CERTIFICATION REPORT

Certification file:	**TUVIT-DSZ-CC-9222**
Product / system:	function library IAIK-JCE CC Core, Version 3.1
Product manufacturer:	Stiftung Secure Information and Communication Technologies SIC, until 2003-12-15 IAIK - Institute for applied information processing and communications - Graz university of technology Inffeldgasse 16a 8010 Graz Austria
Customer:	see above
Evaluation facility:	TÜViT, evaluation body for IT security
Evaluation report:	*Version 1.0 as of 2004-05-19* Document-number: *20603040_TÜV_050.01* Author: Volker Nies
Result:	EAL3+
Evaluation stipulations:	none
Certifier:	Dr. Silke Götze
Certification stipulations:	none

Essen, 2004-05-28

Dr. Ernst-Hermann Gruschwitz Dr. Silke Götze

Contents

Part A

Certificate and Background of the Certification

Part A presents a copy of the issued certificate and summarizes

- information about the certification body,
- the certification procedure, and
- the performance of evaluation and certification.

1 The Certificate

Deutsches IT-Sicherheitszertifikat
anerkannt vom
Bundesamt für Sicherheit in der Informationstechnik

The Certification Body of TÜV Informationstechnik GmbH
hereby certifies that the function library

IAIK-JCE CC Core, Version 3.1

of

Stiftung Secure Information and Communication Technologies SIC, Austria

until 2003-12-15

Institute for Applied Information Processing and Communications (IAIK), Austria

has been evaluated at an accredited and licensed/approved evaluation facility using the *Common Methodology for IT Security Evaluation (CEM) Part 1 Version 0.6* and *CEM Part 2 Version 1.0* for conformance to the *Common Criteria for IT Security Evaluation (CC), Version 2.1* (ISO 15408) with the following results:

SECURITY FUNCTIONALITY
Product specific Security Target
Common Criteria part 2 extended by FCS_RND.1

ASSURANCE PACKAGE
Common Criteria part 3 conformant
EAL 3 augmented by AVA_VLA.4, ADV_IMP.1, ADO_DEL.2, ADV_LLD.1, ALC_TAT.1, AVA_MSU.2

This certificate applies only to the specific version and release of the product in its evaluated configuration and in conjunction with the complete certification report. The recommendations and stipulations in the certification report must be respected. The evaluation has been conducted in accordance with the provisions of the certification scheme of TÜV Informationstechnik GmbH and the conclusions of the evaluation facility in the evaluation technical report are consistent with the evidence adduced.

The security target, against which the product has been evaluated, is part of the certification report. The rating of the strength of cryptographic mechanisms suitable for encryption and decryption is excluded from the recognition by BSI. A copy of the certificate and of the certification report is available from the product manufacturer or from the certification body.

This certificate is not an endorsement of the IT product by TÜV Informationstechnik GmbH or by any other organisation that recognises or gives effect to this certificate, and no warranty of the IT product by TÜV Informationstechnik GmbH or by any other organisation that recognises or gives effect to this certificate is either expressed or implied.

Certificate-Registration-No.
TUVIT-DSZ-CC-9222-2004

Essen, 2004-05-28 **sign. Dr. Gruschwitz**
Certification Body

TÜV Informationstechnik GmbH - Subsidiary of the RWTÜV Group • Langemarckstraße 20 • 45141 Essen, Germany
☎ +49 201 8999-580 • 🖷 +49 201 8999-555 • tuvit@tuvit.de • www.certuvit.de
accredited for IT security certifications under DAR-registration no. DAT-ZE-014/99-00 by
Deutsche Akkreditierungsstelle Technik e.V. (DATech)

2 Certification Body – CERTÜVIT

CERTÜVIT, the Certification Body of *TÜV Informationstechnik GmbH*[1] – a subsidiary of the RWTÜV Group - was established in 1998 and offers a variety of services in the context of security evaluation and validation.

CERTÜVIT was accredited in September 1999 for certification of IT security products according to ITSEC and Common Criteria by *Deutsche Akkreditierungsstelle für Informations- und Telekommunikationstechnik (Dekitz)* now *Deutsche Akkreditierungsstelle Technik e.V. (DATech), Frankfurt/Main* under DAR-registration no. DAT-ZE-014/99-00 and performs its projects under a quality management system certified against ISO 9001 by *Germanischer Lloyd, Hamburg*.

CERTÜVIT is accredited by *Bundesamt für Sicherheit in der Informationstechnik*[2] to issue the "German IT Security Certificate" which is recognised by BSI as equivalent to the "German IT Security Certificate" of BSI.

3 Specifications of the Certification Procedure

The certification body conducts the certification procedure according to the criteria laid down in the following:

- DIN EN 45011
- TÜVIT Certification Scheme
- TÜVIT Certification Conditions
- Regulations on the "German IT Security Certificate" issued by the BSI and accepted in the contract of BSI and TÜVIT as of December 2nd, 1997 (renewed on the 20th of November 2002).
- Common Criteria for Information Technology Security Evaluation (CC) part 1-3, version 2.1, August 1999.
- Common Methodology for Information Technology Security Evaluation (CEM) part 1, version 0.6, January 1997.
- Common Methodology for Information Technology Security Evaluation (CEM) part 2, version 1.0, August 1999.
- Application Notes and Interpretations of the Scheme (AIS), published by BSI.

[1] in the following termed shortly TÜVIT

[2] in the following termed shortly BSI

4 Recognition Agreements

In order to avoid multiple certification of the same product by different certification bodies a mutual recognition of IT security certificates – as far as such certificates are based on ITSEC or CC - under certain conditions was agreed. The CERTÜViT certificates are recognized by BSI – the national German certification body in international agreements – to be equivalent to its own certificates.

4.1 CC - Certificates

An arrangement (Common Criteria Arrangement) on the mutual recognition of certificates based on the CC evaluation assurance levels up to and including EAL4 was signed between the national participants of Australia and New Zealand, Austria, Canada, Finland, France, Germany, Greece, Hungary, Israel, Italy, Japan, The Netherlands, Norway, Spain, Sweden, Turkey, United Kingdom and the United States.

4.2 ITSEC/CC - Certificates

The SOGIS-Agreement on the mutual recognition of certificates based on ITSEC was signed by the national bodies of Finland, France, Germany, Greece, Italy, The Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and the United Kingdom. The arrangement on the mutual recognition of IT security certificates based on the CC was extended by these participants up to and including the evaluation assurance level EAL7.

5 Performance of Evaluation and Certification

The certification body monitors each individual evaluation to ensure uniform procedures, interpretations of the criteria, and ratings. The product IAIK-JCE CC Core, Version 3.1 has undergone the certification procedure at TÜViT certification body. It was an initial certification.

The evaluation of the product IAIK-JCE CC Core, Version 3.1 was conducted by the evaluation body for IT-security of TÜViT and concluded on May 19, 2004. The TÜViT evaluation facility is recognised by BSI.

The sponsor as well as the developer is Stiftung Secure Information and Communication Technologies and the distributor is also Stiftung Secure Information and Communication Technologies.

The certification was concluded with

- the comparability check and
- the preparation of this certification report.

This work was concluded on May 28, 2004. The confirmation of the evaluation assurance

level (EAL) only applies on the condition that

- all stipulations regarding generation, configuration and operation, as given in part B of this report, are observed,
- the product is operated – where indicated – in the environment described.

This certification report applies only to the version of the product indicated here. The validity of the certificate can be extended to cover new versions and releases of the product, provided the applicant applies for re-certification of the modified product, in accordance with the procedural requirements, and provided the evaluation does not reveal any security deficiencies.

With regard to the meaning of the evaluation assurance levels (EAL) and the strength of function (SoF), please refer to part C of this report.

6 Publication

The following Certification Results consist of pages B-1 to B-26. The product IAIK-JCE CC Core, Version 3.1 will be included in the BSI list of certified products which is published at regular intervals (e.g. in the Internet at http://www.bsi.bund.de) and the TÜViT certification lists (http://www.certuvit.de).

Further copies of this certification report may be ordered from the sponsor of the product. The certification report may also be obtained in electronic form at the internet address of CERTÜViT as stated above.

Part B

Certification Result

The following results represent a summary of

- the security target of the sponsor for the target of evaluation,
- the relevant evaluation results from the evaluation facility, and
- complementary notes and stipulations of the certification body.

Contents of the Certification Result

1 Executive Summary

1.1 Target of Evaluation and Evaluation Background

The TOE, the library IAIK-JCE CC Core, version 3.1, is a pure Java software delivered to users as part of a toolkit. This toolkit consists of a Java library in form of JAR files, documentation and demo code. The TOE provides components usable to develop applications including functionality to create and verify digital signatures, to encrypt and to decrypt data, and to generate random numbers. The sponsor, vendor and distributor is "Stiftung Secure Information and Communication Technologies SIC" Austria until 2003-12-15 "Institute for Applied Information Processing and Communications (IAIK), Austria".

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.

The TOE was evaluated against the claims of the Security Target[3] (attached in part D) by the "*evaluation body of TÜV Informationstechnik GmbH*" (TÜViT). The evaluation was completed on May 19, 2004. TÜViT's evaluation body is recognised by BSI.

1.2 Assurance Package

The TOE security assurance requirements are based entirely on the assurance components and classes defined in Part 3 of the Common Criteria (see part C of this report or [CC] Part 3 for details). The TOE meets the assurance requirements of assurance level EAL 3 augmentet by AVA_VLA.4, ADV_IMP.1, ADO_DEL.2, ADV_LLD.1, ALC_TAT.1, and AVA_MSU.2 (Evaluation Assurance Level 3+).

1.3 Strength of Functions

The TOE's strength of functions is rated "high" (SOF-high).

1.4 Functionality

The TOE's security requirements have been taken from CC part 2 extended by the component FCS_RND.1 taken from AIS 31 and can be categorized in the categories cryptographic support and user data protection:

[3] hereinafter called ST

Cryptographic operation FCS_COP.1/SHA	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***.

Cryptographic operation 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].
Cryptographic operation 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] und 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***.
Import of user data without security attributes FDP_ITC.1	The TSF shall enforce the ***JVM-Policy*** when importing user data, controlled under the SFP, from outside of the TSC. The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC. The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: ***none***.

The security functional requirements are met by suitable IT security functions realized by the TOE:

TSF	Functionality	Description
TSF.Hash	Message digest computation	The following hash algorithms are implemented: • SHA-1 hash algorithm according to FIPS PUB 180-1. • SHA-256 hash algorithm according to FIPS PUB 180-2. • SHA-384 hash algorithm according to FIPS PUB 180-2. • SHA-512 hash algorithm according to FIPS PUB 180-2. • RIPEMD-160 hash algorithm according to ISO/IEC 10118-3:1998.
TSF.Cipher	Encryption and decryption	The TOE offers functionality to decrypt and encrypt data. These functions can be subdivided into symmetric and asymmetric functions. The following symmetric algorithms are implemented: • AES data encryption and decryption in ECB/CBC/OFB/CFB/CTR Mode with 128, 192, 256 bit key size according to FIPS PUB-197. • Triple-DES data encryption and decryption in ECB/CBC/OFB/CFB Mode with 112, 168 bit key size according to FIPS PUB 46-3. • RC2 data encryption and decryption in ECB/CBC/OFB/CFB Mode with 128-1024 bit key size according to RFC2268. • ARCFOUR data encryption and decryption with 128-2048 bit key size according to [IETF-Draft-Kaukonen]. The following asymmetric algorithms are implemented: • 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. • 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.

TSF	Functionality	Description
TSF.Signature	Signature generation and verification	The TOE can be used to generate and validate digital signatures according to the following schemes: • 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 SHA-1, SHA-256, SHA-384, SHA-512, and RIPEMD-160. • 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 SHA-1, SHA-256, SHA-384, SHA-512, and RIPEMD-160.
TSF.Random	Random number generation	TSF.Random implements two random number generators: • Class K3 random number generator as defined in example E.5 of AIS20 with hash functions SHA-1, SHA-256, SHA-384, SHA-512, and RIPEMD-160. The implementation is according to example E.5 of AIS20. • Class K4 random number generator as defined in AIS20. The implementation is according to FIPS PUB 186-2 and uses with hash functions SHA-1, SHA-256, SHA-384, SHA-512, and RIPEMD-160.
TSF.MAC	Message authentication code computation	The TOE implements a HMAC, which is based on a shared secret and a secure hash function, in compliance with the following standard: • 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.

1.5 Summary of Threats and Organisational Security Policies (OSPs)

The threats countered by the TOE and the environment are the following:

Threats	Description
T.SignatureForgery	S.Attacker could forge O.Signature or recover O.PrivateKey from O.Signature.
T.DeduceData	S.Attacker could deduce O.Data from O.CipherText.
T.DeduceKey	S.Attacker could deduce O.SecretKey from O.CipherText.
T.DeduceRandomSeed	S.Attacker could deduce O.RandomSeed.
T.PredictRandomNumber	S.Attacker could predict the next generated O.RandomNumber.
T.MACForgery	S.Attacker could forge O.MAC or recover O.SecretKey.
T.HashForgery	S.Attacker could find collisions to O.Hash.

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

1.6 Special Configuration Requirements

The TOE is delivered from the manufacturer as one fixed configuration to the customer. The TOE is used in terms of developing a surrounding application which calls up the functionality provided by the TOE without changing the TOE configuration. Details on installation of the TOE can be found in "*install.html: How to install IAIK-JCE*" which is part of the HTML-documentation [GD_API] which is delivered with the TOE.

1.7 Assumptions about the Operating Environment

The assumptions about the environment of use of the TOE and about the intended usage of the TOE covers physical and procedural measures. It is assumed, that the TOE is protected against reading or modifying data by unauthorized persons or processes, that the developers and administrators are qualified to use the TOE, that IT environment provides suitable seeds for the RandomNumberGenerator and is responsible for the key management, the Java VM works according the JVM Specification and the JCE framework works according JCE 1.2.

All in all there have been 7 assumptions given, detailed in chapter 4.

1.8 Independence of the Certifier

Within the last two years, the certifier did not render any consulting- or other services for the company ordering the certification and there was no relationship between them which might have an influence on his assessment.

The certifier did not participate at any time in test procedures for the product which forms the basis of the certification.

1.9 Disclaimers

The certification results only apply to the version of the product indicated in the certificate and on the condition that all the stipulations are kept with regard to generation, configuration and operation as detailed in this certification report. This certificate is not an endorsement of the IT product by TÜV Informationstechnik GmbH or any other organisation that recognises or gives effect to this certificate, and no warranty of the IT product by TÜV Informationstechnik GmbH or any other organisation that recognises or gives effect to this certificate, is either expressed or implied.

2 Identification of the TOE

The TOE is delivered to the user in its fixed configuration, which cannot be altered by the user/administrator. The TOE is delivered on a CD-ROM as part of the IAIK-JCE toolkit. In detail the following components are delivered to the user/administrator:

No.	Type	Denotation	Way of delivery
1.	SW	TOE (IAIK-JCE CC Core): as part of a ZIP file `iaikjce31cc.zip`.	CD-ROM: distribution by public mail or a private distribution service.
2.	DOC	HTML guidance documentation [GD_HTML]: as part of a ZIP file `iaikjce31cc.zip`.	
3.	DOC	API documentation [GD_API]: as part of a ZIP file `iaikjce31cc.zip`.	
4.	DOC	Security Target [ST]: as part of a ZIP file `iaikjce31cc.zip`.	
5.	DATA	Text file `iaikjce31cc.zip.sha1` with the SHA-1 hash value over the `iaikjce31cc.zip` file.	

No.	Type	Denotation	Way of delivery
6.	DATA	SHA-1 hash value of the `iaikjce31cc.zip` file.	distribution by fax, signed e-mail, telephone or personal appearance.
7.	DOC	Guidance documentation subdocument [GD]: *IAIK – Guidance Document: Integrity Verification Guidance*, version 1.1, date 22.04.2004.	

After delivery the TOE only features one fixed configuration, which cannot be altered by the user/administrator. After delivery the integrity of the `iaikjce31cc.zip` file including the TOE has to be verified by calculating the SHA-1 hash value of this file by any independent trustworthy program that can calculate SHA-1 hash values and comparing the result with the delivered hash value of this file.

3 Security Policy

The TOE 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

It supplements the security functionality of the default Java Runtime Environment as defined in the JCA and JCE specification. Applications can access the cryptographic algorithms of the IAIK provider. Apart from the implicit policies that

- hash functions are collision resistant one-way functions
- only private key holder can create signatures
- only symmetric key holder can encrypt and decrypt data
- random numbers are statistically inconspicuous and it is practically impossible to determine predecessor or successor random numbers of a known random number sequence

which are given by those functionalities no explicit security policy was defined.

4 Assumptions and Clarification of Scope

4.1 Usage Assumptions

Assumption	Definition
A.Train	Administrators (S.Admin) are assumed to be suitably qualified to set up the system and to verify the TOE integrity.
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.

4.2 Environmental Assumptions

The specific conditions listed below are assumed to exist in the TOE environment. These assumptions include essential environmental constraints on the use of the TOE.

Assumption	Definition
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.
A.SeedManagement	SeedGeneration. The IT-Environment must provide a suitable seed for the RandomNumberGenerator. Furthermore it must ensure that the seed is kept secret.
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**.

Assumption	Definition
A.Java_Spec	Java Specification. The Java VM in use 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 the following APIs: • J2SE 1.4.x [JavaAPI.14] • J2SE 1.3.x [JavaAPI1.3] • J2SE 1.2.x [JavaAPI1.2]
A.JCE_Spec	JCE Specification. JCE framework, which is needed if Java API in use is older than version 1.4 [JavaAPI.14] (1.1.x [JavaAPI1.1], 1.2.x [JavaAPI1.2], 1.3.x [JavaAPI1.3]), works according to the JCE 1.2 [JCE1.2-REF], JCE 1.2.1 [JCE1.2.1-REF], JCE 1.2.2 [JCE1.2.2-REF] specification.

4.3 Clarification of Scope

The TOE provides components usable to develop applications including functionality to create and verify digital signatures, to encrypt and to decrypt data, and to generate random numbers. Therefore first a developer has to install and to use the TOE in order to create a surrounding application which builds the basis for the usage of the TOE by an end-user.

The person who is in charge to perform the TOE installation and TOE configuration represents the administrator of the TOE, while the developer of the surrounding application is identified as the user in terms of the person who uses the TOE for developing his application. In practice these tasks are often exercised by the same person. Due to the fact that the developer is trusted to perform security critical operations within the TOE and to set TOE parameters he incorporates not only the user role but also the administrator role.

According to assumption A.Protection, the environment must protect the TOE against side channel attacks and must assure the protection of the key material.

The TOE is delivered as a part of the IAIK-JCE toolkit which offers more functionality that is **not** part of the TOE. For instance:

- key sizes smaller than the minimum and bigger than the maximum defined by the TSF in section 1.4
- PCBC as a mode of operation for all symmetric block ciphers
- additional ciphers like DES, IDEA, or Blowfish
- more hash algorithms like MD2, MD5 or RipeMd128

- X.509 Certificate parsing and creation
- CRL parsing and creation
- OCSP protocol classes

The security functions (TSF) which are part of the TOE are listed in section 1.4.

5 Architectural Information

The TOE implements a Java Cryptographic Service Provider which is called IAIK provider.

The TOE consists of the two subsystems IAIK and UTILITIES. The first subsystem IAIK contains all modules/packages that implement JCA/JCE compliant engines and parameters for algorithms that are included in the TSF and one additional module which is responsible for registering these engines.

Subsystem IAIK		
Module Name	**Package Name**	**Module Description**
CIPHER	iaik.security.cipher	Symmetric Cipher engines; contains implementations of the TSF ciphers AES, Triple-DES, RC2 and ARCFOUR; also contains parameter implementations for these ciphers
MAC	iaik.security.mac	Message Authentication Code engines; contains the implementation of the TSF MAC algorithm HMAC
MAIN	iaik.security.provider	Contains the IAIK provider main class responsible for registering all crypto engines
MESSAGE DIGEST	iaik.security.md	MessageDigest engines; contains implementations of the TSF hash algorithms SHA-1, SHA-256, SHA-384, SHA-512 RIPEMD-160
RSA SIGNATURE	iaik.security.rsa	RSA based Signature engines; contains implementations of the TSF PKCS#1 RSA based signature algorithms
SECURE RANDOM	iaik.security.random	secureRandom engines; contains of the TSF PRNG algorithms

Subsystem IAIK		
PKCS#1	iaik.pkcs.pkcs1	PKCS#1 Cipher engine; contains the implementation for the TSF cipher RSA; also contains parameter and function implementations as used by PKCS#1 based signature and cipher engines

The second subsystem is named UTILITIES. It consists of a set of modules/packages providing a certain number of utilities required by the subsystem IAIK.

6 Documentation

The following documentation is provided with the product by the developer to the consumer:

- *IAIK - Guidance Document - Integrity Verification Guidance IAIK-JCE CC Core 3.1, Vs. 1.1 [GD]*
- *a complete HTML documentation containing the following HTML documents [GD_HTML]:*

HTML Document	**Version**
IAIK-JCE 3.1 with IAIK-JCE CC Core 3.1 API Documentation [GD_API][4]	3.1
Changes.html: IAIK-JCE Version History	56
CCCore.html: IAIK-JCE CC Core 3.1	12
Copyright.html: IAIK Java Crypto-Software Development Kit Licence Agreement	26
Demo.html: IAIK-JCE Demo Programs	12
Features.html: IAIK-JCE: Features	20
idea.html: Idea Patent Regulations	3
install.html: How to install IAIK-JCE	23

[4] API documentation related to the TOE consists of single HTML documents corresponding to the TOE Java classes that are part of the API. Each API document has the same name as the *Java* class it corresponds to (except that it is suffixed with ".html" instead of ".java") and includes the same version number as the Java class.

HTML Document	**Version**
Readme.html: IAIK-JCE 3.1 with IAIK-JCE CC Core 3.1	25
Problems.html: Problem Solving	19
QuickStart.html: Quick Start Guide for IAIK-JCE	4
Speed.html: IAIK-JCE Performance	11
Using.html: Using the IAIK-JCE	15
Version.html: IAIK-JCE Library Versions	26

7 IT Product Testing

The developer's strategy was to test the TOE against the specification of all TSF and the related subfunctions. Two basic kinds of tests were conducted:

- Known-answer tests: for these tests there exist several sets of input data and for each set the expected result is known. These tests aim at comparing the results of the TOE with expected results.
- API tests: this type of test ensures that the security functions behave according to the API specification ([JCA1.4-API], [JCE1.4-API]), the TOE API specification [GD_API] and especially the functional specification [FSP] and the guidance documentation ([GD], [GD_HTML]).

The known-answer-tests were implemented into 50 individual test classes in order to verify that the algorithms of the TSF have been implemented correctly. The API tests were implemented into 6 individual test classes to ensure that the security functions behaves according to the API specification, the functional specification and the guidance documentation.

The evaluator's objective was to test independently the functionality of the TOE systematically against the TSF description in [ST] and [FSP] as well as to verify the developer’s test results by a sample chosen from the developer's tests contained in [TD]. Therefore the entire security functionality of the TOE was tested. The evaluator tested a subset of all security functions with approximately 211 individual test scenarios. The evaluator performed all of the developer tests documented in the developers test documentation [TD] except for the tests concerning the code coverage analysis because these code coverage tests last some days or longer depending on the used platform. Therefore no further selection criteria are applied. Overall the TSF were tested systematically against the security target [ST], the functional specification [FSP] and the

high-level design [HLD]. The tests demonstrate that the security functions perform as specified.

The developer and the evaluator tests were performed on the following platforms:

- Operating system: Windows 2000 SP4,
- Processor: Intel Pentium 4,
- Clock speed: 2 GHz,
- Hard disk size: 40 GB,
- RAM: 768 MB,
- Monitor, mouse, and keyboard.

The installed Java runtime environments were:

- SUN JRE 1.1.8_010 serves as Java version 1.1.8,
- SUN JRE 1.2.2_017 serves as Java version 1.2.2,
- SUN JRE 1.3.1_10 serves as Java version 1.3.1,
- SUN JRE 1.4.0_04 serves as Java version 1.4.0,
- SUN JRE 1.4.1_06 serves as Java version 1.4.1,
- SUN JRE 1.4.2_03 serves as Java version 1.4.2.

The following test resources were used for the evaluator's testing in the TOE's development environment.

Name of the Tool	Version	Date/time	Description
Jakarta Ant	1.5.4		Ant is a Java-based build tool. Its purpose is similar to the MAKE tool known from Unix systems. Scripts for the Ant tool are XML documents. The tests use an Ant script to compile and run the tests.
Apache Xalan	2.5.2		Xalan is a XSLT processor compliant with the XSLT standard from W3C. Ant Uses Xalan to transform the test results of JUnit tests from an XML format into HTML documents.
JUnit	3.8.1		Test framework, that allows to run the tests automatically from a variety of environments.

Name of the Tool	Version	Date/time	Description
JCoverage	1.05		Code coverage analysis tool for Java.
ANT Script Build XML		01.03.2004/ 07:59	Custom ANT script to run the tests with and without Jcoverage.
SUN JCE (SunJCE and SunRsaSign provider) provider which is included in SUN JDK 1.4.2_03	SUN JDK 1.4.2_03		Reference products used for dedicated algorithms to compare the results of the TOE with the results of that reference implementation.
SUN JCE (SunJCE and SunRsaSign provider) provider which is included in SUN JDK 1.5.0 beta 1	SUN JDK 1.5.0		
GUN Crypto library	2.0.1		
Bouncy Castle crypto library	1.1.9		
Cryptix JCE		17.03.2003	

Additionally the evaluator used the following test tools and environment at the TÜViT GmbH:

- Standard-PC (Model: Compaq Deskpro, Serial-No.: 3872C782) with operating system Windows 2000 Professional SP2, processor Intel Pentium III, clock speed 450 MHz, 192 MB RAM, monitor, mouse, and keyboard.
- Software tools:

Name of the Tool	Version	Date/time	Description
OpenSSL	0.9.7c	30.09.2003	Reference product used for dedicated algorithms to compare the results of the TOE with the results of that reference implementation.
AISRandomTest		19.02.2004, 15:48	Test tool that runs the AIS20 tests.

The test results obtained for all of the performed tests turned out to be as expected. No errors or other flaws occurred with regard to the specified security functionality and mechanisms.

It is obvious that the identified vulnerabilities of the developer are not exploitable in praxis if the assumptions and security objectives for the environment contained in [ST] are regarded.

This is in the exclusive responsibility of the environment (developer, application, Java VM) of the TOE. No special penetration testing was necessary to approve or disapprove the developer vulnerability analysis, as the arguments of the developer are clear to the evaluator.

The evaluator did not find any further vulnerability in addition to those found by the developer. Therefore, no special penetration testing based on the independent vulnerability analysis was necessary.

The evaluator has reviewed the developer's vulnerability analysis and the evaluator's independent vulnerability analysis with the result that there are no exploitable vulnerabilities and no residual vulnerabilities existing for the TOE in its intended environment with regard to an attacker possessing a *high* attack potential.

8 Evaluated Configuration

The TOE is delivered in one fixed configuration and no further generation takes place. Therefore the evaluated configuration is identified by the version number:

- IAIK-JCE CC Core, Version 3.1,
  (SHA-1 hash value: a15f150a01bc019c40361c20d8b0b9f4ecc1e242)

9 Results of the Evaluation

The Evaluation Technical Report [ETR] was provided by TÜViT's evaluation body according to the requirements of the Scheme, the Common Criteria [CC], the Methodology [CEM] and the Application Notes and Interpretations of the Scheme [AIS].

The verdicts for the CC, part 3 assurance classes and components (according to EAL3 and the class ASE for the Security Target Evaluation) augmented by AVA_VLA.4, ADV_IMP.1, ADO_DEL.2, ADV_LLD.1, ALC_TAT.1, AVA_MSU.2 are summarised in the following table:

EAL3 assurance classes and components		**Verdict**
Security Target evaluation	**CC Class ASE**	PASS
TOE description	ASE_DES.1	PASS
Security environment	ASE_ENV.1	PASS
ST introduction	ASE_INT.1	PASS
Security objectives	ASE_OBJ.1	PASS
PP claims	ASE_PPC.1	n.a.[5]
IT security requirements	ASE_REQ.1	PASS
Explicitly stated IT security requirements	ASE_SRE.1	PASS
TOE summary specification	ASE_TSS.1	PASS

[5] n.a. = not applicable

Configuration Management	**CC Class ACM**	PASS
Authorisation controls	ACM_CAP.3	PASS
TOE CM coverage	ACM_SCP.1	PASS
Delivery and operation	**CC Class ADO**	PASS
Delivery procedures	ADO_DEL.2	PASS
Installation, generation, and start-up procedures	ADO_IGS.1	PASS
Development	**CC Class ADV**	PASS
Informal functional specification	ADV_FSP.1	PASS
Security enforcing high-level design	ADV_HLD.2	PASS
Implementation representation	ADV_IMP.1	PASS
Low Level Design	ADV_LLD.1	PASS
Informal correspondence demonstration	ADV_RCR.1	PASS
Guidance documents	**CC Class AGD**	PASS
Administrator guidance	AGD_ADM.1	PASS
User guidance	AGD_USR.1	PASS
Life cycle support	**CC Class ALC**	PASS
Identification of security measures	ALC_DVS.1	PASS
Tools and techniques	ALC_TAT.1	PASS
Tests	**CC Class ATE**	PASS
Analysis of coverage	ATE_COV.2	PASS
Testing: high-level design	ATE_DPT.1	PASS
Functional testing	ATE_FUN.1	PASS
Independent testing – sample	ATE_IND.2	PASS
Vulnerability assessment	**CC Class AVA**	PASS
Examination of guidance	AVA_MSU.2	PASS
Strength of TOE security function evaluation	AVA_SOF.1	PASS
Developer vulnerability analysis	AVA_VLA.4	PASS

No Protection Profile (PP) compliance claims were made in the ST. Thus, the component ASE_PPC.1 is not applicable. All other assurance components were assessed with the verdict PASS. This includes that all evaluator action elements being part of the assurance components are also assessed with PASS. Therefore, the TOE as defined in the security target is considered to be Part 3 conformant.

The security target, chapter 5 claims, that the TOE will fulfil the following TOE security functional requirements, which were exclusively taken from [CC] part 2:

Component ID	Component title
FCS_COP.1/SHA-1	Cryptographic operation
FCS_COP.1/SHA-256	Cryptographic operation
FCS_COP.1/SHA-384	Cryptographic operation
FCS_COP.1/SHA-512	Cryptographic operation
FCS_COP.1/RIPEMD-160	Cryptographic operation
FCS_COP.1/AES	Cryptographic operation
FCS_COP.1/TripleDES	Cryptographic operation
FCS_COP.1/RC2	Cryptographic operation

Component ID	Component title
FCS_COP.1/ARCFOUR	Cryptographic operation
FCS_COP.1/RSACipher	Cryptographic operation
FCS_COP.1/RSACipherOAEP	Cryptographic operation
FCS_COP.1/RSASignature	Cryptographic operation
FCS_COP.1/RSASignaturePSS	Cryptographic operation
FCS_RND.1/HashRandom	Cryptographic operation

Section 1.4 lists the security functional requirements in detail.

The evaluation performed in accordance to EAL3+ has shown that the TOE security functional requirements are correctly realised by the TOE security functions. Thus, in realising these functional requirements, it is assured that the TOE will meet the security objectives claimed in the ST.

The evaluation has shown that for TSF.Hash, TSF.MAC, and TSF.Random the strength of the TOE security function is SOF-high. TSF.Cipher is based on cryptographic algorithms, that are not subject to an SOF-claim, the minimum key size used has been found to be conform to SOF-high. TSF.Signature is based on algorithms that are published in the [BANZ] as suitable for the qualified electronic signature and therefore fulfil the requirements for SOF-high.

The sponsor must advise the certification authority about any modification of the TOE or its guidance documentation. The certification authority will then check whether the certification results are still valid and, if necessary, initiate all further steps concerning a re-evaluation. The results of the evaluation are only applicable to "*IAIK-JCE CC Core,Version 3.1*". The validity can be extended to new versions and releases of the product, provided the sponsor applies for re-certification of the modified product, in accordance with the procedural requirements, and the evaluation does not reveal any security deficiencies.

10 Evaluation stipulations, comments and recommendations

There are no recommendations & hints necessary for the user (except those provided by the guidance documents).

11 Certification stipulations and notes

There are no stipulations or notes.

12 Security Target

The security target for "*IAIK-JCE CC Core,Version 3.1*" as of 2004-05-05, version 1.2 is included in part D of this certification report.

13 Definitions

13.1 Acronyms

ADM	Administrator Guidance
AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher Block Chaining
CC	Common Criteria for Information Technology Security Evaluation (referenced to as [CC])
CEM	Common Methodology for Information Technology Security Evaluation (referenced to as [CEM])
CFB	Cipher Feedback
DES	Data Encryption Standard
EAL	Evaluation Assurance Level
ECB	Electronic Codebook
FIPS	Federal Information Processing Standard
FIPS PUB	Federal Information Processing Standard Publication
FSP	Functional Specification
GHz	Giga Hertz
HLD	High-level Design
HMAC	Hashed Message Authentication Code
HTML	Hypertext Markup Language
IAIK	Institute for Applied Information Processing and Communications
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IGS	Installation, Generation and Start-up
ISO	International Organization for Standardization
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extension
JRE	Java Runtime Environment
MAC	Message Authentication Code
OAEP	Optimal Asymmetric Encryption Padding
OFB	Output Feedback
OSP	Organisational Security Policy
PKCS	Public Key Cryptography Standards

PP	Protection Profile
PRNG	Pseudo Random Number Generator
PSS	Provably Secure Encoding Method for Digital Signatures
RAM	Random Access Memory
RC2	Ron's Code 2
RC4	Ron's Code 4
RFC	Request For Comment
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RSA	Rivest-Shamir-Adleman
SF	Security Function
SFP	Security Function Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SOF	Strength of Function
SS	Sub-system
ST	Security Target
TOE	Target Of Evaluation
TSC	TSF Scope of Control
TSF	TOE Security Functions
TSP	TOE Security Policy
USR	User Guidance
VLA	Vulnerability Analysis
VM	Virtual Machine
W3C	World Wide Web Consortium
XML	Extensible Markup Language

13.2 Glossary

Augmentation - The addition of one or more assurance component(s) from Part3 to an EAL or assurance package.

Extension - The addition to an ST or PP of functional requirements not contained in Part 2 and/or assurance requirements not contained in Part 3 of the CC.

Formal - Expressed in a restricted syntax language with defined semantics based on well-established mathematical concepts.

Informal - Expressed in natural language.

Object - An entity within the TSC that contains or receives information and upon which subjects perform operations.

Protection Profile - An implementation-independent set of security requirements for a category of TOEs that meet specific consumer needs.

Security Function - A part or parts of the TOE that have to be relied upon for enforcing a closely related subset of the rules from the TSP.

Security Target - A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.

Semiformal - Expressed in a restricted syntax language with defined semantics.

Strength of Function - A qualification of a TOE security function expressing the minimum efforts assumed necessary to defeat its expected security behaviour by directly attacking its underlying security mechanisms.

SOF-basic - A level of the TOE strength of function where analysis shows that the function provides adequate protection against casual breach of TOE security by attackers possessing a low attack potential.

SOF-medium - A level of the TOE strength of function where analysis shows that the function provides adequate protection against straightforward or intentional breach of TOE security by attackers possessing a moderate attack potential.

SOF-high - A level of the TOE strength of function where analysis shows that the function provides adequate protection against deliberately planned or organised breach of TOE security by attackers possessing a high attack potential.

Subject - An entity within the TSC that causes operations to be performed.

Target of Evaluation - An IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation.

TOE Security Functions - A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the TSP.

TOE Security Policy - A set of rules that regulate how assets are managed, protected and distributed within a TOE.

TSF Scope of Control - The set of interactions that can occur with or within a TOE and are subject to the rules of the TSP.

14 Bibliography

[AIS] Application Notes and Interpretations of the Scheme (AIS), published by BSI.

[BANZ] *Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen) vom 2. Januar 2004 – Geeignete Algorithmen zur Erfüllung der Anforderungen nach §17 Abs. 1 bis 3 SigG vom 22. Mai 2001 in Verbindung mit Anlage 1 Abschnitt I Nr. 2 SigV vom 22. November 2001,* Bundesanzeiger Nr. 30 – S. 2537-2538 vom 13. Februar 2004.

[CC] ISO/IEC 15408, Information technology – Security techniques – Evaluation criteria for IT security,
ISO/IEC 15408-1:1999 (E), Part 1: Introduction and general model
ISO/IEC 15408-2:1999 (E), Part 2: Security functional requirements
ISO/IEC 15408-3:1999 (E), Part 3: Security assurance requirements

[CEM] Common Methodology for Information Technology Security Evaluation, Part 1: Introduction and general model, version 0.6, revision 11.01.1997,
Part 2: Evaluation Methodology, version 1.0, revision August 1999

[ETR] Evaluation Technical Report, version 1.0, 2004-05-19, TÜV Informationstechnik GmbH

[FIPS PUB] *U.S. Department Of Commerce, Federal Information Processing Standards Publicatios,* U.S. Department Of Commerce, (http://csrc.nist.gov/publications)

[GD] *IAIK - Guidance Document - Integrity Verification Guidance, IAIK-JCE CC Core 3.1, 2004-04-22*

[GD_API] *IAIK-JCE 3.1 with IAIK-JCE CC Core 3.1 API Documentation*

[GD_HTML] Complete HTML guidance documentation containing single HTML documents, see chapter 6

[GD_Inst] *install.html: How to install IAIK-JCE*

[IETF-Kaukonen] *K.Kaukonen, R.Thayer: A Stream Cipher Encryption Algorithm "Arcfour"*, IETF draft (Internet Draft: draft-kaukonen-cipher-arcfour-03.txt), 14 July 1999

[ISO/IEC10118-3] *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 Dedicated hash-functions, Reference number: ISO/IEC FDIS 10118-3:2003(E)

[PKCS#1v.2.1] *PKCS#1 v2.1: RSA Cryptography Standard RSA Laboratories*; June 14, 2002 (http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/)

[PKCS#1v1.5] *PKCS#1 v1.5: RSA Encryption Standard RSA Laboratories*; 1 November 1, 1993 (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)

[RFC 2268] *R. Rivest: A Description of the RC2(r) Encryption Algorithm*, Network Working Group, March 1998 (http://www.ietf.org/rfc/rfc2268.txt)

[SigG] *Gesetz über Rahmenbedingungen für elektronische Signaturen und zur Änderung weiterer Vorschriften* vom 16.05.2001, Bundesgesetzblatt 2001, Teil I, Nr. 22, Seite 876 (ausgegeben zu Bonn am 21.05.2001)

[SigV] *Verordnung zur digitalen Signatur (Signaturverordnung – SigV)* vom 16.11.2001, Bundesgesetzblatt 2001, Teil I, Nr. 59, Seite 3074 (ausgegeben zu Bonn am 21.11.2001)

[ST] *IAIK Security Target - IAIK-JCE CC Core 3.1, Version* 1.2, 2004-05-05

Part C

Excerpts from the Criteria

The excerpts from the criteria are dealing with

- caveats on evaluation results
- assurance categorization
- evaluation assurance levels
- strength of security function
- vulnerability analysis

CC Part 1:

Conformance results (section 5.4 of CC part 1 with final interpretation 008)

„The conformance result indicates the source of the collection of requirements that is met by a TOE or PP that passes its evaluation. This conformance result is presented with respect to Part 2 (functional requirements), Part 3 (assurance requirements) and, if applicable, to a pre-defined set of requirements (e.g., EAL, Protection Profile).

The conformance result consists of one of the following:

> **Part 2 conformant** - A PP or TOE is Part 2 conformant if the functional requirements are based only upon functional components in Part 2.
>
> **Part 2 extended** - A PP or TOE is Part 2 extended if the functional requirements include functional components not in Part 2.

plus one of the following:

> **Part 3 conformant** - A PP or TOE is Part 3 conformant if the assurance requirements are based only upon assurance components in Part 3.
>
> **Part 3 extended** - A PP or TOE is Part 3 extended if the assurance requirements include assurance requirements not in Part 3.

Additionally, the conformance result may include a statement made with respect to sets of defined requirements, in which case it consists of one of the following:

> ***Package name* Conformant** - A PP or TOE is conformant to a pre-defined named functional and/or assurance package (e.g. EAL) if the requirements (functions or assurance) include all components in the packages listed as part of the conformance result.
>
> ***Package name* Augmented** - A PP or TOE is an augmentation of a predefined named functional and/or assurance package (e.g. EAL) if the requirements (functions or assurance) are a proper superset of all components in the packages listed as part of the conformance result.“

CC Part 3:

Assurance categorisation

The assurance classes, families, and the abbreviation for each family are shown in *Table 1*.

Assurance Class	Assurance Family	Abbreviated Name
Class ACM: Configuration management	CM automation	ACM_AUT
	CM capabilities	ACM_CAP
	CM scope	ACM_SCP
Class ADO: Delivery and operation	Delivery	ADO_DEL
	Installation, generation and start-up	ADO_IGS
Class ADV: Development	Functional specification	ADV_FSP
	High-level design	ADV_HLD
	Implementation representation	ADV_IMP
	TSF internals	ADV_INT
	Low-level design	ADV_LLD
	Representation correspondence	ADV_RCR
	Security policy modeling	ADV_SPM
Class AGD: Guidance documents	Administrator guidance	AGD_ADM
	User guidance	AGD_USR
Class ALC: Life cycle support	Development security	ALC_DVS
	Flaw remediation	ALC_FLR
	Life cycle definition	ALC_LCD
	Tools and techniques	ALC_TAT
Class ATE: Tests	Coverage	ATE_COV
	Depth	ATE_DPT
	Functional tests	ATE_FUN
	Independent testing	ATE_IND
Class AVA: Vulnerability assessment	Covert channel analysis	AVA_CCA
	Misuse	AVA_MSU
	Strength of TOE security functions	AVA_SOF
	Vulnerability analysis	AVA_VLA

Table 1: Assurance family breakdown and mapping

Evaluation assurance levels (chapter 6)

“The Evaluation Assurance Levels (EALs) provide an increasing scale that balances the level of assurance obtained with the cost and feasibility of acquiring that degree of assurance. The CC approach identifies the separate concepts of assurance in a TOE at the end of the evaluation, and of maintenance of that assurance during the operational use of the TOE.

It is important to note that not all families and components from Part 3 are included in the EALs. This is not to say that these do not provide meaningful and desirable assurances.

Instead, it is expected that these families and components will be considered for augmentation of an EAL in those PPs and STs for which they provide utility."

Evaluation assurance level (EAL) overview

„Table 2 represents a summary of the EALs. The columns represent a hierarchically ordered set of EALs, while the rows represent assurance families. Each number in the resulting matrix identifies a specific assurance component where applicable.

As outlined in the next section, seven hierarchically ordered evaluation assurance levels are defined in the CC for the rating of a TOE's assurance. They are hierarchically ordered inasmuch as each EAL represents more assurance than all lower EALs. The increase in assurance from EAL to EAL is accomplished by *substitution* of a hierarchically higher assurance component from the same assurance family (i.e. increasing rigour, scope, and/or depth) and from the *addition* of assurance components from other assurance families (i.e. adding new requirements).

These EALs consist of an appropriate combination of assurance components as described in chapter 2 of CC Part 3. More precisely, each EAL includes no more than one component of each assurance family and all assurance dependencies of every component are addressed.

While the EALs are defined in the CC, it is possible to represent other combinations of assurance. Specifically, the notion of "augmentation" allows the addition of assurance components (from assurance families not already included in the EAL) or the substitution of assurance components (with another hierarchically higher assurance component in the same assurance family) to an EAL. Of the assurance constructs defined in the CC, only EALs may be augmented. The notion of an "EAL minus a constituent assurance component" is not recognised by the CC as a valid claim. Augmentation carries with it the obligation on the part of the claimant to justify the utility and added value of the added assurance component to the EAL. An EAL may also be extended with explicitly stated assurance requirements."

Assurance Class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Configuration Management	ACM_AUT				**1**	1	**2**	2
	ACM_CAP	**1**	**2**	**3**	**4**	4	**5**	5
	ACM_SCP			**1**	**2**	**3**	3	3
Delivery and Operation	ADO_DEL		**1**	1	**2**	2	2	**3**
	ADO_IGS	**1**	1	1	1	1	1	1
Development	ADV_FSP	**1**	1	1	**2**	**3**	3	**4**
	ADV_HLD		**1**	**2**	2	**3**	**4**	**5**
	ADV_IMP				**1**	**2**	**3**	3
	ADV_IMT					**1**	**2**	***3***
	ADV_LLD				**1**	1	**2**	2
	ADV_RCR	**1**	1	1	1	**2**	2	**3**
	ADV_SPM				**1**	**3**	3	3
Guidance Documents	AGD_ADM	**1**	1	1	1	1	1	1
	AGD_USR	**1**	1	1	1	1	1	1
Life Cycle Support	ALC_DVS			**1**	1	1	**2**	2
	ALC_FLR							
	ALC_LCD				**1**	**2**	2	**3**
	ALC_TAT				**1**	**2**	**3**	3
Tests	ATE_COV		**1**	**2**	2	2	**3**	3
	ATE_DPT			**1**	1	**2**	2	**3**
	ATE_FUN		**1**	1	1	1	**2**	2
	ATE_IND	**1**	**2**	2	2	2	2	**3**
Vulnerability Assessment	AVA_CCA					**1**	**2**	2
	AVA_MSU			**1**	**2**	2	**3**	3
	AVA_SOF		**1**	1	1	1	1	1
	AVA_VLA		**1**	1	**2**	**3**	**4**	4

Table 2: Evaluation assurance level summary

Evaluation assurance level 1 (EAL1) - functionally tested

“EAL1 is applicable where some confidence in correct operation is required, but the threats to security are not viewed as serious. It will be of value where independent assurance is required to support the contention that due care has been exercised with respect to the protection of personal or similar information.

EAL1 provides an evaluation of the TOE as made available to the customer, including independent testing against a specification, and an examination of the guidance documentation provided. It is intended that an EAL1 evaluation could be successfully conducted without assistance from the developer of the TOE, and for minimal outlay. An evaluation at this level should provide evidence that the TOE functions in a manner consistent with its documentation, and that it provides useful protection against identified threats.“

Evaluation assurance level 2 (EAL2) - structurally tested

“EAL2 requires the co-operation of the developer in terms of the delivery of design information and test results, but should not demand more effort on the part of the developer than is consistent with good commercial practice. As such it should not require a substantially increased investment of cost or time.

EAL2 is therefore applicable in those circumstances where developers or users require a low to moderate level of independently assured security in the absence of ready availability of the complete development record. Such a situation may arise when securing legacy systems, or where access to the developer may be limited.“

Evaluation assurance level 3 (EAL3) - methodically tested and checked

“EAL3 permits a conscientious developer to gain maximum assurance from positive security engineering at the design stage without substantial alteration of existing sound development practices.

EAL3 is applicable in those circumstances where developers or users require a moderate level of independently assured security, and require a thorough investigation of the TOE and its development without substantial re-engineering.“

Evaluation assurance level 4 (EAL4) - methodically designed, tested, and reviewed

“EAL4 permits a developer to gain maximum assurance from positive security engineering based on good commercial development practices which, though rigorous, do not require substantial specialist knowledge, skills, and other resources. EAL4 is the highest level at which it is likely to be economically feasible to retrofit to an existing product line.

EAL4 is therefore applicable in those circumstances where developers or users require a moderate to high level of independently assured security in conventional commodity TOEs and are prepared to incur additional security-specific engineering costs.“

Evaluation assurance level 5 (EAL5) - semiformally designed and tested

“EAL5 permits a developer to gain maximum assurance from security engineering based upon rigorous commercial development practices supported by moderate application of specialist security engineering techniques. Such a TOE will probably be designed and developed with the intent of achieving EAL5 assurance. It is likely that the additional costs attributable to the EAL5 requirements, relative to rigorous development without the application of specialised techniques, will not be large.

EAL5 is therefore applicable in those circumstances where developers or users require a high level of independently assured security in a planned development and require a rigorous development approach without incurring unreasonable costs attributable to specialist security engineering techniques."

Evaluation assurance level 6 (EAL6) - semiformally verified design and tested

"EAL6 permits developers to gain high assurance from application of security engineering techniques to a rigorous development environment in order to produce a premium TOE for protecting high value assets against significant risks.

EAL6 is therefore applicable to the development of security TOEs for application in high risk situations where the value of the protected assets justifies the additional costs."

Evaluation assurance level 7 (EAL7) - formally verified design and tested

"EAL7 is applicable to the development of security TOEs for application in extremely high risk situations and/or where the high value of the assets justifies the higher costs. Practical application of EAL7 is currently limited to TOEs with tightly focused security functionality that is amenable to extensive formal analysis."

Strength of TOE security functions (AVA_SOF)

AVA_SOF Strength of TOE security functions

"Strength of function analysis addresses TOE security functions that are realised by a probabilistic or permutational mechanism (e.g. a password or hash function). Even if such functions cannot be bypassed, deactivated, or corrupted, it may still be possible to defeat them by direct attack. A level or a specific metric may be claimed for the strength of each of these functions. Strength of function analysis is performed to determine whether such functions meet or exceed the claim. For example, strength of function analysis of a password mechanism can demonstrate that the password function meets the strength claim by showing that the password space is sufficiently large."

Three levels exist: SOF-basic, SOF-medium, and SOF-high.

Vulnerability analysis (AVA_VLA)

AVA_VLA Vulnerability analysis

"Vulnerability analysis consists of the identification of flaws potentially introduced in the different refinement steps of the development. It results in the definition of penetration tests through the collection of the necessary information concerning: (1) the completeness of the TSF (does the TSF counter all the postulated threats?) and (2) the dependencies between all security functions. These potential vulnerabilities are assessed through penetration testing to determine whether they could, in practice, be exploitable to compromise the security of the TOE."

Application notes

"A vulnerability analysis is performed by the developer in order to ascertain the presence of security vulnerabilities, and should consider at least the contents of all the TOE deliverables including the ST for the targeted evaluation assurance level. The developer is required to document the disposition of identified vulnerabilities to allow the evaluator to make use of that information if it is found useful as a support for the evaluator's independent vulnerability analysis.

The intent of the developer analysis is to confirm that no identified security vulnerabilities can be exploited in the intended environment for the TOE and that the TOE is resistant to obvious penetration attacks.

Obvious vulnerabilities are considered to be those that are open to exploitation that requires a minimum of understanding of the TOE, skill, technical sophistication, and resources. These might be suggested by the TSF interface description. Obvious vulnerabilities include those in the public domain, details of which should be known to a developer or available from an evaluation authority.

Performing a search for vulnerabilities in a systematic way requires that the developer identify those vulnerabilities in a structured and repeatable way, as opposed to identifying them in an ad-hoc fashion. The associated evidence that the search for vulnerabilities was systematic should include identification of all TOE documentation upon which the search for flaws was based.

Independent vulnerability analysis goes beyond the vulnerabilities identified by the developer. The main intent of the evaluator analysis is to determine that the TOE is resistant to penetration attacks performed by an attacker possessing a low (for AVA_VLA.2), moderate (for AVA_VLA.3) or high (for AVA_VLA.4) attack potential. To accomplish this intent, the evaluator first assesses the exploitability of all identified vulnerabilities. This is accomplished by conducting penetration testing. The evaluator should assume the role of an attacker with a low (for AVA_VLA.2), moderate (for

AVA_VLA.3) or high (for AVA_VLA.4) attack potential when attempting to penetrate the TOE. Any exploitation of vulnerabilities by such an attacker should be considered by the evaluator to be “obvious penetration attacks” (with respect to the AVA_VLA.*.2C elements) in the context of the components AVA_VLA.2 through AVA_VLA.4.”

Part D

Security Target

Attached is the Security Target for IAIK-JCE CC Core 3.1

Author: Institute for applied information processing and communications

Date: 2004-05-05

Version: 1.2

IAIK – Security Target

Version 1.2

IAIK-JCE CC Core 3.1

05 May 2004

Table of Contents:

List of Tables

List of Figures

1 ST Introduction

1.1 ST Identification

Title:	IAIK-JCE CC Core Security Target
Version:	1.2
Date:	05 May 2004
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:	3.1
Assurance level:	EAL 3+ The **TOE** meets the assurance requirements of assurance level EAL 3 augmented by AVA_VLA.4, ADV_IMP.1, ADO_DEL.2, ADV_LLD.1, ALC_TAT.1 and AVA_MSU.2.
Strength of functions:	The **TOE**s strength of functions is rated high (SOF high).
Evaluation Body:	TÜV Informationstechnik GmbH Langemarckstraße 20 45141 Essen, Germany
TOE documentation:	HTML Documentation - IAIK-JCE 3.1 with IAIK-JCE CC Core 3.1: Readme.html, File Revision 25 and linked documents IAIK – Guidance Document, Integrity Verification Guidance Version 1.1, 2004-04-22

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 ST is CC part 2 [CC2] extended (by FCS_RND.1) and CC Part 3 [CC3] conformant. The Evaluation Assurance Level is EAL3 augmented by AVA_VLA.4, ADV_IMP.1, ADO_DEL.2, ADV_LLD.1, ALC_TAT.1, AVA_MSU.2.
This ST does not claim conformance with any Protection Profile.

2 TOE Description

2.1 Product type

The **TOE** is pure Java software 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 TOEs cryptographic algorithms through the JCA/JCE framework.

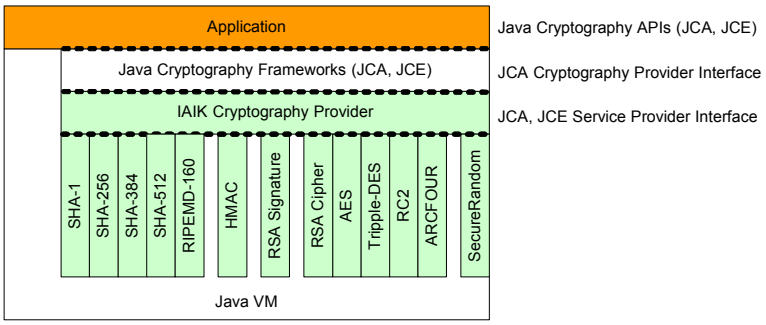

Figure 1: The TOE and its environment

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) 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 add 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 which 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 qualified electronic signatures as defined in the legislation of the European Union [EU_directive], [SigG] and [SigV]. However, for the generation of a qualified electronic signature, it may be required to use an external secure signature creation device (SSCD) for the private key operation. The **TOE** can calculate the hash value in this case but the incorporation of the SSCD is up to the application.
Most of the signature algorithms support smaller key sizes as well, but they are not suitable for use in an environment which 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. Moreover, it ensures that key material is protected and not revealed to any other application or other entities.
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 cipher 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 which 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. Two use cases can be identified:

- Advanced and Qualified Electronic Signatures.
  The **TOE** is used together with a secure signature creation device to create qualified electronic signatures or advanced electronic signatures. In this case, the **TOE** is used only to calculate the hash of the data to be signed (and only if the SSCD 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 SSCD 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 which is not part of the evaluation. For example:

- The **TOE** supports more key sizes than the minimum and maximum key size which are described in chapter 2.3.3 and 2.3.4 for RSA-Signatures and RSA-Encryption. The maximum key size depends on the system resources only.
- The **TOE** also supports PCBC as mode of operation for all symmetric block ciphers.

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

- Additional ciphers like DES, IDEA, or Blowfish
- More hash algorithms like MD2, MD5 or RipeMd128

- X.509 Certificate parsing and creation
- CRL parsing and creation
- OCSP protocol classes

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 Java only and runs in the same instance of the Java VM as the environment. All components communicate by Java 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], [JCE1.2.2-REF] or [JCE1.4-REF])
- JVM Specification 1.2 [JVMSpec2] with one of the following APIs:
  - J2SE 1.4.x [JavaAPI1.4]
  - J2SE 1.3.x [JavaAPI1.3] and JCE 1.2.x ([JCE1.2-REF], [JCE1.2.1-REF], [JCE1.2.2-REF] or [JCE1.4-REF])
  - J2SE 1.2.x [JavaAPI1.2] and JCE 1.2.x ([JCE1.2-REF], [JCE1.2.1-REF], [JCE1.2.2-REF] or [JCE1.4-REF])

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

2.4 Qualified Electronic Signatures

The **TOE** is aimed to be compliant with the requirement specified for products for qualified electronic signatures in the German Digital Signature Act [SigG] § 17 and the Digital Signature Ordinance [SigV] § 15.
If the application attempts to generate a qualified electronic signature, it may use the **TOE** to calculate the hash value over the signed data. After receiving the hash value from the **TOE**, the application forwards this hash value to a SSCD. The **TOE** does not communicate with the SSCD. This is the job of the application.
The **TOE** specifically supports several algorithms which are relevant with respect to qualified signatures [SigG-Alg]. The relevant hash functions are:

- SHA-1 hash function
- SHA-256
- SHA-384
- SHA-512
- RIPEMD-160 hash function

The relevant signature algorithms for qualified signatures are:

- RSA according to PKCS#1 v1.5
- RSA-PSS according to PKCS#1 v2.1

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.	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 Java VM in use 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 the following APIs: • J2SE 1.4.x [JavaAPI.14] • J2SE 1.3.x [JavaAPI1.3] • J2SE 1.2.x [JavaAPI1.2]	OE.ExecutionEnvironment
A.JCE_Spec	JCE Specification. JCE framework, which is needed if Java API in use is older than version 1.4 [JavaAPI.14] (1.1.x [JavaAPI1.1], 1.2.x [JavaAPI1.2], 1.3.x [JavaAPI1.3]), works according to the JCE 1.2 [JCE1.2-REF], JCE 1.2.1 [JCE1.2.1-REF], 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 3 Threats

3.3 Organization Security Policies

There are no organisational 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. Since the current evaluation level EAL3+, the attacker has a high level potential attack and no time limit.

Table 5 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 7 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 9 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. (see ADO_DEL.2 in chap. 6.2)	A.Train
OE.EnvironmentIntegrity	Access Protected. The Environment shall ensure that only S.Application has access to the **TOE**.	A.Protection
OE.KeyProtection	Key Protection. The Environment must protect the keys from unauthorized access.	A.KeyManagement
OE.CorrectKeys	Correct Keys. The Environment must provide well formed and valid keys to the **TOE**.	A.KeyManagement
OE.SuitableSeed	Suitable Seed. The Environment must provide a suitable seed to the TOE.	A.SeedManagement
OE.SeedProtection	Seed Protection. The Environment must protect the seed from unauthorized access.	A.SeedManagement
OE.ExecutionEnvironment	Execution Environment. The Environment 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 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 11 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***.

Cryptographic operation 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].

Cryptographic operation 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 access to any copies of key material, 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.2 ADO_IGS.1

ADV	ADV_FSP.1 ADV_HLD.2 ADV_RCR.1 ADV_IMP.1 ADV_LLD.1
AGD	AGD_ADM.1 AGD_USR.1
ALC	ALC_DVS.1 ALC_TAT.1
ATE	ATE_COV.2 ATE_DPT.1 ATE_FUN.1 ATE_IND.2
AVA	AVA_MSU.2 AVA_SOF.1 AVA_VLA.4

Table 13 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)

Detection of modification ADO_DEL.2

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

5.2.3 Development (ADV)

Informal functional specification (ADV_FSP.1)

Subset of the implementation of the TSF (ADV_IMP.1)

Security enforcing high-level design (ADV_HLD.2)

Descriptive low-level design (ADV_LLD.1)

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)

Well-defined development tools (ALC_TAT.1)

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)

Validation of analysis (AVA_MSU.2)

Strength of TOE security function evaluation (AVA_SOF.1)

Highly resistant (AVA_VLA.4)

5.3 Security Requirements for the Environment

5.3.1 General Requirements for the Environment

R.EnvironmentIntegrity
The Environment shall ensure that only S.Application has access to the **TOE**.

R.KeyProtection
The Environment must protect the keys from unauthorized access.

R.CorrectKeys
The Environment must provide well formed and valid keys to the **TOE**.

R.SuitableSeed:
The TSF 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 must protect the seed from unauthorized access.

R.ExecutionEnvironment
The Environment must provide an execution environment that meets the requirements (see chap. 2.3.7).

R.EnvironmentProtection
The Environment 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: The IT Environment is the S.JavaVM and/or the S.Application.

Subset residual information protection (FDP_RIP.1)

FDP_RIP.1.1

The TSF 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 TSF 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 TSF 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 47-3***.

FCS_CKM.1/RC2

The TSF 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 TSF 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 TSF 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 TSF 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 TSF 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 TSF 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 TSF 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 TSF 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):
A pure Java implementation of the ***SHA-1*** hash algorithm according to ***FIPS PUB 180-1***.

FCS_COP.1/SHA-256 (SOF-high):
A pure Java implementation of the ***SHA-256*** hash algorithm according to ***FIPS PUB 180-2***.

FCS_COP.1/SHA-384 (SOF-high):
A pure Java implementation of the ***SHA-384*** hash algorithm according to ***FIPS PUB 180-2***.

FCS_COP.1/SHA-512 (SOF-high):
A pure Java implementation of the ***SHA-512*** hash algorithm according to ***FIPS PUB 180-2***.

FCS_COP.1/RIPEMD-160 (SOF-high):
A pure Java implementation of the ***RIPEMD-160*** hash algorithm according to ***ISO/IEC 10118-3:1998***.

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:

A pure Java 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***.

FCS_COP.1/TripleDES:
A pure Java 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***.

FCS_COP.1/RC2:
A pure Java implementation of ***RC2 data encryption and decryption*** in ***ECB/CBC/OFB/CFB Mode*** with ***128-1024*** bit key size according to ***RFC2268***.

FCS_COP.1/ARCFOUR:
A pure Java implementation of ***ARCFOUR data encryption and decryption*** with ***128-2048*** bit key size according to **[IETF-Draft-Kaukonen]**.

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:
A pure Java 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***.

FCS_COP.1/RSACipherOAEP:
A pure Java 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***.

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 pure Java 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***.

FCS_COP.1/RSASignaturePSS:
A pure Java 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***.

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):
A pure Java implementation of a class K3 ***secure random number generator*** as defined in AIS20. **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):
A pure Java implementation of a class K4 ***secure random number generator*** as defined in AIS20**. 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):
A pure Java 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***.

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.2	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. The TOE itself, which is a jar archive containing the compiled Java code, is signed, as required by the JCE specification [JCE1.4-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.
	ADO_IGS.2	
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), HLD (high level design), LLD (low level design) and IMP (implementation). It stepwise refines the design by leading from the JCA/JCE API (FSP) to the JCE/JCA SPI (HLD) to TOE subsystems (HLD), modules (LLD) and final classes (IMP) which are presented at Java Source Code level.
	ADV_IMP.1	The IMP (Implementation Representation) describes the structure of the TOE source code. In addition it provides information about how to compile the source code.
	ADV_LLD.1	The LLD (Low Level Design) discusses the modules of the TOE subsystems and presents all external (to the environment and other subsystems) and internal interfaces (among the modules of a subsystem) module interfaces. The presentation is based on the Javadoc output of the corresponding classes. The LLD also shows how the several modules depend on each other.
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.
	ALC_TAT.1	The IAIK-JCE CC Core library is written in the Java programming language in a code fully compatible to version 1.1. Java is a well defined programming language. Critical constructs, such as threads or constructs from the Reflection API, are not used within the IAIK-JCE CC Core library.
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.2	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 systematic analysis of the TOE for vulnerabilities. The attacker is assumed to have a high attack potential an practically unlimited time.
	AVA_SOF.1	
	AVA_VLA.4	

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.Cipher. 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.Cipher. 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 15 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 17 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 19 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 21 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 qualified electronic signatures as defined in the legislation of the European Union [EU_directive], [SigG] and [SigV] the selected evaluation level is EAL3 augmented by AVA_VLA.4, ADV_IMP.1, ADO_DEL.2, ADV_LLD.1, ALC_TAT.1 and AVA_MSU.2 and the selected strength of functions is high (SOF-high).

8.3 TOE Summary Specification Rationale

8.3.1 TOE Security Functions Rationale

The security functions of the **TOE** reach SOF-high.
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 his attacks. The attacker is assumed to use equipment which 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-	128 bit	FCS_COP.1/HMAC	high

	1, SHA-256, SHA 384, SHA 512, RipeMD-160		FDP_ITC.1	

Table 22 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 FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/SHA-256	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/SHA-384	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/SHA-512	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RIPEMD-160	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/AES	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/TripleDES	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RC2	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/ARCFOUR	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RSACipher	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RSACipherOAEP	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RSASignature	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_COP.1/RSASignaturePSS	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2, FCS_CKM.4
FCS_CKM.4/AES	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/TripleDES	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/RC2	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/ARCFOUR	[FDP_ITC.1 or FCS.CKM.1],

	FMT_MSA.2
FCS_CKM.4/RSACipher	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/RSACipherOAEP	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/RSASignature	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/RSASignaturePSS	[FDP_ITC.1 or FCS.CKM.1], FMT_MSA.2
FCS_CKM.4/HMAC	[FDP_ITC.1 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
Assurance Requirements	
ACM_CAP.3	ACM_SCP.1, ALC_DVS.1
ACM_SCP.1	ACM_CAP.3
ADO_IGS.1	AGD_ADM.1
ADV_FSP.1	ADV_RCR.1
ADV_HLD.2	ADV_FSP.1, ADV_RCR.1
AGD_ADM.1	ADV_FSP.1
AGD_USR.1	ADV_FSP.1
ATE_COV.2	ADV_FSP.1, ATE_FUN.1
ATE_DPT.1	ADV_HLD.1, ATE_FUN.1
ATE_IND.2	ADV_FSP.1, AGD_ADM.1, AGD_USR.1, ATE_FUN.1
AVA_MSU.2	ADO_IGS.1, ADV_FSP.1, AGD_ADM.1, AGD_USR.1

AVA_SOF.1	ADV_FSP.1, ADV_HLD.1
AVA_VLA.4	ADV_FSP.1, ADV_HLD.2, AGD_ADM.1, AGD_USR.1

Table 23 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.

- **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.

- **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 IT 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 IT 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.

- **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 IT 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 IT 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.

- **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 IT 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 IT 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.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

- **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 initialisation:**

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

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 25 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) 15408:1999, Common Criteria for Information Technology Security Evaluation (Comprising Parts 1-3, [CC1], [CC2], [CC3]CC 2.1), Common Criteria Implementation Board (CCIB) and the International Standards Organization (ISO), JTC1/SC27/WG3 available online at http://www.commoncriteria.de, cited 15 January 2004
[CC1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and General Model CCIMB-99-031, Version 2.1, August 1999, Common Criteria Implementation Board (CCIB) available online at http://www.commoncriteria.de, cited 15 January 2004
[CC2]	Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Requirements CCIMB-99-032, Version 2.1, August 1999, Common Criteria Implementation Board (CCIB) available online at http://www.commoncriteria.de, cited 15 January 2004
[CC3]	Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Requirements CCIMB-99-033, Version 2.1, August 1999, Common Criteria Implementation Board (CCIB) available online at http://www.commoncriteria.de, cited 15 January 2004
[CRYPTO SPEC]	Java Cryptography Architecture, API Specification & Reference, SUN Microsystems, Inc. http://java.sun.com/security/index.html, October 2003, cited 15 January 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, cited 15 January 2004
[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, http://csrc.nist.gov/publications/fips/fips180-1/fip180-1.pdf, cited 15 January 2004
[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, cited 15 January 2004
[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, cited 15 January 2004
[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 2003Dedicated hash-functions, Reference number: ISO/IEC FDIS 10118-3:2003(E), Final Draft: http://www.ncits.org/ref-docs/FDIS_10118-3.pdf, cited 15 January 2004
[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, cited 15 January 2004
[JavaAPI1.2]	Java 2 Platform, Standard Edition, v1.2.2 API Specification, SUN Microsystems, Inc., 1999, http://java.sun.com/products/jdk/1.2/docs/api/index.html, cited 15 January 2004
[JavaAPI1.3]	Java 2 Platform, Standard Edition, v 1.3.1 API Specification, SUN Microsystems, Inc., 2001, http://java.sun.com/j2se/1.3/docs/api/index.html, cited 15 January 2004
[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/, cited 15 January 2004
[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
[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
[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
[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, cited 15 January 2004
[PKCS#1v1.5]	PKCS#1 v1.5: RSA Encryption Standard RSA Laboratories; 1 November 1, 1993 http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/, cited 15 January 2004
[PKCS#1v2.1]	PKCS#1 v2.1: RSA Cryptography Standard RSA Laboratories; 14 June 14, 2002 http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/, cited 15 January 2004
[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
[SigG]	Gesetz ueber Rahmenbedingungen fuer elektronische Signaturen und zur Aenderung weiterer Vorschriften, BGBl. I, S. 876, the German Bundestag, 16 Mai 2001 (German Digital Signature Act), 16. Mai 2001 http://www.bmwa.bund.de/Navigation/Service/Gesetze/rechtsgrundlagen-informationsgesellschaft.html**Fehler! Hyperlink-Referenz ungültig.**, cited 15 January 2004
[SigG-Alg]	Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Uebersicht ueber geeignete Algorithmen), Regulierungsbehoerde für Telekommunikation und Post, 13 February 2004 in Bundesanzeiger Nr. 30, S. 2537-2538
[SigV]	Verordnung zur elektronischen Signatur, BGBl. I S. 3074, the German Government, 16 November 2001 (Digital Signature Ordinance [(Signaturverordnung – SigV])), 16. November 2001 http://www.bmwa.bund.de/Navigation/Service/Gesetze/rechtsgrundlagen-informationsgesellschaft.html, 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.