS_CKM.1d (PSK for anonymous file encryption)	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	Yes, by FCS_COP.1f Yes, by FCS_CKM.4a,b
FCS_CKM.4a (memory)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	Yes, by FCS_CKM.1a,b,c and FCS_KDF_EXT.1 except for passwords (chosen by the user) and passphrases (generated by the TSF, FIA_SOS.2).
FCS_CKM.4b (storage)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	Yes, by FCS_CKM.1b,c
FCS_COP.1a (symmetric encryption and decryption)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1a,b and FCS_KDF_EXT.1 Yes, by FCS_CKM.4a,b
FCS_COP.1b (digital signature verification)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No, satisfied by FMT_SMF.1 and FDP_CRT_EXT.1 instead. The reason is that public key is used for signature verification. Public key is not generated by the TOE. It is imported in

		certificate from outside of the TOE, but not restricted by any access control or information flow control SFP.
	FCS_CKM.4	No, no key destruction is needed since public key does not contain any secret information
FCS_COP.1c (asymmetric encryption)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No, satisfied by FMT_SMF.1 and FDP_CRT_EXT.1 instead. The reason is that public key is used for asymmetric encryption. Public key is not generated by the TOE. It is imported in certificate from outside of the TOE, but not restricted by any access control or information flow control SFP.
	FCS_CKM.4	No, no key destruction is needed since public key does not contain any secret information
FCS_COP.1d (hash)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	No, since no key is needed for the hash operation No, no key destruction is needed since there is no key associated with the hash operation
FCS_COP.1e (keyed hash)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1c and FCS_KDF_EXT.1 Yes, by FCS_CKM.4a,b
FCS_COP.1f (anonymous file encryption and decryption)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, by FCS_CKM.1d Yes, by FCS_CKM.4a,b
FCS_KDF_EXT.1	-	-
FDP_ACC.1a (key store)	FDP_ACF.1	Yes, by FDP_ACF.1a
FDP_ACF.1a (key store)	FDP_ACC.1, FMT_MSA.3	Yes, by FDP_ACC.1a No, since there is no default value for the key store password.
FDP_ACC.1b (key file)	FDP_ACF.1	Yes, by FDP_ACF.1b
FDP_ACF.1b (key file)	FDP_ACC.1, FMT_MSA.3	Yes, by FDP_ACC.1b No, the passphrase is generated by the TOE each time a key file is to be exported

		and therefore there is no need of initialization.
FDP_CRT_EXT.1	FIA_X509_EXT.1	Yes, by FIA_X509_EXT.1
FDP_ETC.2	[FDP_ACC.1 or FDP_IFC.1]	Yes, by FDP_ACC.1b
FDP_ITC.2	[FDP_ACC.1 or FDP_IFC.1] [FTP_ITC.1 or FTP_TRP.1] FPT_TDC.1	Yes, by FDP_ACC.1b No, confidentiality and integrity of imported keys are instead satisfied by FCS_COP.1a and FCS_COP.1e No, consistency is not an issue since the key file is both generated and interpreted by the TOE.
FIA_SOS.2	-	-
FIA_X509_EXT.1	-	-
FMT_MSA.1	[FDP_ACC.1 or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	Yes, by FDP_ACC.1a No, the TOE is not aware of security roles; it relies on the TOE environment for user access control (OE.SINGLE) Yes, by FMT_SMF.1
FMT_SMF.1	-	-
FPT_TEE.1	-	-
FPT_TST.1	-	-
FCS_COP.1g (digital signature generation)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	No, dependency is not fulfilled because this SFR is enforced by the operational environment.
FCS_COP.1h (asymmetric decryption)	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	No, dependency is not fulfilled because this SFR is enforced by the operational environment.

6.3 Security Assurance Requirements

The security assurance requirements of this Protection Profile are those defined in CC part 3 for the assurance level EAL3 augmented with ALC_FLR.2.

Assurance class	**Assurance components**
ADV: Development	ADV_ARC.1 Security architecture description
	ADV_FSP.3 Functional specification with complete summary
	ADV_TDS.2 Architectural design

AGD: Guidance documents	AGD_OPE.1 Operational user guidance
	AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.3 Authorization controls
	ALC_CMS.3 Implementation representation CM coverage
	ALC_DEL.1 Delivery procedures
	ALC_DVS.1 Identification of security measures
	ALC_FLR.2 Flaw reporting procedures
	ALC_LCD.1 Developer defined life-cycle model
ASE: Security Target evaluation	ASE_CCL.1 Conformance claims
	ASE_ECD.1 Extended components definition
	ASE_INT.1 ST introduction
	ASE_OBJ.2 Security objectives
	ASE_REQ.2 Derived security requirements
	ASE_SPD.1 Security problem definition
	ASE_TSS.1 TOE summary specification
ATE: Tests	ATE_COV.2 Analysis of coverage
	ATE_DPT.1 Testing: basic design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
AVA: Vulnerability assessment	AVA_VAN.2 Vulnerability analysis

6.4 Security Assurance Requirements Rationale

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.

EAL 3 is augmented with ALC_FLR.2 to ensure that instructions and procedures for the reporting and remediation of identified security flaws are in place.

7. Abbreviations

CA	Certification Authority
CC	Common Criteria
COTS	Commercial Off-The-Shelf
CRL	Certificate Revocation List
CSEC	Swedish Certification Body for IT Security
EAL	Evaluation Assurance Level
FEK	File Encryption Key
KSEK	Key Store Encryption Key
OSP	Organizational Security Policy
PP	Protection Profile
PSK	Pre-Shared Key
RFC	Request for Comment
SAR	Security Assurance Requirement
SFP	Security Function Policy
SFR	Security Functional Requirement
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality

8. References

[CC] Common Criteria for Information Technology Security Evaluation. Part 1: Introduction and general model, April 2017, Version 3.1 Revision 5, CCMB-2017-04-001; Part 2: Security functional components, April 2017, Version 3.1 Revision 5, CCMB-2017-04-002; Part 3: Security assurance components, April 2017, Version 3.1 Revision 5, CCMB-2017-04-003.

[CEM] Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, April 2017, Version 3.1 Revision 5, CCMB-2017-04-004.

[cPPND] Collaborative Protection Profile for Network Devices, Version 1.0, 27-Feb-2015.

[ISO15446] Technical Report ISO/IEC TR 15446, Information technology – Security techniques – Guide for the production of Protection Profiles and Security Targets, Third edition 2017-10.

[RFC 5280] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280, May 2008.

[RFC 6960] Internet X.509 Public Key Infrastructure Online Certificate Status Protocol – OCSP, RFC 6960, June 2013.

[SP-188] CSEC (Swedish Certification Body for IT Security) 188 Scheme Crypto Policy, the latest version applies.