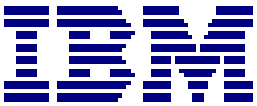

IBM Global Security Kit (GSKit) 8.0.14
Security Target

Version:	3.5
Status:	Released
Last Update:	2012-03-27

Trademarks

atsec is a trademark of atsec GmbH.

IBM, the IBM logo, GSKit, iKeyman and ICC are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Sun Solaris is a trademark of Sun Microsystems, Inc., in the United States, other countries, or both.

Microsoft Windows 95, Microsoft Windows 98, Microsoft Windows ME, Microsoft Windows NT, Microsoft Windows 2000, Windows 2000 Professional and Advanced Server, and Microsoft Windows XP are trademarks of Microsoft in the United States, other countries, or both.

HP-UX is a trademark of Hewlett Packard, in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds.

Other companies, products, and service names may be trademarks or service marks of others.

Legal Notice

This document is provided AS IS with no express or implied warranties. Use the information in this document at your own risk.

This document may be reproduced or distributed in any form without prior permission provided the copyright notice is retained on all copies. Modified versions of this document may be freely distributed provided that they are clearly identified as such, and this copyright is included intact.

Copyright © 2012 by atsec information security and IBM Corporation

Revision History

Version	Author(s)	Date	Changes to Previous Revision	Application Notes
3.5	Ochel	2012-03-27	Removed draft note / confidentiality label / internal document history, and updated copyright notice, for release – no content/version change!	

Copyright © 2012 by atsec information security and IBM Corporation

Table of Contents

Copyright © 2012 by atsec information security and IBM Corporation

Copyright © 2012 by atsec information security and IBM Corporation

List of Tables

List of Figures

1 Introduction

This Security Target is for the re-evaluation of a previously evaluated product, GSKit 7.0.4.11 (NIAP's VID 10180).

1.1 Security Target Identification

Title: IBM Global Security Kit (GSKit) 8.0.14 Security Target

Version: 3.5

Status: Released

Date: 2012-03-27

Sponsor: IBM Corporation

Developer: IBM Corporation

Keywords: GSKit, TLS, ICC, OCSP, MSCAPI, MS CNG, PKCS#11, PKCS#12.

The TOE comprises the Global Security Kit (GSKit) Version 8.0.14.21, including the SSL API and Key Management API and CLI. GSKit encapsulates the IBM Crypto for C (ICC) Version 8.0 as an algorithm factory.

1.2 TOE Overview

This security target documents the security characteristics of the Global Security Kit (GSKit). GSKit is a set of tools and C/C++ programming interfaces that can be used to add secure channels using the TLS protocol to TCP/IP applications (products). It provides the cryptographic functions, the protocol implementation, and key generation and management functionality for this purpose. GSKit ships only compiled object code and header files. The GSKit binaries are available for a number of platforms, and the evaluated version of GSKit is available on IBM AIX, RedHat Enterprise Linux, Sun Solaris, and Microsoft Windows (for more information on underlying platforms, refer to section 1.3.4.4).

GSKit is only for IBM internal use and is not offered for sale as a standalone product, i.e., it is designed for the use in other IBM products only.

The TOE consists of GSKit, providing the following security functionality:

- Secure channels

  The TOE allows consuming applications to implement TLS functionality. (The SSL protocol versions supported by GSKit are not available in the evaluated configuration). GSKit supports both TLS client and server functionality.

  The TLS functionality is offered via an API (called SSL API) for TLSv1, 1.1, and 1.2 (as defined in [TLSv1], [TLSv1.1], and [TLSv1.2], with [TLS_AES] support for TLSv1 and 1.1, and with certain extensions from [RFC6066]).

  The TOE supports a wide variety of cipher suites (identified in section 1.3.2.1) for the encryption of payload, enforces server authentication, and optionally can enforce client authentication; including certificate validation with optional revocation checking via CRLs and/or OCSP.

  The TOE can query OCSP responders in the operational environment using OCSP as defined in [RFC2560] and [RFC6277], or using the lightweight OCSP profile defined in [RFC5019]. The TOE can also retrieve and validate X.509 version 1 and 2 CRLs from the operational environment using LDAP, flat files or HTTP.

- Key and certificate generation and management

  The TOE implements both a key management API and command line interface (CLI) to generate keys and certificate requests, and manage (import, export, define the trust status, etc.) keys and certificates. Key data (i.e. keys, certificates, and related information) is stored in a so-called keystore, a file stored in the operational environment. The TOE ensures by cryptographic means that the integrity and, where appropriate, the confidentiality of the data stored in the keystore is protected. Alternatively, PKCS#11 devices or MSCAPI/MSCNG cryptographic service providers can be used for key and certificate storage and for performing cryptographic primitives.

Copyright © 2012 by atsec information security and IBM Corporation

The TOE is capable of certificate generation, but not of certificate management (including the issuance of CRLs, management of revocation status, etc.) for PKIs in any larger scale. The evaluated configuration assumes that the operational environment will manage (and provide, where appropriate) certificates for the operation of GSKit within a proper and trustworthy certificate management infrastructure.

- Self-tests

  The cryptographic module within GSKit implements self-tests as required by [FIPS140-2].

GSKit encapsulates the IBM Crypto for C (ICC) cryptographic software module. ICC Version 8.0 has been validated under the Federal Information Processing Standard (FIPS) 140-2 [FIPS140-2] for an overall Security level 1 [FIPS140-2], with certificate number 1433. The module provides a variety of FIPS 140-2 validated cryptographic algorithms, as well as some algorithms that are not standardized by the cryptographic algorithm validation program. For a complete list of algorithms that are available and used in the evaluated configuration, refer to section 1.3.2.2.

The TOE must be operated in GSKit's FIPS and CC mode. In FIPS and CC mode, GSKit's functions are restricted so that:

- Only the use of TLS, not SSL, is possible.
- Only TLS ciphersuites whose cipherspec parts consist of cryptographic algorithms that are FIPS-approved (as listed in Annex A of FIPS 140-2 [FIPS140-2]) and/or NIST-recommended can be used. (Also, the cryptographic algorithms used for key establishment are approved as per Annex D and the Implementation Guidance [FIPS140IG].)
- Only a FIPS-approved and/or NIST-recommended random number generator is used.
- Server authentication is mandatory. "Total anonymity mode" for TLS is not supported.

The product that deploys the TOE must initialize and use the TOE in FIPS mode and in CC mode.

Operating systems that the TOE (or, the application consuming the TOE) runs on have to be configured to prevent remote login (by disabling all services that offer remote login) since this is required by the FIPS 140-2 security policy for ICC [ICCSEC]. Furthermore, the operational environment must be configured in a way that the TOE binaries, the services offered, TSF data, and the keystore are protected against unauthorized access. If an external (hardware or software) cryptographic module in the operational environment is used instead of ICC, the operational environment must ensure that the module is FIPS 140-1 or 140-2 validated and authenticates GSKit before allowing access to cryptographic material and functions.

If the TOE is configured to use CRLs, the operational environment must provide an LDAP or HTTP server or flat files for CRL retrieval. If the TOE is configured to use OCSP, an OCSP responder must be provided in the operational environment. The operational environment is responsible for ensuring that connections to these services are appropriately protected against replay and denial of service attacks, and that they provide reliable and trustworthy revocation information.

As far as CCEVS's technology types for CC evaluated products are concerned, the GSKit library is of the type "Miscellaneous" (see also the discussion in section 1.3.1).

1.3 TOE Description

The TOE consists of the IBM Global Security Kit (GSKit).

The TOE's main function is to provide a secure channel to another trusted IT-product. This is shown in Figure 1. Optional components have been drawn with dotted lines and white components are part of the operational environment. The TOE components have been colored dark grey and the box surrounding them shows the TOE boundary. The secure channel set up by the TOE is depicted as a pipe named TLS.

Copyright © 2012 by atsec information security and IBM Corporation

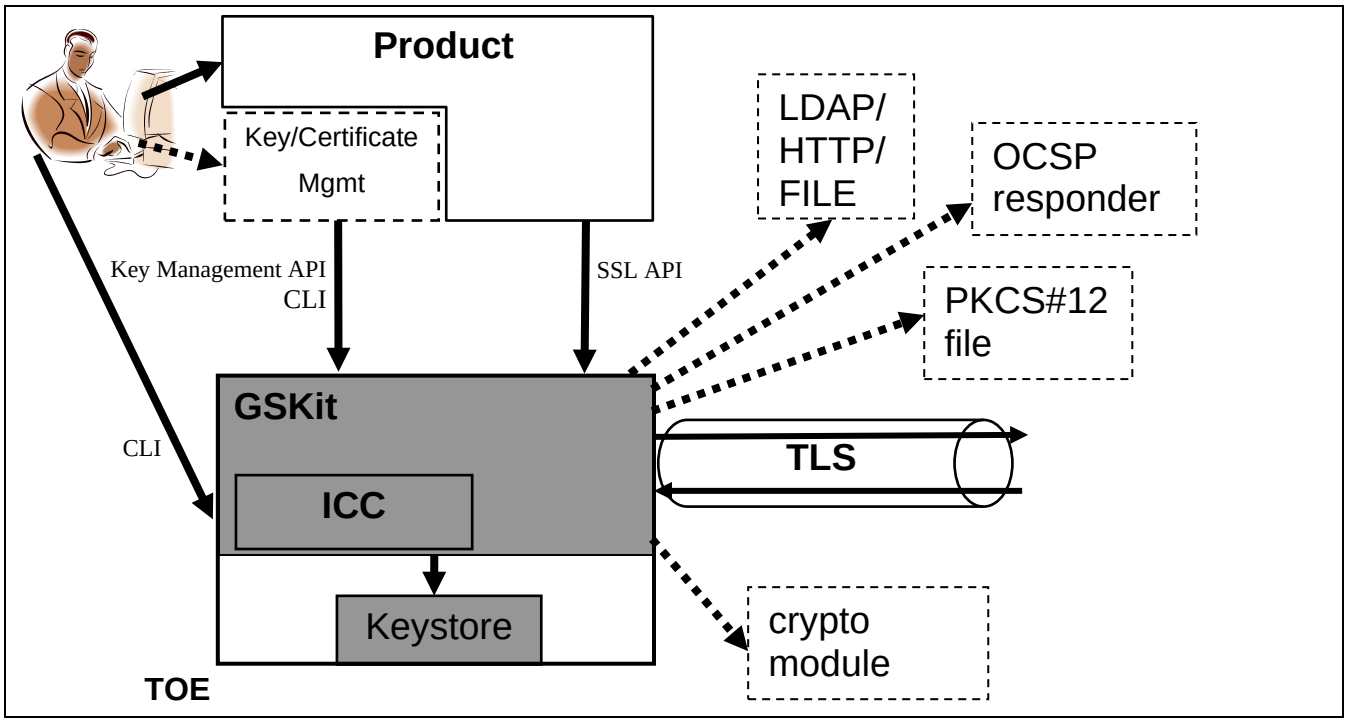

Figure 1*: GSKit Overview*

The following interfaces are offered to the user:

- SSL API: this is the main API of the product. For the evaluated configuration of the TOE, only secure connections using TLS are supported (this is enforced by operating the TOE in FIPS mode).
- Key management API and command line interface (CLI): through this API and the CLI, key and certificate generation and management functionality (like validation or deletion of certificates) can be accessed. A product that uses the TOE shall access the keystore through these interfaces only.

The TOE encapsulates ICC for software-based cryptographic functions and key generation.

Certificate validation may be done with the help of OCSP or CRLs. If CRLs are to be used for certificate validation, LDAP or HTTP resources or a flat file containing the revocation list must be provided by the operational environment. Likewise, for the usage of OCSP, an OCSP responder must be provided by the environment.

Alternatively to using the cryptographic and key storage mechanisms implemented in the TOE, the TOE is able to use cryptographic modules in the operational environment via PKCS#11 and, on Windows, via the MSCAPI/MSCNG. Such modules must be FIPS 140-1 or -2 validated. Key management tools other than the CLI, e.g., iKeyman, are not part of the TOE.

The classification of the different data types is given in the following:

- **User data**: User data is information stored in TOE resources that can be operated upon by users and upon which the TSF places no special meaning. In this category falls all data that is not listed under TSF data since according to the CC, all data is either user or TSF data.
- **TSF data**: TSF data is information used by the TSF in making TSP decisions:
  - **critical** security parameters,
    - key data,
    - random number seed,
    - random numbers used for cryptographic operations,
    - passwords
  - configuration parameters,
  - certificates,
  - CRLs,

Copyright © 2012 by atsec information security and IBM Corporation

- OCSP responses, and
- the **security attributes** given below for the objects they belong to:
  - for the TOE: the configuration parameters FIPS mode and CC mode,
  - for TLS connections:
    - ciphersuite used
    - session keys
    - mode (client mode or server mode)
    - in server mode: authentication type (whether client authentication is required or not)
    - whether CRLs are used for certificate verification,
    - whether OCSP is used for certificate status validation,
    - whether Certificate Status Request extension is used or not,
    - whether Server Name Indication is critical or not,
  - for the keystore: the password and the hash value,
  - for crypto modules accessed via PKCS#11 or the MSCAPI: the PIN or password,
  - for PKCS#12 formatted files: the password,
  - for communication partners: the public key certificate,
  - for certificates: the trust status.

1.3.1 Product Type

GSKit is a library that can be used as a component within a larger product; it is intended to be integrated into such a product to provide this product with the capability to use a TLS-protected communication channel to another product, providing authentication of the communication partners and protection against disclosure or undetected modification of the data transferred over this trusted channel. Furthermore, GSKit offers a command line interface which can be used for accessing key generation and management functions of GSKit, for example by a human user rather than by a program.

For (optional) client and (mandatory) server authentication, X.509 [X.509] certificates are used. These certificates can be imported from the environment. In addition to that, GSKit offers certificate generation and management functionality.

GSKit requires that the product it is part of, as well as the underlying operating system hosting this product, provide the protection of the GSKit executables, security attributes and data, especially cryptographic keys and certificates. This includes a correct configuration of the TOE and that the processing resources of the TOE must be located within controlled access facilities.

1.3.2 Summary of Security Features and Logical Scope

The security features provided by GSKit are described in the following paragraphs.

1.3.2.1 Secure Channel

The TOE offers a secure channel for the confidentiality and integrity protection of data transmitted over that channel.

The secure channel functionality of the TOE is available through the TOE's SSL API. The operational environment is responsible to limit access to this API to the users and roles in the operational environment authorized to use those functions.When demanded by the product that integrates the TOE, the secure channel is established by GSKit using the TLS protocol. Where the standards defining the TLS protocol suites leave several options, the TOE does not always provide all possible options. These restrictions are described as follows.

The SSL protocol has been excluded from the TOE as mandated by the Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program. The use of only TLS is ensured by initializing the TOE in FIPS mode. This has to be done by the operational environment.

Copyright © 2012 by atsec information security and IBM Corporation

The TLS connections support mandatory server authentication and optional (configurable as mandatory) client authentication. Total anonymity mode as defined for TLS, meaning that not even the server of the connection has to authenticate, cannot be used in the evaluated configuration.

For authentication, X.509 [X.509] certificates are used; version 1, 2, and 3 certificates are supported for root and end user certificates, and version 3 certificates are supported for intermediate CA certificates.

The following ciphersuites are supported by the TOE (see chapter 1.3.2.2 for further details on cryptographic algorithms):

a) 3DES-based:

- TLS_RSA_WITH_3DES_EDE_CBC_SHA = { 0x00,0x0A } [TLSv1]
- TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA = { 0xC0, 0x08 } [RFC4492]
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA = {0xC0, 0x12} [RFC4492]

b) AES CBC- and GSM-based (RSA):

- TLS_RSA_WITH_AES_128_CBC_SHA = { 0x00, 0x2F } [RFC3268]
- TLS_RSA_WITH_AES_256_CBC_SHA = { 0x00, 0x35 } [RFC3268]
- TLS_RSA_WITH_AES_128_CBC_SHA256 = { 0x00, 0x3C } [TLSv1.2]
- TLS_RSA_WITH_AES_256_CBC_SHA256 = { 0x00, 0x3D } [TLSv1.2]
- TLS_RSA_WITH_AES_128_GCM_SHA256 = { 0x00, 0x9C } [RFC5288]
- TLS_RSA_WITH_AES_256_GCM_SHA384 = { 0x00, 0x9D } [RFC5288]
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA = { 0xC0, 0x09 } [RFC4492]
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA = { 0xC0, 0x0A } [RFC4492]
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 = {0xC0,0x23} [RFC5289]
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 = {0xC0,0x24} [RFC5289]

c) AES CBC- and GCM- based (ECDHE):

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 = {0xC0,0x2F} [RFC5289]
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 = {0xC0,0x27} [RFC5289]
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 = {0xC0,0x30} [RFC5289]
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 = {0xC0,0x28} [RFC5289]

d) Suite B compliant profile cipher suites for TLS 1.2 (per [RFC6460]:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 = {0xC0,0x2B} [RFC5289]
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 = {0xC0,0x2C} [RFC5289]

If no common ciphersuite is found, no TLS connection is set up, i.e. a session with the ciphersuite SSL_NULL_WITH_NULL_NULL is not supported.

The check that a certificate is valid and has not been revoked can be done with the help of CRLs or OCSP. CRLs have to be conformant to X.509 [X.509], version 1 and 2 CRLs are supported. The use of CRLs is not mandatory but checking the revocation status for certificates is recommended in the TLS standards. If CRLs are used, the environment has to provide an LDAP server, or an HTTP server, or the CRL via a flat file available on the underlying system. Current and correct CRLs must be provided by the environment. If CRLs are used and no CRL can be retrieved, the validation check will fail and the certificate's revocation status will be considered undetermined.

As an alternate means to CRL validation, the TOE can use OCSP as defined in [RFC2560] (and augmented by [RFC6277]) or [RFC5019] (lightweight OCSP profile) to obtain the revocation status of a certificate. The TOE can be configured to use CRL checking as a fall-back if no valid OCSP response can be obtained from the operational environment.

The use of CRL checking and/or OCSP responses in CC mode is optional.

The TOE furthermore supports the following TLS extensions:

- Server Name Indication [RFC6066]
- Certificate Status Request [RFC6066]
- Signature and Hash Algorithms as part of the TLS 1.2 specification [RFC 5246]

Copyright © 2012 by atsec information security and IBM Corporation

- Renegotiation Indication [RFC 5746]

GSKit supports compression as specified in [RFC3749].

Please note: Traditionally and when operated outside the evaluated configuration, GSKit supports versions of SSL in addition to TLS. This is the reason why – in the guidance and this ST – certain interfaces and functions offered by the TOE, such as the SSL API and the SSL environment initialization function, carry the name "SSL" instead of TLS, although their operation in the evaluated configuration (in "CC mode") is nevertheless restricted to the TLS protocol versions.

1.3.2.2 Cryptographic operations

The TOE offers generation of symmetric keys, generation of asymmetric key pairs, symmetric encryption/decryption, asymmetric encryption/decryption,generation/verification of digital signatures, data authentication, secure message digest algorithms, and random number generation. These cryptographic services are provided by the ICC component and are used by the TOE for TLS.

The ICC module that is part of the TOE has been FIPS 140-2 level 1 [FIPS140-2] validated under Certificate No. 1433 [ICC1433]. A subset of its cryptographic algorithms used within the TOE is FIPS-Approved and/or NIST-allowed; these algorithms are listed in table 2-1.

The TOE uses only FIPS-Approved and/or NIST-allowed cryptographic algorithms for the cipherspec part of the TLS cipher suites and uses non-FIPS approved algorithms for other functions, like RSA key exchange for a TLS session, when operated in compliance with this Security Target.

The TOE's DH and ECDH key establishment mechanisms, as well as MD5 and MD2 are not FIPS 140-2 [FIPS140-2] approved. DH and ECDH key establishment, and MD5, are required for key exchange during the setup of TLS sessions with corresponding cipher suites. MD5 and MD2 are used for the verification of certificates that have not been generated by the TOE. None of these functions are part of the cipherspecs of the TLS cipher suites.

The use of only FIPS-Approved and/or NIST-allowed cryptographic algorithms for the cipherspecs of TLS cipher suites (which leads to the supported cipher suites listed in chapter 1.3.2.1) is assured by initializing the TOE in FIPS mode. This initialization has to be done by the operational environment. The TOE then only provides the cipher suites listed above and initializes its ICC component also in FIPS mode which assures that only the FIPS approved random number generator is used.

The algorithms used by the TOE and whether they have been CAVS validated or not is shown in the following table.

Alternatively to using the cryptographic mechanisms implemented in the TOE, the TOE is able to use cryptographic modules in the operational environment via PKCS#11 and, on Windows, hardware or software cryptographic service providers via the MSCAPI/MSCNG, for cryptographic primitives. MSCNG is available only on Windows versions starting with Windows Vista and Server 2008, and is currently only used by GSKit (if configured) to perform ECDSA operations and store related keys and certificates. (See also Table 2 in section 1.3.2.4 for an overview of external crypto modules, key stores, and file formats supported.)

Cryptographic operation	**Algorithms implemented by the TOE**	
	FIPS 140-2 validated	**Non-FIPS 140-2 validated**
Asymmetric key generation	**RSA; 1024, 1536, 2048, 3072, 4096 Bits** as defined in ANSI X9.31 [X9.31] **ECDSA: 224, 256, 384, and 512 Bits** for curves generated over prime fields, as defined in FIPS 186-3 [FIPS186-3]	
Pair-wise key establishment		**ECDH** (Ephemeral Unified Model and One-Pass Diffie Hellman) as defined in NIST SP 800-56A [SP800-56A]

Copyright © 2012 by atsec information security and IBM Corporation

Cryptographic operation	Algorithms implemented by the TOE	
	FIPS 140-2 validated	**Non-FIPS 140-2 validated**
Symmetric data encryption / decryption	**TDEA with three independent keys (also called Triple DES or 3 key Triple DES)** – **CBC mode** as defined in FIPS 46-3 [FIPS46-3] (TDEA) and FIPS 81 [FIPS81] (CBC mode) **AES – CBC mode** ; 128 and 256 Bits as defined in FIPS 197 [FIPS197] **AES – GCM** as defined in [FIPS197] (AES) and [SP800-38D] (GCM)	
Asymmetric data encryption/decryption		**RSA** as defined in RFC 2437 [RFC2437], RFC 3447 [RFC3447], and RFC 2313 [RFC2313]; ***used by the TOE only for the key exchange of session keys as defined for TLS – acceptable per [FIPS140IG]***
Digital signature generation/ verification	**DSA (1024 bit modulus) with SHA-1** as defined in FIPS 186-3 [FIPS186-3] and FIPS 180-3 [FIPS180-3] (SHA); ***used by the TOE for signature verification only***. **RSA with SHA-1, SHA 224, 256, 384, or 512** as defined in RFC 2437 [RFC2437] and RFC 2313 [RFC2313] (RSA) and FIPS 180-3 [FIPS180-3] (SHA) **ECDSA with SHA-1, SHA 224, 256, 384, or 512** as defined in FIPS 186-3 [FIPS186-3] (ECDSA) and FIPS 180-3 [FIPS180-3] (SHA)	**DSA (1536, 2048 and 3072 bit moduli) with SHA-1** as defined in FIPS 186-3 [FIPS186-3] and FIPS 180-3 [FIPS180-3] (SHA); ***used by the TOE for signature verification only***. **RSA with MD5** as defined in RFC 2437 [RFC2437], RFC 3447 [RFC3447], and RFC 2313 [RFC2313] (RSA) and RFC 1321 [RFC1321]; ***only signature verification is used by the TOE (for the validation of certificates that are signed using RSA with MD5)*** **RSA with MD2** as defined in RFC 2437 [RFC2437], RFC 3447 [RFC3447], and RFC 2313 [RFC2313] (RSA) and RFC 1319 [RFC1319] (MD2); ***only signature verification is used by the TOE (for the validation of certificates that are signed using RSA with MD2)***
Digest (Hashfunction)	**SHA-1, SHA 224, 256, 384, and 512** as defined in FIPS 180-3 [FIPS180-3]	**MD5** as defined in RFC 1321 [RFC1321]
Data authentication	**HMAC SHA-1, SHA 224, 256, 384, or 512** as defined in FIPS 198 [FIPS198] (HMAC) and FIPS 180-3 [FIPS180-3] (SHA)	
Random number generation	**DRBG 800-90** in accordance with [SP800-90]	

Copyright © 2012 by atsec information security and IBM Corporation

Table 1: Cryptographic algorithms supported by the TOE

1.3.2.3 Self-tests

GSKit offers self-tests for the ICC component: some of ICC's cryptographic functions and the integrity of ICC can be tested. The self-tests have been analyzed as part of the FIPS 140-2 level 1 [FIPS140-2] validation.

The ICC self-tests are automatically executed upon the first call to the SSL environment initialization command of the TOE. This command (function) must be called by the application that deploys the TOE to initialize the environment before setting up a trusted channel. If the self-tests fail, the SSL environment initialization will fail and the user will be notified.

Furthermore, the TOE checks the integrity of the trust status of certificates and of certificate request data when accessing data in the keystore and will notify the user of errors if these checks fail.

1.3.2.4 Key Management

GSKit provides a local command line interface and an API for key generation and management of asymmetric key pairs as well as generation and management of certificates, including the storage of any such key material. GSKit uses a native keystore (also referred to as "CMS keystore") to store key data, or can alternatively access external hardware or software key stores via PKCS#11, PKCS#12, and (on Windows) the MSCAPI and MSCNG. For the generation of certificates, certificate requests conformant to PKCS#10 are used.

GSKit's native CMS key stores come in different versions:

- CMS V3 is deprecated and not supported in the evaluated configuration
- CMS V4 is a proprietary format
- CMS V5 is, in fact, implemented according to the PKCS#12 standard

For key and certificate management, and when initializing an SSL environment, GSKit always requires the user to specify the keystore (or cryptographic module) to be used. As a result, multiple keystores and modules can exist in the operational environment, allowing the user to specify which one is being used by GSKit. GSKit will always operate on only one keystore or module at a time.

GSKit supports im- and export of private elliptic curve keys per [RFC5915].

	CMS V4	**CMS V5**	**PKCS#11**	**PKCS#12 / CMSv5**	**MSCAPI**	**MSCNG**
Certificate / key storage?	Yes	Yes	Yes	Yes	Yes	Yes
Execution of cryptographic primitives?	No	No	Yes	No	Yes	Yes
Password / PIN supported?	Yes	Yes	Yes	Yes	Yes	Yes
Password / PIN enforced?	GSKit	GSKit	operational environment	operational environment	operational environment	operational environment
Certificate trust attributes?	trust status	not used	CKA_TRUSTED CKA_PRIVATE	not used	not used	not used

Table 2: Support of internal and external cryptographic modules

Protection of stored data

For CMS keystore contents, GSKit implements password protection for confidentiality and integrity. The password must be provided before any access to the keystore database. Each certificate, certificate/private key pair, or certificate request data is stored in one so-called data record.

Both in CMS V4 and V5, GSKit uses TDEA (Triple DES) and HMAC SHA-1 to provide this protection (pbeWithSHAAnd3KeyTripleDESCBC). CMS V4 only encrypts and hashes keys in this fashion, and adds an additional HMAC-SHA-1 hash that is generated over all records, using the user-supplied

Copyright © 2012 by atsec information security and IBM Corporation

password as the HMAC Key. This hash is then stored in the keystore, in a special record that exists once per keystore file, a so-called header record. CMS V5 encrypts and hashes all records in the keystore individually, and does not apply an additional overall hash to the keystore.

The CMS keystore password protection does not limit unauthorized users from reading and writing the keystore file. GSKit relies on the underlying operating system for these protections, but access to sensitive data is effectively controlled because all sensitive data in the keystore is encrypted, all records hashed, and the index to all records is hashed. This ensures that any modification to the file is detectable. If tampering is detected, GSKit will deny access to the keystore (the behavior is similar to receipt of an incorrect password).

The protection of data stored in non-native key stores is provided by the respective keystore, i.e. the operational environment.

If a cryptographic module in the operational environment requires a PIN or password, GSKit can temporarily store the user-supplied PIN in memory during run-time and hand it down to the module when required. This is not, however, a requirement. It is possible that a cryptographic module provides its own trusted path for user authentication, in which case an “unlocked” module may be available to GSKit without GSKit having to provide a credential to the module. GSKit therefore does not require its users to provide a PIN or password for the use of external modules.

Certificate trust

Stored with each certificate in CMS V4 keystores is its trust status, i.e. a security attribute that determines whether a certificate is trusted or not. If a certificate is not trusted, it is neither considered a valid CA nor a valid end user certificate. Setting the trust status of cryptographic material stored in CMS V4 keystores can thus be used by the TOE users to validate or invalidate certificates.

In CC mode, GSKit may only use certificates held on PKCS#11 devices that have the CKA_TRUSTED and/or CKA_PRIVATE attribute set. If the provider accessed via PKCS#11 does not support any of these Certificate Object Attributes, then it is not considered a source of trusted certificates.

For all other keystore types, trust is implicit, i.e. certificates are implicitly trusted if they are stored in these keystores.

CC mode, and the operational environment

The TOE may be initialized in a mode that affects the behavior of the GSKit-provided keystore:

- Non Common Criteria (non-CC) mode: CMS keystores have optional password protection.
- Common Criteria (CC) mode: CMS keystores are password protected and private keys are encrypted, using the password to generate the key. The password is checked against a defined password quality policy when entered the first time to ensure that a secure password has been chosen.CMS V3 keystores are not supported.

The evaluated configuration has to initialize the TOE in CC mode; non-CC mode must not be used. This has to be set by the user deploying the TOE.

Further assumptions on the operational environment:

- The operational environment is responsible for limiting access to the TOE functions (CLI and API) for key/certificate generation and management to the users and roles in the operational environment authorized to perform those management functions. Likewise, access to stash files must be restricted to authorized users in the operational environment. (Stash files can be used to “stash” passwords for keystores to avoid the need for manual intervention during the start of GSKit. When a keystore is to be accessed, GSKit will obtain the password for the keystore from a stash file if present, and otherwise use the password provided by the calling application. While the use of stash files is allowed in the evaluated configuration, care must be taken to restrict access to stash files to authorized users only.)
- Users are responsible for protecting any passwords and PINs for access to the CMS keystore or external modules appropriately.
- External key stores and cryptographic service providers must be FIPS 140-1 or 140-2 validated.

1.3.3 Software and Guidance

The Target of Evaluation is based on the following system software:

Copyright © 2012 by atsec information security and IBM Corporation

- GSKit

The following documents comprise the TOE guidance:

- [GSKCAPI]: GSKCapiCmd User's Guide
- [GSKCCMODE]: Global Security Kit Common Criteria Mode Operating Guidance
- [GSKKEY]: IBM Global Security Kit Key Management for C Programmer's Guide
- [GSKINST]: Global Security Kit Install and Packaging Guide
- [GSKSSL]: IBM Global Security Kit – Secure Socket Layer for C Programmer's Guide
- [GSKTRUST]: IBM Global Security Kit Trust Policy Design for GSKit
- [DOD]: Global Security Kit Delivery Procedure Additional Guidance

GSKit is only for IBM internal use and is not offered for sale as a standalone product, i.e., it is designed for the use in other IBM products. There is a formal process in place by which GSKit binaries, header files and documentation can be requested within IBM. After approval, permissions to access the TOE software and guidance over an IBM internal secured network connection are given and the TOE can be securely retrieved by the product team.

1.3.4 Physical Scope

1.3.4.1 Overview

The following figure provides an overview of the TOE within its intended environment. Dotted lines mark optional components which may be left out. White boxes show components that are part of the environment. The TOE components have been colored dark grey and the box surrounding them shows the TOE boundary. The secure channel set up by the TOE is the box named TLS. The TOE can use either its internal ICC software crypto module or an external cryptographic module in the operational environment for cryptographic operations, and in lieu of the TOE keystore again the external module can be used for key and certificate storage.

Copyright © 2012 by atsec information security and IBM Corporation

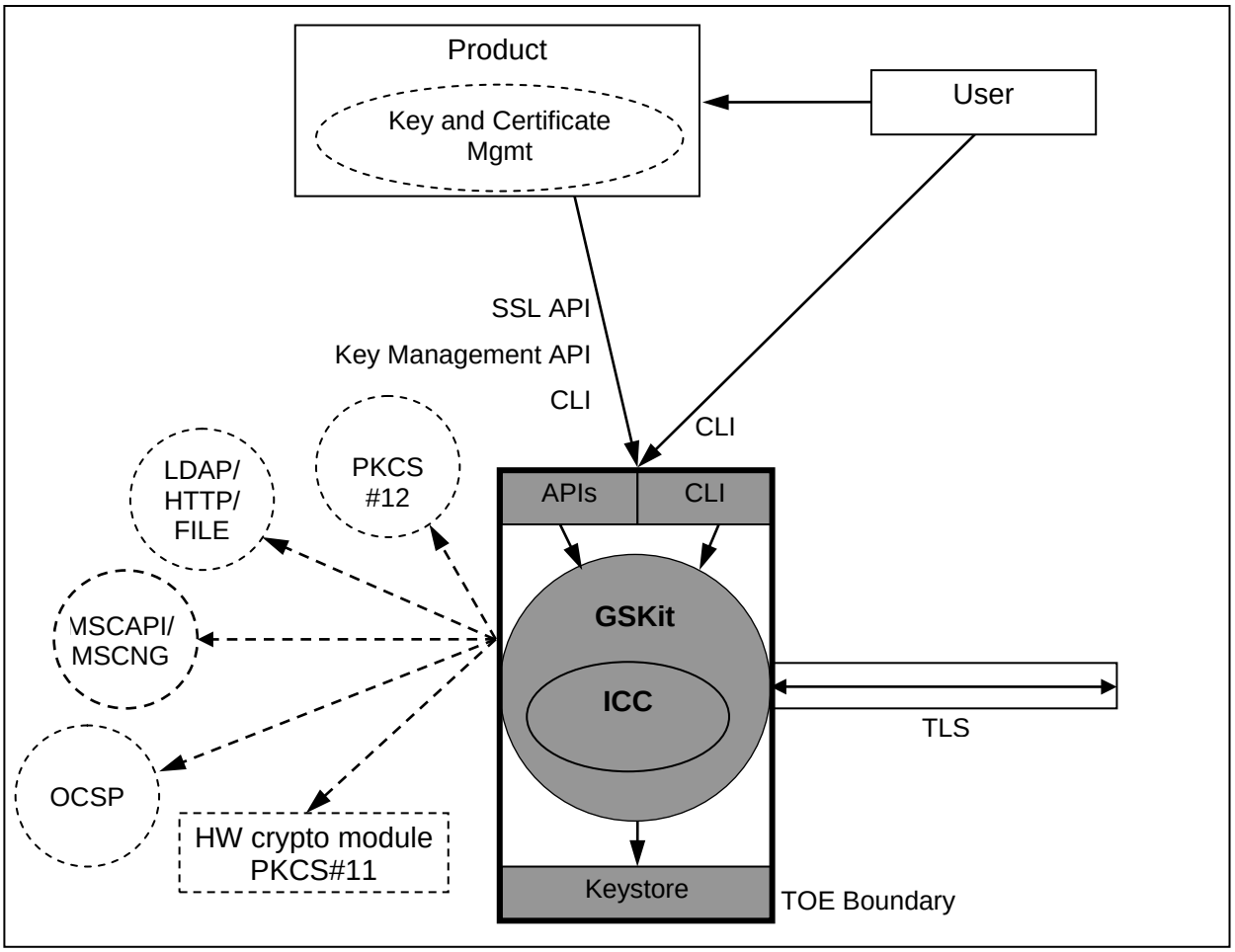

Figure 2*: TOE Interfaces and Boundaries*

1.3.4.2 TOE and User Interfaces

GSKit is not an end product by itself but integrated into an application running in the same process. GSKit offers three interfaces to the user: the Key Management API and CLI for key and certificate management functions as well as the SSL API. The CLI is the only interface to human users. The APIs and also the CLI are used by IT products (applications) that want to integrate and use the functions of the TOE. It is imperative that application integrators follow the guidance provided for integrating GSKit within the boundaries of its evaluated configuration in order to prevent security functionality, which has been subject to evaluation, from being undermined accidentally or on purpose.

The SSL API is used by an IT product in the operational environment to request the TOE to setup or to use a TLS connection. A TLS connection goes from the TOE to a remote trusted IT product, allowing them to exchange data over this trusted channel. The setup of a TLS connection can be initiated either by the TOE (then the TOE is in client mode and the remote trusted IT product in server mode) or by the remote trusted IT product (then the TOE is in server mode and the remote trusted IT product in client mode).

The Key Management API and the CLI are for key and certificate generation and management.

GSKit implements the LDAP API to access a TOE external LDAP client as an optional part of its certificate management functions (to retrieve a current and correct CRL). The LDAP client used for this purpose is not part of the TOE but an optional part of the operational environment. Alternatively, GSKit can use a statically linked LDAP client library that ships with the TOE. In either case, the interface to the LDAP client is not offered to the user; the information needed by the LDAP client to connect to the correct LDAP server is specified in the function calls of the SSL API. Likewise, GSKit can issue HTTP requests to web servers to retrieve CRLs or interpret a FILE URI to locate a CRL in the file system of its host system.

Copyright © 2012 by atsec information security and IBM Corporation

The TOE uses interfaces to the underlying operating system to store TSF data in the keystore (files in the operating system's filesystem), retrieve the system time and to access the TCP/IP stack. The TCP/IP stack is accessed using one of two methods controlled by the product using GSKit. In the first method, the application program passes a file descriptor directly to GSKit for use in writing and reading to the TCP/IP stack. In the second method, the application program passes a pointer to callback routines for reading and writing.

1.3.4.3 Auditing

The TOE does not generate audit records. For functionality invoked via the TOE's APIs or command line interface, the operational environment of the TOE is expected to implement the generation of audit records for the TOE, compliant with Exception (2) identified in CCEVS' Policy Letter #15. The guidance provided to consumers / integrators provides further information for developers on the support that needs to be provided by the operational environment.

1.3.4.4 Operating Systems and TOE Security Architecture

The operating system is not part of the TOE. It provides identification and authorization for TOE users, and a time source to be used by the TOE for certificate validation. The operating system is also responsible to protect the access to TOE services, the GSKit software, TSF data, and the keystore. As such, it provides significant support for the security of the TSF and TSF data: the TOE itself has only limited ability to protect against the corruption of TSF (i.e. the ICC module implements basic self-checks to satisfy FIPS 140-2 requirements). The underlying operating system is responsible for preventing bypass of the TSF or compromise of the TOE and its data by restricting access to authorized users only.

The TOE is provided in form of multiple binaries that have been compiled for the different operating systems supported by GSKit and the hardware architectures these operating systems run on. The TOE does not interface hardware directly, but receives direct support for its security functions from the operating system. A direct dependency of the TSF on the underlying operating system is for the provision of a reliable time source for certificate validation. Indirect dependencies exist as elaborated above.

The table below identifies the GSKit binaries that are part of the evaluated configuration, and for each of them the operating system versions they have been tested on and the additional operating system versions that the evaluation results can be extended to. For each GSKit binary, the operating system versions listed for a binary in the "Compatibility" column have been found to provide the necessary support functions in the same way as the operating system version that is actually used for testing (column "Test Environment") during the evaluation. Compatibility is claimed in accordance with CCEVS Precedent PD-0084.

TOE binary version:	Tested on:	Test results also valid for:
Sun Solaris 32-bit built on Solaris 9 on Ultrasparc architecture	Solaris 10 Release 11/06 on Ultrasparc architecture	Solaris 9, 11 on Ultrasparc architecture
Sun Solaris 64-bit built on Solaris 9 on Ultrasparc architecture	Solaris 10 Release 11/06 on Ultrasparc architecture	Solaris 9, 11 on Ultrasparc architecture
IBM AIX 32-bit built on AIX 5.3 TL07 on PPC64 architecture	AIX 6.1 on PPC64 architecture	AIX 5.3, 7 on PPC64 architecture
IBM AIX 64-bit built on AIX 5.3 TL07 on PPC64 architectures	AIX 6.1 on PPC64 architecture	AIX 5.3, 7 on PPC64 architecture
Microsoft Windows 32-bit built on Windows Server 2008 on x86_64 architectures	Windows Server 2008 on x86_32 and x86_64 architecture	Windows NT, XP, Vista, 7, Server 2000, Server 2003 on x86_32 and x86_64 architectures
Microsoft Windows 64-bit built on Windows Server 2008 on x86_64 architecture	Windows Server 2008 on x86_64 architecture	Windows Vista, 7, Windows Server 2003 on x64_64 architecture

Copyright © 2012 by atsec information security and IBM Corporation

TOE binary version:	Tested on:	Test results also valid for:
Linux 32-bit built on SLES 10 on x86_64 architecture	RHEL 5.0 on x86_32 and X86_64 architecture	RHEL6 and SLES 10, 11 on x86_32 and x86_64 architectures
Linux 64-bit built on SLES 10 on x86_64 architecture	RHEL 5.0 on x86_64 architecture	RHEL6 and SLES 10, 11 on x86_64 architecture
Linux 32-bit built on SLES 10 on PPC64 architectures	RHEL 5.0 on PPC64 architecture	RHEL6 and SLES 10, 11 on PPC64 architecture
Linux 64-bit built on SLES 10 on PPC64 architectures	RHEL 5.0 on PPC64 architecture	RHEL6 and SLES 10, 11 on PPC64 architecture
Linux 32-bit built on SLES 10 on zSeries-64 architecture	RHEL 5.0 on zSeries-64 architecture	RHEL6 and SLES 10, 11 on zSeries-64 architecture
Linux 64-bit built on SLES10 on zSeries-64 architecture	RHEL 5.0 on zSeries-64 architecture	RHEL6 and SLES 10, 11 on zSeries-64 architecture

Table 3: Operating systems supported by the evaluated configuration

The underlying operating system has to be configured to disallow remote login (by disabling all services that provide remote login). This is due to the FIPS 140-2 level 1 security policy [ICCSEC]. It is the consumer's responsibility to ensure that the evaluated configuration of GSKit is only executed on operating systems that have been covered by the FIPS 140-2 validation of GSKit's ICC module.

1.3.4.5 Certificates

Certificates are either provided by the operational environment or generated by the TOE.

The TOE lacks important features that would allow its use for the management of Public Key Infrastructures, for example the generation of CRLs.

To this extent, the TOE is able to provide basic certificate generation features, for example for the use in closed environments. Further needed certificate management functionality, especially the generation of CRLs if used for certificate validation or functions needed by the product deploying the TOE, has to be provided by the operational environment.

1.3.4.6 Keystore

The password-protected CMS keystore can be used by the TOE for the storage of key data. The password used to protect the keystore must be generated by the TOE; user-defined passwords must not be used. The TOE checks passwords provided by users as part of keystore creation or password change against a defined quality metric, but that metric does not test the randomness of the password. Only generation of the password by the TOE guarantees proper randomness of the password in addition to meeting the defined quality metric. The password must not be revealed to anyone. In order to ensure the secure operation of the keystore, the TOE has to be configured by the product deploying it to be in CC mode (see chapter 1.3.2.4 for an explanation of CC mode).

The files used to implement the keystore are located in the operating system. The operational environment is responsible for ensuring that all access to these files is mediated via the GSKit Key Management API and for protecting these files from unauthorized access.

1.3.4.7 CRLs

CRLs can be used for certificate validation. If CRLs are used for certificate validation, it is the responsibility of the operational environment to provide a current and correct CRL.

CRLs can be retrieved using an LDAP client and server, an HTTP server, or a flat file in the operational environment. For LDAP or HTTP, an LDAP server, or an HTTP server, must be provided by the operational environment. In case of LDAP, the TOE accesses the LDAP client via the LDAP client's API, either using a statically bound library that is part of the TOE, or a dynamically linked library in the operational environment. Then the LDAP client establishes a TCP/IP connection to the LDAP server, retrieves the CRL, and passes this CRL back to the TOE. In case of HTTP, the TOE

Copyright © 2012 by atsec information security and IBM Corporation

issues HTTP requests to the web server, and supports CRL caching per [RFC3280] and [RFC5280]. In case of flat files, the TOE retrieves the CRL from the underlying system's file system. The retrieved CRL is finally used for certificate validation.

The connection between the LDAP client and the LDAP server, or between the TOE and the HTTP server, must be protected against DoS attacks or replay attacks by means of the operational environment.

1.3.4.8 OCSP

GSKit supports the Online Certificate Status Protocol as specified in RFC 2560 [RFC2560], and the Lightweight OCSP Profile specified in [RFC5019], to query a responder in the operational environment for the revocation status of a certificate. If OCSP is used for certificate validation, it is the responsibility of the operational environment to provide a trusted OCSP responder.

GSKit supports the following algorithms for OCSP in accordance with [RFC6277]:

- RSA with SHA1, SHA224, SHA256, SHA384, SHA512,
- ECDSA with SHA1, SHA224, SHA256, SHA384, SHA512
- DSA with SHA1

1.3.4.9 Crypto Modules

In addition to its own cryptographic software module, ICC, and its native key store, the CMS keystore, GSKit also provides support for the use of external crypto modules for cryptographic operations as discussed in section 12 and storage of key material as discussed in section 14.

Such modules must be validated according to FIPS 140-1 or 140-2 and must be operated in their Approved mode of operation. The cryptographic operations accessed via PKCS#11 or the MSCAPI/MSCNG are the same as provided by ICC (see section 1.3.2.2).

GSKit implements PKCS#11 version 2.20 [PKCS#11-2.10], including SHA-224 as defined in Amendment 3 [PKCS#11A3R1].

Copyright © 2012 by atsec information security and IBM Corporation

2 CC Conformance Claim

This ST is *CC Part 2 extended* [CC] and *CC Part 3 conformant* [CC]. The CC version of [CC] is 3.1R3.

This ST is EAL4-conformant as defined in [CC] Part 3.

No conformance to a Protection Profile is claimed.

Copyright © 2012 by atsec information security and IBM Corporation

3 TOE Security Environment

3.1 Threats

The assumed security threats are listed below.

The **assets** to be protected comprise the information stored, processed or generated by the TOE. This information contains key data, data for cryptographic operations (like for example seeds for random data needed for key generation) as well as the data to be transferred over the trusted channel.

The purpose of the TOE is to provide a secure channel between itself and a remote trusted IT product. This secure channel is used by a product (application) of the operational environment. Threats concerning information confidentiality and integrity of the TOE code and TSF data are mainly countered by the operational environment; the TOE only has to provide a correct implementation of the secure channel such that the data transmitted over it is protected and to secure the data that is to be stored in the keystore.

The **threat agents** can be categorized as either:

1. A user (not a user of the TOE) who gets access to data communicated over unprotected networks (“remote attacker”). This person has no access to the TOE interfaces and no access to the system the TOE is integrated into.
2. A user (not a user of the TOE) who gets access to the TOE code, TSF data, or the keystore (e. g. on a backup tape or by getting access to the disk where it is stored on).
3. An authorized user of the operational environment who has been granted the right to access the TOE using one or many of the interfaces (SSL API, command line interface, Key Management API). Authorized users include administrators of the TOE and the operational environment, and are assumed to be competent and trustworthy. They will not intentionally subvert of bypass TOE security functions, but may do so by accident and may have to be held responsible for their actions.

3.1.1 Threats

T.DATA_COMPROMISE An unauthorized user is able to eavesdrop on, or manipulate without detection, the data exchanged between the TOE, or an application integrating the TOE, and a remote trusted IT product. This includes the case where an unauthorized user has gained access to a consuming application and attempts to use this as a path to gain access to data streams of other applications using the TOE.

T.UNAUTHORIZED An unauthorized user gains access to the TOE, TSF data, or the keystore, allowing the user to manipulate the TSF or TSF data or to decrypt traffic encrypted by the TOE that the user is not authorized to see in plaintext.

3.2 Assumptions

3.2.1 Intended usage of the TOE

A.AMBIGUOUS It is assumed that the operational environment will provide mechanisms to audit all security-relevant TSF actions initiated via the TOE’s APIs or performed on key stores through the command line interface in order to provide accountability for user actions.

A.FIPS140 It is assumed that the TOE and its operational environment will be operated in a configuration compliant with the FIPS validated configuration of the ICC component as described in the FIPS 140-2 Level 1 Security Policy for ICC [ICCSEC].

Copyright © 2012 by atsec information security and IBM Corporation

3.2.2 Environment of use of the TOE

3.2.2.1 Physical

None. (See A.INTEGRATE for physical protection of the TOE.)

3.2.2.2 Personnel

A.ADMIN Those responsible for the administration of the TOE and the operational environment are competent and trustworthy individuals, capable of managing the TOE and the operational environment and the security of the information it contains.

A.INTEGRATE Those who integrate the TOE as part of a larger product or system are assumed to follow the guidance provided with the TOE with respect to the configuration of the TOE, the use of TOE functions and interfaces and the protection of the TOE code and data. They are trustworthy and do not try to subvert or bypass TOE security functions.

3.2.2.3 Connectivity

A.PKI The operational environment provides public key infrastructure services not supplied by the TOE (such as, registration authorities, processing of certificate requests by a certification authority, issuance of CRLs, operation of OCSP responders, etc.), as needed for the intended usage of the application integrating the TOE, in a reliable and trustworthy fashion.

3.3 Organizational Security Policies

P.SELFSEC The TOE shall implement self-tests for its cryptographic module and failure detection with preservation of a secure state for the trusted channel component

P.TRANSFER The TOE shall be able to exchange TSF data with entities in the operational environment.

Copyright © 2012 by atsec information security and IBM Corporation

4 Security Objectives

4.1 Objectives for the TOE

O.KEYSTORE	The TOE must protect private keys stored in the keystore from unauthorized disclosure. In addition to that, the whole of the data stored in the keystore must be protected against undetected modification.
O.SECCHANNEL	The TOE shall offer the functionality to communicate with a remote trusted IT product using a trusted channel that protects the confidentiality and integrity of user data being transmitted over this channel, and that provides the ability to authenticate the communication partners before transmitting user data that requires protection.
O.SELFSEC	The TOE must implement self-tests for the ICC cryptographic module and failure detection with preservation of a secure state for the trusted channel component.
O.SESSIONDATA	The TOE must protect secret TSF data (private/secret keys, session keys, and critical security parameters) of its individual trusted channel sessions and must ensure that no such data is made available to other trusted channel sessions also provided by the TOE.
O.TRANSFER	The TOE must be able to consistently interpret TSF data that is exchanged with entities in the operational environment via PKCS#11, PKCS#12, or the MSCAPI/MSCNG.

4.2 Objectives for the Operational Environment

OE.ADMIN	Those responsible for the administration of the TOE must be trained such that they are capable of managing the TOE and the security of the information it contains; this includes making sure that any backup copies of TSF data are appropriately protected. Administrators are trustworthy.
OE.AUDIT.1	The operational environment must implement the generation of audit records for all security-relevant TSF actions initiated via the TOE's APIs or performed on key stores through the command line interface.
OE.AUDIT.2	The operational environment must provide means for users to review all audit records.
OE.AUDIT.3	Integrators of the TOE must provide guidance to end users to ensure that the operational environment will be configured so that audit records generated by the TOE are protected against unauthorized access.
OE.PKI	The operational environment is responsible for the provision of public key infrastructure services not supplied by the TOE (such as, registration authorities, processing of certificate requests by a certification authority, issuance of CRLs, operation of OCSP responders, etc.), as needed for the intended usage of the application integrating the TOE, in a reliable and trustworthy fashion.
OE.CRYPTO	If the TOE is configured to use a cryptographic module or services provider in the operational environment for the storage of cryptographic keys and certificates and/or the provision of cryptographic operations, the environment must provide such a module accessible via PKCS#11 or the MSCAPI/MSCNG. The module must authenticate GSKit before granting access to stored key material.
OE.DATA	The operational environment (e.g. the operating system) must protect the TOE code from unauthorized modification and user data and TSF data from unauthorized disclosure, access, and modification. This

Copyright © 2012 by atsec information security and IBM Corporation

	includes also the protection of the TOE against other applications in the operational environment.
OE.FIPS140	If crypto modules are employed in the operational environment, they must be FIPS 140-1 or 140-2 validated and operated in their Approved mode of operation.
OE.INTEGRATE	Those who integrate the TOE as part of a larger product or system are following the guidance provided with the TOE with respect to the configuration of the TOE, the use of TOE functions and interfaces and the protection of the TOE code and data. They are trustworthy and do not try to subvert or bypass TOE security functions.
OE.CRL	If a CRL is required by the TOE, the operational environment must provide a valid, current CRL conformant to [X.509]. If CRLs are provided via LDAP client and server or HTTP server, then the connection to the server must be protected against replay and DoS attacks. Alternatively, a CRL can be provided by the operational environment in a flat file on the system that the TOE is installed on.
OE.MODE	The TOE must be initialized to operate in FIPS mode and CC mode.
OE.OS	On the system the TOE is running on, only operating systems that are covered by the FIPS 140-2 level 1 certificate for the ICC component may be used.
OE.OSLOGIN	Remote login to the operating system must be disabled (by disabling all services that offer remote login).
OE.PASSWORD	Users knowing the password to protect the keystore or an external key store must apply protection measures that prohibit unauthorized access to or disclosure of password information.
OE.PHYSEC	The processing resources of the TOE must be located within controlled access facilities such that they are protected against unauthorized physical access.
OE.USER	The environment must only allow authorized users to access the TOE.

4.3 Security Objectives Rationale

4.3.1 Coverage

The following table provides a mapping of TOE objectives to threats and policies, showing that each objective is at least covered by one threat or policy.

Objective	Threat / Policy
O.KEYSTORE	T.UNAUTHORIZED
O.SECCHANNEL	T.DATA_COMPROMISE
O.SELFSEC	P.SELFSEC
O.SESSIONDATA	T.DATA_COMPROMISE
O.TRANSFER	P.TRANSFER

Table 4: Mapping Objectives to Threats and Policies

The following table provides a mapping of the objectives for the operational environment to assumptions, threats and policies, showing that each objective is at least covered by one assumption, threat or policy.

Env. Objective	Threat / Assumption / Policy
OE.ADMIN	A.ADMIN
OE.AUDIT.1	A.AMBIGUOUS
OE.AUDIT.2	A.AMBIGUOUS

Copyright © 2012 by atsec information security and IBM Corporation

Env. Objective	Threat / Assumption / Policy
OE.AUDIT.3	A.AMBIGUOUS
OE.PKI	A.PKI
OE.CRL	A.PKI
OE.CRYPTO	T.UNAUTHORIZED P.TRANSFER
OE.DATA	T.UNAUTHORIZED
OE.FIPS140	A.FIPS140
OE.INTEGRATE	A.INTEGRATE
OE.MODE	A.FIPS140
OE.OS	A.FIPS140
OE.OSLOGIN	A.FIPS140
OE.PASSWORD	T.UNAUTHORIZED
OE.PHYSEC	T.UNAUTHORIZED
OE.USER	T.UNAUTHORIZED

Table 5: Mapping Objectives for the Environment to Threats, Assumptions and Policies

4.3.2 Sufficiency

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

Threat	Rationale for security objectives
T.DATA_COMPROMISE	The threat of data compromise is addressed by: • The TOE objective that states that the TOE shall offer a product of the operational environment the functionality to communicate with a remote trusted IT product using a trusted channel that protects the confidentiality and integrity of user data being transmitted over this channel and provides the ability to authenticate the communication partners before transmitting user data that requires protection (O.SECCHANNEL). • The TOE objective that states that the TOE must protect secret TSF data (private/secret keys, session keys, and critical security parameters) of its individual trusted channel sessions and must ensure that no such data is made available to other trusted channel sessions also provided by the TOE. (O.SESSIONDATA).
T.UNAUTHORIZED	The threat of unauthorized access to the TOE and TSF data is addressed by the objective O.KEYSTORE which states that the TOE must protect private keys stored in the keystore from unauthorized disclosure, and that the whole of the key data stored in the keystore must be protected against undetected modification. This is supported by various objectives for the operational environment: • The environment objective that states that the environment must only allow authorized users to access the TOE (OE.USER). • The environment objective that states that the operational environment (e.g. the operating system) must protect the TOE code from unauthorized modification and user data and TSF data from

Copyright © 2012 by atsec information security and IBM Corporation

	unauthorized disclosure, access, and modification. This includes also the protection of the TOE against other applications in the operational environment. (OE.DATA). • The environment objective that states that the processing resources of the TOE must be located within controlled access facilities such that they are protected against unauthorized physical access (OE.PHYSEC). • The environment objective that states that users knowing the password to protect the keystore must apply protection measures that prohibit unauthorized access to or disclosure of password information (OE.PASSWORD).

Table 6: Sufficiency analysis for threats

The following rationale provides justification that the security objectives for the environment are suitable to cover each individual assumption, that each security objective for the environment that traces back to an assumption about the environment of use of the TOE, when achieved, actually contributes to the environment achieving consistency with the assumption, and that if all security objectives for the environment that trace back to an assumption are achieved, the intended usage is supported:

Assumption	Rationale for security objectives
A.ADMIN	This assumption is addressed by the environment objective that states that those responsible for the administration of the TOE must be trained such that they are capable of managing the TOE and the security of the information it contains; this includes making sure that backup copies of TSF data are appropriately protected. Administrators are trustworthy. (OE.ADMIN).
A.AMBIGUOUS	The assumption to implement audit mechanisms for actions provided by the TSF is addressed by the operational environment: The objective OE.AUDIT.1 expresses the requirement for the integrators of the TOE to create audit records for TSF actions when the TOE's APIs are being used, or when the TOE's CLI is used to modify stored key material. OE.AUDIT.3 requires that the operational environment protects these records against unauthorized access, and OE.AUDIT.2 provides review capabilities to authorized users.
A.PKI	This assumption is addressed by the environment objective that states that the operational environment is responsible for the provision of PKI services as needed in the operational environment (OE.PKI). The requirements for the provision of CRLs are further spelled out in OE.CRL, which that states that if a CRL is required by the TOE, the operational environment must provide a valid, current CRL, and – if applicable, as for LDAP and HTTP – TCP/IP connections used for retrieving CRLs must be an internal communication link within a protected network. The environment provides a valid, current CRL conformant to [X.509].
A.FIPS140	The assumption on compliance with ICC's Security Policy is addressed by: • The environment objective that states that the TOE must be initialized to operate in FIPS mode and CC mode (OE.MODE). • The environment objective that states that remote login to the operating system must be disabled (by disabling all services that offer remote login) (OE.OSLOGIN).

Copyright © 2012 by atsec information security and IBM Corporation

	• Choosing one of the operating systems that have been used for testing of the ICC FIPS component during the FIPS 140-2 level 1 [FIPS140-2] validation (OE.OS). • OE.FIPS140 requires consumers to ensure that external modules are operated in a FIPS 140-1 or -2 validated mode, providing for the necessary tamper resistance.
A.INTEGRATE	This assumption is addressed by the environment objective that states that those who integrate the TOE as part of a larger product or system are following the guidance provided with the TOE with respect to the configuration of the TOE, the use of TOE functions and interfaces and the protection of the TOE code and data. They are trustworthy and do not try to subvert or bypass TOE security functions (OE.INTEGRATE).

Table 7: Sufficiency analysis for assumptions

The following rationale provides justification that the security objectives are suitable to implement each individual organizational security policy and that each security objective tracing back to a policy, when achieved, actually contributes to the implementation of that policy:

Policy	Rationale for security objectives
P.SELFSEC	This policy is fulfilled by the TOE objective that states that the TOE must implement self-tests for cryptographic modules and failure detection with preservation of a secure state for the trusted channel component (O.SELFSEC).
P.TRANSFER	The objective O.TRANSFER requires the TOE to be able to interpret data that is exchanged via interfaces following the specifications of PKCS#11, the MSCAPI/MSCNG, and PKCS#12. This is supported by OE.CRYPTO, which requires that cryptographic modules or services providers used in the operational environment support PKCS#11 or the MSCAPI/MSCNG.

Table 8: Sufficiency analysis for OSPs

Copyright © 2012 by atsec information security and IBM Corporation

5 Extended Components Definition

5.1 Class FPT: Protection of the TSF

5.1.1 External TSF data transfer (FPT_ETT_GSK)

Family Behavior

This family provides requirements that address protection of TSF data when it is transferred outside the control of the TOE.

Component leveling

FPT_ETT_GSK.1 Basic external TSF data protection, requires that TSF data be protected when transmitted outside the control of the TOE.

Management: FPT_ETT_GSK.1

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

a) The management of the list of types of TSF data that must be protected when it is transmitted outside of the TSF;

b) management of the types of modification against which the TSF should protect;

c) management of the mechanism used to provide the protection of the data transmitted outside of the TOE.

Audit: FPT_ETT_GSK.1

There are no auditable events foreseen.

5.1.1.1 FPT_ETT_GSK.1 Basic external TSF data self-protection

Hierarchical to: No other components.

Dependencies: No dependencies

FPT_ETT_GSK.1.1 The TSF shall protect [assignment: *list of TSF data*] from [selection: *disclosure, undetected modification*] when it is transmitted outside of the TOE.

FPT_ETT_GSK.1.2 The TSF shall use [assignment: *list of standards that are applicable*].

5.2 Class FTP: Trusted path/channels

5.2.1 Inter-TSF trusted channel (FTP_ITC)

Component leveling

FTP_ITC_PD-0108.1 Inter-TSF trusted channel, requires that the TSF provide a trusted communication channel between itself and another trusted IT product.

Management: FTP_ITC_PD-0108.1

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

a) Configuring the actions that require trusted channel, if supported.

Audit: FTP_ITC_PD-0108.1

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

a) Minimal: Failure of the trusted channel functions.

b) Minimal: Identification of the initiator and target of failed trusted channel functions.

c) Basic: All attempted uses of the trusted channel functions.

Copyright © 2012 by atsec information security and IBM Corporation

d) Basic: Identification of the initiator and target of all trusted channel functions.

5.2.1.1 FTP_ITC_PD-0108.1 Inter-TSF trusted channel

Hierarchical to: No other components.

Dependencies: No dependencies

FTP_ITC_PD-0108.1.1 The TSF shall provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

FTP_ITC_PD-0108.1.2 The TSF shall permit [selection: *the TSF, the remote trusted IT product*] to initiate communication via the trusted channel.

FTP_ITC_PD-0108.1.3 The TSF shall use a trusted channel for the following functions [assignment: *list of functions for which a trusted channel is required*].

Note: Use of the explicitly specified requirement replacement does not imply a responsibility for the TSF under evaluation to ensure that a remote TSF performs a particular action; rather, the TSF under evaluation is only required to use the channel for the indicated purpose if the channel is initiated. There is no requirement for the evaluator to verify that the remote TSF initiates the channel.

Note: When this modified version of FTP_ITC.1.3 is used, there should also be an accompanying note that explains that the rationale for this explicit requirement is that it corrects an error identified by CCEVS in the requirement and an interpretation is being created by NIAP to correct the offending wording.

5.3 Extended Components Definition Rationale

One extended security component is defined in this Security Target:

FPT_ETT_GSK.1 Basic external TSF data transfer protection

In addition to that, the security functional requirement FTP_ITC.1 has been replaced by FTP_ITC_PD-0108.1 according to [PD-0108].

The reasons for using these explicit SFRs are given in the following.

5.3.1 FPT_ETT_GSK.1 Basic external TSF data self-protection

This security functional requirement has been defined as an extension to part 2 of the Common Criteria to address a situation where a TOE needs to export (and later re-import and use) TSF data outside of the TOE scope of control. The TOE needs to ensure that the data is protected by itself when it is not under the control of the TOE. A typical example is critical TSF data that needs to be stored for later retrieval outside the TOE scope of control (even though the keystore is part of the TOE, the files that are used to implement it are located in the operational environment). Requirements may exist to protect the TSF data from disclosure and/or to allow the TOE to detect (upon re-import) unauthorized modifications made to the TSF data.

Although cryptographic methods will usually be used to provide this functionality, other methods (e. g. a non-cryptographic transformation using a secret only known to the TOE) are not excluded and therefore no dependency to cryptographic functions is defined.

There has been found no SFR in [CC] part 2 that is appropriate for this functionality. There are FPT_ITC.1 for confidentiality protection and FPT_ITI.1 for integrity protection but these SFRs have been found to be not appropriate since they deal with transmission of data between the TOE and other entities whereas in this context, data has to be stored outside the TOE. Furthermore, SFRs of class user data protection (FDP) do not apply since FPT_ETT_GSK.1 aims to protect TSF data rather than user data.

5.3.2 FTP_ITC_PD-0108.1 Inter-TSF trusted channel

The SFR FTP_ITC.1 has been replaced by FTP_ITC_PD-0108.1 as described in PD-0108 [PD-0108]. PD-0108 addresses the element FTP_ITC.1.3. A problem arises when the functions for which either the TSF or a remote trusted IT product must use a trusted channel would have to be specified

Copyright © 2012 by atsec information security and IBM Corporation

since FTP_ITC.1.3 only refers to the local TOE and specifically refers to the initiation and not to the use of the trusted channel.

PD-0108 corrects this error identified by CCEVS in the requirement and provides an interpretation created by NIAP to correct the offending wording.

Copyright © 2012 by atsec information security and IBM Corporation

6 Security Requirements

6.1 TOE Security Functional Requirements

The following table shows the Security functional requirements for the TOE, and the operations performed on the components according to CC part 2: iteration (Iter.), refinement (Ref.), assignment (Ass.) and selection (Sel.). Typographical distinctions are used throughout this chapter in the reproduction of the SFRs to further identify these operations: assignments and selections are formatted in **bold**, refinements in ***bold and italics***, and iterations by appending an alphabetic counter (a, b, c, …) to the component identifier.

Security Functional Class	Security Functional Requirement	Component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
Cryptographic Support (FCS)	FCS_CKM.1: Cryptographic key generation	FCS_CKM.1a	CC Part 2	Yes	Yes	Yes	No
		FCS_CKM.1b	CC Part 2	Yes	Yes	Yes	No
	FCS_COP.1: Cryptographic operation	FCS_COP.1a	CC Part 2	Yes	No	Yes	No
		FCS_COP.1b	CC Part 2	Yes	No	Yes	No
		FCS_COP.1c	CC Part 2	Yes	Yes	Yes	No
		FCS_COP.1d	CC Part 2	Yes	No	Yes	No
		FCS_COP.1e	CC Part 2	Yes	No	Yes	No
		FCS_COP.1f	CC Part 2	Yes	No	Yes	No
		FCS_COP.1g	CC Part 2	Yes	No	Yes	No
		FCS_COP.1h	CC Part 2	Yes	No	Yes	No
User Data Protection (FDP)	FDP_ACC.1: Subset access control	FDP_ACC.1	CC Part 2	No	No	Yes	No
	FDP_ACF.1: Security attribute based access control	FDP_ACF.1	CC Part 2	No	No	Yes	No
	FDP_DAU.1: Basic data authentication	FDP_DAU.1	CC Part 2	No	Yes	Yes	No
	FDP_RIP.2: Full residual information protection	FDP_RIP.2a	CC Part 2	Yes	Yes	No	Yes
		FDP_RIP.2b	CC Part 2	Yes	Yes	No	Yes
	FDP_UCT.1: Basic data exchange confidentiality	FDP_UCT.1	CC Part 2	No	No	Yes	Yes
	FDP_UIT.1: Data exchange integrity	FDP_UIT.1	CC Part 2	No	No	Yes	Yes
Identification and Authentication (FIA)	FIA_SOS.1: Verification of secrets	FIA_SOS.1	CC Part 2	No	Yes	Yes	No
	FIA_SOS.2: TSF Generation of secrets	FIA_SOS.2	CC Part 2	No	No	Yes	No
	FIA_UAU.2: User authentication before any action	FIA_UAU.2	CC Part 2	No	Yes	No	No
Security Management (FMT)	FMT_MSA.3: Static attribute initialization	FMT_MSA.3	CC Part 2	No	No	Yes	Yes
	FMT_SMF.1: Specification of Management Functions	FMT_SMF.1	CC Part 2	No	Yes	Yes	No
Protection of the TSF (FPT)	FPT_ETT_GSK.1 Basic external TSF data transfer protection	FPT_ETT_GSK.1a	ECD	Yes	Yes	Yes	Yes
		FPT_ETT_GSK.1b	ECD	Yes	Yes	Yes	Yes
	FPT_FLS.1: Failure with preservation of secure state	FPT_FLS.1	CC Part 2	No	No	Yes	No
	FPT_TDC.1: Inter-TSF basic	FPT_TDC.1a	CC Part 2	Yes	Yes	Yes	No

Copyright © 2012 by atsec information security and IBM Corporation

Security Functional Class	Security Functional Requirement	Component	Source	Operations			
				Iter.	Ref.	Ass.	Sel.
	TSF data consistency	FPT_TDC.1b	CC Part 2	Yes	Yes	Yes	No
		FPT_TDC.1c	CC Part 2	Yes	Yes	Yes	No
		FPT_TDC.1d	CC Part 2	Yes	Yes	Yes	No
		FPT_TDC.1e	CC Part 2	Yes	Yes	Yes	No
	FPT_TST.1: TSF testing	FPT_TST.1	CC Part 2	No	Yes	Yes	Yes
Trusted Path/Channels (FTP)	FTP_ITC_PD-0108.1: Inter-TSF trusted channel	FTP_ITC_PD-0108.1	[PD-0108]	No	No	Yes	Yes

Table 9: Security Functional Requirements of the TOE

6.1.1 Cryptographic Support (FCS)

6.1.1.1 Cryptographic key generation (FCS_CKM.1a)

FCS_CKM.1.1 The TSF shall generate symmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm

a) TLSv1 and TLSv1.1 symmetric key and secret generation,

b) TLSv1.2 symmetric key and secret generation

and specified cryptographic key sizes

a) 168 Bits (three independent TDEA keys), 128 and 256 Bits (AES key), and 160 Bits (HMAC SHA-1 secret) for TLSv1 and TLSv1.1,

b) 168 Bits (three independent TDEA keys), 128 and 256 Bits (AES key), and 256 Bits (HMAC SHA-256 secret) for TLSv1.2

that meet the following:

a) conformant to RFC 2246 [TLSv1] and RFC 4346 [TLSv1.1] with RFC 3268 [TLS_AES] (TLSv1 and TLSv1.1 symmetric key and secret generation),

b) conformant to RFC 5246 [TLSv1.2] (TLSv1.2 symmetric key and secret generation).

6.1.1.2 Cryptographic key generation (FCS_CKM.1b)

FCS_CKM.1.1 The TSF shall generate ***asymmetric*** cryptographic keys in accordance with a specified cryptographic key generation algorithm

a) RSA

b) ECDSA

and specified cryptographic key sizes

a) 1024 and 2048 Bits

b) 224, 256, 384, and 512 Bits for curves generated over prime fields

that meet the following:

a) conformant to ANSI X9.31 [X9.31],

b) conformant to FIPS 186-3 [FIPS186-3] and FIPS 140-2 [FIPS140-2] approved.

Copyright © 2012 by atsec information security and IBM Corporation

6.1.1.3 Cryptographic operation (FCS_COP.1a)

FCS_COP.1.1 The TSF shall perform **symmetric encryption and symmetric decryption** in accordance with a specified cryptographic algorithm

a) TDEA with three independent keys (CBC mode) ,

b) AES (CBC mode)

c) AES (GCM mode)

and cryptographic key sizes

a) 168 Bits (TDEA),

b) 128, 256 Bits (AES)

c) 128, 256 Bits (AES)

that meet the following:

a) conformant to FIPS 46-3 [FIPS46-3] (TDEA), conformant to FIPS 81 [FIPS81] (CBC mode),

b) conformant to FIPS 197 [FIPS197] (AES, CBC mode),

c) conformant to FIPS 197 [FIPS197] (AES) and [SP800-38D] (GCM)

and FIPS 140-2 [FIPS140-2] approved.

6.1.1.4 Cryptographic operation (FCS_COP.1b)

FCS_COP.1.1 The TSF shall perform **digest generation and verification** in accordance with a specified cryptographic algorithm

a) SHA-1, -224, -256, -384, and -512

b) MD5

and cryptographic key sizes **none** that meet the following:

a) conformant to the Secure Hash Standard (SHS) as defined in FIPS 180-3 [FIPS180-3] and FIPS 140-2 [FIPS140-2] approved (SHA-1, -224, -256, -384, and -512),

b) conformant to RFC 1321 [RFC1321] (MD5).

6.1.1.5 Cryptographic operation (FCS_COP.1c)

FCS_COP.1.1 The TSF shall perform **data authentication** in accordance with a specified cryptographic algorithm **HMAC SHA-1, -224, -256, -384, and -512** and cryptographic ***hash*** sizes **160, 224, 256, 384, and 512 Bits accordingly** that meet the following

conformant to FIPS 198 [FIPS198] (HMAC) and FIPS 180-3 [FIPS180-3] (SHA-1, -224, -256, -384, and -512).

6.1.1.6 Cryptographic operation (FCS_COP.1d)

FCS_COP.1.1 The TSF shall perform **encryption and decryption of session key related data** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes **1024, 1536, 2048, 3072, 4096, or 8192 Bits** that meet the following:

conformant to RFC 2437 [RFC2437], RFC 3447 [RFC3447], and RFC 2313 [RFC2313] (RSA) and
encryption/decryption of session key related data as defined in RFC 2246 [TLSv1], RFC 4346 [TLSv1.1], and RFC 5246 [TLSv1.2].

Copyright © 2012 by atsec information security and IBM Corporation

6.1.1.7 Cryptographic operation (FCS_COP.1e)

FCS_COP.1.1 The TSF shall perform **generation of random numbers** in accordance with a specified cryptographic algorithm **Deterministic Random Bit Generator** and cryptographic key sizes **none** that meet the following:

the requirements in [SP800-90] and FIPS 140-2 [FIPS140-2] approved.

6.1.1.8 Cryptographic operation (FCS_COP.1f)

FCS_COP.1.1 The TSF shall perform **digital signature generation for certificate signing** in accordance with a specified cryptographic algorithm

a) RSA with SHA-1, -224, -256, -384, and -512 message digest

b) ECDSA with SHA-1, -224, -256, -384, and -512 message digest

and cryptographic key sizes

a) 1024, 1536, 2048, 3072 or 4096 Bits (RSA)

b) 224, 256, 384, and 512 Bits for curves generated over prime fields (ECDSA prime moduli)

that meet the following:

conformant to FIPS 186-3 [FIPS186-3] (RSA and ECDSA) and FIPS 180-3 [FIPS180-3] (SHA-1, -224, -256, -384, and -512) respectively, and FIPS 140-2 level 1 [FIPS140-2] approved.

6.1.1.9 Cryptographic operation (FCS_COP.1g)

FCS_COP.1.1 The TSF shall perform **digital signature verification for certificate validation** in accordance with a specified cryptographic algorithm

a) RSA with MD2 message digest,

b) RSA with MD5 message digest,

c) RSA with SHA-1, -224, -256, -384 or -512 message digest,

d) DSA with SHA-1 message digest,

e) ECDSA with SHA-1, -224, -256, -384, and -512 message digest,

and cryptographic key sizes

a) 1024, 1536, 2048, 3072, 4096, or 8192 Bits (RSA),

b) 1024, 1536, 2048, 3027, 4096, or 8192 Bits (RSA),

c) 1024, 1536, 2048, 3072, 4096, or 8192 Bits (RSA),

d) 1024, 1536, 2048, or 3072 Bits (DH prime moduli),

e) 224, 256, 384, and 512 Bits for curves generated over prime fields (ECDSA prime moduli)

that meet the following:

a) conformant to RFC 2437 [RFC2437] , RFC 3447 [RFC3447], and RFC 2313 [RFC2313] (RSA) and RFC 1319 [RFC1319] (MD2),

b) conformant to RFC 2437 [RFC2437] , RFC 3447 [RFC3447], and RFC 2313 [RFC2313] (RSA) and RFC 1321 [RFC1321] (MD5),

c) conformant to FIPS 186-3 [FIPS186-3] (RSA) and FIPS 180-3 [FIPS180-3] (SHA-1, -224, -256, -384, and -512) respectively, and FIPS 140-2 level 1 [FIPS140-2] approved

Copyright © 2012 by atsec information security and IBM Corporation

d) conformant to FIPS 186-3 [FIPS186-3] (DSA) and FIPS 180-3 [FIPS180-3] (SHA-1) respectively, and DSA with 1024 Bits modulus and SHA-1 FIPS 140-2 level 1 [FIPS140-2] approved,

e) conformant to FIPS 186-3 [FIPS186-3] (ECDSA) and FIPS 180-3 [FIPS180-3] (SHA-1, -224, -256, -384, and -512) respectively, and FIPS 140-2 level 1 [FIPS140-2] approved.

6.1.1.10 Cryptographic operation (FCS_COP.1h)

FCS_COP.1.1 The TSF shall perform **pair-wise key establishment (key agreement)** in accordance with a specified cryptographic algorithm

a) ECDH

and cryptographic key sizes

a) 224, 256, 384, and 512 Bits for curves generated over prime fields (ECDH prime moduli)

that meet the following:

a) NIST SP 800-56A [SP800-56A],

and FIPS 140-2 [FIPS140-2] approved.

6.1.2 User Data Protection (FDP)

6.1.2.1 FDP_ACC.1 Subset access control

FDP_ACC.1.1 The TSF shall enforce the **TLS connection policy** on **external entities as subjects, TLS (as defined in [TLSv1] or [TLSv1.1] with [TLS_AES], or as defined in [TLSv1.2]) sessions as objects, and session establishment as the operation**.

6.1.2.2 FDP_ACF.1 Security attribute based access control

FDP_ACF.1.1 The TSF shall enforce the **TLS connection policy** to objects based on the following:

- **the public key certificate (as defined in ITU-T Recommendation X.509 [X.509]) as security attribute associated with the communication partner as subject,**
- **on TLS connections as objects with the associated security attributes of cipher suite used, mode (client mode or server mode), in server mode: authentication type (whether client authentication is required or not), and whether a CRL and/or OCSP responses are used for certificate validation**.

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

a) if operated in client mode, the session is established only if the certificate (and its path) presented by the communication partner can be validated by the TOE in accordance with [X.509] and [RFC3280] and the Standard Certificate Policy described in section 7.1.3.2, and when the communication partner proves that it has the associated private key.

b) If operated in server mode and the TOE is not configured for client authentication, no rule applies.

c) If operated in server mode and the TOE is configured for client authentication, the session is established if the certificate (and its path) presented by the communication partner can be validated by

Copyright © 2012 by atsec information security and IBM Corporation

the TOE in accordance with [X.509] and [RFC3280] and the Standard Certificate Policy described in section 7.1.3.2 of this ST, and when the communication partner proves that it has the associated private key. If the client presents a certificate that cannot be validated by the TOE or that has expired, the connection is refused. If the client does not present any certificate an indicator is set that allows the application using the TOE to decide if the connection is established.

FDP_ACF.1.3 The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **none**.

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the following additional rules:

a) The TOE will not establish a TLS session if the communication partner does not support a cipher suite provided by the TOE. The following cipher suites are provided as defined in the identified standards:

- **TLS_RSA_WITH_3DES_EDE_CBC_SHA = { 0x00,0x0A } [TLSv1]**
- **TLS_RSA_WITH_AES_128_CBC_SHA = { 0x00, 0x2F } [RFC3268]**
- **TLS_RSA_WITH_AES_256_CBC_SHA = { 0x00, 0x35 } [RFC3268]**
- **TLS_RSA_WITH_AES_128_CBC_SHA256 = { 0x00, 0x3C } [RFC3268]**
- **TLS_RSA_WITH_AES_256_CBC_SHA256 = { 0x00, 0x3D } [RFC3268]**
- **TLS_RSA_WITH_AES_128_GCM_SHA256 = { 0x00, 0x9C } [RFC3268]**
- **TLS_RSA_WITH_AES_256_GCM_SHA384 = { 0x00, 0x9D } [RFC3268]**
- **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 = {0xC0,0x2F} [RFC5289]**
- **TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 = {0xC0,0x27} [RFC5289]**
- **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 = {0xC0,0x30} [RFC5289]**
- **TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 = {0xC0,0x28} [RFC5289]**
- **TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA = { 0xC0, 0x09 } [RFC4492]**
- **TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA = { 0xC0, 0x0A } [RFC4492]**
- **TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA = {0xC0, 0x12} [RFC4492]**
- **TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA = { 0xC0, 0x08 } [RFC4492]**
- **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 = {0xC0,0x2B} [RFC5289]**
- **TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 = {0xC0,0x23} [RFC5289]**

Copyright © 2012 by atsec information security and IBM Corporation

- **TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 = {0xC0,0x2C} [RFC5289]**
- **TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 = {0xC0,0x24} [RFC5289]**

**b) If OCSP usage has been configured for certificate validation:
if the TOE is in client mode or
if the TOE is in server mode and configured for client authentication, the connection is denied if the OCSP response is not conformant with [RFC2560], [RFC2560] augmented by [RFC6277], or [RFC5019], and with the Standard OCSP Policy described in section 7.1.3.2 of this ST, or if the OCSP response indicates that the certificate has been revoked.**

**c) If CRL usage has been configured for certificate validation, and OCSP has not been configured for usage or the revocation status is undetermined after using OCSP:
if the TOE is in client mode or
if the TOE is in server mode and configured for client authentication, the connection is denied if the CRL is not conformant with [X.509] and the Standard CRL Policy described in section 7.1.3.2 of this ST, or if the CRL indicates that the certificate has been revoked.**

d) If the use of the Certificate Status Request extension [RFC6066] is configured in client mode, the connection is denied if the OCSP response included in the server's response is not conformant with [RFC2560], [RFC2560] augmented by [RFC6277], or [RFC5019], and with the Standard OCSP Policy described in section 7.1.3.2 of this ST, or if the OCSP response indicates that the certificate has been revoked.

e) If the TOE is configured to handle the Server Name Indication extension [RFC6066] as critical, then the connection is denied if the TOE is either in client mode and the server does not satisfy the request, or the TOE is in server mode and cannot match a client request.

f) If the TOE operates in client mode and the server does not offer the renegotiation_info extension [RFC5746], or if the TOE operates in server mode and the client does not offer the renegotiation_info extension or TLS_EMPTY_RENEGOTIATION_INFO_SCSV Signaling Cipher Suite Value [RFC5746], the connection is denied.

g) A renegotiation is aborted if the peer (server if in client mode, and vice versa) presents a different certificate or identity.

h) If the TOE is operated in Suite B mode, a connection that does not follow the Suite B compliant profile [RFC6460] for session establishment is denied.

Application note: The rules for the communication partner connecting to the channel set up by the TOE are not laid out in FDP_ACF.1.2 since it is not within the scope of the TOE to care about the connection policy of the remote program. Nevertheless, the default rules can easily be derived from FDP_ACF.1.2 by choosing the alternate role (TLS client or server) for the communication partner.

Application note: The TLS standards do not require that the connection establishment is terminated in the case where a server is configured for client authentication and the client does not present any certificate. This allows applications to handle this case individually e. g. by presenting an error message to the client or offering the client a reduced set of services. The TOE allows the application to check if the client has been successfully authenticated and then lets the application decide whether to shut down the connection, inform the client about the required authentication or offer the client a reduced set of services. Quite a number of web based

Copyright © 2012 by atsec information security and IBM Corporation

applications want to adapt their services depending on the fact if the client has been authenticated or not.

6.1.2.3 Basic data authentication (FDP_DAU.1)

FDP_DAU.1.1 The TSF shall provide a capability to generate evidence that can be used as a guarantee of the validity of **the X.509 [X.509] conformant public key certificate *by signing it using a digital signature***.

FDP_DAU.1.2 The TSF shall provide **all subjects** with the ability to verify evidence of the validity of the indicated information ***by verifying a certificate's digital signature***.

6.1.2.4 Full residual information protection (FDP_RIP.2a)

FDP_RIP.2.1 The TSF shall ensure that ***the state information of the random number generator*** is made unavailable upon the **deallocation of the *random number generator* from** all objects.

6.1.2.5 Full residual information protection (FDP_RIP.2b)

FDP_RIP.2.1 The TSF shall ensure that any previous ***sensitive*** information content of ***the buffer space memory (in particular, all cleartext critical security parameters, random numbers and cryptographic keys)*** is made unavailable upon the **deallocation of the *buffer space memory* from** all objects.

6.1.2.6 Basic data exchange confidentiality (FDP_UCT.1)

FDP_UCT.1.1 The TSF shall enforce the **TLS connection policy** to **transmit and receive** user data in a manner protected from unauthorized disclosure.

6.1.2.7 Data exchange integrity (FDP_UIT.1)

FDP_UIT.1.1 The TSF shall enforce the **TLS connection policy** to **transmit and receive** user data in a manner protected from **modification, insertion and replay** errors.

FDP_UIT.1.2 The TSF shall be able to determine on receipt of user data, whether **modification, insertion or replay** has occurred.

6.1.3 Identification and Authentication

6.1.3.1 Verification of secrets (FIA_SOS.1)

FIA_SOS.1.1 The TSF shall provide a mechanism to verify that secrets ***used to protect the keystore*** meet **the following metric:**

- **The minimum password length is 14 characters**
- **A password needs to have at least one lower case character, one upper case character, and one digit or special character.**
- **Each character should not occur more than three times in a password.**
- **No more than two consecutive characters of the password should be identical.**

6.1.3.2 TSF Generation of secrets (FIA_SOS.2)

FIA_SOS.2.1 The TSF shall provide a mechanism to generate secrets that meet **the following metric:**

- **the password length is at least 14 characters**
- **the characters are chosen from the set of upper case alpha, lower case alpha, numeric and punctuation characters i.e. 95 characters of the 7-bit ASCII set,**

Copyright © 2012 by atsec information security and IBM Corporation

- **the password is cryptographically random, i.e. generated with the FIPS 140-2 [FIPS140-2] approved DRBG 800-90**.

FIA_SOS.2.2 The TSF shall be able to enforce the use of TSF generated secrets for

no TSF functions.

6.1.3.3 User authentication before any action (FIA_UAU.2)

FIA_UAU.2.1 The TSF shall require each user to be successfully authenticated before allowing ***any TSF-mediated access to the keystore*** on behalf of that user.

6.1.4 Security Management (FMT)

6.1.4.1 Static attribute initialization (FMT_MSA.3)

FMT_MSA.3.1 The TSF shall enforce the **TLS connection policy** to provide **permissive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow the **Crypto-Officer** to specify alternative initial values to override the default values when an object or information is created.

Application note: The TOE will allow a connection when the rules of the TLS connection policy are satisfied. It is up to the policy enforced by the operational environment to define additional restrictions. The operational environment is also responsible for enforcing those additional restrictions.

6.1.4.2 Specification of Management Functions (FMT_SMF.1)

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions ***via the key management API and CLI***:

- **for keystores: create, delete, changepw, list;**
- **for certificates: create, add, delete, (view) details, export, receive, import, extract, rename, list, modify, sign, validate, set certificate trust status;**
- **for certificate requests: create, delete, (view) details, extract, list, recreate;**
- **for passwords: create**.

6.1.5 Protection of the TSF (FPT)

6.1.5.1 Basic external TSF data self-protection (FPT_ETT_GSK.1a)

FPT_ETT_GSK.1.1 The TSF shall protect **a private key** from **disclosure and undetected modification** when it is transmitted outside of the TOE ***for storage in a keystore***.

FPT_ETT_GSK.1.2 The TSF shall use **the password privacy mode defined in PKCS #12 [PKCS#12] (SHA-1 and 3-key TDEA CBC mode)**.

6.1.5.2 Basic external TSF data self-protection (FPT_ETT_GSK.1b)

FPT_ETT_GSK.1.1 The TSF shall protect **a certificate with the associated trust status and certificate request data** from **undetected modification** when it is transmitted outside of the TOE ***for storage in a CMS V4 keystore***.

FPT_ETT_GSK.1.2 The TSF shall use **password based HMAC SHA-1 RFC 2104 [RFC2104] (SHA-1).**

6.1.5.3 FPT_FLS.1 Failure with preservation of secure state

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur:

Copyright © 2012 by atsec information security and IBM Corporation

a) TLS handshake failure,

b) failing to correctly decrypt TLS messages received from the communication partner,

c) data integrity failure for TLS messages received from the communication partner,

d) TLS protocol detected buffer sequence errors,

e) integrity errors detected when reading keys from the key file,

f) internal failure intercepted by the TOE,

g) failure of the self tests.

6.1.5.4 Inter-TSF basic TSF data consistency (FPT_TDC.1a)

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret

a. public key certificates,

b. and certificate requests

when shared between the TSF and another trusted IT product ***via the key management API or CLI***.

FPT_TDC.1.2 The TSF shall use

a. the standards X.509 as defined in the ITU-T Recommendation X.509 [X.509] and RFC 3280 [RFC3280];
certificates of the versions 1, 2, and 3 are supported (public key certificates),

b. the standard PKCS #10 as defined in [RFC2986] (certificate requests)

when interpreting the TSF data from another trusted IT product.

6.1.5.5 Inter-TSF basic TSF data consistency (FPT_TDC.1b)

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret

a. public key certificates

b. certificate revocation lists (CRLs),

when shared between the TSF and another trusted IT product ***in a TLS session***.

FPT_TDC.1.2 The TSF shall use

a. the standards X.509 as defined in the ITU-T Recommendation X.509 [X.509] and RFC 3280 [RFC3280];
certificates of the versions 1, 2, and 3 are supported (public key certificates),

b. the standards X.509 as defined in the ITU-T Recommendation X.509 [X.509] and RFC 3280 [RFC3280] following ITU-T Recommendation X.509 [X.509] for the handling of unknown critical extensions in a CRL;
CRLs of the versions 1 and 2 are supported (CRLs),

when interpreting the TSF data from another trusted IT product.

Application Note: Please refer to chpt. 6 for a discussion on the specifics of certificate path validation with optional CRLs.

6.1.5.6 Inter-TSF basic TSF data consistency (FPT_TDC.1c)

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret

Copyright © 2012 by atsec information security and IBM Corporation

a. public key certificates

b. private keys

when shared between the TSF and another trusted IT product ***by importing/exporting them from/to files***.

FPT_TDC.1.2 The TSF shall use **PKCS#12 [PKCS#12] employing password privacy mode and password integrity mode using the same password and pbeWithSHAAnd3-KeyTriple-DES-CBC, supporting KeyBags, PKCS-8ShroudedKeyBags, and/or CertBags,** when interpreting the TSF data from another trusted IT product.

6.1.5.7 Inter-TSF basic TSF data consistency (FPT_TDC.1d)

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret

a. the revocation status of a certificate

when shared between the TSF and another trusted IT product ***via communication with an OCSP responder or via the TLS Certificate Status Request extension***.

FPT_TDC.1.2 The TSF shall use **the Online Certificate Status Protocol [RFC2560], and optionally the Algorithm Agility augmentation [RFC6277], Lightweight OCSP Profile [RFC5019] and the TLS Certificate Status Request extension [RFC6066]** when interpreting the TSF data from another trusted IT product.

6.1.5.8 Inter-TSF basic TSF data consistency (FPT_TDC.1e)

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret

a. public key certificates

b. private keys

c. security attributes (such as the trust status for certificates)

when shared between the TSF and ***a trusted crypto module to retrieve and/or store public key certificates, private keys and security attributes***.

FPT_TDC.1.2 The TSF shall use **PKCS#11 v2.20 [PKCS#11-2.20] augmented up to amendment 3, the MSCAPI [MSCAPI], or MSCNG [MSCNG]** when interpreting the TSF data from another trusted IT product.

6.1.5.9 TSF testing (FPT_TST.1)

FPT_TST.1.1 The TSF shall run a suite of self tests **at the conditions**

- **first call to the SSL environment initialization command of the TOE**

to demonstrate the correct operation of **the TOE ICC component**.

FPT_TST.1.2 The TSF shall provide authorized users the capability to verify the integrity of **private keys and certificates in the keystore** ***by verifying the integrity of the protection measures applied in FPT_ETT_GSK.1b when accessing this data in the keystore***.

FPT_TST.1.3 The TSF shall provide authorized users with the capability to verify the integrity of **the dynamic parts of the ICC module** ***at the conditions identified in FPT_TST.1.1***.

6.1.6 Trusted Path/Channels (FTP)

6.1.6.1 Inter-TSF trusted channel (FTP_ITC_PD-0108.1)

FTP_ITC_PD-0108.1.1 The TSF shall provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and

Copyright © 2012 by atsec information security and IBM Corporation

provides assured identification of its end points and protection of the channel data from modification or disclosure***, specifically by implementing any of the following protocols:***

- ***TLS 1 [TLSv1] with secure renegotiation [RFC5746],***
- ***TLS 1.1 [TLSv1.1] with secure renegotiation [RFC5746],***
- ***TLS 1.2 [TLSv1.2] with secure renegotiation [RFC5746], and***
- ***TLS 1.2 [TLSv1.2] with the Suite B compliant profile [RFC6460] and secure renegotiation [RFC5746]***.

FTP_ITC_PD-0108.1.2 The TSF shall permit **the TSF (when operated in client mode) or the remote IT product (when operated in server mode)** to initiate communication via the trusted channel.

FTP_ITC_PD-0108.1.3 The TSF shall use a trusted channel for the following functions: **data transfer**.

Application Note: The rationale for this explicit requirement is that it corrects an error identified by CCEVS in the requirement and an interpretation is being created by NIAP to correct the offending wording.

6.2 Security Functional Requirements Rationale

6.2.1 Coverage

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

Security Functional Requirements	Objectives
FCS_CKM.1a	O.SECCHANNEL
FCS_CKM.1b	O.SECCHANNEL
FCS_COP.1a	O.SECCHANNEL, O.TRANSFER
FCS_COP.1b	O.SECCHANNEL, O.TRANSFER
FCS_COP.1c	O.KEYSTORE, O.SECCHANNEL
FCS_COP.1d	O.SECCHANNEL
FCS_COP.1e	O.SECCHANNEL
FCS_COP.1f	O.SECCHANNEL
FCS_COP.1g	O.SECCHANNEL
FCS_COP.1h	O.SECCHANNEL
FDP_ACC.1	O.SECCHANNEL
FDP_ACF.1	O.SECCHANNEL
FDP_DAU.1	O.SECCHANNEL
FDP_RIP.2a	O.SESSIONDATA
FDP_RIP.2b	O.SESSIONDATA
FDP_UCT.1	O.SECCHANNEL
FDP_UIT.1	O.SECCHANNEL
FIA_SOS.1	O.KEYSTORE
FIA_SOS.2	O.KEYSTORE
FIA_UAU.2	O.KEYSTORE
FMT_MSA.3	O.SECCHANNEL
FMT_SMF.1	O.KEYSTORE, O.SECCHANNEL, O.TRANSFER
FPT_ETT_GSK.1a	O.KEYSTORE

Copyright © 2012 by atsec information security and IBM Corporation

Security Functional Requirements	Objectives
FPT_ETT_GSK.1b	O.KEYSTORE
FPT_FLS.1	O.SELFSEC
FPT_TDC.1a	O.SECCHANNEL
FPT_TDC.1b	O.SECCHANNEL
FPT_TDC.1c	O.TRANSFER
FPT_TDC.1d	O.SECCHANNEL
FPT_TDC.1e	O.TRANSFER
FPT_TST.1	O.KEYSTORE, O.SELFSEC
FTP_ITC_PD-0108.1	O.SECCHANNEL

Table 10: Mapping SFRs to objectives

6.2.2 Sufficiency

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

Security objectives	Rationale
O.KEYSTORE	Before the keystore can be accessed, the password must be provided (FIA_UAU.2). This password is generated by the TOE (FIA_SOS.2); users are advised to only use TOE generated passwords as keystore passwords. Furthermore, a quality metric on the passwords is given by a password policy (this password policy cannot guarantee the same randomness of the passwords as the generated passwords) and a newly entered password is checked against that password policy (FIA_SOS.1). Private keys are encrypted using TDEA before being stored in the keystore and are decrypted after retrieval. The password based encryption scheme PKCS #12 Password Based Encryption with SHA-1 and TDEA with three independent keys in CBC mode as described in PKCS #12 [PKCS#12] is used to generate a key from the given password (FPT_ETT_GSK.1a). A HMAC-SHA-1 hash is generated over all data records in the keystore using the password as the HMAC Key. This hash is stored in the keystore in a special record, called the header record, that exists once per keystore file. A HMAC-SHA-1 hash of the header using the password as the HMAC Key is also stored in the keystore, in the header record (FCS_COP.1c, FPT_ETT_GSK.1b). This is further modeled as part of FPT_TST.1 (integrity checking for TSF data). Management functions for the keystore in FMT_SMF.1 contribute to this objective.
O.SECCHANNEL	TLS is implemented to be used to set up a trusted channel which provides confidentiality and integrity protection of transferred data using cryptographic methods (FDP_UCT.1, FDP_UIT.1, FTP_ITC_PD-0108.1). X.509 certificates as defined in [X.509] are needed for authentication during setup of the trusted channels. For the setup of the trusted channel, server authentication is mandatory and client authentication optional, meaning that it can be configured as mandatory. Furthermore, the channel is only set up if the communication partners can agree on a ciphersuite. The TOE can use an LDAP server (if this is provided by the operational environment) to obtain a current CRL compliant to [X.509] during the TLS handshake that it then uses for verification. Several well-know CA certificates are always contained in a newly created keystore and renewal certificates can be received during the setup of the TLS session (FDP_ACC.1, FDP_ACF.1, FMT_MSA.3, FPT_TDC.1a , FPT_TDC.1b). The implementation of the cryptographic algorithms needed for the generation of symmetric keys and secrets for MAC computation, for the generation of asymmetric key pairs (FCS_CKM.1a, FCS_CKM.1b), for the cryptographic operations and for the random number generation are provided by the ICC component (FCS_COP.1a, FCS_COP.1b, FCS_COP.1c, FCS_COP.1d, FCS_COP.1e, FCS_COP.1f, FCS_COP.1g, FCS_COP.1h). All cryptographic algorithms given in the cipherspecs of the TLS cipher suites and the random number generator are FIPS 140-2 [FIPS140-2] approved. Certificates can be imported using API calls, the command line interface or during the setup of an TLS connection (FPT_TDC.1a , FPT_TDC.1b). In addition to that, certificates can be generated by the TOE itself by digitally signing a certificate for a public key such that this digital signature can afterwards be used for verifying the authenticity of that key. Certificate request compliant to PKCS #10 [RFC2986] can be

Copyright © 2012 by atsec information security and IBM Corporation

Security objectives	Rationale
	generated and used to request the generation of certificates. These requests may be stored in and retrieved from the keystore (FCS_COP.1f, FDP_DAU.1, FPT_TDC.1b). Certificates signed with RSA/MD2 or RSA/MD5 can be verified but are not generated (FCS_COP.1g). Management functions for the certificates and keys used in TLS sessions in FMT_SMF.1 contribute to this objective.
O.SELFSEC	Self-tests for the cryptographic module ICC are implemented; they are called upon the first call to the SSL environment initialization command of the TOE. This is a function a product using the TOE has to call for the initialization of the TLS (and SSL, when operated outside the evaluated configuration) environment; this initialization is needed for setting up a TLS connection. Since the self-tests of the ICC component are requested when this function is called for the first time, these tests are conducted before the first use of the cryptographic algorithms supplied by the ICC for TLS (FPT_TST.1). Failures during the operation of the trusted channel are detected and a secure state is reached by either terminating the TLS session or the TOE (FPT_FLS.1).
O.SESSIONDATA	The random number generator's state is reset when the module is unloaded and sensitive information in the buffers of a session is cleared before the buffer is released. Neither the buffer content nor the random number generator state are then any longer available (FDP_RIP.2(a), FDP_RIP.2(b)). Cryptographic keys (symmetric keys and asymmetric key pairs) are destroyed when no longer used (FDP_RIP.2(b)).
O.TRANSFER	This policy is modeled in FPT_TDC.1c, which requires that the TOE is able to exchange public key certificates and private keys based on the PKCS#12 format, supported by cryptographic operations modeled in FCS_COP.1a and (b), and in FPT_TDC.1e, which requires the TOE to be able to interpret public key certificates, private keys and security attributes exchanged via PKCS#11 or the MSCAPI/MSCNG. Management functions for the keystore in FMT_SMF.1 contribute to this objective.

Table 11: SFR sufficiency analysis

6.2.3 Security Requirements Dependency Rationale

Dependencies within the EAL4 package selected for the security assurance requirements have been considered by the authors of CC Part 3 and are not analyzed here again.

The security functional requirements in this Security Target do not introduce dependencies on any security assurance requirement; neither do the security assurance requirements in this Security Target introduce dependencies on any security functional requirement.

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

Security Functional Requirement	Dependencies	Resolution
FCS_CKM.1a	[FCS_CKM.2 or FCS_COP.1]	FCS_COP.1a, FCS_COP.1c
	FCS_CKM.4	Cryptographic session keys for the TLS session are protected by the TOE against unauthorized access and are destroyed by the object re-use functions of the TOE.
FCS_CKM.1b	[FCS_CKM.2 or FCS_COP.1]	FCS_COP.1d, FCS_COP.1f, FCS_COP.1g
	FCS_CKM.4	Long living private keys of a public/private key pair will be destroyed by the object re-use function of the TOE when they are kept in memory.
FCS_COP.1a	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1a

Copyright © 2012 by atsec information security and IBM Corporation

Security Functional Requirement	Dependencies	Resolution
	FCS_CKM.4	Cryptographic session keys for the TLS session are protected by the TOE against unauthorized access and are destroyed by the object re-use functions of the TOE.
FCS_COP.1b	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	Since no cryptographic keys are needed for digest generation and verification, the dependencies dealing with keys (FCS_CKM.1 and FCS_CKM.4) or import of user data without security attributes (FDP_ITC.1) cannot be applied.
	FCS_CKM.4	Digests will be destroyed by the object re-use function of the TOE when they are kept in memory.
FCS_COP.1c	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1a
	FCS_CKM.4	Cryptographic secrets are protected by the TOE against unauthorized access and are destroyed by the object re-use functions of the TOE.
FCS_COP.1d	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1b
	FCS_CKM.4	Long living private keys of a public/private key pair will be destroyed by the object re-use function of the TOE when they are kept in memory.
FCS_COP.1e	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	The random number generator (Universal Software Base TRN algorithm) needs no key.
	FCS_CKM.4	Random numbers will be destroyed by the object re-use function of the TOE when they are kept in memory.
FCS_COP.1f	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1b
	FCS_CKM.4	Cryptographic secrets are protected by the TOE against unauthorized access and are destroyed by the object re-use functions of the TOE. Long living private keys of a public/private key pair will be destroyed by the object re-use function of the TOE when they are kept in memory.
FCS_COP.1g	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	This SFR deals with certificate verification using RSA with MD2/MD5. Since the TOE itself does not generate such certificate (but may need to verify them), no key generation (FCS_COP.1) is needed, furthermore, the import of user data without security attributes as given by FTP_ITC.1 is not used since certificates are security attributes for communication partners. Instead the import of such certificates is covered by FTP_TDC.1.
	FCS_CKM.4	Key destruction as provided by FCS_CKM.4 is not needed since certificates only contain public keys, so no secure way to delete them is needed.

Copyright © 2012 by atsec information security and IBM Corporation

Security Functional Requirement	Dependencies	Resolution
FCS_COP.1h	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	The key agreement mechanism specified here either uses ephemeral keys, or keys stored in a certificate. The import of certificates and access to the keystore is covered by the FTP_TDC.1 requirements. If certificates are used that have been provided by the operational environment based on a certificate request generated by the TOE, FCS_CKM.1b applies.
	FCS_CKM.4	Cryptographic secrets are protected by the TOE against unauthorized access and are destroyed by the object re-use functions of the TOE. Long living private keys of a public/private key pair will be destroyed by the object re-use function of the TOE when they are kept in memory.
FDP_ACC.1	FDP_ACF.1	FDP_ACF.1
FDP_ACF.1	FDP_ACC.1	FDP_ACC.1
	FMT_MSA.3	FMT_MSA.3
FDP_DAU.1	No dependencies	
FDP_RIP.2(a)	No dependencies	
FDP_RIP.2(b)	No dependencies	
FDP_UCT.1	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC_PD-0108.1
	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1
FDP_UIT.1	[FTP_ITC.1 or FTP_TRP.1]	FTP_ITC_PD-0108.1
	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1
FIA_SOS.1	No dependencies	
FIA_SOS.2	No dependencies	
FIA_UAU.2	FIA_UID.1	Authentication is required before access to the keystore is granted, i.e. the password to access the keystore must be known. This authentication is not combined with identification since no identity is associated with the password used to protect the keystore. The TOE implements no knowledge of identities; it relies on the operational environment to provide this where needed. So the TOE grants everyone who enters the correct password access to the keystore.

Copyright © 2012 by atsec information security and IBM Corporation

Security Functional Requirement	Dependencies	Resolution
FMT_MSA.3	FMT_MSA.1	The TOE itself has no knowledge of user roles but relies on the environment to provide this. The operational environment is responsible to restrict access to functions related to the management of cryptographic keys and certificates to a role Crypto-Officer. The operational environment may also define additional roles (e. g. the role of a system administrator), but this is dependent on the specific requirements the operational environment needs to satisfy. The definition of the roles is therefore left to the operational environment and therefore dependencies to FMT_MSA.1 and FMT_SMR.1 are not resolved within the security functional requirements for the TOE but need to be resolved within the environment the TOE is integrated into.
	FMT_SMR.1	See above.
FMT_SMF.1	No dependencies	
FPT_ETT_GSK.1a	No dependencies	
FPT_ETT_GSK.1b	No dependencies	
FPT_FLS.1	No dependencies	
FPT_TDC.1a	No dependencies	
FPT_TDC.1b	No dependencies	
FPT_TDC.1c	No dependencies	
FPT_TDC.1d	No dependencies	
FPT_TDC.1e	No dependencies	
FPT_TST.1	No dependencies	
FTP_ITC_PD-0108.1	No dependencies	

Table 12: Dependencies between TOE Security Functional Requirements

6.3 Security Assurance Requirements

The security assurance requirements for the TOE are the Evaluation Assurance Level 4 components as specified in [CC] Part 3. No operations are applied to the assurance components.

6.4 Security Assurance Requirements Rationale

The evaluation assurance level has been chosen commensurate with the threat environment that is experienced by typical consumers of the TOE.

Copyright © 2012 by atsec information security and IBM Corporation

7 TOE Summary Specification

7.1 TOE Security Functionality

7.1.1 Key Management (KEYMAN)

As a general note, when performing key and certificate management, and when performing a TLS environment initialization, the TOE always requires the user to specify the keystore (or crypto module) to be used. In essence, this means that multiple keystores and modules can exist in the environment, and it is up to the user to identify which of these to operate on. The TOE's functions always operate on one keystore or module at a time.

7.1.1.1 KEYMAN.1: Key generation

The following services are provided for key generation:

- symmetric key and secret generation: a symmetric key or a secret for MAC computation is generated. The supported keys are 168 Bits TDEA keys, 128 or 256 Bits AES keys, and 160 or 256 Bits secrets for HMACs created based on SHA-1 or SHA-256.

  The keys are generated conformant to TLSv1 [TLSv1] and TLSv1.1 [TLSv1.1] with [TLS_AES], and to TLSv1.2 [TLSv1.2] (which includes AES) respectively. This is done as follows: first, random numbers as exchanged in the ClientHello and ServerHello messages and in a premaster secret generated by the client (which consists of the protocol version number and a random number). Then, a master secret is computed from the premaster secret and the random numbers of the ClientHello and ServerHello messages. This master secret is finally used for the generation of keys and secrets for MAC computation.

  The operations used are (see CRYPTO.6, CRYPTO.2):

  - o random number generation for the random numbers used in the ClientHello message, the ServerHello messages and the premaster secret,
  - o RSA encryption/decryption for the encryption/decryption of the premaster secret (see CRYPTO.4),
  - o MD5 and SHA-1 in TLSv1 and TLSv1.1, and SHA-256 in TLSv1.2, for the computation of the master secret, and
  - o MD5 and SHA-1 in TLSv1 and TLSv1.1, and SHA-256 in TLSv1.2, for the generation of keys and secrets for MAC computation, using the master secret.
- asymmetric key generation: an asymmetric key pair is generated. The supported key sizes are 1024, 1536, 2048, 3072 and 4096 Bits RSA keys (conformant to ANSI X9.31 [X9.31]). Each prime is about half the size of the modulus; two independently generated random numbers are used to ensure that the probability of using close primes is negligible; checks include Miller-Rabin tests (3 rounds for 2048 bit modulus, 6 rounds for 1024 bit modulus) and small-factor tests.

  The TOE also generates keys over the elliptic curves specified in [FIPS186-3].

7.1.1.2 KEYMAN.2: Not defined

KEYMAN.2 is intentionally not defined.

7.1.1.3 KEYMAN.3: Keystore access protection

Access to any CMS keystore is password-protected: the password must be entered correctly before any access is granted to the keystore. Note that this password is also used to generate the keys for the encryption and signing of data records as described in KEYMAN.4 and is enforced by a password policy as described in KEYMAN.5.

7.1.1.4 KEYMAN.4: Keystore content protection

Private keys that are to be stored in a CMS keystore are encrypted and hashed prior to storage and decrypted (and the hash sum verified) after retrieval. This ensures their confidentiality and integrity.

Copyright © 2012 by atsec information security and IBM Corporation

In CMS V4, certificates together with their trust status and certificate requests are stored unencrypted for faster access. For integrity protection, the TOE generates a HMAC-SHA-1 hash over all data records by hashing all records using the password as the HMAC Key. This hash is then stored in the keystore in a special record that exists once in every keystore file, a so-called header record. A HMAC-SHA-1 hash of the header using the password as the HMAC Key is also stored in the header record in the keystore.

In CMS V5, all key material is protected using the Privacy Mode and Password Integrity Mode defined in PKCS#12.

Both CMS keystore versions implement encryption and hashing as defined inPKCS #12, with SHA-1 and TDEA with three independent keys in CBC mode. CMS V4 only applies this protection to keys, as described above, and then generates a SHA-1 hash over the complete keystore, while CMS V5 applies it to all cryptographic material stored in the keystore, and does not generate an additional has over the complete keystore.

7.1.1.5 KEYMAN.5: Keystore password quality

The TOE enforces a password policy, describing the quality of the passwords that are used for the password based encryption scheme to encrypt private keys prior to storage in a CMS keystore and to decrypt them after retrieval.

The rules of the password policy are the following:

- The minimum password length is 14 characters.
- A password needs to have at least one lower case character, one upper case character, and one digit or special character.
- Each character of the password should not occur more than three times in a password.
- No more than two consecutive characters of the password should be identical.

Furthermore, the password characters must be randomly chosen.

The TOE offers functionality to generate random passwords using the TOE's cryptographic random number generator, consisting of alphanumeric and punctuation characters. Users must use this functionality to generate the passwords that are used for keystores in order to achieve sufficiently random passwords (since it is assumed that human users would not be able to manually define random passwords with sufficient entropy).

When passwords are generated via the Key Management CLI, the CLI will verify that those generated password also meet the policy described above before returning them to the user, in order to reduce the likelihood that the randomly generated password might be susceptible to dictionary attacks.

When passwords are generated via the Key Management API, this policy check is not automatically performed before the password is returned. Rather, a separate API call has to be issued to check the password’s compliance with the above policy. (This is pointed out in the guidance.)

7.1.1.6 KEYMAN.6: Not defined

KEYMAN.6 is intentionally not defined.

7.1.1.7 KEYMAN.7: Certificate generation and validation

Public key certificates that are conformant to the Standard X.509 [X.509] are used for TLS connections; certificates of the versions 1, 2, and 3 are supported. For intermediate CAs, only version 3 certificates are supported.

These certificates can be imported and exported using the Key Management API and the CLI. Furthermore, public key certificates can be imported during the setup of an TLS connection. This allows the TOE to use a larger external public key infrastructure to establish trust in the authenticity of public keys received from external parties.

Several well-known CA certificates (IBM, Thawte, and Verisign) are always contained in a newly created keystore and renewal certificates can be imported as described above.

Public key certificates can be generated using RSA, DSA, or ECDSA, with SHA-1, -224, -256, -384, and -512; the generated certificates are conformant to X.509 [X.509]; certificates of versions 1, 2, and

Copyright © 2012 by atsec information security and IBM Corporation

3 are supported. To generate such a certificate, the public key is digitally signed such that this digital signature can afterwards be used for verifying the authenticity of that key.

Certificate request can be generated and used to request the signing of the public key used to create a certificate. These certificate requests are conformant to PKCS #10 [RFC2986]. They can be stored in or retrieved from the keystore.

The certification path validation algorithm is done as described in [RFC3280]. In addition to the algorithms and key lengths supported for certificate generation, in order to be able to deal with certificates that are signed using RSA with 4096 or 8192 bits, and/or using MD2 or MD5, the TOE is able to verify such signatures, but the MD2 and MD5 algorithms, as well as 4096 and 8192 bit-RSA keys are not used by the TOE for certificate generation.

This is supported by CRYPTO.5.

7.1.1.8 KEYMAN.8: PKCS#12 file im- and export

The TOE imports cryptographic material into its keystore from PKCS#12-formatted files [PKCS#12] employing password privacy mode and password integrity mode using the same password and pbeWithSHAAnd3-KeyTriple-DES-CBC, supporting KeyBags, PKCS-8ShroudedKeyBags, and/or CertBags.

The TOE exports cryptographic material from its keystore into PKCS#12-formatted files employing privacy mode and password integrity mode using the same password and pbeWithSHAAnd3-KeyTriple-DES-CBC, supporting KeyBags, PKCS-8ShroudedKeyBags, and/or CertBags.

7.1.1.9 KEYMAN.9: PKCS#11 and MSCAPI/MSCNG support

The TOE implements the PKCS#11 protocol specified in [PKCS#11-2.20] augmented up to amendment 3 to access cryptographic modules in the operational environment in order to retrieve and/or store public key certificates, private keys and security attributes, and to request the execution of cryptographic primitives.

On Windows, the TOE also supports the MSCAPI/MSCNG to access hardware or software cryptographic service providers in the operational environment for these operations.

In CC mode, i.e. in the evaluated configuration, GSKit uses only certificates held on PKCS#11 FIPS 140-2 certified devices that have the CKA_TRUSTED attribute set or are PIN/password protected (as indicated by the CKA_PRIVATE attribute). For access to cryptographic services providers via the MSCAPI/MSCNG on Windows, GSKit relies on the access control mechanisms enforced by Windows.

If the crypto module in the operational environment requires a PIN or password, GSKit can temporarily store the user-supplied credential in memory during run-time and hand it down to the module when required.

This functionality satisfies the following SFRs:

- KEYMAN.1:
  - FCS_CKM.1 (Cryptographic key generation)
    - FCS_CKM.1a applies to the generation of symmetric keys
    - FCS_CKM.1b applies to the generation of asymmetric keys
- KEYMAN.3:
  - FIA_UAU.2 (User authentication before any action)
- KEYMAN.4:
  - FCS_COP.1 (Cryptographic operation)
    - FCS_COP.1a applies to TDEA encryption
    - FCS_COP.1c applies to HMAC SHA-1 / -256
  - FPT_ETT_GSK.1 (Basic external TSF data transfer protection)
    - FPT_ETT_GSK.1a applies to private keys
    - FPT_ETT_GSK.1b applies to certificates
  - FPT_TST.1 (TSF protection)
- KEYMAN.5:

Copyright © 2012 by atsec information security and IBM Corporation

  - FIA_SOS.2 (TSF Generation of secrets)
  - FIA_SOS.1 (Verification of secrets)
- KEYMAN.7:
  - FPT_TDC.1a (Inter-TSF basic TSF data consistency)
  - FCS_COP.1 (Cryptographic operation)
    - FCS_COP.1f for RSA, DSA, ECDSA with SHA
    - FCS_COP.1g for (additional) MD2 and MD5 and RSA 4096 bit keys
  - FDP_DAU.1 (Basic data authentication)
- KEYMAN.8:
  - FCS_COP.1 (Cryptographic operation)
    - FCS_COP.1a for TDEA
    - FCS_COP.1b for SHA and MD5
  - FPT_TDC.1c (Inter-TSF basic TSF data consistency)
- KEYMAN.9:
  - FPT_TDC.1e (Inter-TSF basic TSF data consistency)

7.1.2 Cryptographic Algorithms (CRYPTO)

7.1.2.1 CRYPTO.1: Symmetric algorithms

The algorithms for symmetric encryption and decryption are:

- TDEA (Triple DES) with three independent keys – CBC mode; 168 Bits as defined in FIPS 46-3 [FIPS46-3] (TDEA) and FIPS 81 [FIPS81] (CBC mode)
- AES - CBC mode; 128 & 256 Bits as defined in FIPS 197 [FIPS197] (AES, CBC mode)
- AES – GCM mode; 128 & 256 Bits as defined in FIPS 197 [FIPS197] and [SP800-38D] (GCM)

All algorithms are FIPS 140-2 [FIPS140-2] approved.

7.1.2.2 CRYPTO.2: Digest generation/verification

The algorithms for digest generation and verification are:

- SHA-1, -228, -256, -384, and -512 as defined in FIPS 180-3 [FIPS180-3]
- MD5 as defined in RFC 1321 [RFC1321]

The SHA algorithms are FIPS 140-2 [FIPS140-2] approved.

MD5 is not used within the cipherspec parts of the TLS ciphersuites since it is not FIPS approved but it is used together with HMAC for the setup of TLSv1 and TLSv1.1 connections (see also [FIPS140IG]).

7.1.2.3 CRYPTO.3: Data authentication

The algorithm for data authentication is HMAC with SHA-1, -228, -256, -384, and -512; 160, 228, 256, 384, or 512 Bits as defined in FIPS 198 [FIPS198] (HMAC) and FIPS 180-3 [FIPS180-3] (SHA-1).

7.1.2.4 CRYPTO.4: Session key agreement

The algorithm used for the encryption/decryption needed for exchange of the premaster secret used to generate the session key for the TLS connection when using RSA is:

- RSA: 1024, 1536, 2048, 3072, 4096, or 8192 Bits as defined in [TLSv1] and [TLSv1.1], or as in [TLSv1.2]

RSA encryption/decryption which is not FIPS 140-2 [FIPS140-2] approved is used for this purpose; the generation of the session key is an operation covered by the key exchange algorithm parts (not the cipherspec parts) given in TLS ciphersuites.

Copyright © 2012 by atsec information security and IBM Corporation

The TOE also supports ECDH-based key agreement as required by supported TLS cipher suites, with key sizes of 224, 256, 384 and 512 bits (curves generated over prime fields) for ECDH. ECDH is implemented according to [SP800-56A] and are FIPS 140-2 [FIPS140-2] approved.

7.1.2.5 CRYPTO.5: Signature generation and verification

The TOE can generate and verify electronic signatures in support of certificate creation and verification.

RSA, and ECDSA signatures can be generated and verified, with SHA-1, -224, -256, -384, or -512 bit message digests as defined in FIPS 180-3 [FIPS180-3] (SHA); 1024, 1536, 2048, 3072, 4096, or 8192 (verification only) Bits for RSA and as defined in RFC 2437 [RFC2437], RFC 3447 [RFC3447], and RFC 2313 [RFC2313] (RSA), 1024, 1536, 2048, or 3072 bits for DSA and as defined in [FIPS186-3], and 224, 256, 384 and 521 bits (curves generated over prime fields) for ECDSA implemented according to [FIPS186-3]. DSA is only used for signature verification, not generation.

Signatures using other algorithms or key sizes are not generated by the TOE but they can be verified, the additionally supported algorithms and key sizes are

- MD5 as defined in RFC 1321 [RFC1321] (MD5), and
- MD2 as defined in RFC 1319 [RFC1319] (MD2).

The RSA, DSA and ECDSA signature generation/verification with SHA is FIPS 140-2 [FIPS140-2] approved except for DSA modulus sizes of 1536, 2048, and 3072 bits; the RSA signature verification with MD2 or MD5 is not.

7.1.2.6 CRYPTO.6: Random number generation

A random number is generated using ICC's DRBG 800-90, which implements a deterministic random bit generation compliant with [SP800-90]

The DRBG and SHA-1 are FIPS 140-2 [FIPS140-2] approved.

This functionality satisfies the following SFRs:

- CRYPTO.1: FCS_COP.1 (Cryptographic operation)
  - FCS_COP.1a
- CRYPTO.2: FCS_COP.1 (Cryptographic operation)
  - FCS_COP.1b
- CRYPTO.3: FCS_COP.1 (Cryptographic operation)
  - FCS_COP.1c
- CRYPTO 4: FCS_COP.1 (Cryptographic operation)
  - FCS_COP.1d
  - FCS_COP.1h
- CRYPTO.5: FCS_COP.1 (Cryptographic operation)
  - FCS_COP.1f for signature generation
  - FCS_COP.1g for signature verification
- CRYPTO.6: FCS_COP.1 (Cryptographic operation)
  - FCS_COP.1e for random number generation

7.1.3 Secure Channels (SECCHAN)

7.1.3.1 SECCHAN.1: TLS

GSKit offers the possibility to set up secure/trusted channels using the TLS protocol versions 1, 1.1 and 1.2. SSL is disabled by initializing the TOE in FIPS mode. These channels offer confidentiality and integrity protection of the data transmitted over them. In other words: modification of the transmitted data, insertion of additional data, or later replay of captured data is prevented by using cryptographic means (encryption, keyed MACs). This functionality is accessible through the SSL API

Copyright © 2012 by atsec information security and IBM Corporation

of GSKit. It is both possible to write client- and server-applications using this API and the libraries are capable of multi-threading.

The TOE implements the TLS protocol as defined in [TLSv1] and [TLSv1.1] with the extension of the AES ciphersuites as defined in RFC 3268 [TLS_AES], or as defined in [TLSv1.2]. When there are multiple options, not all options are implemented. This applies to the supplied cipher suites (not all possible cipher suites are supported), the authentication (total anonymity mode must not be used, i.e. the server always has to authenticate), and the key agreement using RSA and DH (FORTEZZA is not supported).

The cipherspec parts of TLS cipher suites provided by the TOE in FIPS mode contain only the algorithms that are FIPS 140-2 [FIPS140-2] approved (see the security functions of CRYPTO). The cipher suites listed in section 1.3.2.1 are supported.

When establishing a TLS session using GSKit, the TOE handles a number of security attributes in a permissive way:

- ciphersuite: Unless otherwise specified, all cipher specs available in the evaluated configuration are allowed.
- Certificate Status Request extension (client mode only): Unless otherwise specified, this extension is not expected/necessary in order to establish a session.
- CRL usage: Unless otherwise specified, CRLs are not being checked.
- OCSP usage: Unless otherwise specified, OCSP is not being used.
- Server Name Indication extension: Unless specified as critical, the extension is not required.
- Extended Random Extension: Unless specified as critical, the extension is not required.

7.1.3.2 SECCHAN.2: Certificate validation in TLS sessions

For authentication, X.509 certificates as defined in the X.509 standard [X.509] are used.

The TOE may operate in server or client mode.

When the TOE is in client mode, server authentication (with a valid X.509 certificate and the associated private key) is required for the successful completion of the TLS channels. This means that the session is only established if the certificate of the communication partner can be validated by the TOE and the communication partner proves that it has the associated private key.

When the TOE is in server mode, it can be configured whether the TOE requires client authentication. If the TOE is configured to require client authentication, client authentication (with a valid X.509 certificate and the associated private key) is needed for the successful establishment of the TLS channels without restrictions, i.e. the TLS session is established if the certificate of the communication partner can be validated by the TOE and the communication partner proves that it has the associated private key. If the communication partner presents a certificate that the TOE cannot validate or a certificate the TOE verifies as invalid (due to expiration or due to being revoked), the session is terminated. If the TOE is configured to require client authentication and the client does not present any certificate the TOE establishes the session and sets a flag indicating that the communication partner did not present a certificate. The application using the TOE can check this flag and decide to terminate the session, send an error message to the communication partner and then terminate the session or to continue the session with a set of services adapted to the fact that the communication partner did not authenticate itself. If the TOE requires no client authentication, no authentication of the communication partner is performed.

The validity of a certificate stored in the TOE's keystore is managed by setting its trust status. The trust status of a certificate is a security attribute that tells whether a certificate is trusted or not. If a certificate is not trusted, it is not considered a valid certificate (i.e. it is invalid).

The path validation algorithm implemented by the TOE is compliant to ITU-T Technical Recommendation X.509 [X.509] and also follows the PKIX path validation logic as specified in [RFC3280].

The TOE does the path validation with respect to the current date and time and for the validation of a certificate, the entire chain is followed to the root of the signing tree, thus any intermediate certificates are considered in the chain validation.

OCSP

Copyright © 2012 by atsec information security and IBM Corporation

The TOE can be configured to use an OCSP responder in the operational environment to query the revocation status of a certificate. The OCSP responder is located either by being pre-configured or via the AIA extension. Responses will be verified and must be signed by an instance that is an OCSP signing authority for the certificate in question known to the TOE, the CA that issued the certificate in question, or an instance who has the id-ad-ocspSigning extension in its certificate and has been issued by the CA that issued the certificate in question.

Through the Certificate Status Request extension defined in [RFC6066], OCSP data can also be delivered during the TLS handshake. If GSKit is operating in client mode, this means it will accept OCSP responses this way from servers, and if it is operating in server mode, it will offer OCSP responses retrieved from an OCSP server to the client if requested.

The TOE supports the following three trust models, as specified in [RFC2560], for the OCSP responder:

- The Responder could be explicitly trusted (i.e., as a trust anchor).
- The CA that issued the certificate signs the OCSP response. In this case the CA uses the same key to sign the OCSP response and the certificate in question.
- The CA issues a certificate to the OCSP responder (Delegated OCSP Responder Model). In this case, the CA uses the same key to sign the OCSP Responder certificate and the certificate in question.

Revocation checking of an OCSP responder is performed per [RFC2560] or [RFC5019]. The AIA or CDP extensions are used to check the revocation status except in the case where the 'id-pkix-ocsp-nocheck' extension is set.

The thisUpdate, nextUpdate and producedAt time elements in the OCSP response are used as follows in determining the validity of the OCSP response:

- The producedAt time is ignored.
- The thisUpdate time is ignored.
- If the nextUpdate time is less than the current time, the response is considered old but still usable as long as no other non-stale information is found by all forms and sources of revocation checking/information. If out of date information is used, GSKit retains the certificates' status as UNDETERMINED for the application. If the nextUpdate time is empty, the response is used as if the nextUpdate time was greater than the current time.

In all cases, GSKit never caches responses and always requests fresh information from the OCSP responder when using [RFC2560], and caches responses until the nextUpdate time when using [RFC5019]. The TOE offers the option of nonce checking to allow or disallow the use of responses that have been cached by the responder and had been previously received, and to prevent replay attacks on the network. GSKit supports nonce checking for requests that it issues and/or for responses received from the responder – it is the application's choice whether or not to enable these.

If the status of a certificate is undetermined due to an unsuccessful attempt to query the OCSP responder, and if CRL checking is enabled, the TOE will attempt to verify the validity of the certificate by using CRLs as described below. If CRL checking is not configured and the certificate status is undetermined, the TOE sets a corresponding marker that can be queried by the applications using the TOE.

The TOE supports the following signature algorithms for OCSP in accordance with [RFC6277]:

- RSA with SHA1, SHA224, SHA256, SHA384, SHA512,
- ECDSA with SHA1, SHA224, SHA256, SHA384, SHA512
- DSA with SHA1

CRLs

The TOE can be configured to do certificate validation with the help of X.509 [X.509] conformant CRLs; CRLs of versions 1 and 2 are supported. The use of CRLs is not mandatory. CRLs can be used both in server mode with client authentication enabled, and in client mode.

If CRLs are to be used for certificate validation, an LDAP client is used to query an LDAP server for CRLs, an HTTP server is queried, or a file is retrieved from the underlying file system, depending on the URI provided in the certificate to be checked. An LDAP server, or HTTP server, must be provided by the environment and the environment must provide current and correct CRLs. In case of LDAP, the TOE accesses the LDAP client via the LDAP client's API. Then the LDAP client establishes a

Copyright © 2012 by atsec information security and IBM Corporation

TCP/IP connection to the LDAP server, retrieves one or more CRLs as result of its query, and passes the CRLs back to the TOE, which uses them as part of the certificate path validation.

If CRLs are configured to be part of the certificate path validation, the following behavior is of particular relevance to their evaluation:

- If the query to the application-specified distribution point fails, the revocation status for the certificate is “revoked”.
- If the query to the application-specified distribution point succeeds and no CRL is returned, the revocation status for the certificate is “undetermined”.
- If multiple CRLs are returned as result of the query, all of them are evaluated until a valid and correctly issued CRL revokes the certificate in question or all CRLs have been processed.
- A CRL is rejected if its validation fails, for example due to a bad signature.
- An indirect CRL is rejected.
- The following cases do not automatically lead to a CRL validation failure:
    - If an unrecognized critical extension is encountered in the crlExtensions field and the certificate is listed in the CRL, the certificate revocation status is “revoked”. If the certificate is not listed in the CRL, the CRL is rejected.
    - If an unrecognized critical entry extension is encountered in the crlEntryExtensions field and the certificate is listed in the CRL, the certificate revocation status is “revoked”. If the certificate is not listed in the CRL, the CRL is rejected.
    - If a CRL is encountered containing an out of date ‘NextUpdate’ value and the certificate is listed in the CRL, the certificate revocation status is “revoked”. If the certificate is not listed in the CRL, the CRL is rejected.

A rejected CRL yields a revocation status of “undetermined”.

Please note that the TOE, being an TLS implementation, does not differentiate between the status “undetermined” and “not revoked” for certificates and establishes a trusted channel if the certificate itself is determined to be valid.

A certificate revocation status of “revoked” results in termination of the TLS session with the entity that provided the certificate.

Distribution points in certificates are evaluated as follows: if a CDP is specified it will only be used if it is an X.500 Directory, FILE or HTTP name since only LDAP, files and HTTP are supported for CRLs. If the CDP is in a format other than an X.500 directory , file URI or HTTP name, the Issuer Name will be used unless the CDP is marked critical in which case GSKit will fail the validation.

For HTTP distribution points, GSKit supports CRL caching per [RFC3280] and [RFC5280], sections 6.3. If enabled, GSKit will interpret HTTP messages with information on how long to cache the CRL provided, and use the HTTP protocol to ask the server for a CRL that is newer than the CRL already cached in GSKit, upon which the server will either return a new CRL or a message signifying that no newer CRL is available (see also [RFC2068], chapter 13, for HTTP caching).

The supported extensions are specified in the certificate, OCSP and CRL policies, which comply with X.509 [X.509]. [GSKTRUST] describes which attributes are evaluated as follows:

Standard Certificate Policy

Supported Fields:

- *OuterSigAlgID*
- *Signature*
- *Version*
- *SerialNumber*
- *InnerSigAlgID*
- *Issuer*
- *Validity*
- *SubjectName*
- *SubjectPublicKeyInfo*

Copyright © 2012 by atsec information security and IBM Corporation

- *IssuerUniqueID*
- *SubjectUniqueID*

Supported Extensions:

- *AuthorityKeyID*
- *AuthorityInfoAccess*
- *SubjectKeyID*
- *IssuerAltName*
- *SubjectAltName*
- *KeyUsage*
- *BasicConstraints*
- *PrivateKeyUsage*
- *CRLDistributionPoints*
  - *DistributionPoint*
    - *DistributionPointName (X.500 Name and LDAP/HTTP/FILE Format URI only)*
    - *NameRelativeToCRLIssuer -- (not supported)*
    - *Reasons -- (ignored)*
    - *CRLIssuer fields*
- *InhibitAnyPolicy*
- *Extended KeyUsage*
- *NameConstraints*
- *CertificatePolicies*
  - *PolicyInformation*
    - *PolicyIdentifier – (including anyPolicy)*
    - *PolicyQualifiers – (not supported)*
- *PolicyMappings*
- *PolicyConstraints*

Standard CRL Policy

Supported Fields:

- *OuterSigAlgID*
- *Signature*
- *Version*
- *InnerSigAlgID*
- *Issuer*
- *ThisUpdate*
- *NextUpdate*
- *RevokedCertificate*
  - *UserCertificate*
  - *RevocationDate*

Supported CRLEntry Extensions:

- *certificateIssuer*

Supported CRL Extensions:

- *AuthorityKeyID*
- *IssuerAltName*
- *CRLNumber*

Copyright © 2012 by atsec information security and IBM Corporation

- *IssuingDistributionPoint*
  - *DistributionPoint*
  - *DistributionPointName*
    - *FullName (X.500 Name and LDAP/HTTP/FILE Format URI only)*
    - *NameRelativeToCRLIssuer -- (not supported)*
  - *Reasons -- (ignored)*
  - *CRLIssuer*
  - *OnlyContainsUserCerts -- (not supported)*
  - *OnlyContainsCACerts -- (not supported)*
  - *OnlySomeReasons -- (not supported)*
  - *IndirectCRL*

Standard OCSP Policy

Request Supported Fields:

- *Signature (Optional)*
- *Version (Version 1 Only)*
- *RequesterName (Optional)*
- *RequestList (single request only)*
  - *CertID*
  - *singleRequestExtensions (not supported)*
- *RequestExtensions*
  - *Nonce (if enabled)*

Response Supported Fields:

- *ResponseStatus*
- *Response*
  - *responseType (id-pkix-ocsp-basic)*
  - *BasicOCSPResponse*
    - *Signature*
    - *Certs*
      - *Extensions*
      - *extendedKeyUsage*
        - *id-kp-OCSPSigning*
      - *id-pkix-ocsp-nocheck*
    - *ResponseData*
      - *Version (Version 1 Only)*
      - *ResponderID (by name or by hash)*
      - *ProducedAt (ignored)*
      - *Responses (multiple responses supported)*
        - *SingleResponse*
          - *certID*
          - *certStatus*
            - *RevokedInfo (ignored)*
          - *thisUpdate (ignored)*
          - *nextUpdate*
          - *singleExtensions (ignored)*
      - *responseExtensions*

Copyright © 2012 by atsec information security and IBM Corporation

- *Nonce (if enabled)*

A TLS connection is not set up if the communication partners cannot find a common cipher suite supported by the TOE. This ensures that there cannot be any unprotected connection in the case that the remote application does not support the cipher suites provided by the TOE in FIPS mode.

This is supported by CRYPTO.5.

7.1.3.3 SECCHAN.3: Residual information protection

Sensitive data must be protected between different sessions. Unprotected (i.e. cleartext) critical security parameters and random numbers are considered such sensitive data and are cleared before the buffer is released. The mechanism for this is that the TSF marks all critical data objects. Before releasing the buffers, the TSF clears all the objects that are marked as critical.

The random number generator clears its state automatically when the module is removed from memory.

Cryptographic material and/or passwords are also erased from memory when:

- executing key management API calls to end operations, close sessions, etc.
- the capicmd CLI clears the passwords that was entered by the user from memory after execution of the requested commands

This functionality satisfies the following SFRs:

- SECCHAN.1:
  - FDP_UCT.1 (Basic data exchange confidentiality)
  - FDP_UIT.1 (Data exchange integrity)
  - FTP_ITC_PD-0108.1 (Inter-TSF trusted channel)
  - FDP_ACF.1 (Security attribute based access control)
- SECCHAN.2:
  - FDP_ACC.1 (Subset access control)
  - FDP_ACF.1 (Security attribute based access control)
  - FMT_MSA.3 (Static attribute initialization)
  - FPT_TDC.1b (Inter-TSF basic TSF data consistency)
  - FPT_TDC.1d (Inter-TSF basic TSF data consistency) – OCSP
- SECCHAN.3:
  - FDP_RIP.2 (Full residual information protection)
    - FDP_RIP.2(a) for the random number generation
    - FDP_RIP.2(b) for the buffers

7.1.4 Self-tests and Failure Handling (STATE)

7.1.4.1 STATE.1: Initialization and keystore access self-tests

The ICC component is able to do self-tests upon initialization. This is not directly available to the user or program (since the ICC API has no external interface) but the self-tests are called automatically with the first call to the SSL environment initialization of the TOE; calling this environment initialization function does the initialization that is needed for setting up a TLS connection. So the self-tests of the ICC component are invoked and conducted before the first use of the cryptographic algorithms of the ICC component for setting up an TLS connection.

If the self-tests fail, the user is notified that the SSL environment initialization failed.

The self-tests check cryptographic operations and the software integrity (an RSA signature verification). The following self-tests are done:

Copyright © 2012 by atsec information security and IBM Corporation

- **Integrity test** of a digital signature (a SHA-256 hash signed with a 2048-bit RSA key) that has been computed over the dynamic parts of the ICC component and that is provided with the static parts of the ICC component.
- **Known answer tests** for encryption/decryption of
  - TDEA in CBC mode
  - AES in CBC mode
  - AES in GCM mode
  - SHA-1, -224, -256, -384, -512 message digest generation (signed with RSA)
  - DSA 1024 bit key (verify signature)
  - RSA 1024 bit key (verify signature & encrypt/decrypt)
  - EC curves B-233, P-384 (verify signatures)
  - HMAC-SHA-1, -224, -256, -384, and -512
  - SHA-1, -224, -256, 384, and -512 digests
  - DRBG (Self tests specified in SP800-90 are carried out on all DRBG modes before use. DRBG modes used as RNG sources in FIPS mode will be 256 bit strength. (HMAC-SHA256 or AES-256 SP800-90). The DRBG's in ICC autoreseed from the internal TRNG at intervals specified in SP800-90 and obtaining new instances of DRBG's will trigger automatic self tests at intervals specified in SP800-90.)
- **Pairwise consistency tests** for
  - RSA signature generation and verification
  - DSA signature generation and verification
  - EC signature generation and verification
- **Pairwise consistency tests** for public/private key generation: the consistency of the keys is tested by the generation and verification of a digital signature. If the signature cannot be verified, the test will fail.
- **Continuous DRBG tests** for the random number generator: generated random number blocks or random number bits sequences are compared. The test will fail if an equal block or bit sequence is generated. A continuous longer term statistical test based on a compression function is used as an ongoing entropy check. > 0.5bits/bit entropy is guaranteed.

Furthermore, the TOE performs the following check when accessing data in the keystore, and will notify the user of errors if these checks fail:

- The integrity of the trust status of certificates and of certificate request data is verified.

7.1.4.2 STATE.2: TLS session failure handling

The TOE will cause a termination of the TLS session (in case of 1, 2, 3, 4 and 6 below), Inability to establish a TLS session (in case of 5 and 6 below), or a disabling of all ICC cryptographic functions (in case 7), thus entering a secure state, if one of the following events occurs:

1. TLS handshake failure, including decryption of encrypted handshake messages and data integrity
2. failing to correctly decrypt TLS application data messages received from the communication partner
3. data integrity failure for TLS application data messages received from the communication partner (including attempted replay of messages)
4. TLS protocol detected buffer sequence errors
5. integrity failures of private keys and certificates read from the keystore
6. internal failure intercepted by the TOE
7. failure of the ICC component's self tests

This functionality satisfies the following SFRs:

- STATE.1: FPT_TST.1 (TSF testing)
- STATE.2: FPT_FLS.1 (Failure with preservation of secure state)

Copyright © 2012 by atsec information security and IBM Corporation

7.1.5 Security Function Management (MGMT)

7.1.5.1 MGMT.1: Key management interfaces

The TOE implements management functions for the management of keys, keystores, certificates, and passwords via its key management API and CLI.

The key management API is a C language API exposing key and certificate management functionality to applications integrating the TOE. In particular, this includes:

- keystore operations
- key and certificate creation
- key and certificate management
- key and certificate information retrieval

A command line utility, GSKCapiCmd, is provided, which is essentially a wrapper around the key management API that provides command line access to the API's functionality to users.

This functionality satisfies the following SFRs:

- MGMT.1: FMT_SMF.1

Copyright © 2012 by atsec information security and IBM Corporation

8 Abbreviations, Terminology and References

8.1 Abbreviations

AES	Advanced Encryption Standard
AIX	Advanced Interactive Executive
ANSI	American National Standards Institute
API	Application Programming Interface
CC	Common Criteria
CD	Compact Disc
CLI	Command Line Interface
CMS	Certificate Management System, as in CMS keystore
CRL	Certificate Revocation List
CSP	Cryptographic Service Provider
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
EE	End-entiry
FIPS	Federal Information Processing Standard
HCM	Hardware Cryptographic Module
ID	Identifier
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISO	International Standards Organization
MSCAPI	Microsoft Crypto API
MSCNG	Microsoft Cryptography API: Next Generation
OSP	Organizational Security Policy
PDF	Portable Data Format
PP	Protection Profile
RSA	Rivest-Shamir-Adleman
SHA	Secure Hash Algorithm
SFR	Security Functional Requirement
SOF	Strength of Function
SSL	Secure Sockets Layer
ST	Security Target
TDEA	Triple Data Encryption Algorithm
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOE	Target of Evaluation

Copyright © 2012 by atsec information security and IBM Corporation

TRNG	True random number generator
TSF	TOE Security Functions
URI	Uniform Resource Identifier

8.2 Terminology

This section contains definitions of technical terms that are used with a meaning specific to this document. Terms defined in the [CC] are not reiterated here, unless stated otherwise.

GSKit: This term serves as an abbreviation for Global Security Kit Version 7d, which is the target of this evaluation.

Key Management API and CLI: GSKit API and CLI used by programs that want to deploy the key/certificate generation and management functionality of GSKit.

ICC: This is an abbreviation for IBM Crypto for C, a library that is used within GSKit.

CMS keystore: A CMS keystore is a collection of flat files in the operational environment (on the underlying system) managed by the TOE to provides secure storage for key and certificate data.

OCSP: Online Certificate Status Protocol, defined in [RFC2560].

SHA-2: Common denominator for the SHA-224, SHA-256, SHA-384 and SHA-512 hash algorithms.

SSL: Secure Sockets Layer; this protocol is implemented by the GSKit and available through the SSL API, but only when GSKit is operated outside of the evaluated configuration. In the evaluated configuration, the use of SSL is disabled.

SSL API: GSKit API used by programs that want to deploy the functionality of SSL or TLS.

TLS: Transport Layer security; this protocol is implemented by the GSKit and available through the SSL API.

User: A user is a human or a product/application using the TOE or the operational environment, e.g. a system user uses the underlying operating system, a TOE user the TOE.

8.3 References

[CC] Common Criteria for Information Technology Security Evaluation. Part 1-3. July 2009. Version 3.1 Revision 3.

[CEM] Common Methodology for Information Technology Security Evaluation. July 2009. Version 3.1 Revision 3.

[DOD] Global Security Kit Delivery Procedure Additional Guidance, Version 7.0 as of 2006-07-20

[PD-0108] CCEVS Precedence PD-0108: FTP_ITC.1.3 Specifies The Functions For Which A Trusted Channel Is Provided, Effective Date: 2004-07-19, Last Modified Date: 2004-08-26.

[EPRAND] EUROPEAN PATENT APPLICATION EP 1 081 591 A2, Random number generator, Application number: 00114754.5, Date of publication: 07.03.2001 Bulletin 2001/10.

[FIPS46-3] FIPS PUB 46-3: DATA ENCRYPTION STANDARD (DES), October 25, 1999.

[FIPS81] FIPS PUB 81: DES MODES OF OPERATION, Issued December 2, 1980, including CHANGE NOTICES 2 and 3

[FIPS140-2] FIPS PUB 140-2: SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES, Issued May 25, 2001, including CHANGE NOTICES (12-03-2002)

[FIPS140IG] Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program, Initial Release: March 28, 2003, Last Update: August 04, 2009.

[FIPS180-3] FIPS PUB 180-3: Secure Hash Standard, October 2008.

Copyright © 2012 by atsec information security and IBM Corporation

[FIPS186-3]	FIPS PUB 186-3: Digital Signature Standard (DSS), June 2009.
[FIPS197]	FIPS PUB 197: Specification for the ADVANCED ENCRYPTION STANDARD (AES), November 26, 2001.
[FIPS198]	FIPS PUB 198: The Keyed-Hash Message Authentication Code (HMAC), March 6, 2002.
[GSKCAPI]	IBM Global Security Kit GSKCapiCmd User's Guide GSKit Version 8, Version 1.4
[GSKCCMODE]	Global Security Kit Common Criteria Mode Operating Guidance for Version 7d Edition March 10, 2005
[GSKTRUST]	IBM Global Security Kit Certificate Validation and Trust Policy Design for version 8, April 22, 2009
[GSKKEY]	IBM Global Security Kit Key Management for C Programmer's Guide Version 8, Edition March 30, 2009.
[GSKINST]	IBM Global Security Kit Global Security Kit Install and Packaging Guide, Version 8, June 3, 2009.
[GSKSSL]	IBM Global Security Kit, Secure Socket Layer for C Programmer's Guide, Version 8, Edition June 3, 2009
[ICCSEC]	IBM Crypto for C (ICC) Version 8.0.0 FIPS 140-2 Non-Proprietary Security Policy, version 1.2, September 30, 2010. Obtained on 2010-12-02 from http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp1433.pdf.
[ICC1433]	FIPS 140-2 Validation Certificate No. 1433. IBM Crypto for C by IBM Corporation (When operated in FIPS mode). Obtained on 2010-12-02 from http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140crt/140crt1433.pdf.
[MSCAPI]	Crypt32.dll Versions. http://msdn.microsoft.com/en-us/library/aa379884%28v=VS.85%29.aspx
[MSCNG]	CNG Features. http://msdn.microsoft.com/en-us/library/bb204775%28v=VS.85%29.aspx
[PKCS#11-2.10]	PKCS #11 v2.10: Cryptographic Token Interface Standard. RSA Laboratories, December 1999.
[PKCS#11-2.20]	PKCS #11 v2.20: Cryptographic Token Interface Standard. RSA Laboratories, 28 June 2004.
[PKCS#11A3R1]	PKCS #11 v2.20 Amendment 3 Revision 1: Additional PKCS#11 Mechanisms. RSA Laboratories. 11 January 2007.
[PKCS#12]	PKCS 12 v1.0: Personal Information Exchange Syntax, RSA Laboratories, June 24, 1999.
[RFC1319]	RFC 1319: The MD2 Message-Digest Algorithm, including the Erratum for RFC 1319, April 1992
[RFC1321]	RFC 1321: The MD5 Message-Digest Algorithm, including the Erratum for RFC 1321, April 1992
[RFC2068]	RFC 2068: Hypertext Transfer Protocol -- HTTP/1.1. January 1997.
[RFC2104]	RFC 2104: HMAC-Keyed-Hashing for Message Authentication, February 1997.
[RFC2313]	RFC 2313: PKCS#1: RSA Cryptography Specification, Version 1.5, March 1998.
[RFC2437]	RFC 2437: PKCS #1: RSA Cryptography Specifications, Version 2.0, October 1998.
[RFC2560]	RFC 2560: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP. June 1999
[RFC2986]	RFC 2986: PKCS #10: Certification Request Syntax Specification, Version 1.7, November 2000
[RFC3268]	RFC 3268: Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS). June 2002.
[RFC3280]	RFC 3280: Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL), obsoletes RFC 2459, April 2002.

Copyright © 2012 by atsec information security and IBM Corporation

[RFC3447]	RFC 3447: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. February 2003.
[RFC3749}	RFC 3749: Transport Layer Security Protocol Compression Methods. May 2004.
[RFC4492]	RFC 4492: Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). May 2006.
[RFC5019]	RFC 5019: The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments. September 2007.
[RFC5280]	RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. May 2008.
[RFC5288]	RFC 5288: AES Galois Counter Mode (GCM) Cipher Suites for TLS. August 2008.
[RFC5289]	RFC 5289: TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM). August 2008.
[RFC6460]	RFC 6460: Suite B Profile for Transport Layer Security (TLS). September 2011.
[RFC5746]	RFC 5746: Transport Layer Security (TLS) Renegotiation Indication Extension. Feburary 2010.
[RFC5915]	RFC 5915: Elliptic Curve Private Key Structure. June 2010.
[RFC6066]	RFC 6066: TLS Extension Definitions. January 2011.
[RFC6277]	RFC 6277: Online Certificate Status Protocol Algorithm Agility. June 2011.
[SP800-38D]	NIST Special Publication 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. November 2007.
[SP800-56A]	NIST Special Publication 800-56A: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised). March, 2007.
[SP800-90]	NIST Special Publication 800-90: Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised). March 2007.
[SSLv3]	Alain O. Freier, Philip Karlton, Paul C. Kocher: The SSL Protocol, Version 3; IETF Memo, Internet Draft, November 1996.
[TLSv1]	T. Dierks, C.Allen: The TLS Protocol Version 1.0; RFC 2246, January 1999.
[TLSv1.1]	RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1. April 2006.
[TLSv1.2]	RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2. August 2008.
[TLS_AES]	P. Chown: Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS); RFC 3268, June 2002.
[X.509]	ITU-T RECOMMENDATION X.509 | ISO/IEC 9594-8: INFORMATION TECHNOLOGY - OPEN SYSTEMS INTERCONNECTION - THE DIRECTORY: PUBLIC-KEY AND ATTRIBUTE CERTIFICATE FRAMEWORKS.
[X9.31]	American National Standards Institute, ANSI X9.31-1998: Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry (rDSA), 1998.

Copyright © 2012 by atsec information security and IBM Corporation