IAIK – Security Target

Version 1.8

IAIK-JCE CC Core 3.15

16 March 2007

Table of Contents:

List of Tables

List of Figures

1 ST Introduction

1.1 ST Identification

Title:	IAIK-JCE CC Core Security Target
Version:	1.8
Date:	16 March 2007
Authors:	SIC Stiftung secure information and communication technologies.[1] IAIK Institute for applied information processing and communications – Graz university of technology.
TOE:	IAIK-JCE CC Core
TOE version:	3.15
Assurance level:	EAL 3 The **TOE** meets the assurance requirements of assurance level EAL 3.
Strength of functions:	The **TOE**s strength of functions is rated high (SOF high).
CC Identification:	Common Criteria for Information Technology Security Evaluation Version 2.3, August 2005 [CC]
Evaluation Body:	TÜV Informationstechnik GmbH Langemarckstraße 20 45141 Essen, Germany
TOE documentation:	HTML Documentation - IAIK-JCE 3.15 with IAIK-JCE CC Core 3.15: Readme.html, File Revision 30 and linked documents IAIK – Guidance Document, Integrity Verification Guidance Version 1.7

1.2 ST Overview

The IAIK-JCE CC Core is a set of APIs and implementations of cryptographic functionality.
Including:

- hash functions
- signature schemes
- block ciphers
- stream ciphers
- asymmetric ciphers
- message authentication codes
- random number generators

[1] The IAIK has established the "Stiftung Secure Information and Communication Technologies" (SIC). Stiftung SIC is a non-profit organisation which was established as a foundation for public utility, aiming at encouraging independent scientific research, development as well as teaching and knowledge transfer in the fields of applied information processing, communication and information security. On December 15, 2003 all rights regarding our crypto toolkits were transferred from IAIK to Stiftung SIC.

It supplements the security functionality of the default Java™ Runtime Environment. The IAIK-JCE CC Core is delivered to the customer as part of the IAIK-JCE toolkit, which extends the CC Core by additional algorithms, features and protocols.

1.3 CC Conformance

The **TOE** is CC part 2 [CC2] extended (by FCS_RND.1) and CC Part 3 [CC3] conformant. The Evaluation Assurance Level is EAL3.
This ST does not claim conformance with any Protection Profile.

2 TOE Description

2.1 Product type

The **TOE** is software written solely in the Java™ Programming Language and delivered to users as part of a toolkit. This toolkit consists of a Java™ library in form of JAR file, documentation and demo code. The **TOE** provides components usable to develop applications including functionality to create and verify digital signatures as well as encrypting and decrypting data.
The **TOE** is conformant to the Java™ Cryptographic Architecture (JCA) and Java™ Cryptographic Extensions (JCE) and implements a Cryptographic Service Provider as defined there. Applications access the cryptographic functionality of this provider through the JCA and JCE framework.

2.2 TOE structure

This section explains the structure of the **TOE**, its relationship and boundary to other components. Figure 1 shows a Java™ Virtual Machine VM running an application that uses the **TOE**s cryptographic algorithms through the JCA/JCE framework.

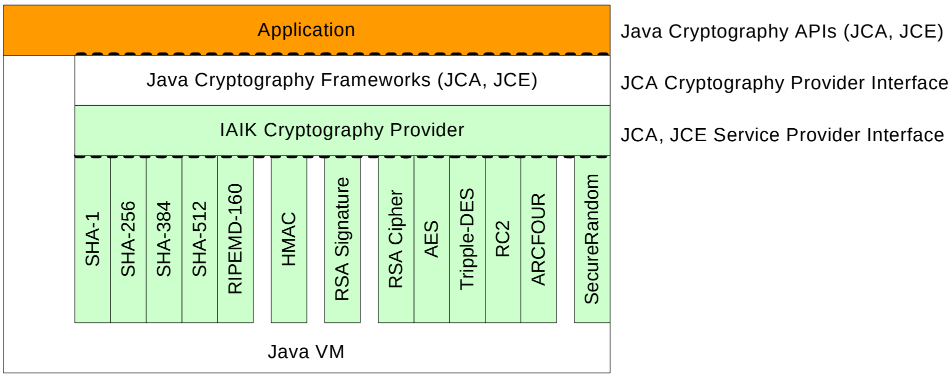

Figure 1: The TOE and its environment

The **TOE** implements a Java™ Cryptographic Service Provider (used interchangeably with "provider" in this document) as defined in the JCA and JCE specification by SUN Microsystems. This provider implementation is called IAIK provider. The IAIK provider can be registered in the JCA framework. Thereafter, applications can access the cryptographic algorithms of the IAIK provider. For each cryptographic primitive the JCA and JCE provide a separate service provider interface (SPI), which is an abstract class. Each concrete implementation of a cryptographic algorithm must implement the SPI and thus derive the abstract class. For instance, the class of the **TOE** which contains the actual SHA-1 implementation extends the abstract class `MessageDigestSPI`. The **TOE** implements hash algorithms (also called message digest in the JCA context), signature algorithms (includes signature generation and verification), MACs, pseudo random generators and ciphers (includes block ciphers as well as stream ciphers and asymmetric ciphers).
The application can request an implementation of a certain algorithm from the JCA framework using static methods in framework classes. For example, to get an implementation of the SHA-1 hash algorithm of the IAIK provider, the application

calls `MessageDigest.getInstance("SHA-1", "IAIK")`. The names of the algorithms, like “`SHA-1`”, are defined in the developers manual. The name of the provider is fixed to `IAIK` for the IAIK provider. The result is a `MessageDigest` object, which contains the SHA-1 implementation of the IAIK provider. The class `MessageDigest` of the JCA framework provides a common interface to all hash algorithms. For signature and cipher implementations, the workflow is similar. For a more detailed description of the JCA/JCE framework, please refer to [CRYPTO SPEC].
The **TOE** (IAIK-JCE CC Core) is delivered to the customer as part of the IAIK-JCE toolkit. This toolkit adds more algorithms, features and protocols to the **TOE** functionality.

2.3 General TOE functionality

The **TOE** provides cryptographic hash, message authentication code (MAC), digital signature and encryption related functionality, as well as deterministic random number generators DRNG.

2.3.1 Hash related functionality

The **TOE** provides implementations of algorithms used to calculate hash functions. There are several uses cases, when it is necessary to calculate a cryptographic hash function only, for instance, when using dedicated cryptographic hardware, like smart cards, to create digital signatures. Some of these hardware modules are not capable of computing the hash itself and therefore need the **TOE** to perform this task. Furthermore, the computation of a hash function will be used whenever the creation of the signature is a multistep process, where hashes of data to be signed are incorporated into a new structure (like CMS or XML-Dsig).
The **TOE** implements the following hash algorithms:

- SHA-1 [FIPS PUB 180-1]
- Ripemd-160 [ISO/IEC 10118-3]
- SHA-256 [FIPS PUB 180-2]
- SHA-384 [FIPS PUB 180-2]
- SHA-512 [FIPS PUB 180-2]

2.3.2 MAC related functionality

To compute a message authentication code the **TOE** uses the HMAC algorithm as defined in [RFC 2104]. This HMAC uses the following cryptographic hash functions:

- SHA-1 [FIPS PUB 180-1]
- Ripemd-160 [ISO/IEC 10118-3]
- SHA-256 [FIPS PUB 180-2]
- SHA-384 [FIPS PUB 180-2]
- SHA-512 [FIPS PUB 180-2]

as described in the previous chapter. The application must provide the secret key of size (128 + k * 8) bit ≤ blocksize of the used hash function, with [k=0,1,2,...]. Smaller key sizes are supported as well, but they are not suitable for use in an environment that requires a high strength of functions.

2.3.3 Digital Signature related functionality

The **TOE** provides implementations of algorithms used to generate and verify digital signatures. Specifically, the **TOE** provides implementations of hash functions, asymmetric encryption algorithms and padding schemes and implements specific signature schemes. The included hash functions are the same as those listed in the previous section about hash functionality.
The **TOE** implements signature generation and verification according to the following digital signature schemes:

- RSA with SHA-1, SHA-256, SHA-384, SHA-512 or RIPEMD-160 according to [PKCS#1v1.5], with key lengths of 1024 + k * 64 [k=0,1,2,...] bit. The maximum key size is 8192 bit.
- RSA-PSS with SHA-1, SHA-256, SHA-384, SHA-512 or RIPEMD-160 according to [PKCS#1v2.1], with key lengths of 1024 + k * 64 [k=0,1,2,...] bit. The maximum key size is 8192 bit.

The **TOE** is designed to meet the requirements of an application for the generation and the verification of electronic signatures as defined in the legislation of the European Union [EU_directive].
Most of the signature algorithms support smaller key sizes as well, but they are not suitable for use in an environment that requires a high strength of functions.

2.3.4 Encryption functionality

The **TOE** implements several algorithms that can be used for data encryption and decryption. Key management is out of scope of the **TOE**. The application provides the keys to the **TOE**. The **TOE** does not modify the keys it gets from the application. The access protection of the memory where the keys reside must be provided by the environment, especially the Java™ VM.
The **TOE** implements the following block ciphers:

- AES 128, 192, 256 bit [FIPS PUB 197]
- Triple-DES 112, 168 bit [FIPS 46-3]
- RC2 128-1024 bit [RFC 2268]

Each of these block ciphers can be used with the following modes of operation:

- ECB
- CBC
- OFB
- CFB

In addition, AES supports the CTR mode.
The **TOE** implements the following stream ciphers:

- ARCFOUR 128 – 2048 bit according to [IETF-Draft-Kaukonen]. This algorithm is assumed to be compatible with RC4™ from RSA Security Inc..

The **TOE** implements the following asymmetric ciphers:

- RSA 1024 + k * 64 [k=0,1,2,...] bit according to [PKCS#1v1.5]. The maximum key size is 8192 bit.
- RSA-OAEP 1024 + k * 64 [k=0,1,2,...] bit according to [PKCS#1v2.1]. The maximum key size is 8192 bit.

Most of the encryption algorithms support smaller key sizes as well, but they are not suitable for use in an environment that requires a high strength of functions.

2.3.5 Random Number Generator related functionality

The **TOE** contains two random number generators based on one of the following hash functions: SHA-1 [FIPS PUB 180-1], SHA-256 [FIPS PUB 180-2], SHA-384 [FIPS PUB 180-2], SHA-512 [FIPS PUB 180-2] or RIPEMD-160 [ISO/IEC 10118-3]. The random number generator must be initialized with a random seed with adequate entropy.

2.3.6 TOE Boundary

In principle, the **TOE** has two boundaries. The first is the interaction with the Java™ VM and its Java™ Runtime Environment JRE and JCE classes. The **TOE** assumes that these operate compliant with the Java™ Language Specification 2.0 and the Java™ Virtual Machine Specification, Second Edition.
The second boundary is between the **TOE** and the application. It is worth to note that this is not a direct boundary. The application only accesses classes of the JCA and JCE framework directly, and these classes forward requests to the **TOE**. The JCA and JCE classes are part of the environment and the **TOE** assumes that they operate according to the JCA Specification of Java™ 1.1 [JCA1.1-REF] (or later) and the JCE Specification 1.2 [JCE1.2-REF] (or later). The **TOE** does not have any direct interfaces to any component other than the application, the JCA and JCE classes or the JRE classes (like e.g. the operating system or other applications). The **TOE** does also not initiate any I/O operations like file access or network connections.

The **TOE** is able to support the generation of digital signatures. From the application's point of view, two use-cases can be identified:

- Electronic Signatures with Signature Creation Device.
  The **TOE** is used together with a signature creation device (SCD) to create electronic signatures. In this case, the **TOE** is used only to calculate the hash of the data to be signed (and only if the SCD is unable to do so by itself). The **TOE** calculates the hash and returns it to the application. The application can pass this hash value to the SCD to process the private key operation. It may access such cryptographic hardware e.g. via the PKCS#11 API. Prompting a PIN or pass phrase for access to the private key, will usually be done with a smart card reader or HSM which has its own key pad for entering this authentication data. Displaying data to be signed or verified is out of scope of the **TOE**.
- Conventional Signatures.
  The **TOE** is used without hardware support to create electronic signatures. In this case, all calculations required to create the signature are done within the **TOE**. In specific, the Java™ VM (with its JRE classes) executes the code of the **TOE,** which implements all required cryptographic algorithms.

Furthermore, the **TOE** has functionality that is not part of the evaluation:

- The **TOE** supports more key sizes than the minimum and maximum key size, which are described in chapters 2.3.2, 2.3.3 and 2.3.4. The maximum key size depends on the system resources only. Moreover, the **TOE** can also work with odd key sizes, e.g. 1025 bit RSA keys.
- The **TOE** supports CTR as mode of operation for Triple DES and RC2.

In addition, the IAIK-JCE toolkit, which contains the **TOE,** offers more functionality that is not part of the **TOE**:

- Additional cipher algorithms
- Additional modes of operation for block ciphers
- Additional padding schemes
- Additional signature schemes
- Additional hash functions
- Additional random generators
- Additional MAC algorithms
- Key generation
- Key-Pair generation
- Key agreement algorithms
- Algorithm parameter generation
- Key storage
- ASN.1 support
- Suopport for additional PKCS standards
- X.509 support for certificates, CRLs and OCSP

2.3.7 TOE Environment

The application, the JRE classes, the JCA and JCE framework and the Java™ VM constitute the environment of the **TOE**. The **TOE** is written in the Java™ Programming Language only and runs in the same instance of the Java™ VM as the environment. All components communicate by method calls executed by the Java™ VM. No other communication techniques are used at the interfaces. In particular, the **TOE** does not perform any I/O operation, like file or network access. The **TOE** requires the Java™ VM in use to operate as defined in one of the following specifications:

- JVM Specification 1.0.2 [JVMSpec1] with the Java™ Platform 1.1 API [JavaAPI1.1] and JCE 1.2.x ([JCE1.2-REF], [JCE1.2.1-REF] or [JCE1.2.2-REF])
- JVM Specification 1.2 [JVMSpec2] with one of the following APIs:
  - J2SE 5.0 [JavaAPI5]
  - J2SE 1.4.x [JavaAPI1.4]
  - J2SE 1.3.x [JavaAPI1.3] and JCE 1.2.x ([JCE1.2-REF], [JCE1.2.1-REF] or [JCE1.2.2-REF])
  - J2SE 1.2.x [JavaAPI1.2] and JCE 1.2.x ([JCE1.2-REF], [JCE1.2.1-REF] or [JCE1.2.2-REF])

Only the administrator can install and modify the environment and the **TOE**.

3 TOE Security Environment

The statement of **TOE** security environment describes the security aspects of the environment in which the **TOE** is intended to be used and the manner in which it is expected to be employed.
To this end, the statement of **TOE** security environment identifies and lists the assumptions made on the operational environment (including physical and procedural measures), states the intended method of use of the product, defines the threats that the product is designed to counter.

3.1 Assumptions

Assumption	Definition	Security Objectives
A.Protection	Protection. The **TOE** and its environment are protected in such a way that it is impossible for S.Attacker to read or modify any data managed by the **TOE**, i.e. objects defined in chapter 3.4.2.	OE.EnvironmentIntegrity, OE.EnvironmentProtection
A.Train	Administrators (S.Admin) are assumed to be suitably qualified to set up the system and to verify the **TOE** integrity.	OE.TOEIntegrity
A.Manual	S.Developer uses the **TOE** in the right way as described in the manual. In order to reach SOF high, the S.Developer must use the key sizes recommend in the manual.	OE.ExecutionEnvironment, OE.TOE_Usage
A.SeedManagement	SeedGeneration. The IT-Environment must provide a suitable seed for the RandomNumberGenerator. Furthermore it must ensure that the seed is kept secret.	OE.SuitableSeed, OE.SeedProtection

A.KeyManagement	Key Management. The IT-Environment is responsible for key management. Key management is out of scope of the **TOE**. O.PrivateKey and O.SecretKey, needed for computation of O.CipherText, O.MAC and O.Signature, must be provided by S.Application. The **TOE** does not generate or destruct keys. Given key material won't be modified or stored by the **TOE**.	OE.KeyProtection, OE.CorrectKeys
A.Java_Spec	Java™ Specification. The S.Admin or S.Developer has to install a Java™ VM that works according the JVM Specification V 1.0.2 [JVMSpec1] with the API of Java™ 1.1 [JavaAPI1.1] or JVM 1.2 [JVMSpec2] with one of the following APIs: • J2SE 5.x [JavaAPI5] • J2SE 1.4.x [JavaAPI1.4] • J2SE 1.3.x [JavaAPI1.3] • J2SE 1.2.x [JavaAPI1.2]	OE.ExecutionEnvironment
A.JCE_Spec	JCE Specification. If the Java™ API in use is older than version 1.4 [JavaAPI1.4] (1.1.x [JavaAPI1.1], 1.2.x [JavaAPI1.2] or 1.3.x [JavaAPI1.3]) the S.Admin/S.Developer has to install a JCE framework that works according to the JCE 1.2 [JCE1.2-REF], JCE 1.2.1 [JCE1.2.1-REF] or JCE 1.2.2 [JCE1.2.2-REF] specification.	OE.ExecutionEnvironment

Table 1 Assumptions

3.2 Threats

Threat	Definition	Security Objectives
T.SignatureForgery	S.Attacker could forge O.Signature or recover O.PrivateKey from O.Signature.	OT.SignatureSecure, OE.EnvironmentProtection
T.DeduceData	S.Attacker could deduce O.Data from O.CipherText.	OT.CipherSecure, OE.EnvironmentProtection

T.DeduceKey	S.Attacker could deduce O.SecretKey from O.CipherText.	OT.CipherSecure, OE.EnvironmentProtection
T.DeduceRandomSeed	S.Attacker could deduce O.RandomSeed.	OT.RandomSecure, OE.EnvironmentProtection
T.PredictRandomNumber	S.Attacker could predict the next generated O.RandomNumber.	OT.RandomSecure
T.MACForgery	S.Attacker could forge O.MAC or recover O.SecretKey.	OT.MACSecure, OE.EnvironmentProtection
T.HashForgery	S.Attacker could find collisions to O.Hash..	OT.HashSecure

Table 2 Threats

3.3 Organization Security Policies

There are no organizational security policies with which the **TOE** must comply.

3.4 Subjects, Objects

3.4.1 Subjects

Subject	Definition
S.Admin	User who is in charge to perform the **TOE** installation and **TOE** configuration.
S.Developer	User who is in charge to use the **TOE** for developing his Application (S.Application).
S.Application	The surrounding application which is using the **TOE**.
S.JavaVM	Java™ Virtual Machine.
S.Attacker	A human or a process outside the **TOE** whose main goal is to access Application sensitive information. For functions with SOF-high claim the attacker has a high attack potential and no time limit. For all other functions the **TOE** has no obvious vulnerabilities that are exploitable by attackers possessing low attack potential.

Table 3 Subjects

3.4.2 Objects

Object	Definition
O.Data	Private data obtained from the S.Application (e.g. Data to be signed).
O.MAC	MAC generated by the **TOE**.
O.Hash	Hash generated by the **TOE**.
O.Signature	Signature generated by the **TOE**.
O.CipherText	The cipher text generated by the **TOE**.
O.PrivateKey	Private Key Data which the **TOE** uses to generate O.Signature (e.g. RSA Private key).

O.SecretKey	Secret Key Data which the **TOE** uses to encrypt O.Data and/or decrypt O.CipherText (e.g. AES key).
O.RandomSeed	The seed (initial state) used by the DRNG
O.RandomNumber	The random number generated by the **TOE**

Table 4 Objects

4 Security Objectives

4.1 Security Objectives for the TOE

Security Objective	Definition	Threats
OT.SignatureSecure	Signature Secure. The **TOE** shall generate and validate O.Signature. The **TOE** uses robust algorithms to ensure that the signature cannot be forged or O.PrivateKey cannot be reconstructed from O.Signature.	T.SignatureForgery
OT.CipherSecure	Data Privacy. The **TOE** shall generate secure O.CipherText from O.Data by encryption with O.SecretKey or O.Data from O.CipherText by decryption with O.SecretKey. The use of robust algorithms and appropriate key sizes ensures that O.SecretKey, O.Data or O.CipherText cannot be deduced.	T.DeduceData, T.DeduceKey
OT.RandomSecure	The **TOE** shall generate unpredictable O.RandomNumber. O.RandomSeed cannot be deduced.	T.DeduceRandomSeed T.PredictRandomNumber
OT.MACSecure	MAC Secure. The **TOE** shall generate and validate O.MAC. It uses robust MAC algorithms, that cannot be forged. Furthermore O.SecretKey cannot be extracted from O.MAC.	T.MACForgery
OT.HashSecure	Secure hash algorithms. The **TOE** shall generate secure O.Hash.	T.HashForgery

Table 5 Security Objectives for the TOE

4.2 Security Objectives for the Environment

Security Objective	Definition	Assumptions / Threats
OE.TOEIntegrity	S.Admin or S.Developer must be sufficiently trained to set up the system and shall verify the integrity of the **TOE** by comparing the SHA-1 fingerprint of the delivered ZIP file with the fingerprint obtained by an independent secure delivery from the manufacturer.	A.Train
OE.EnvironmentIntegrity	Access Protected. The Environment (S.Admin and S.Developer) shall ensure that only S.Application, S.Developer, S.Admin or S.JavaVM has access to the **TOE**.	A.Protection
OE.KeyProtection	Key Protection. The Environment (S.Developer, S.Application and S.JavaVM) must protect the keys from unauthorized access.	A.KeyManagement
OE.CorrectKeys	Correct Keys. The Environment (S.Application) must provide well formed and valid keys to the **TOE**.	A.KeyManagement
OE.SuitableSeed	Suitable Seed. The Environment (S.Application) must provide a suitable seed to the **TOE**.	A.SeedManagement
OE.SeedProtection	Seed Protection. The Environment (S.Application and S.JavaVM) must protect the seed from unauthorized access.	A.SeedManagement

OE.ExecutionEnvironment	Execution Environment. The Environment (S.Admin and S.JavaVM) must provide an execution environment that meets the requirements (see chap. 2.3.7).	A.Java_Spec, A.JCE_Spec, A.Manual
OE.EnvironmentProtection	Side Channel. The Environment (S.Admin and S.JavaVM) must protect the **TOE** against side channel attacks.	A.Protection, T.SignatureForgery, T.DeduceData, T.DeduceKey, T.DeduceRandomSeed, T.MACForgery
OE.TOE_Usage	**TOE** Usage. The S.Application uses the **TOE** according to the manual.	A.Manual

Table 6 Security Objectives for the Environment

5 IT Security Requirements

5.1 TOE Security Functional Requirements

This chapter defines the functional requirements for the **TOE** using functional components drawn from [CC2] and the extended component FCS_RND.1/HashRandom.
The minimum strength level for the **TOE** security functional requirements FCS_COP.1/SHA-1, FCS_COP.1/SHA-265, FCS_COP.1/SHA-384, FCS_COP.1/SHA-512, FCS_COP.1/RIPEMD-160, FCS_RND.1/HashRandom, FCS_RND.1/FipsRandom and FCS_COP.1/HMAC is **SOF-high**.
According to [CC1] the strength of cryptographic algorithms is outside the scope of the CC evaluation.

5.1.1 Cryptographic support (FCS)

Cryptographic operation FCS_COP.1/SHA-1

The TSF shall perform ***Secure hash computation*** in accordance with a specified cryptographic algorithm ***SHA-1*** and cryptographic key sizes ***none*** that meet the following: ***FIPS PUB 180-1***.

Cryptographic operation FCS_COP.1/SHA-256

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

Cryptographic operation FCS_COP.1/SHA-384

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

Cryptographic operation FCS_COP.1/SHA-512

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

Cryptographic operation FCS_COP.1/RIPEMD-160

The TSF shall perform ***Secure hash computation*** in accordance with a specified cryptographic algorithm ***RIPEMD-160*** and cryptographic key sizes ***none*** that meet the following: ***ISO/IEC 10118-3:1998***.

Cryptographic operation FCS_COP.1/AES

The TSF shall perform ***Data encryption and decryption*** in accordance with a specified cryptographic algorithm ***AES ECB/CBC/OFB/CFB/CTR Mode*** and cryptographic key sizes ***128 bit, 192 bit, 256 bit*** that meet the following: ***FIPS PUB-197***.

Cryptographic operation FCS_COP.1/TripleDES

The TSF shall perform ***Data encryption and decryption*** in accordance with a specified cryptographic algorithm ***Triple-DES ECB/CBC/OFB/CFB Mode*** and cryptographic key sizes ***112 bit, 168 bit*** that meet the following: ***FIPS PUB 46-3***.

Cryptographic operation FCS_COP.1/RC2

The TSF shall perform ***Data encryption and decryption*** in accordance with a specified cryptographic algorithm ***RC2 ECB/CBC/OFB/CFB Mode*** and cryptographic key sizes ***128 - 1024 bit*** that meet the following: **RFC 2268**.

Cryptographic operation FCS_COP.1/ARCFOUR

The TSF shall perform ***Data encryption and decryption*** in accordance with a specified cryptographic algorithm ***ARCFOUR*** and cryptographic key sizes ***128 - 2048 bit*** that meet the following: **[IETF-Draft-Kaukonen]**.

Cryptographic operation FCS_COP.1/RSACipher

The TSF shall perform ***Data encryption and decryption*** in accordance with a specified cryptographic algorithm ***RSA*** and cryptographic key sizes ***(1024 + k * 64) bit - 8192 bit max., [k=0,1,2,...]*** that meet the following: ***PKCS#1 v1.5***.

Cryptographic operation FCS_COP.1/RSACipherOAEP

The TSF shall perform ***Data encryption and decryption*** in accordance with a specified cryptographic algorithm ***RSA*** and cryptographic key sizes ***(1024 + k * 64) bit – 8192 bit max., [k=0,1,2,...]*** that meet the following: ***PKCS#1 v2.1 OAEP***.

Cryptographic operation FCS_COP.1/RSASignature

The TSF shall perform ***Digital signature generation and verification*** in accordance with a specified cryptographic algorithm ***RSA*** signature and cryptographic key sizes ***(1024 + k * 64) bit – 8192 bit max., [k=0,1,2,...]*** that meet the following: ***PKCS#1 v1.5 in combination with FCS_COP.1/SHA-1, FCS_COP.1/SHA-256, FCS_COP.1/SHA-384, FCS_COP.1/SHA-512 and FCS_COP.1/RIPEMD-160***.

Cryptographic operation FCS_COP.1/RSASignaturePSS

The TSF shall perform ***Digital signature generation and verification*** in accordance with a specified cryptographic algorithm ***RSA*** signature and cryptographic key sizes ***(1024 + k * 64) bit – 8192 bit max., [k=0,1,2,...]*** that meet the following: ***PKCS#1 v2.1 PSS in combination with FCS_COP.1/SHA-1, FCS_COP.1/SHA-256, FCS_COP.1/SHA-384, FCS_COP.1/SHA-512 and FCS_COP.1/RIPEMD-160***.

Quality metrics for random numbers FCS_RND.1/HashRandom

The TSF shall provide a mechanism to generate random numbers that meet ***the functionality class K3 according to AIS20***.

The TSFs shall be able to enforce the use of TSF-generated random numbers for ***TSF.Random***.

Note: The implementation must be according to example E.5 of AIS20. Possible hash functions for generating random numbers are: SHA-1 [FIPS PUB 180-1], RIPEMD-

160 [ISO/IEC 10118-3], SHA-256 [FIPS PUB 180-2], SHA-384 [FIPS PUB 180-2], and SHA-512 [FIPS PUB 180-2].

Quality metrics for random numbers FCS_RND.1/FipsRandom

The TSF shall provide a mechanism to generate random numbers that meet ***the functionality class K4 according to AIS20***.

The TSFs shall be able to enforce the use of TSF-generated random numbers for ***TSF.Random***.

Note: The implementation must be according to FIPS PUB 186-2. Possible hash functions for generating random numbers are: SHA-1 [FIPS PUB 180-1], RIPEMD-160 [ISO/IEC 10118-3], SHA-256 [FIPS PUB 180-2], SHA-384 [FIPS PUB 180-2], and SHA-512 [FIPS PUB 180-2].

Cryptographic operation FCS_COP.1/HMAC

The TSF shall perform ***MAC generation and verification*** in accordance with a specified cryptographic algorithm ***HMAC with SHA-1, SHA-256, SHA-384, SHA-512, RipeMD-160*** and cryptographic key sizes ***(128+k*8)bit <= block size of the used hash function [k=0,1,2,...]*** that meet the following: ***RFC 2104***.

5.1.2 User Data Protection (FDP)

Import of user data without security attributes FDP_ITC.1

- **FDP_ITC.1.1**

The TSF shall enforce the ***JVM-Policy*** when importing user data, controlled under the SFP, from outside of the TSC.

- **FDP_ITC.1.2**

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

- **FDP_ITC.1.3**

The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: ***none***.

Note: ***JVM-Policy***: The private keys and user data, used for computation, are given as arguments to the **TOE.** The **TOE** does not provide any functionality to access any copies of key material used by the **TOE**, not even the application has access to these copies. Since the environment, especially the **S.JavaVM**, protects access to the memory where these copies reside, there are no means for attackers to get access to these copies. Moreover, the **S.JavaVM** guarantees that memory areas are zeroed out before they are reclaimed and assigned for reuse. With this zero-out, any key copies are destructed. The **TOE** does not modify the original key objects nor does it destruct them, it only accesses them in a read-only fashion.

5.2 TOE Security Assurance Requirements

Assurance Class	Assurance Components
ACM	ACM_CAP.3 ACM_SCP.1

ADO	ADO_DEL.1 ADO_IGS.1
ADV	ADV_FSP.1 ADV_HLD.2 ADV_RCR.1
AGD	AGD_ADM.1 AGD_USR.1
ALC	ALC_DVS.1
ATE	ATE_COV.2 ATE_DPT.1 ATE_FUN.1 ATE_IND.2
AVA	AVA_MSU.1 AVA_SOF.1 AVA_VLA.1

Table 7 Assurance Requirements (EAL3)

5.2.1 Configuration management (ACM)

Authorisation controls (ACM_CAP.3)

TOE CM coverage (ACM_SCP.1)

5.2.2 Delivery and operation (ADO)

Delivery procedures ADO_DEL.1

Installation, generation, and start-up procedures (ADO_IGS.1)

5.2.3 Development (ADV)

Informal functional specification (ADV_FSP.1)

Security enforcing high-level design (ADV_HLD.2)

Informal correspondence demonstration (ADV_RCR.1)

5.2.4 Guidance documents (AGD)

Administrator guidance (AGD_ADM.1)

User guidance (AGD_USR.1)

5.2.5 Life cycle support (ALC)

Identification of security measures (ALC_DVS.1)

5.2.6 Tests (ATE)

Analysis of coverage (ATE_COV.2)

Testing: high-level design (ATE_DPT.1)

Functional testing (ATE_FUN.1)

Independent testing – sample (ATE_IND.2)

5.2.7 Vulnerability assessment (AVA)

Examination of guidance (AVA_MSU.1)

Strength of TOE security function evaluation (AVA_SOF.1)

Developer vulnerability analysis (AVA_VLA.1)

5.3 Security Requirements for the Environment

5.3.1 General Requirements for the Environment

R.EnvironmentIntegrity
The Environment (S.Admin and S.Developer) shall ensure that only S.Application, S.Developer, S.Admin or S.JavaVM has access to the **TOE**.

R.KeyProtection

The Environment (S.Developer, SApplication and S.JavaVM) must protect the keys from unauthorized access.

R.CorrectKeys

The Environment (SApplication) must provide well-formed and valid keys to the **TOE**.

R.SuitableSeed:

The Environment (SApplication) has to provide a random seed offering suitable entropy and the length of the seed should be at least half the size of the hash value; e.g. s0 should have at least 80 bits if the pseudo random is based on the SHA-1 hash, which produces 160 bit hash values.

R.SeedProtection

The Environment (SApplication and S.JavaVM) must protect the seed from unauthorized access.

R.ExecutionEnvironment

The Environment (S.Admin and S.JavaVM) must provide an execution environment that meets the requirements (see chap. 2.3.7).

R.EnvironmentProtection

The Environment (S.Admin and S.JavaVM) must protect the **TOE** against side channel attacks.

R.TOE_Usage

The S.Application uses the **TOE** according to the S.Manual.

R.TOEIntegrity

S.Admin or S.Developer must be sufficiently trained to set up the system and shall verify the integrity of the **TOE** by comparing the SHA-1 fingerprint of the delivered ZIP file with the fingerprint obtained by an independent secure delivery from the manufacturer.

5.3.2 Security Requirements for the IT Environment

Note: In this chapter (within the Security Requirements for the IT environment drawn from CC Part 2 [CC2]) the phrases "The TSF shall" are replaced by "The IT environment shall" to clearly indicate that the IT environment, not the **TOE**, will meet the requirement. The IT Environment is the S.JavaVM and/or the S.Application.

Subset residual information protection (FDP_RIP.1)

FDP_RIP.1.1

The IT environment shall ensure that any previous information content of a resource is made unavailable upon the de-allocation of the resource from the following objects: ***seed***.

Cryptographic key generation (FCS_CKM.1)

FCS_CKM.1/AES

The IT environment shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***FIPS PUB 197*** and specified cryptographic key sizes ***128, 192, 256 bit*** that meet the following: ***FIPS PUB 197***.

FCS_CKM.1/TripleDES

The IT environment shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***FIPS 46-3*** and specified cryptographic key sizes ***112, 168 bit*** that meet the following: ***FIPS 46-3***.

FCS_CKM.1/RC2

The IT environment shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***RFC 2268*** and specified cryptographic key sizes ***128-1024 bit*** that meet the following: ***RFC 2268***.

FCS_CKM.1/ARCFOUR

The IT environment shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***IETF-Draft-Kaukonen*** and specified cryptographic key sizes ***128-2048 bit*** that meet the following: ***IETF-Draft-Kaukonen***.

FCS_CKM.1/RSACipher

The IT environment shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***PKCS#1v1.5*** and specified cryptographic key sizes ***(1024 + k * 64) – 8192 bit, [k=0,1,2,...]*** that meet the following: ***PKCS#1v1.5***.

FCS_CKM.1/RSACipherOAEP

The IT environment shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***PKCS#1v2.1*** and specified cryptographic key sizes ***(1024 + k * 64) – 8192 bit, [k=0,1,2,...]*** that meet the following: ***PKCS#1v2.1***.

FCS_CKM.1/RSASignature

The IT environment shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***PKCS#1v1.5*** and specified cryptographic key sizes ***(1024 + k * 64) – 8192 bit, [k=0,1,2,...]*** that meet the following: ***PKCS#1v1.5***.

FCS_CKM.1/RSASignaturePSS

The IT environment shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***PKCS#1v2.1*** and specified cryptographic key sizes ***(1024 + k * 64) – 8192 bit, [k=0,1,2,...]*** that meet the following: ***PKCS#1v2.1***.

FCS_CKM.1/HMAC

The IT environment shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***not applicable*** and specified cryptographic key sizes of ***at least 128 bit*** that meet the following: ***RFC2104***.

Note: As described in RFC 2104 any byte array can be used as key. Thus there is no need to specify a key generation algorithm. The key length should be at least 128 bit to prevent a brute-force key search.

Cryptographic key destruction (FCS_CKM.4)

FCS_CKM.4/AES

The IT environment shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method ***zeroing out memory areas*** that meets the following: ***JVMSpec1, JVMSpec2***.

FCS_CKM.4/TripleDES

Analogous to FCS_CKM.4/AES.

FCS_CKM.4/RC2

Analogous to FCS_CKM.4/AES.

FCS_CKM.4/ARCFOUR

Analogous to FCS_CKM.4/AES.

FCS_CKM.4/RSACipher

Analogous to FCS_CKM.4/AES.

FCS_CKM.4/RSACipherOAEP

Analogous to FCS_CKM.4/AES.

FCS_CKM.4/RSASignature

Analogous to FCS_CKM.4/AES.

FCS_CKM.4/RSASignaturePSS

Analogous to FCS_CKM.4/AES.

FCS_CKM.4/HMAC

Analogous to FCS_CKM.4/AES.

6 TOE Summary Specification

6.1 TOE Security Functions

6.1.1 TSF.Hash (SOF-high)

The **TOE** is capable of computing a cryptographic hash function (also called message digest in this context). A message digest algorithm represents the functionality of an one-way hash function for computing a fixed sized data value (message digest, hash) from input data of arbitrary size. The length of the resulting hash value usually is shorter than the length of the input data. Using a one-way hash function will make it easy to compute the hash from the given data, but hard to go the reverse way for calculating the input data when only the hash is known. Furthermore, a proper hash function should avoid any collision, meaning that it has to be hard to find two different messages producing the same hash value. The following hash algorithms are implemented:

FCS_COP.1/SHA-1 (SOF-high):
An implementation of the ***SHA-1*** hash algorithm according to ***FIPS PUB 180-1*** solely written in the Java™ Programming Language.

FCS_COP.1/SHA-256 (SOF-high):
An implementation of the ***SHA-256*** hash algorithm according to ***FIPS PUB 180-2*** solely written in the Java™ Programming Language.

FCS_COP.1/SHA-384 (SOF-high):
An implementation of the ***SHA-384*** hash algorithm according to ***FIPS PUB 180-2*** solely written in the Java™ Programming Language.

FCS_COP.1/SHA-512 (SOF-high):
An implementation of the ***SHA-512*** hash algorithm according to ***FIPS PUB 180-2*** solely written in the Java™ Programming Language.

FCS_COP.1/RIPEMD-160 (SOF-high):
An implementation of the ***RIPEMD-160*** hash algorithm according to ***ISO/IEC 10118-3:1998*** solely written in the Java™ Programming Language.

6.1.2 TSF.Cipher

The **TOE** offers functionality to decrypt and encrypt data. These functions can be subdivided into symmetric and asymmetric functions.

6.1.2.1 Symmetric Functions:

The **TOE** provides symmetric block ciphers for data encryption and decryption. Symmetric ciphers use a shared secret for decryption and encryption. Additionally these ciphers can be used in various modes of operations like ECB, CBC, OFB and CFB. The following symmetric algorithms are implemented:

FCS_COP.1/AES:

An implementation of ***AES data encryption and decryption*** in ***ECB/CBC/OFB/CFB/CTR Mode*** with ***128, 192, 256*** bit key size according to ***FIPS PUB-197*** solely written in the Java™ Programming Language.

FCS_COP.1/TripleDES:
An implementation of ***Triple-DES data encryption and decryption*** in ***ECB/CBC/OFB/CFB Mode*** with ***112, 168*** bit key size according to ***FIPS PUB 46-3*** solely written in the Java™ Programming Language.

FCS_COP.1/RC2:
An implementation of ***RC2 data encryption and decryption*** in ***ECB/CBC/OFB/CFB Mode*** with ***128-1024*** bit key size according to ***RFC2268*** solely written in the Java™ Programming Language.

FCS_COP.1/ARCFOUR:
An implementation of ***ARCFOUR data encryption and decryption*** with ***128-2048*** bit key size according to **[IETF-Draft-Kaukonen]** solely written in the Java™ Programming Language.

6.1.2.2 Asymmetric Functions:

In contrast to the symmetric ciphers, asymmetric techniques use two different keys to encrypt and decrypt the data. The **TOE** implements the following asymmetric encryption schemes:

FCS_COP.1/RSACipher:
An implementation of ***RSA data encryption and decryption*** with (***1024 + k * 64) – 8192 bit max., [k=0,1,2,...]*** bit key size according to ***PKCS#1 v1.5*** solely written in the Java™ Programming Language.

FCS_COP.1/RSACipherOAEP:
An implementation of ***RSA data encryption and decryption*** with (***1024 + k * 64) – 8192 bit max.,[k=0,1,2,...]*** bit key size according to ***PKCS#1 v2.1 OAEP*** solely written in the Java™ Programming Language.

6.1.3 TSF.Signature

The **TOE** can be used to generate and validate digital signatures according to the following schemes:

FCS_COP.1/RSASignature:
A implementation of ***RSA signature generation and verification*** with (***1024 + k * 64) – 8192 bit max., [k=0,1,2,...]*** bit key size according to ***PKCS#1 v1.5 in combination with FCS_COP.1/SHA-1, FCS_COP.1/SHA-256, FCS_COP.1/SHA-384, FCS_COP.1/SHA-512 and FCS_COP.1/RIPEMD-160*** solely written in the Java™ Programming Language.

FCS_COP.1/RSASignaturePSS:
A implementation of ***RSA signature generation and verification*** with (***1024 + k * 64) – 8192 bit max., [k=0,1,2,...]*** bit key size according to ***PKCS#1 v2.1 PSS in combination with FCS_COP.1/SHA-1, FCS_COP.1/SHA-256, FCS_COP.1/SHA-***

384, FCS_COP.1/SHA-512 and FCS_COP.1/RIPEMD-160 solely written in the Java™ Programming Language.

6.1.4 TSF.Random (SOF-high)

The **TOE** offers deterministic random number generators (DRNG). The application has to provide a random seed, offering suitable entropy.

FCS_RND.1/HashRandom (SOF-high):
An implementation of a class K3 ***secure random number generator*** as defined in AIS20 solely written in the Java™ Programming Language. **The implementation is according to *example E.5 of AIS20.*** Possible hash functions for generating random numbers are: SHA-1 [FIPS PUB 180-1], RIPEMD-160 [ISO/IEC 10118-3], SHA-256 [FIPS PUB 180-2], SHA-384 [FIPS PUB 180-2] und SHA-512 [FIPS PUB 180-2].

FCS_RND.1/FipsRandom (SOF-high):
An implementation of a class K4 ***secure random number generator*** as defined in AIS20 solely written in the Java™ Programming Language**. The implementation is according to *FIPS PUB 186-2.*** Possible hash functions for generating random numbers are: SHA-1 [FIPS PUB 180-1], RIPEMD-160 [ISO/IEC 10118-3], SHA-256 [FIPS PUB 180-2], SHA-384 [FIPS PUB 180-2] und SHA-512 [FIPS PUB 180-2].

6.1.5 TSF.MAC (SOF-high)

Message Authentication Codes (MACs) are used to guarantee the integrity and authenticity of a message. The **TOE** uses a HMAC, which is based on a shared secret and a secure hash function, in compliance with the following standard:
FCS_COP.1/HMAC (SOF-high):
An implementation of ***HMAC generation and verification*** using ***SHA-1, SHA-256, SHA-384, SHA-512, RipeMD-160*** as hash functions with key sizes of ***(128+k*8)bit <= block size of the used hash function [k=0,1,2,...]*** according to ***RFC 2104*** solely written in the Java™ Programming Language.

6.2 Assurance Measures

Assurance requirements		Measures
Configuration management	ACM_CAP.3	IAIK maintains a central CM server located in a locked server room. The CM tool in use is *Microsoft Visual SourceSafe*. Each version of each configuration item is archived and maintained in the central *Visual SourceSafe* database. Each item may be uniquely identified by its name (full path name) within the corresponding project folder and each version of that *item* by its *version number*, its creation or modification *date* and *time*, and, if available, its *label* (a *label* is not required on each version, however, the evaluated version of the IAIK-JCE CC Core is tagged with a unique identifier (as all other release versions, too)). *Version number*, *date* and *time* are assigned automatically; A *label* is set by the user. Authorisation controls to both, the central server and the user workstations, are managed by the security functions of the particular operating system. Access to the *SourceSafe* database is password protected to authorized developers only.
	ACM_SCP.1	The CM system tracks the **TOE** implementation representation as well as documentation and test material. The implementation is archived as encapsulated release versions containing the entire Java™ source code.
Delivery and operation	ADO_DEL.1	IAIK delivers the library and all documentation in terms of a single ZIP archive on a CD. Furthermore the CD contains a SHA-1 fingerprint of this ZIP file and of all relevant parts contained within the ZIP. Thereby it is possible to check the integrity of some unzipped parts after the installation. A tool to validate these hash values will be included as well. All these hash values will be published on IAIKs web server (https access) and additionally will be sent to the

		customers with a signed e-mail , with fax or handed over personally.
	ADO_IGS.1	The **TOE** itself, which is a jar archive containing the compiled Java™ code, is signed, as required by the JCE specification [JCE1.4-REF] and [JCE5-REF]. There is no installation of the **TOE** in the conventional meaning. The administrator simply has to unzip the **TOE** and put it on the “right place”. The “right place” depends on the application using the **TOE**.
Development	ADV_FSP.1	IAIK provides a functional specification of the **TOE**. The usage of the security functions of the **TOE** is primarily prescribed by the JCA/JCE architecture, which defines most of the interfaces. There exist different versions of this architecture. The various original specifications of this architecture are added to the delivered documents. All other interfaces that are not compliant to the JCA/JCE architecture are described additionally.
	ADV_HLD.2	The HLD (High Level Design) description introduces the subsystems of the **TOE**. According to the JCA/JCE provider definition each subsystem is defined as a collection of Java™ packages. There are only two subsystems, IAIK and UTILITIES. Subsystem IAIK implements all **TOE** security functions (see chapter 6.1 of this document). Subsystem UTILITIES provides a set of utilities that are used by subsystem IAIK. The HLD also presents all external (to the environment) and internal interfaces (among the subsystems) of the two subsystems. The presentation is based on the Javadoc output of the corresponding classes. With respect to the functional specification, the HLD introduces the JCA/JCE SPI as main interface between final application (end user, API) and **TOE** subsystems, and discusses where the **TOE** extends the/differs from the JCA/JCE reference API/SPI.
	ADV_RCR.1	The RCR (Representation Correspondence) shows the correspondence between the TSS security functions (as presented in chapter 6.1 of this document), the FSP (functional specification) and HLD (high level design). It stepwise refines the design by leading from the JCA/JCE API (FSP) to the JCE/JCA SPI (HLD) to **TOE** subsystems (HLD).
Guidance documents	AGD_ADM.1	There is no separate administrator manual. All required information on how to install the **TOE** are within the user manual.
	AGD_USR.1	IAIK provides a user manual, which contains all necessary information about the **TOE** installation and usage (required by the application programmers).

Life cycle support	ALC_DVS.1	The protection of the development environment is guaranteed by physical, procedural and personal measures.
Tests	ATE_COV.2	The tests are explained in a test specification document. It describes the source of test data and how the tests are organized. Moreover, there is a test suite for the complete **TOE** which include tests of all interfaces. This test suite runs automatically and applies test vectors for each TSF. The test vectors consist of input data and expected output data. Standard vectors were taken where available. The tests have been monitored with a tool that measures the code coverage of the test suite. The evaluators will have access to the test suite to verify it.
	ATE_DPT.1	
	ATE_FUN.1	
	ATE_IND.2	
Vulnerability assessment	AVA_MSU.1	The AVA (Vulnerability Assessment) analyses the **TOE** for vulnerabilities. It starts with an investigation of the guidance documentation to ensure if it is complete and consistent. Then, there follows a consideration of the strength of the used security functions. The document closes with an analysis of the **TOE** for vulnerabilities.
	AVA_SOF.1	
	AVA_VLA.1	

7 PP Claims

This chapter is not applicable to this ST (see chapter 1.3).

8 Rationale

8.1 Security Objectives Rationale

This chapter shall demonstrate that the stated security objectives are traceable to all of the aspects identified in the **TOE** security environment and are suitable to cover them.

Policy /Threat/ Assumption:	**Objectives:**	**Comment:**
Security Objectives for the **TOE**		
T.DeduceData	OT.CipherSecure	This objective ensures that data cannot be deduced from O. CipherText. By the use of appropriate cipher algorithms, which are generally known as secure, it is not possible to deduce data from the cipher text.
T.DeduceKey	OT.CipherSecure	This objective ensures that keys cannot be deduced from O. CipherText. By the use of appropriate cipher algorithms, which are generally known as secure, it is not possible to deduce the key from the cipher text.
T.DeduceRandomSeed	OT.RandomSecure	This objective ensures that the random seed cannot be deduced. By the use of an appropriate random number generation algorithm, which is generally known as secure, it is not possible to deduce the random seed.

T.PredictRandomNumber	OT.RandomSecure	This objective ensures that the next generated random number cannot be predicted. By the use of an appropriate random number generation algorithm, which is generally known as secure, it is not possible to predict the random number.
T.HashForgery	OT.HashSecure	This objective ensures that the S.Attacker cannot find collisions. By the use of an appropriate hash algorithm, which is generally known as secure, it is not possible to find collisions.
T.MACForgery	OT.MACSecure	This objective ensures that the S.Attacker cannot forge O.MAC or recover O.SecretKey from O.MAC. By the use of an appropriate mac algorithm, which is generally known as secure, it is not possible to forge the mac or to recover the key.
T.SignatureForgery	OT.SignatureSecure	This objective ensures that O.Signature cannot be forged and O.PrivateKey cannot be recovered from O.Signature. By the use of an appropriate signature algorithm, which is generally known as secure, it is not possible to forge the signature or to recover the key.
Security Objectives for the Environment		
A.Protection	OE.EnvironmentIntegrity, OE.EnvironmentProtection	These objectives ensure that S.Attacker cannot read or modify any data.

A.Java_Spec	OE.ExecutionEnvironment	This objective ensures that the Java™ version in use meets the required specification.
A.JCE_Spec	OE.ExecutionEnvironment	This objective ensures that the JCE version in use meets the required specification.
A.KeyManagement	OE.KeyProtection, OE.CorrectKeys	These objectives ensure an appropriate key management.
A.Manual	OE.ExecutionEnvironment, OE.TOE_Usage	These objectives ensure that the **TOE** is used and behaves according to the manual.
A.Train	OE.TOEIntegrity	This objective ensures that the integrity of the **TOE** can be verified at any time. This can be attained by a suitably qualified S.Admin.
A.SeedManagement	OE.SuitableSeed, OE.SeedProtection	These objectives ensure an appropriate seed management.
T.DeduceData	OE.EnvironmentProtection	This objective ensures that data cannot be read or modified by S.Attacker before the **TOE** receives the data.
T.DeduceKey	OE.EnvironmentProtection	This objective ensures that the environment protects the key.
T.DeduceRandomSeed	OE.EnvironmentProtection	This objective ensures that the environment protects the seed.
T.MACForgery	OE.EnvironmentProtection	This objective ensures that the data cannot be read or modified by S.Attacker before the **TOE** receives the data.
T.SignatureForgery	OE.EnvironmentProtection	This objective ensures that the data cannot be read or modified by S.Attacker before the **TOE** receives the data.

Table 8 Mapping the TOE Security Environment to Security Objectives

Objective:	**Policies/ Threats/ Assumptions:**	**Comment:**
Security Objectives for the **TOE**		
OT.CipherSecure	T.DeduceData, T.DeduceKey	These threats are countered by the use of ciphers which are known as secure.
OT.HashSecure	T.HashForgery	This threat is countered by the use of hash algorithms which are known as secure.
OT.MACSecure	T.MACForgery	This threat is countered by the use of secure mac algorithms.
OT.SignatureSecure	T.SignatureForgery	This threat is countered by the use of secure signature algorithms.
OT.RandomSecure	T.PredictRandomNumber, T.DeduceRandomSeed	This threat is countered by the use of secure random number generation algorithms.
Security Objectives for the Environment		
OE.TOEIntegrity	A.Train	This assumption assures the integrity of the **TOE**.
OE.EnvironmentIntegrity	A.Protection	This assumption assures the integrity of the environment.
OE.CorrectKeys	A.KeyManagement	This assumption assures that the environment provides correct keys to the **TOE**.
OE.SuiteableSeed	A.SeedManagement	This assumption assures that the environment provides suitable seeds to the **TOE**.
OE.ExecutionEnvironment	A.Java_Spec, A.JCE_Spec, A.Manual	These assumptions assure that the provided execution environment meets the required specification.
OE.KeyProtection	A.KeyManagement	This assumption assures the protection of the key material.
OE.SeedProtection	A.SeedManagement	This assumption assures the protection of the seed.
OE.EnvironmentProtection	A.Protection, T.DeduceData, T.DeduceKey, T.DeduceRandomSeed, T.SignatureForgery, T.MACForgery	This assumption assures that the environment is protected and helps to avert these threats.
OE.TOE_Usage	A.Manual	This assumption assures that the **TOE** is used in an appropriate way.

Table 9 Tracing of Security Objectives to the TOE Security Environment

8.2 Security Requirements Rationale

The **TOE** security objectives concern the provision of "secure" cryptographic functionality as further specified in the security functional requirements. They contain no specific strength-related properties. Therefore, strength of function claim SOF-high is consistent with the security objectives for the **TOE**.

8.2.1 Functional Security Requirements Rationale

8.2.1.1 Functional Security Requirements Rationale for the TOE

Objectives:	Requirements:	Comments:
OT.CipherSecure	FCS_COP.1/AES, FCS_COP.1/TripleDES, FCS_COP.1/RSACipher, FCS_COP.1/RSACipherOAEP FCS_COP.1/RC2, FCS_COP.1/ARCFOUR, FDP_ITC.1	The use of the left-mentioned cryptographic operations ensures that the generated O.CipherText is secure. FDP_ITC.1 is needed to import cryptographic keys for the operation.
OT.HashSecure	FCS_COP.1/SHA-1, FCS_COP.1/SHA-256, FCS_COP.1/SHA-384, FCS_COP.1/SHA-512, FCS_COP.1/RIPEMD-160	The use of the left-mentioned hash functions ensures that the generated O.Hash is secure.
OT.MACSecure	FCS_COP.1/HMAC, FDP_ITC.1	The use of this cryptographic function ensures that the generated O.MAC is secure. FDP_ITC.1 is needed to import cryptographic keys for the operation.
OT.SignatureSecure	FCS_COP.1/RSASignature, FCS_COP.1/RSASignaturePSS, FDP_ITC.1	The use of the left-mentioned cryptographic operations ensures that the generated O.Signature is secure. FDP_ITC.1 is needed to import cryptographic keys for the operation.
OT.RandomSecure	FCS_RND.1/HashRandom FCS_RND.1/FipsRandom	The use of the left-mentioned functions ensures that the generated O.RandomNumber is secure.

Table 10 Functional Security Requirements Rationale for the TOE

8.2.1.2 Functional Security Requirements Rationale for the environment

Objectives:	Requirements:	Comments:
OE.TOEIntegrity	R.TOEIntegrity	If the left-mentioned requirement is met, the **TOE** integrity is ensured.
OE.ExecutionEnvironment	R.ExecutionEnvironment	If the left-mentioned requirement is met, the execution environment fulfils the required conditions as described in chapter 2.3.7.
OE.CorrectKeys	R.CorrectKeys, FCS_CKM.1	If the left-mentioned requirements are met, the use of correct keys is ensured.
OE.SuitableSeed	R.SuitableSeed	If the left-mentioned requirement is met, the use of applicable seeds is ensured.
OE.SeedProtection	R.SeedProtection, FDP_RIP.1	If the left-mentioned requirement is met, the seed protection is ensured.
OE.EnvironmentIntegrity	R.EnvironmentIntegrity	If the left-mentioned requirement is met, the environment integrity is ensured.
OE.TOE_Usage	R.TOE_Usage	If the left-mentioned requirement is met, the right **TOE** usage is ensured.
OE.KeyProtection	R.KeyProtection, FCS_CKM.4.1	If the left-mentioned requirements are met, the protection of the key is ensured.
OE.EnvironmentProtection	R.EnvironmentProtection	If the left-mentioned requirement is met, the environment protection is ensured.

Table 11 Functional Security Requirements Rationale for the environment

8.2.2 Security Assurance Requirements Rationale

To meet the requirements of an application for the generation and the verification of electronic signatures, the selected evaluation level is EAL3, and the selected strength of functions is high (SOF-high).

8.3 TOE Summary Specification Rationale

8.3.1 TOE Security Functions Rationale

The **TOE** security functions TSF.Hash, TSF.Random and TSF.MAC reach SOF-high. For these security functions, the **TOE** should be able to resist attacks from attackers with sophisticated knowledge.
Given that the **TOE** is generally available, the attacker is assumed to have unlimited time to set up attacks. The attacker is assumed to use equipment that is state of the art. The data which is processed by the **TOE** is assumed to be of high importance.
To counter these threats, the **TOE** uses cryptographic functions.

Security Functions:	Mechanism:	Min.-Key-Size:	Security Functional Requirements:	SOF:
TSF.Hash	SHA-1 SHA-256 SHA-384 SHA-512 RIPEMD-160		FCS_COP.1/SHA-1 FCS_COP.1/SHA-256 FCS_COP.1/SHA-384 FCS_COP.1/SHA-512 FCS_COP.1/RIPEMD-160	high high high high high
TSF.Cipher	AES TripleDES RC2 ARCFOUR RSA PKCS#1 v1.5 RSA PKCS#1 v2.1 OAEP	128 bit 112 bit 128 bit 128 bit 1024 bit 1024 bit	FCS_COP.1/AES FCS_COP.1/TripleDES FCS_COP.1/RC2 FCS_COP.1/ARCFOUR FCS_COP.1/RSACipher FCS_COP.1/RSACipherOAEP FDP_ITC.1	
TSF.Signature	RSA PKCS#1 v1.5 with SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD-160 RSA PKCS#1 v2.1 PSS with SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD-160	1024 bit 1024 bit	FCS_COP.1/RSASignature FCS_COP.1/RSASignaturePSS FDP_ITC.1	
TSF.Random			FCS_RND.1/HashRandom FCS_RND.1/FipsRandom	high high
TSF.MAC	HMAC with SHA-1, SHA-256, SHA 384, SHA 512, RipeMD-160	128 bit	FCS_COP.1/HMAC FDP_ITC.1	high

Table 12 Assurance Security Requirements Rationale

There is a one to one correspondence between the TSF and the SFR with the exception of FDP_ITC.1. This requirement is needed to import keys for cryptographic operations and is implicitly fulfilled by the corresponding TSF (Cipher, Signature, MAC). That means that the TSF are suitable to meet the security functional requirements and work together without any conflict.

8.4 Dependency Rationale

Requirement:	Dependencies:
Functional Requirements	
FCS_COP.1/SHA-1	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/SHA-256	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/SHA-384	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/SHA-512	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RIPEMD-160	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/AES	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/TripleDES	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RC2	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/ARCFOUR	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RSACipher	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RSACipherOAEP	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RSASignature	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RSASignaturePSS	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_CKM.4/AES	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/TripleDES	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/RC2	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/ARCFOUR	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/RSACipher	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/RSACipherOAEP	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/RSASignature	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2

FCS_CKM.4/RSASignaturePSS	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/HMAC	[FDP_ITC.1 or FDP_ITC.2 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.1/AES	[FCS_CKM.2 or FCS_COP.1], FCS_CKM.4, FMT_MSA.2
FCS_CKM.1/TripleDES	[FCS_CKM.2 or FCS_COP.1], FCS_CKM.4, FMT_MSA.2
FCS_CKM.1/RC2	[FCS_CKM.2 or FCS_COP.1], FCS_CKM.4, FMT_MSA.2
FCS_CKM.1/ARCFOUR	[FCS_CKM.2 or FCS_COP.1], FCS_CKM.4, FMT_MSA.2
FCS_CKM.1/RSACipher	[FCS_CKM.2 or FCS_COP.1], FCS_CKM.4, FMT_MSA.2
FCS_CKM.1/RSACipherOAEP	[FCS_CKM.2 or FCS_COP.1], FCS_CKM.4, FMT_MSA.2
FCS_CKM.1/RSASignature	[FCS_CKM.2 or FCS_COP.1], FCS_CKM.4, FMT_MSA.2
FCS_CKM.1/RSASignaturePSS	[FCS_CKM.2 or FCS_COP.1], FCS_CKM.4, FMT_MSA.2
FCS_CKM.1/HMAC	[FCS_CKM.2 or FCS_COP.1], FCS_CKM.4, FMT_MSA.2
FDP_ITC.1	[FDP_ACC.1 or FDP_IFC.1], FMT_MSA.3
FCS_RND.1/HashRandom	FPT_TST.1
FCS_RND.1/FipsRandom	FPT_TST.1
FCS_COP.1/HMAC	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FDP_RIP.1	-
Assurance Requirements	
ACM_CAP.3	ACM_SCP.1, ALC_DVS.1
ACM_SCP.1	ACM_CAP.3
ADO_DEL.1	-
ADO_IGS.1	AGD_ADM.1
ADV_FSP.1	ADV_RCR.1
ADV_HLD.2	ADV_FSP.1, ADV_RCR.1
ADV_RCR.1	-
AGD_ADM.1	ADV_FSP.1
AGD_USR.1	ADV_FSP.1
ALC_DVS.1	-
ATE_COV.2	ADV_FSP.1, ATE_FUN.1
ATE_DPT.1	ADV_HLD.1, ATE_FUN.1
ATE_FUN.1	-
ATE_IND.2	ADV_FSP.1, AGD_ADM.1, AGD_USR.1, ATE_FUN.1
AVA_MSU.1	ADO_IGS.1, ADV_FSP.1, AGD_ADM.1, AGD_USR.1
AVA_SOF.1	ADV_FSP.1, ADV_HLD.1
AVA_VLA.1	ADV_FSP.1, ADV_HLD.1, AGD_ADM.1, AGD_USR.1

Table 13 Functional and Assurance Requirements Dependencies

TSF.Hash

FCS_COP.1/SHA-1:

- **FDP_ITC.1/SHA-1 Import of user data without security attributes:**

The computation of SHA-1 does not require the import of user data in terms of cryptographic keys.

- **FDP_ITC.2/SHA-1 Import of user data with security attributes:**

The computation of SHA-1 does not require the import of user data in terms of cryptographic keys. This functional component is not needed.

- **FCS_CKM.1/SHA-1 Cryptographic key generation:**

There are no cryptographic keys required and thus there is no requirement for this security functional component.

- **FCS_CKM.4/SHA-1 Cryptographic key destruction:**

Since the computation of SHA-1 does not require any cryptographic keys this component can be omitted.

- **FMT_MSA.2/SHA-1 Secure security attributes:**

The hash computation does not require cryptographic keys and therefore no management of the security attributes. This security functional component is not needed.

FCS_COP.1/SHA-256:
Analogous to the points as described in FCS_COP.1/SHA-1.

FCS_COP.1/SHA-384:
Analogous to the points as described in FCS_COP.1/SHA-1.

FCS_COP.1/SHA-512:
Analogous to the points as described in FCS_COP.1/SHA-1.

FCS_COP.1/RIPEMD-160:
Analogous to the points as described in FCS_COP.1/SHA-1.

TSF.Cipher

FCS_COP.1/AES:

- **FDP_ITC.1/AES Import of user data without security attributes:**

See chapter 5.1.2 FDP_ITC.1.

- **FDP_ITC.2/AES Import of user data with security attributes:**

The **TOE** does not use import of user data with security attributes. This functional component is not needed.

- **FCS_CKM.1/AES Cryptographic key generation:**

The **TOE** does not generate keys itself. The environment is responsible for the key generation, so the requirement is included in chapter 5.3 "Security Requirements for the Environment".

- **FCS_CKM.4/AES Cryptographic key destruction:**

The **TOE** does not destroy keys itself. The environment is responsible for the key destruction, so the requirement is included in chapter 5.3 "Security Requirements for the Environment".

- **FMT_MSA.2/AES Secure security attributes:**

There are no security attributes related with the cryptographic keys (see FDP_ITC.1/AES).

FCS_COP.1/TripleDES:

Analogous to the points as described in FCS_COP.1/AES.

FCS_COP.1/RC2:

Analogous to the points as described in FCS_COP.1/AES.

FCS_COP.1/ARCFOUR:

Analogous to the points as described in FCS_COP.1/AES.

FCS_COP.1/RSACipher:

Analogous to the points as described in FCS_COP.1/AES.

FCS_COP.1/RSACipherOAEP:

Analogous to the points as described in FCS_COP.1/AES.

TSF.Signature

FCS_COP.1/RSASignature:

- **FDP_ITC.1/RSASignature Import of user data without security attributes:**

See chapter 5.1.2 FDP_ITC.1.

- **FDP_ITC.2/ RSASignature Import of user data with security attributes:**

The **TOE** does not use import of user data with security attributes. This functional component is not needed.

- **FCS_CKM.1/RSASignature Cryptographic key generation:**

The **TOE** does not generate keys itself. The environment is responsible for the key generation, so the requirement is included in chapter 5.3 “Security Requirements for the Environment”.

- **FCS_CKM.4/RSASignature Cryptographic key destruction:**

The **TOE** does not destroy keys itself. The environment is responsible for the key destruction, so the requirement is included in chapter 5.3 “Security Requirements for the Environment”.

- **FMT_MSA.2/RSASignature Secure security attributes:**

There are no security attributes related with the cryptographic keys (see FDP_ITC.1/RSASignature).

FCS_COP.1/RSASignaturePSS:

Analogous to the points as described in FCS_COP.1/RSASignature.

TSF.Random

FCS_RND.1/HashRandom:

- **FPT_TST.1/HashRandom TSF testing**

This dependency is intended for true random number generators (TRNG). Since the **TOE** implements a deterministic random number generator (DRNG) and the seed handling is done outside the **TOE** this functional requirement is not required.

FCS_RND.1/FipsRandom:

- **FPT_TST.1/FipsRandom TSF testing**

This dependency is intended for true random number generators (TRNG). Since the **TOE** implements a deterministic random number generator (DRNG) and the seed handling is done outside the **TOE** this functional requirement is not required.

TSF.MAC

FCS_COP.1/HMAC:

- **FDP_ITC.1/HMAC Import of user data without security attributes:**

See chapter 5.1.2 FDP_ITC.1.

- **FDP_ITC.2/ HMAC Import of user data with security attributes:**

The **TOE** does not use import of user data with security attributes. This functional component is not needed.

- **FCS_CKM.1/HMAC Cryptographic key generation:**

The **TOE** does not generate keys itself. The environment is responsible for the key generation, so the requirement is included in chapter 5.3 “Security Requirements for the Environment”.

- **FCS_CKM.4/HMAC Cryptographic key destruction:**

The **TOE** does not destroy keys itself. The environment is responsible for the key destruction, so the requirement is included in chapter 5.3 “Security Requirements for the Environment”.

- **FMT_MSA.2/HMAC Secure security attributes:**

There are no security attributes related with the cryptographic keys (see FDP_ITC.1/HMAC).

FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4/AES:

- **FDP_ITC.1/AES**

See chapter 5.1.2 FDP_ITC.1.

- **FDP_ITC.2/AES**

In this context, the user data are the keys, which do not have security attributes. This functional component is not needed.

- **FCS_CKM.1/AES**

Is included in chapter 5.3 “Security Requirements for the Environment”.

- **FMT_MSA.2/AES**

There are no security attributes related with the cryptographic keys and therefore this functional component is not needed.

FCS_CKM.4/TripleDES:

Analogous to the points as described in FCS_CKM.4/AES.

FCS_CKM.4/RC2:

Analogous to the points as described in FCS_CKM.4/AES.

FCS_CKM.4/ARCFOUR:

Analogous to the points as described in FCS_CKM.4/AES.

FCS_CKM.4/RSACipher:

Analogous to the points as described in FCS_CKM.4/AES.

FCS_CKM.4/RSACipherOAEP:

Analogous to the points as described in FCS_CKM.4/AES.

FCS_CKM.4/RSASignature:

Analogous to the points as described in FCS_CKM.4/AES.

FCS_CKM.4/RSASignaturePSS:

Analogous to the points as described in FCS_CKM.4/AES.

FCS_CKM.4/HMAC:
Analogous to the points as described in FCS_CKM.4/AES.

FCS_CKM.1 Cryptographic key generation

FCS_CKM.1/AES:

- **FCS_COP.1/AES**

Is included in chapter 5.1.1 “Cryptographic support” as security requirement for the **TOE**.

- **FCS_CKM.4/AES**

Is included in chapter 5.3 “Security Requirements for the Environment”.

- **FMT_MSA.2/AES**

There are no security attributes related with the cryptographic keys and therefore this functional component is not needed.

FCS_CKM.1/TripleDES:
Analogous to the points as described in FCS_CKM.1/AES.

FCS_CKM.1/RC2:
Analogous to the points as described in FCS_CKM.1/AES.

FCS_CKM.1/ARCFOUR:
Analogous to the points as described in FCS_CKM.1/AES.

FCS_CKM.1/RSACipher:
Analogous to the points as described in FCS_CKM.1/AES.

FCS_CKM.1/RSACipherOAEP:
Analogous to the points as described in FCS_CKM.1/AES.

FCS_CKM.1/RSASignature:
Analogous to the points as described in FCS_CKM.1/AES.

FCS_CKM.1/RSASignaturePSS:
Analogous to the points as described in FCS_CKM.1/AES.

FCS_CKM.1/HMAC:
Analogous to the points as described in FCS_CKM.1/AES.

FDP_ITC.1 Import of user data without security attributes

- **FDP_ACC.1 Subset access control:**

The **TOE** does not provide any access control itself. The access control is subject to the S.JavaVM. This functional component is therefore not required.

- **FDP_IFC.1 Subset information flow control:**

For our purposes, no information control is needed and therefore this functional component is not required.

- **FMT_MSA.3 Static attribute initialization:**

For our purposes, no attributes are needed and therefore this functional component is not needed.

Security Assurance Requirements Dependencies Rationale

Within each EAL, all dependencies are met by definition of the EALs. Here, EAL3 was chosen.

8.5 Security Functional Requirements Grounding in Objectives

Requirements:	Objectives:
FCS_COP.1/SHA-1	OT.HashSecure
FCS_COP.1/SHA-256	OT.HashSecure
FCS_COP.1/SHA-384	OT.HashSecure
FCS_COP.1/SHA-512	OT.HashSecure
FCS_COP.1/RIPEMD-160	OT.HashSecure
FCS_COP.1/AES	OT.CipherSecure
FCS_COP.1/TripleDES	OT.CipherSecure
FCS_COP.1/RC2	OT.CipherSecure
FCS_COP.1/ARCFOUR	OT.CipherSecure
FCS_COP.1/RSACipher	OT.CipherSecure
FCS_COP.1/RSACipherOAEP	OT.CipherSecure
FCS_COP.1/RSASignature	OT.SignatureSecure
FCS_COP.1/RSASignaturePSS	OT.SignatureSecure
FCS_COP.1/HMAC	OT.MACSecure
FCS_RND.1/FipsRandom	OT.RandomSecure
FCS_RND.1/HashRandom	OT.RandomSecure
FDP_RIP.1.1	OE.KeyProtection
FCS_CKM.4.1	OE.KeyProtection

Table 14 Requirements to Objectives Mapping

9 Appendix A – References

[AIS20]	Bundesamt für Sicherheit in der Informationstechnik (BSI), Application Notes and Interpretation of the Scheme (AIS), AIS 20, Version 1.0: Functionality classes and evaluation methodology for deterministic random number generators, Bundesamt für Sicherheit in der Informationstechnik (BSI), December 1999
[AIS31]	Bundesamt für Sicherheit in der Informationstechnik (BSI), Application Notes and Interpretation of the Scheme (AIS), AIS 31, Version 1.0: Functionality classes and evaluation methodology for physical random number generators, Bundesamt für Sicherheit in der Informationstechnik (BSI), September 2001
[CC]	ISO International Standard (IS) 15408:2005, Common Criteria for Information Technology Security Evaluation (Comprising Parts 1-3, [CC1], [CC2], [CC3] CC 2.3), Common Criteria Implementation Board (CCIB) and the International Standards Organization (ISO), JTC1/SC27/WG3 available online at http://www.commoncriteriaportal.org
[CC1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and General Model CCIMB-2005-08-001, Version 2.3, August 2005, Common Criteria Implementation Board (CCIB) available online at http://www.commoncriteriaportal.org
[CC2]	Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Requirements CCIMB-1005-08-002, Version 2.3, August 2005, Common Criteria Implementation Board (CCIB) available online at http://www.commoncriteriaportal.org
[CC3]	Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Requirements CCIMB-2005-08-003, Version 2.3, August 2005, Common Criteria Implementation Board (CCIB) available online at http://www.commoncriteriaportal.org
[CRYPTO SPEC]	Java™ Cryptography Architecture, API Specification & Reference, SUN Microsystems, Inc. http://java.sun.com/security/index.html, July 2004
[EU_directive]	Directive 1999/93/EC of the european parlament and of the council, 13 December 1999, on a Community framework for electronic signatures
[FIPS 46-3]	U.S. Department Of Commerce, Federal Information Processing Standards Publication: Data Encryption Standard (DES), FIPS PUB 46-3, U.S. Department Of Commerce, 199925 October 251999, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
[FIPS PUB 180-1]	U.S. Department Of Commerce, Federal Information Processing Standards Publication: Secure Hash Standard, FIPS PUB 180-1, U.S. Department Of Commerce, 171995 April 1995 17

[FIPS PUB 180-2]	U.S. Department Of Commerce, Federal Information Processing Standards Publication: Secure Hash Standard, FIPS PUB 180-2, U.S. Department Of Commerce, 262001 November 2001 26, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
[FIPS PUB 197]	U.S. Department Of Commerce, Federal Information Processing Standards Publication: Advanced Encryption Standard, FIPS PUB 197, U.S. Department Of Commerce, 26 November 2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[IETF-Draft-Kaukonen]	K.Kaukonen, R.Thayer: A Stream Cipher Encryption Algorithm "Arcfour", IETF draftInternet Draft: draft-kaukonen-cipher-arcfour-03.txt, 14 July 1999.
[ISO/IEC 10118-3]	ISO/IEC 10118-3:2003, Information technology -- Security techniques -- Hash-functions -- Part 3: Dedicated hash-functions, ISO/IEC, JTC 1/SC 27, 14 November 2003
[JavaAPI1.1]	Java™ Platform 1.1 Core API Specification, SUN Microsystems, Inc., Palo Alto, California, 1995-1999, http://java.sun.com/products/archive/jdk/1.1/index.html
[JavaAPI1.2]	Java™ 2 Platform, Standard Edition, v1.2.2 API Specification, SUN Microsystems, Inc., 1999, http://java.sun.com/products/archive/j2se-eol.html
[JavaAPI1.3]	Java™ 2 Platform, Standard Edition, v 1.3.1 API Specification, SUN Microsystems, Inc., 2001, http://java.sun.com/products/archive/j2se-eol.html
[JavaAPI1.4]	Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification, SUN Microsystems, Inc., 2003, http://java.sun.com/j2se/1.4.2/docs/api/
[JavaAPI5]	Java™ 2 Platform, Standard Edition, v 5.0 API Specification, SUN Microsystems, Inc., 2004, http://java.sun.com/j2se/1.5.0/docs/api/
[JCA1.1-API]	Java™ Cryptography Architecture API, JavaDoc of package java.security, Java™ Platform API Specification, version 1.1, SUN Microsystems, Inc.
[JCA1.1-REF]	Java™ Cryptography Architecture API Specification & Reference, Java™ Specification, version 1.1, SUN Microsystems, Inc.
[JCA1.2-API]	Java™ Cryptography Architecture API, JavaDoc of package java.security, Java™ Platform API Specification, version 1.2, SUN Microsystems, Inc.
[JCA1.2-REF]	Java™ Cryptography Architecture API Specification & Reference, Java™ 2 SDK, Standard Edition, v 1.2, SUN Microsystems, Inc.
[JCA1.3-API]	Java™ Cryptography Architecture API, JavaDoc of package java.security, Java™ Platform API Specification, version 1.3, SUN Microsystems, Inc.
[JCA1.3-REF]	Java™ Cryptography Architecture API Specification & Reference, Java™ 2 SDK, Standard Edition, v 1.3, SUN Microsystems, Inc.

[JCA1.4-API]	Java™ Cryptography Architecture API, JavaDoc of package java.security, Java™ Platform API Specification, version 1.4, SUN Microsystems, Inc., http://java.sun.com/j2se/1.4.2/docs/api/java/security/package-summary.html
[JCA5-API]	Java™ Cryptography Architecture API, JavaDoc of package java.security, Java™ Platform API Specification, version 5.0, SUN Microsystems, Inc., http://java.sun.com/j2se/1.5.0/docs/api/java/security/package-summary.html
[JCA1.4-PROV]	How to Implement a Provider for the Java™ Cryptography Architecture, Java™ Platform Specification, version 1.4, SUN Microsystems, Inc., http://java.sun.com/j2se/1.4.2/docs/guide/security/HowToImplAProvider.html
[JCA1.4-REF]	Java™ Cryptography Architecture API Specification & Reference, Java™ Platform Specification, version 1.4, SUN Microsystems, Inc., http://java.sun.com/j2se/1.4/docs/guide/security/CryptoSpec.html
[JCA5-REF]	Java™ Cryptography Architecture API Specification & Reference, Java™ Platform Specification, version 5.0, SUN Microsystems, Inc., http://java.sun.com/j2se/1.5/docs/guide/security/CryptoSpec.html
[JCE1.4-API]	Java™ Cryptography Extension API, JavaDoc of package java.security, Java™ Platform API Specification, version 1.4, SUN Microsystems, Inc., http://java.sun.com/j2se/1.4.2/docs/api/javax/crypto/package-summary.html
[JCE1.4-PROV]	How to Implement a Provider for the Java™ Cryptography Extension, Java™ Platform Specification, version 1.4, SUN Microsystems, Inc., http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/HowToImplAJCEProvider.html
[JCE5-REF]	Java™ Cryptography Extension (JCE) API Specification & Reference, Java™ 2 Standard Edition, version 5.0, SUN Microsystems, Inc. http://java.sun.com/j2se/1.5.0/docs/guide/security/jce/JCERefGuide.html
[JCE1.4-REF]	Java™ Cryptography Extension (JCE) API Specification & Reference, Java™ 2 Standard Edition, version 1.4, SUN Microsystems, Inc. http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html
[JCE1.2.2-REF]	Java™ Cryptography Extension (JCE) API Specification & Reference, version 1.2.2, SUN Microsystems, Inc., http://java.sun.com/products/jce/

[JCE1.2.1-REF]	Java™ Cryptography Extension (JCE) API Specification & Reference, version 1.2.1, SUN Microsystems, Inc., http://java.sun.com/products/jce/
[JCE1.2-REF]	Java™ Cryptography Extension (JCE) API Specification & Reference, version 1.2, SUN Microsystems, Inc., http://java.sun.com/products/jce/
[JVMSpec1]	Tim Lindholm, Frank Yellin: Tim Lindholm, Frank Yellin: The Java™ Virtual Machine Specification, Addison-Wesley Pub Co, September 1996, ASIN: 020163452X http://java.sun.com/docs/books/vmspec/index.html
[JVMSpec2]	Tim Lindholm, Frank Yellin: Tim Lindholm, Frank Yellin: The Java™ Virtual Machine Specification (2nd Edition), Addison-Wesley Pub Co, 2nd edition , April 1999, ISBN: 0201432943 http://java.sun.com/docs/books/vmspec/index.html
[PKCS#1v1.5]	PKCS#1 v1.5: RSA Encryption Standard RSA Laboratories; 1 November 1, 1993 http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/
[PKCS#1v2.1]	PKCS#1 v2.1: RSA Cryptography Standard RSA Laboratories; 14 June 14, 2002 http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/
[RFC 2104]	H. Krawczyk, M. Bellare, R. Canetti: HMAC: Keyed-Hashing for Message Authentication, Network Working Group, February 1997, http://www.ietf.org/rfc/rfc2104.txt, cited 15 January 2004
[RFC 2268]	R. Rivest: A Description of the RC2(r) Encryption Algorithm, Network Working Group, March 1998, http://www.ietf.org/rfc/rfc2268.txt, cited 15 January 2004

10 Appendix C – Acronyms

A.XXX	Assumption
CC	Common Criteria for Information Technology Security Evaluation (referenced to as [CC])
CEM	Common Methodology for Information Technology Security Evaluation
CGA	Certificate Generation Application
CMS	Cryptographic Message Syntax
EAL	Evaluation Assurance Level
O.XXX	Objects (Assets)
OT.XXX	Security Objective for the **TOE**
OE.XXX	Security Objective for the Environment
PP	Protection Profile
SF	Security Function
SFR	Security Functional Requirement
SOF	Strength of Function
SSCD	Secure Signature Creation Device
ST	Security Target
T.XXX	Threat
TOE	Target of Evaluation
TSC	TSF Scope of Control
TSF	**TOE** Security Functions
TSP	**TOE** Security Policy
XML	Extensible Markup Language

11 Appendix E - Definition of the Family FCS_RND

Definition of a metric for Random Numbers is not provided in any of the classes of CC part 2. Therefore the component FCD_RND.1 of the German certification scheme document AIS 31 “A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators” of BSI has been selected here.

11.1 FCS_RND generation of random numbers

Family behaviour

This family defines quality metrics for generating random numbers intended for cryptographic purposes.

Component levelling

FCS_RND.1 The generation of random numbers using TSFs requires the random numbers to meet the defined quality metrics.

Management: FCS_RND.1

No management functions are provided for.

Logging: FCS_RND.1

There are no events identified that should be auditable if FCS_RND generation of random numbers data generation is included in the PP/ST.

FCS_RND.1 Quality metrics for random numbers

Is hierarchical to: no other components.

FCS_RND.1.1 The TSFs shall provide a mechanism for generating random numbers that meet [assignment: *a defined quality metric*].

FCS_RND.1.2 The TSFs shall be able to enforce the use of TSF-generated random numbers for [assignment: *list of TSF functions*].

Dependencies: FPT_TST.1 TSF testing.