CryptoServer

User’s Guide for CryptoServer Se/CSe in FIPS Mode

http://hsm.utimaco.com

Imprint

Copyright 2016	Utimaco IS GmbH, Germany Germanusstr. 4 52080 Aachen, Germany
Phone	+49 (0)241 / 1696-200
Fax	+49 (0)241 / 1696-199
Internet	http://hsm.utimaco.com
e-mail	hsm@utimaco.com
Document Number	2011-0003
Document Version	2.2.0
Date	May 10th, 2016
Status	Released
Author	Rainer Herbertz Sven Kaltschmidt Gabriele Wedel Gesa Ott
All Rights reserved	No part of this documentation may be reproduced in any form (printing, photocopy or according to any other process) without the written approval of Utimaco IS GmbH or be processed, reproduced or distributed using electronic systems. Utimaco IS GmbH reserves the right to modify or amend the documentation at any time without prior notice. Utimaco IS GmbH assumes no liability for typographical errors and damages incurred due to them. All trademarks and registered trademarks are the property of their respective owners.

Table of Contents

1 Introduction

This document gives comprehensive guidelines on the cryptographic usage of Utimaco's hardware security module **CryptoServer** (**CryptoServer Se** and **CryptoServer CSe**) if run in FIPS mode. Here, FIPS mode means FIPS approved mode of operation according to [FIPS140-2].

It should be read carefully by all persons who are allowed to assume the role of a *Cryptographic User, User or Key Manager* for the CryptoServer.

The security-relevant administrative commands that are offered by the CryptoServer, e. g. for loading, replacing or deleting firmware, or for user management, are not described in this document: Since these administrative services can only be performed by persons who are allowed to assume the Administrator *or* Security Officer *role, detailed command descriptions of security-relevant administrative commands are not described in this user's guide.*
A guide for administrators of the CryptoServer is provided by document [CSFIPS-AdmGuide].

In FIPS mode, the CryptoServer has a well-defined external software interface offering administration, configuration and cryptographic services. Only after an appropriate authentication a user is allowed to use the designated services:

- A *Cryptographic User* is allowed to use the offered key management and cryptographic services.

  By configuration the user role *'Cryptographic User'* can be split into two different sub-roles:
  - a *Key Manager* is allowed to use the offered key management services
  - a *User* is allowed to use the offered cryptographic services
- a *Security Officer* is allowed to use the offered local configuration services (incl local user management) and to set the 'TRUSTED' attribute of a wrapping key, and
- an *Administrator* is allowed to use the offered global configuration services.

The mandatory authentication protects the CryptoServer against non-authorized access.

This document provides a rough survey on the complete CryptoServer system - hardware, software and its security architecture – while giving guidelines on its administration and the usage of the cryptographic interface. It follows a rough overview of the various chapters:

- In chapter 2 the different roles that can be assumed towards the CryptoServer are outlined.
- Chapter 3 gives basic background information e. g. about the CryptoServer's various states and modes, its authentication concept and the Secure Messaging mechanism which offers far reaching protection for messages to and from the module.
- Chapter 4 gives practical guidance through typical administrative tasks, such as getting the indicators for CryptoServer's state or quitting error states. This chapter can also be used directly for help, even if you have not read the previous chapters before. You will find there direct links to chapter 5, the descriptions of administrative commands.
- In chapter 5 the usage of the CryptoServer Administration Tool **CSADM** will be explained. CSADM is a command line utility provided by Utimaco which must run on

the PC communicating with the CryptoServer; it is designed to facilitate the execution of administrative tasks. In that chapter you will find information about its installation and syntax and a detailed description of every single administration command that can be executed by a user in Cryptographic User role, and its command execution.

- In chapter 6, a survey about the cryptographic commands that are offered by the CryptoServer and which can be used via various API libraries will be given.
- Chapter 9 gives help and practical guidance in critical situations like alarm, or in case the CryptoServer is not showing the expected reaction, or no reaction at all. This chapter can also be used directly, without having studied the other more comprehensive chapters before.
- Chapter 10 lists all built-in Elliptic curves in approved FIPS-mode.
- Chapter 11 lists all firmware modules that are mandatory for a CryptoServer in approved FIPS-mode.
- A reference list is provided in chapter 12.

2 Roles for Operators

A CryptoServer in FIPS mode knows different roles for operators:

1. **Administrator role**

   An *Administrator* is allowed to perform administrative services for firmware module management (load/replace/delete firmware module), MBK management (generate or import Master Backup Keys), global configuration management, user management (add or delete user) on the CryptoServer and the initial process of first personalization of the CryptoServer in order to enter FIPS mode. He is not allowed to perform cryptographic services!

2. **Security Officer role**

   A *Security Officer* is allowed to perform Key Group (PKCS#11: 'slot') specific configuration and user management services and is allowed to set the TRUSTED attribute of a wrapping key. He is not allowed to perform cryptographic services!

3. **Cryptographic User role**

   A *Cryptographic User* is allowed to perform cryptographic and key management services. By configuration the user role *'Cryptographic User'* can be split into two different sub-roles:

   - A *Key Manager* is allowed to use the offered key management services like key generation, key backup and restore, key import and export and key attribute setting.
   - A *User* is allowed to use the offered cryptographic services like en-/decryption, signing/verifying data, hashing and random number generation.

   None of these operators is allowed to perform security-relevant administrative services!

   See section 2.1 below.

Additionally all users are allowed to perform non-security-relevant services like requests for status or logging information.

In this document guidance for a *Cryptographic User* (also *User* and *Key Manager*) is given. The *Administrator's* and *Security Officer's* role is explained insofar as this is helpful for a *Cryptographic User* (e. g. under which circumstances an *Administrator* has to be asked for help). See [CSFIPS-AdmGuide] for an explicit guide for Administrators and *Security Officers*.

2.1 Tasks for a Cryptographic User

For the hardware installation of the CryptoServer PCIe plug-in card on the host PC, or of the CryptoServer LAN, and for physical security advices, see [CSPCIe-InstallManual] or [CSLAN-InstallManual].

Then a *Cryptographic User* may

1. Execute *administrative services* using the **administration tool CSADM**:

The CryptoServer *Administration Tool* CSADM provides any kind of basic administration like file download or deletion, retrieving status information, setting of the CryptoServer's clock, user management a. s. f.

- Most of these commands can only be executed by an *Administrator* since they are security relevant and have to be authenticated respectively.
- Some of these commands do not have to be authenticated at all (e. g. for retrieving status information): These commands can be used by every operator.
- Furthermore every operator can change his own authentication data (password or RSA key).

The CSADM tool is available for various operating systems (Linux, Windows, Solaris…). In chapter 5 the usage of the CSADM tool will be explained, including a detailed description of most CSADM commands that are available for a *Cryptographic User*.

2. Execute PKCS#11 typical *key management services* using the ***PKCS#11 Administration Tool p11toolv2.***

   The *PKCS#11 Administration Tool p11toolv2* provides any kind of basic configuration management and key management:

   - Some of these commands do not have to be authenticated at all (e. g. for retrieving status information): These commands can be performed by every operator.
   - Configuration management (display and seting configuration properties) can only be executed by *Administrators* and *Security Officers*.
   - Key Management commands like key generation or deletion, key export or import, and key backup or restore can only be performed by a *Cryptographic User* or *Key Manager*.
   - Furthermore the standard PKCS#11 users can change their password.

   The *p11toolv2* tool is available for various operating systems (Linux, Windows, Solaris…). In chapter 6 the usage of the *p11toolv2* tool will be explained, including a short description of all *p11toolv2* commands that are available for a *Cryptographic User*.

3. Develop own applications that use *cryptographic services* of the provided **CryptoServer APIs**[1] e.g. for standard cryptographic interfaces like PKCS#11, JCE, OpenSSL etc (see also 8.4):

   An application on the host PC can use one of the CryptoServer APIs to execute cryptographic services on a CryptoServer. These libraries are available for different operating systems; see chapter 8.4 for further details.

In order to execute cryptographic functions the Cryptographic User *must perform an authentication and therefore have an account on the CryptoServer consisting of a user name and his authentication token like either a password or an RSA key. Accounts must be created by an* Administrator *or* Security Officer*.*

If the CryptoServer is installed on the local computer (as PCIe card) commands will be sent from the host to the CryptoServer via the PCIe interface. The CryptoServer

[1] API: Application Programming Interface

processes the command and sends the answer (answer data or error code) back to the host.

If the CryptoServer is part of a CryptoServer LAN, commands will be sent from the host to the CryptoServer via a TCP connection. Generally, the TCP server (‘daemon’) running on the CryptoServer LAN (*csxlan*) forwards incoming commands to the integrated CryptoServer, but a few commands are responded by the CryptoServer LAN itself (e. g. setting of the TCP timeout).

From the user’s point of view it makes no difference whether he accesses a local CryptoServer PCIe or a remote CryptoServer LAN. The Administration Tool CSADM as well as the available APIs are able to send commands either to the local CryptoServer or to the remote CryptoServer LAN (by choosing the appropriate device specifier).

3 Background Information

In this chapter some background information about hardware, software architecture and security mechanisms of the hardware security module CryptoServer will be given.

3.1 Hardware

The CryptoServer is an encapsulated, protected security module which is realized as multi-chip embedded cryptographic module in the sense of FIPS 140-2 **[FIPS140-2]**. The primary purpose for this module is to provide secure cryptographic services like encryption or decryption (for various cryptographic algorithms like Triple-DES, RSA and AES), hashing, signing and verification of data (RSA, ECDSA, DSA), random number generation, on-board secure key generation, key storage and further key management functions in a tamper-protected environment.

In FIPS mode, the module offers a general purpose API with FIPS Approved algorithms for the above mentioned cryptographic services, as well as an administrative interface. A Secure Messaging concept protects the communication to and from the module by message encryption and MAC authentication.

The CryptoServer is available for two different hardware security levels:

3.1.1 CryptoServer Se

The CryptoServer Se meets FIPS 140-2 overall level 3 requirements.

CryptoServer Se may be delivered with an hardware accelerator chip which provides highest performance for RSA operations (models CryptoServer Se1000/Se400).

CryptoServer Se is available in two variants:

- CryptoServer as an expansion card with a PCI Express bus. This is referred to below as the **CryptoServer Se (or CryptoServer)**.
- CryptoServer LAN as a network component (appliance), which can be easily integrated into a network. This is referred to below as **CryptoServer Se LAN (or CryptoServer LAN).**

Illustration 3-2 – CryptoServer Se expansion card with a PCI Express bus

3.1.2 CryptoServer CSe

The CryptoServer CSe meets FIPS 140-2 overall level 3 requirements, with level 4 requirements in section “Physical Security”.

CryptoServer CSe is available in two variants:

- CryptoServer CSe as an expansion card with a PCI Express bus. This is referred to below as the **CryptoServer CSe** (or **CryptoServer PCIe**).
- CryptoServer LAN as a network component (appliance), which can be easily integrated into a network. This is referred to below as **CryptoServer CSe LAN** (or **CryptoServer LAN**).

Illustration 3-3 – CryptoServer CSe expansion card with a PCI Express bus

3.2 Software

Inside the CryptoServer different kinds of software will run to different times:

- Boot loader,
- Operating system (SMOS – **S**mall **M**ultitasking **O**perating **S**ystem) with further firmware modules.

Software module *or* firmware module *in the context of this document denotes an encapsulated software part running on the CryptoServer. A module can have an external interface which can be used by an application from outside the CryptoServer device, and an internal C interface which can be called by other firmware modules.*

The boot loader is an independent firmware module that runs only partially on the CryptoServer, during its boot process, whereas SMOS and the other firmware modules run on the CryptoServer during its normal operational state (when the CryptoServer is up and running and not in any FIPS Error state).

3.2.1 Boot Procedure and FIPS Error States

CryptoServer's boot procedure is divided into two phases, each of them being controlled by one firmware part:

- first boot phase which is controlled by the boot loader
- second boot phase which is controlled by the operating system SMOS

Various tests are part of the boot procedure. Roughly spoken, if one of these tests fails, the CryptoServer will enter a *FIPS Error* state: If the failure is found during the first, boot loader controlled boot phase, the module enters the ***Boot Loader Error* state**; if the failure is found during the second, SMOS controlled boot phase, the module enters the ***OS Error* state**.

If no FIPS Error is found the boot loader will start the operating system firmware module SMOS automatically. If this succeeds the boot loader passes the control to the operating system SMOS and terminates itself. The CryptoServer is considered to be in *operational* state.

After SMOS has been started successfully, it performs further tests and initialization steps. In particular it starts and initializes all firmware modules that are found in the flash file. If this was successful, the CryptoServer is considered to be in (*FIPS mode* and) **normal operational state**, i. e. not in any FIPS error state.

But if during this phase any *FIPS error* is found, the CryptoServer enters ***OS Error* state**. Examples for FIPS errors include: any cryptographic algorithm self-test has failed, or any firmware module that is mandatory in FIPS mode is missing or cannot be successfully initialized (e. g. because its software integrity check fails). In *OS Error* state all started software remains active, but only some non-security relevant commands like status requests can be performed. See [CSFIPS-AdmGuide].

The (second part of the) boot process can even be watched and analyzed for errors:

During start-up of SMOS and other firmware modules the CryptoServer writes log messages to the boot log file.

If no fatal error occurs during the boot phase these log messages can be retrieved later using the administration command *GetBootLog* (see [CSFIPS-AdmGuide]).

3.2.2 CryptoServer's Operator Modi

Depending on the loaded firmware modules, a CryptoServer can either be in FIPS mode (if the respective FIPS firmware modules are loaded and the FIPS mode is set) or in personalization mode (otherwise).

Additionally, depending on the possible occurrence of (FIPS) errors and the question which firmware is actually active, further modi have to be distinguished.

3.2.2.1 FIPS Mode, Personalization Mode

A CryptoServer enters **FIPS mode** after it is set-up and personalized by an Administrator according to instructions given in [CSFIPS-AdmGuide]. This includes that all firmware modules that are necessary in FIPS mode have been loaded and could be started successfully after the setup process. Even if later some of these modules will be deleted (e. g. erroneously), the CryptoServer will stay in FIPS-mode (but fall to a FIPS error state). But there are also circumstances under which a CryptoServer can leave FIPS-mode.

A CryptoServer module that has not entered FIPS-mode (yet) is said to be in ***personalization mode****.*

This can be the case after first shipping of the CryptoServer, or after a physical alarm has happened to the module, or after the module has been cleared manually and intentionally (see 4.5).

In this personalization mode the CryptoServer can be set-up and personalized in order to enter FIPS-mode afterwards. In particular, the necessary firmware modules have to be loaded while the CryptoServer is in personalization mode.

3.2.2.2 Boot Loader Mode, Operational Mode, Maintenance Mode

Independently from this distinction between FIPS-mode and personalization mode, in a particular moment the CryptoServer can either be in *power down mode*, in *boot loader mode*, in *maintenance mode* or in *operational mode*. This distinction between various modes just refers to the firmware which is active at that special moment: no firmware, only boot loader firmware, or operating system SMOS and further firmware modules.

- *The CryptoServer is in* ***power down mode*** *if no firmware is active (CryptoServer is shutdown). In power down mode the CryptoServer is not able to receive any command. A hardware reset has to be performed to get the CryptoServer active again.*

- *The CryptoServer is in* ***boot loader mode*** *if the boot loader is active, i. e. the boot loader is powered up but the CryptoServer's operating system (if loaded at all) has not yet been started.*
- *The CryptoServer is in* ***maintenance mode*** *if its back-up system firmware modules (SYS modules: '*.sys' files) have been started successfully and are active.*
  A CryptoServer in FIPS mode will never reach maintenance mode.
- *The CryptoServer is in* ***operational mode*** *if its operating system as well as the base firmware modules could have been started successfully and are active so that at least basic administration of the CryptoServer can be done over these firmware modules.*

So, if the respective base firmware modules (SMOS, CMDS, UTIL, ADM; see section 11 for a complete list of all firmware modules) are loaded, the CryptoServer is in *operational mode* if at the end of the boot process the boot loader terminates and the operating system module SMOS is started, because SMOS then starts automatically the other firmware modules (if not defect).

With the GetState *command the operator's mode can be retrieved:*

- If the CryptoServer does not answer to the *GetState* command, it is in power down mode.

In all other modi the GetState *command can be performed. A sentence*

- mode = Operational Mode
- mode = Maintenance Mode *or*
- mode = Bootloader Mode

is part of the answer to the command.

For a CryptoServer in FIPS-mode, being in boot loader mode means that the cryptographic module is in FIPS error state (*Boot Loader Error* state, see 3.2.1 above). In this case, only very restricted functionality is available: requests for status and logging information are the only services that can be executed in boot loader mode. To leave the boot loader mode / FIPS error state, it is at least necessary to reset the CryptoServer. See section 4.3 how to quit FIPS error state.

For a CryptoServer in personalization mode, the boot loader mode offers access to basic administrative functionality, to be used for the process to set-up and personalize the CryptoServer in order to enter FIPS-mode afterwards. Apart from basic status requests, this functionality is only available for an operator who has assumed the *Administrator* role.

3.2.2.3 Possible Modes and States

At the customer's site, the following variations of mode and state of a CryptoServer can occur:

- CryptoServer is in power-down mode (see above).
- CryptoServer is in **FIPS mode** and no FIPS error has occurred. This includes that the CryptoServer is in **operational mode**. All administrative and cryptographic services are available.

- CryptoServer is in **FIPS mode** and in **FIPS error state.** This may happen in boot loader mode as well as in operational mode:
    - If the CryptoServer is now in **boot loader mode**, this means that a FIPS error has been noticed during the first phase of the CryptoServer’s boot process, when the boot loader was still active. .
    - If the CryptoServer is now in **operational mode**, this means that a FIPS error has been noticed during the second phase of the CryptoServer’s boot process, when the operating system was already up.

  In both cases only basic non-sensitive services like status requests are available.
- CryptoServer is in **personalization mode** (i. e. not in FIPS mode): The CryptoServer must be personalized to get into FIPS mode; this can only be done by an *Administrator*, see 4.4.

For the respective state indicators see 4.2. For leaving any FIPS error state see 4.3.

3.3 Command Mechanisms and Security Concepts

In this chapter background information will be given about how external commands are processed inside the CryptoServer and by the host PC software respectively. Furthermore, the mechanisms for authentication and/or secure messaging will be described.

3.3.1 External Interface

Some firmware modules (ADM, CMDS, CXI, MBK) offer functionality to the external world, i. e. callable from outside the CryptoServer. Such a command interface is called ***external interface****.*

A CryptoServer external interface expects command data coded together with a command header as a **byte stream** (via PCIe or TCP). Such an external interface can then be attached by a host application directly or by one of the CryptoServer APIs that sends/receives byte streams to/from the CryptoServer’s physical interface. The functions building the external interface of a firmware module will interpret the byte stream as a command like specified in the appropriate interface specification of the module. The answer of the module will be a byte stream, too, specified in the same document.

The different **APIs** (application programmer interfaces) which can be used by the host application generate a comfortable high level interface out of the external CryptoServer byte stream interface. For this it composes the command byte stream from the different logical parts of the command header (like function code FC, sub function code SFC, command data length,...) and the block with the specific command data, and later decomposes the answer byte stream into the different logical parts of the answer header and the block with the specific answer data. This interface can then be used by the host application: It hides the composition and decomposition of the command/answer header byte stream from the application programmer and therefore facilitates the usage of the CryptoServer protocol stack (in particular the usage of the authentication layer).

The task of the composition/decomposition and interpretation of the function specific command/answer data is left for the host application software. This has to be done pursuant to the appropriate module specification.

As explained in chapter 2.1, for a CryptoServer in FIPS-mode, with the respective cryptographic firmware modules loaded, Utimaco provides the following host software to attach to the external byte stream interface:

- For the external interface of the “administrative” firmware modules (CMDS, ADM) this job is done by the *CryptoServer Administration Tool CSADM*. This command line utility will be described in chapter 5 and [CSFIPS-AdmGuide].
- For the external interface to most administrative services *of the firmware module CXI this job i*s done by the *PKCS#11 Administration Tool p11toolv2***.** This command line utility only provides PKCS#11 typical services to Administrators, Security Officers and Key Managers and is described in chapter 6 and [CSP11-ToolGuide].
- As interface *to the cryptographic services, Utimaco* provides various Cryptographic Services Interface Libraries (see chapter 8.4). An applicat*ion on the host PC can attach to the ext*ernal byte stream interface directly or use one of these libraries to execute cryptographic services on a CryptoServer. See chapter *8*.

Inside the CryptoServer, an external command of a module will be executed directly when it is called:

For external commands, there is only single command processing (“store and forward”) available. The commands will be performed one after the other, in that order in which they are received by the CryptoServer.

Responsible for the processing of the protocol stack inside the CryptoServer is the firmware module CMDS:

First CMDS gets the input (command) byte stream from the PCIe interface via the PCIe driver provided by the operating system SMOS. CMDS then processes the command header and, if everything is correct, passes the pure command data to the specific function of the specific firmware module (which are announced through the FC and SFC in the header).

3.3.2 Authentication

The CryptoServer accepts certain external commands only after one (or more) appropriate user(s) have been successfully authenticated

Certain external commands that are sent to the CryptoServer may only be executed if the sender has authenticated the command and a certain *authentication state* has been reached (see below). For this purpose one or more authentication header data blocks can be added to the command data block. These authentication headers will be processed completely by the firmware module CMDS (*command scheduler module*). The addressed firmware module which will only receive the command data block is then responsible for checking the authentication state, if necessary, and to decide about its further execution.

Successful authentication can only be done by certain authorized users which have to be registered at the CryptoServer before. For this purpose the CryptoServer administrates an internal user database. In FIPS-mode, there are the following user roles predefined:

1) Users who are allowed to assume the ***Administrator*** role.

   These users are allowed to perform all commands for CryptoServer administration, global configuration and user management (like loading files or firmware modules, adding or removing users, setting the CryptoServer's time, ...).

2) Users who are allowed to assume the ***Security Officer's*** role.

   These operators are allowed to perform all commands for "local" (Key Group or slot specific) configuration and user management.

3) Users who are allowed to assume the ***Cryptographic User*** role.

   These users are allowed to perform all cryptographic services (like encryption / decryption, MAC calculation, hashing, ...) and key management functions (like key generation, key import/export, ...).

   By configuration (done globally by an Administrator or locally by a Security Officer) the user role *'Cryptographic User'* can be split into two different sub-roles:

   - A *Key Manager* is only allowed to use the offered key management services like key generation, key backup and restore, key import and export and key attribute setting.
   - A *User* is only allowed to use the offered cryptographic services like en-/decryption, signing/verifying, hashing and random number generation.

Moreover, every user has the choice between two mechanisms of authentication: either via password (protected against being eavesdropped by a HMAC algorithm) or via RSA signature.

The following subsections will explain the authentication mechanisms and usage in detail.

3.3.2.1 Authentication State

The CMDS module stores an **authentication state** internally. The stored value will be incremented after every successfull authentication. Depending on the value of the current authentication state it will be decided if a command is allowed to be performed or not:

A successful authentication only changes this authentication state. It is up to each individual command, if called, to check the current authentication state and, depending on its value, to decide if it will continue with execution or not. Thus the meaning of the authentication level within a specific user group depends on the individual command.

For a CryptoServer in FIPS-mode, the responsible firmware module CMDS will store the authentication state on two different levels:

- **Authentication state of a specific (secure messaging) session:**

  First, CMDS stores a *session-individual authentication state*. This is taken from the authentication of the *GetSession* command and will be stored together with the other session-related data until the session ends (about sessions, see next chapter 3.3.3). The authentication state of the session is only valid within this session, and it gets invalid when the session gets invalid.

- **Authentication state of a specific command:**

  Second, a *command-individual authentication state* will be stored together with the command data. This takes the session-individual authentication state, if the command is performed within a session, and adds the authentication of the respective command (if the command has been authenticated). The command-individual authentication state is only valid for this command.

  This command-individual authentication state is what is checked by the respective command and which is thus crucial for the decision if the command is allowed to be executed or not.

3.3.2.2 Authentication Mechanisms

In FIPS-mode, two different mechanisms for command authentication are implemented:

1. **HMAC Password Authentication:**

   For this mechanism an operator *password* will be used. First the host demands an 8 bytes random value from the CryptoServer. Then the host calculates the HMAC value over this random value and the command data block using the password as the HMAC key (hash algorithm for the HMAC calculation is SHA-256). It transfers this hash value to the CryptoServer which recalculates and checks the HMAC with the help of the password which is stored in the user database. Compared with any cleartext password authentication, this mechanism has the following advantages:

   - The password will not be submitted in clear and thus cannot be eavesdropped.
   - Because of the random value the authentication data block cannot be eavesdropped and replayed at a later time.
   - The command data are protected against unnoticed manipulation.

2. **RSA Signature Authentication:**

   For this mechanism the host demands again an 8 bytes random value from the CryptoServer first. Then the host calculates a RSA signature over this random value and the command data block with a private RSA key (PKCS#1 signature over the

SHA-1 hashed data, compliant to [PKCS#1]). This signature will then be transmitted to the CryptoServer which will verify it with the help of the RSA key's public part which is stored in the user database.

Both mechanisms are also qualified for a secure communication via Ethernet.

3.3.2.3 Users and User Permissions

Commands can only be successfully authenticated by authorized **users**. For the management of these *users*, the CryptoServer uses and administrates a **user database** which stores for each user the following data:

- *Name* (which serves as a unique identifier for the user), 8 bytes long.
- *Long Name* (optional, which serves as a unique identifier for the user), up to 255 bytes long.
- *Permissions* of the user. The structure of these user permissions corresponds to the structure of the authentication state, i. e. it consists of eight values each in the range from 0-3, see 3.3.2.1. In FIPS-mode, only some permissions are admissible respectively relevant, see below.
- *Flags* that determines if static login or secure messaging are allowed for this user. In FIPS-mode, these flags are constant:
    - In FIPS-mode, only *single command authentication* is possible. No static login is allowed! Therefore the flag 'no_login' must always be set.
    - *Secure Messaging* is allowed for every user, but the command to open a secure messaging session (*GetSessionKey*) has to be authenticated (see next chapter). Therefore the flag 'sma' must always be set.
- The *authentication mechanism* that has to be used by the user (see above).
- Authentic*ation data* like RSA key or password, depending on the authentication mechanism, see above.
- User Attributes (optional).

The CryptoServer provides the appropriate commands to create or delete a user or to change his/her authentication token (data), see [CSFIPS-AdmGuide]. The commands for creation and deletion of a user can only be done by an *Administrator* (see below).

If a user opens and successfully authenticates a Secure Messaging session (see next section), the session-individual authentication state is set to the user's permissions: The permissions of the user are granted to the whole session.

If a user successfully authenticates any command, the command-individual authentication state is set to the sum of the session-individual authentication state plus the user's permissions. (This includes that, in case the command is *not* performed within a Secure Messaging session, the command-individual authentication state is set to the user's permissions.)

Example:

(Session-individual) authentication state before user's authentication:	01000000
Permissions of the user:	01000001
(Command-individual) authentication state after user's authentication:	02000001

Several users can authenticate one after the other or within one command. Doing this, mixed authentication mechanisms are allowed. All information about the authentication and session is lost if the module is power-cycled.

3.3.2.4 User Roles

In FIPS-mode, there are the following user groups predefined for the access to the external commands of the standard firmware modules:

1) Users who are allowed to assume the ***Administrator*** role.

   These users must have the *user permission* '22000000' (or higher). All commands for CryptoServer administration, global configuration and user management (like *LoadFile*, *AddUser*, *SetTime* ...) can only be performed after an *Administrator's* authentication, i. e. when authentication state '22000000' (or higher) has been reached.

2) Users who are allowed to assume the ***Security Officer*** role.

   These users must have the user permission '00000200' (or higher). The following commands can only be performed after a Security Officer's authentication, i. e. when authentication state '00000200' (or higher) has been reached:

   - Key Group specific configuration management (like getKeyProperty, setKeyProperty, BackupKey,... performed on key group specific configuration properties),
   - Key Group specific user management (AddGroupUser, DelGroupUser for Key Managers and Users), and
   - setting the TRUSTED property of a wrapping key (SetKeyProperty).

3) Users who are allowed to assume the ***User*** role.

   These users must have the *user permission* '00000002' (or higher). All external functions realized by the firmware module CXI which offer cryptographic services (like encryption/decryption, MAC calculation, hashing, ...) can only be performed after a *User's* authentication, i. e. when authentication state '00000002' (or higher) has been reached. If the permission mask of the *Key Manager* role is also set to '00000002' *Users* are additionally allowed to perform all key management services (see next item).

4) Users who are allowed to assume the ***Key Manager*** role.

   These users must have the *user permission* which is defined by the configuration value 'CXI_PROP_CFG_AUTH_KEYM'. The default setting is '00000002' which means that the *User* role and the *Key Manager* role is the same (also called '*Cryptographic Users*') and both *Users* and *Key Managers* are allowed to perform all cryptographic and key management services.

   The configuration value 'CXI_PROP_CFG_AUTH_KEYM' can be set to '00000020' globally by an *Administrator* or individually for each Key Group by the respective *Security Officer*. The local value set for a specific Key Group is valid for all Keys and Storage Objects belonging to the same Key Group; the global value is valid for all Global Keys and Storage Objects and for all Keys and Storage Objects which are assigned to a Key Group with no 'CXI_PROP_CFG_AUTH_KEYM' defined.

   All external functions realized by the firmware module CXI which offer key management functions (like key generation, key import/export, ...) can only be performed after a *Key Manager's* authentication, i. e. when authentication state 'CXI_PROP_CFG_AUTH_KEYM' (or higher) has been reached.

5) Additionally it is possible to create users which can assume both *Administrator* and *User* role, i. e. with user permission '22000002' (or higher).

6) Users who are allowed to assume the *User* role AND the *Key Manager* role are called ***'Cryptographic Users'***.

   By default the required *Key Manager* permission and the required *User* permission are the same ('00000002') which means that the *User* role, the *Key Manager* role and the *Cryptographic User* role is identical.

   When the *Key Manager* Permission mask is configured to '00000020' the Cryptographic User role is split into two different sub-roles *Key Manager* and *User*. Cryptographic Users must then have the permission mask '00000022' to be able to perform all cryptographic AND key management services.

If users with other permissions are created, these additional permissions will be of no use in FIPS mode: Permissions in user groups other than 7, 6, 2, 1 and 0 (e. g. a user permission like '00100000') do not have any influence on the possibilities to authenticate any of the external commands that are given in FIPS-mode.

Upon correct authentication, the (command-individual or session-individual) authentication state will be augmented by the permissions of the user (as stored in the user database) and thus the role (*Administrator, Security Officer, Key Manager, User* or *Cryptographic User*) is selected based on the user name of the operator.

Since the creation of new users also requires an authentication (by one or two users with the permission to manage users: authentication level 2 in user group 7) one initial user 'ADMIN' is preconfigured and uses the 'RSA-Signature' authentication mechanism with the *Default Administrator Key*. ADMIN has the exclusive permission of user management and administration (22000000) and does not require a second user to authenticate administrative commands (2-Persons-Rule).

The default user ADMIN may be deleted from the user database if one or more other user have been created, who – alone or in combination – possess the necessary permission to setup the CryptoServer again.

Basically the user management checks if the combined permission level of remaining users with RSA Signature authentication mechanism still allows administration and user management (≥ 21000000) and denies the deletion of a user eventually.

By this means users can never- accidentally - lock out themselves from the CryptoServer.

3.3.2.5 Consecutive Failed Authentication Attempts

For every user the number of consecutive failed authentication attempts is internally counted and stored as a user attribute (counter for failed authentication attempts, *AuthenticationFailureCounter*). This counter can be retrieved with command *ListUser* (attribute 'Z', see [CSFIPS-AdmGuide]). If the authentication of the user fails the user's *AuthenticationFailureCounter* is incremented by one, if the authentication of the user is successful the user's *AuthenticationFailureCounter* is reset to zero.

By default the allowed number of consecutive failed authentication attempts is not limited, therefore the counter Z has no consequences except for informative purposes. But it is possible for an Administrator to set a maximum value for failed authentication attempts (*MaxAuthFailures*), with 0 ≤ *MaxAuthFailures* ≤ 255. *MaxAuthFailure* = 0 means that there is no limit for failed authentication attempts (which is the default, see above). Any value of *MaxAuthFailures* > 0 means that for each user there are only *(MaxAuthFailures-1)* consecutive failed authentication attempts allowed, but the user will be locked if *MaxAuthFailures* consecutive failed authentication attempts occur (i. e. if the user's *AuthenticationFailureCounter* reaches the value of *MaxAuthFailures*).

If a user is locked no further authentication is possible for this user, consequently this user cannot perform any command which has to be authenticated. Only an Administrator with permission for user management is able to unlock this user by setting the users *AuthenticationFailureCounter* back to zero.

The value of *MaxAuthFailures* can be retrieved with external command *Get Maximum Authentication Failures (*see [CSFIPS-AdmGuide]).

3.3.3 Secure Messaging

The CryptoServer supports ‘Secure Messaging’ for the communication between the CryptoServer and the host: commands sent to the CryptoServer and answer data received from the CryptoServer may be encrypted and integrity-protected with a Triple-DES or AES MAC. For this purpose a secure messaging header data block can be added to the command and answer data block.

To use the secure messaging functionality, two steps are required (each of them being transparent to the user of the CSADM tool since performed within one CSADM command, see below):

1. **Generate a session key.**

   First the external CryptoServer function GetSessionKey is called to generate a Triple-DES or AES session key. The session key will be negotiated with the Diffie-Hellman key establishment protocol (see [PKCS#3]).

   Secure messaging must be allowed for the selected user (user flags). The CryptoServer also returns a session ID and a starting value of a sequence counter.

2. **Send encrypted commands.**

   The host can send commands to the CryptoServer that are encrypted and integrity protected with a Triple-DES or AES MAC using the previously established session key and the secure messaging layer (a software layer which inside the CryptoServer is processed by firmware module CMDS). The respective answers to these commands, which are sent back to the host by the CryptoServer, are always encrypted and protected with a MAC, too. The sequence counter is used as initialization vector for the DES or AES encryption and MAC calculation and is incremented after every command to prevent unauthorized replays of the commands.

*The session key used is identified by a session ID. All commands using the same session ID and the same session key are said to **belong to one session**. In this way a secure channel can be established between the CryptoServer and the host application using the Secure Massaging mechanism*

In FIPS mode the user is responsible to make sure that no 16 bytes TDES session key is used to encrypt more than 2^{20} blocks of data. [2]

After the CryptoServer has generated a session key, it still accepts commands in clear, i. e. without secure messaging. But if the CryptoServer receives an encrypted and MAC-protected command (i. e. a command using the secure messaging software layer), it checks the MAC. The command is rejected if the MAC is invalid.

[2] This is only relevant for an Se because 16 bytes TDES session keys are not available on a CSe.

- A session key automatically becomes invalid if it has not been used to encrypt any command for more than 15 minutes.
- Up to four sessions can be opened simultaneously. If a host application requests a new session key while the maximum of 4 sessions are already active, the oldest session is closed and its session key is invalidated.
- A maximum of 256 session keys may be active at the same time and can be used by different host applications simultaneously (each key identified by its session ID). If a host application requests a new session key while the maximum of 256 sessions are already active, the oldest session is closed and its session key is invalidated.

If the *GetSessionKey* function is authenticated by one or more users using single command authentication, the permission of this user(s) will be granted to the whole session. All commands that are encrypted with this session key have the permission of these users without extra authentication. But outside this session the former authentication state is preserved.

To the user of the CSADM tool, the steps explained above for the usage of secure messaging remain transparent: The user just has to perform the SessionDH *command (see e. g. 0 and 0), together in one command line with the command(s) for which secure messaging should be used.*

If the *SessionDH* command is called over the CSADM tool, CSADM will automatically open the session (by getting the session key), perform the specific given command(s) with secure messaging (i. e. encrypted and MAC-secured) and close the session on the CryptoServer again.

Concerning details about the usage and syntax of secure messaging for cryptographic commands in connection with the respective API libraries, please refer to the corresponding html documentation which can be found on the Product CD which is delivered with the CryptoServer..

3.4 Behavior of CryptoServer Outside the Normal Temperature Range

If the internal temperature of the CryptoServer gets outside of the normal operating temperature range, the CryptoServer will behave in a special way, as shown in the table below:

Temperature	**Behavior of the CryptoServer**
below –13°C	Alarm is triggered (all sensitive data on the security module will be cleared) and the security module is restarted. After the restart the CryptoServer enters *power down mode* by suspending the processor - commands will not be executed any longer.
-13°C to 62°C	Normal operation.
62°C to 66°C	The CryptoServer enters power down mode by suspending the processor. Commands will not be executed any longer.
above 66°C	An alarm is triggered (all sensitive data on the security module will be cleared) and the security module is restarted. After the restart the CryptoServer enters power down mode By suspending the processor, commands will not be executed any longer.

All temperature values in the table are approximate values. The exact temperature values may vary a little because of tolerances of the electronic components and the use of a hysteresis by the comparators.

Note that only the internal temperature of the CryptoServer device is relevant, not the environment temperature. The actual value of the inner temperature can be retrieved with the GetState *administration command.*

Once the CryptoServer has entered power down mode, it does not respond to any request. An attempt to access the CryptoServer will usually result in a kind of timeout error from the device driver.

Before entering power down mode the boot loader writes an entry into the audit log file which can be read out with the administration command GetAuditLog. The only way to get the CryptoServer out of the power down mode is to reset or power-cycle the module.

Resetting the CryptoServer has no effect, if the temperature is still outside the normal range. In case of CryptoServer's power down caused by high temperature, it is recommended to switch the supply power off for some time in order to cool the CryptoServer down.
If even a temperature alarm has been triggered, i. e. the CryptoServer's internal temperature has exceeded 66°C, all sensitive data on the security module are completely cleared and the CryptoServer has to be personalized again. Please call the CryptoServer's System Administrator *for help.*

4 Typical Administration Tasks

In this chapter the most important administration tasks are explained step by step.

Security-relevant administration tasks can only be performed by a user who is at least allowed to assume the Administrator role.

Even if most tasks cannot be performed by a *Cryptographic User* alone, the given information is useful. It is always marked at which point an *Administrator* has to be called for help.

4.1 How to Install the CryptoServer

For the hardware installation of the CryptoServer LAN or CryptoServer PCIe plug-in card on a host PC, and for physical security advices, see [CSPCIe-InstallManual] or [CSLAN-InstallManual].

To ensure the operational safety, please read the installation manual carefully before unpacking and installing the CryptoServer. Keep the manual always to hand.

Furthermore technical data and instructions for the following situations can be found in this installation manual:

- installation of the host software,
- battery replacement,
- de-installation,
- transport and
- storage.

4.2 How to Get State Indicators

In many situations it is vital to be informed about the CryptoServer's actual state and mode.

Precondition:

None.

What to do:

Perform a *GetState* command (see [CSFIPS-AdmGuide]).

CryptoServer's state and mode can be retrieved by analyzing the command output:

1. CryptoServer does not answer to *GetState* command:

   ➔ CryptoServer is in **dead state** or **power down mode**.

   Here either the CryptoServer's Central Processing Unit (CPU) is set into power save mode (dead state) or the whole module is without power. The module is therefore unable to perform any action or service.

   To leave this state, reset or power-cycle the CryptoServer. (One reason for power down mode could be that the CryptoServer's internal temperature has exceeded its maximum operational temperature from +62°C. In this case the module has to be cooled down before a successful reset.)

2. GetState command returns `state = DEFECT:`

```
mode       = Bootloader Mode
state      = DEFECT (0x00000001)
temp       = …
```

   ➔ CryptoServer is in **defect state**.

   The reason for a CryptoServer to be in *defect* state may also be a defect file system. Therefore the module itself cannot necessarily always decide if it is in FIPS-mode or not.

   If the CryptoServer has been in FIPS mode before, the *defect* state is considered to be a FIPS error state.

   The customer is not able to get a *defect* CryptoServer working again. Thus the manufacturer/Utimaco has to be contacted.

3. GetState command returns `state = INITIALIZED:`

   ➔ CryptoServer is in **initialized state**. This implies that the CryptoServer is generally working and not defect. The following sub-states are possible:

1.1 Line '`FIPS mode = ON`' is missing in answer of *GetState* command.

As long as the CryptoServer is not in defect state, the CryptoServer is in FIPS mode if and only if the line '`FIPS mode = ON`' is returned by the GetState command.

➔ CryptoServer is in **Personalization Mode.** This means that the CryptoServer is (not in power-down mode and) NOT in FIPS mode (see chapter 3.2.2). The following sub-states are possible:

1.1.1 GetState command returns `mode = Bootloader Mode`

```
mode       = Bootloader Mode
state      = INITIALIZED (0x00000004)
temp       = 35,5 [C]
alarm      = OFF
bl_ver     = (…)
(…)
```

➔ CryptoServer is in **Boot Loader Mode** (i. e. the bootloader is active).

The CryptoServer is ready for the personalization process, see 4.4. This process can only be performed by an *Administrator*.

1.1.2 GetState command returns `mode = Maintenance Mode`

➔ CryptoServer is in **Maintenance Mode** (i. e. back-up system firmware modules *.sys are running). The following sub-states are possible:

1.1.2.1 GetState command returns `alarm = ON:`

```
mode       = Maintenance Mode
state      = INITIALIZED (0x00027f84)
temp       = 35,0 [C]
alarm      = ON
sens       = 027f
           - Alarm has occurred
           - external Erase is executed

bl_ver     = (…)
(…)
```

➔ CryptoServer is in **alarm state**:

If alarm is given, the CryptoServer is automatically in *initialized* state, maintenance mode and in personalization mode, i. e. it has left FIPS mode. (The alarm has cleared most data.)

The (hexadecimal coded) two bytes behind `'sens = (…)'` display the contents of the sensory register. To help any user to analyze the alarm reasons, the following text explains these contents:

- `'Alarm has occurred'` means that the physical alarm reason is no longer present. (Alternatively, `'Alarm is present'` would indicate that the physical alarm reason is still present.)
- The individual alarm reason is specified in the following line (here: `'external Erase is executed'`).

For more information about possible alarms, see section 9.2.

No command can be performed before the alarm is set back. See section (1) for alarm treatment.

1.1.2.2 GetState command returns `alarm = OFF:`

```
mode       = Maintenance Mode
state      = INITIALIZED (0x00000004)
temp       = 35,5 [C]
alarm      = OFF
bl_ver     = (…)
(…)
```

➔ No alarm is given. This may happen if a *RecoverOS* command has been performed. The CryptoServer is ready for the personalization process, see 4.4. This process can only be performed by an *Administrator*.

1.1.3 GetState command returns `mode = Operational Mode`

```
mode      = Operational Mode
state     = INITIALIZED (0x00000004)
temp      = 35,0 [C]
alarm     = OFF
bl_ver    = (…)
(…)
```

➔ CryptoServer is in **Operational Mode** (i. e. the operating system SMOS is already loaded and active).

In this state usually the first personalization process after delivery is performed. This personalization process can only be performed by an *Administrator*.

1.2 GetState command returns `FIPS mode = ON`:

➔ CryptoServer is in **FIPS-mode**. The following sub-states are possible:

1.2.1 GetState command returns `FIPS error state` and `Bootloader Mode`

```
mode      = Bootloader Mode
state     = INITIALIZED (0x00040004)
FIPS mode = ON
FIPS error state (0xb0070039)
temp      = 34,5 [C]
alarm     = OFF
bl_ver    = (…)
(…)
```

➔ CryptoServer is in **Boot Loader Error State**(i. e. a FIPS error has been noticed during the first phase of the CryptoServer's boot process):

The number behind the '`FIPS error state`' indicator serves for error analysis and serves here just as an example. Only basic status request services are available. For leaving the FIPS error state, see 4.3.

1.2.2 GetState command returns `FIPS error state` and `Operational Mode.`

```
mode      = Operational Mode
state     = INITIALIZED (0x00040004)
FIPS mode = ON
FIPS error state (0xb0830025)
temp      = 34,5 [C]
alarm     = OFF
bl_ver    = (…)
(…)
```

➔ CryptoServer is in **OS Error State** (i. e. a FIPS error has been noticed when the operating system was already up; in particular the OS is still active; the CryptoServer is thus necessarily in operational mode).:

CryptoServer is in FIPS mode, but a FIPS error has been noticed when the operating system was already up and running.

The number behind the ‘`FIPS error state`’ indicator serves for error analysis and serves here just as an example. Only some non-sensitive services (like status request services) are available. For leaving the FIPS error state, see 4.3

(If on the other hand a CryptoServer in FIPS mode is *not* in any error state, the line ‘`FIPS error state`’ is completely left out.):

1.2.3 GetState command returns no `FIPS error state.`

```
mode      = Operational Mode
state     = INITIALIZED (0x00040004)
FIPS mode = ON
temp      = 34,5 [C]
alarm     = OFF
(…)
```

➔ CryptoServer is in (FIPS mode and) **Normal Operational State**:

CryptoServer is in FIPS mode, the operating system is up and running and no FIPS error has occured yet. All administrative and cryptographic services are available.

4.3 How to Quit an Error State

In this chapter it will be explained how to leave a FIPS error state (*Boot Loader Error State* or *OS Error State*). Depending on the error cause this can require various activities. Depending on the error reason, in some cases it will be necessary to call an *Administrator* for help.

Precondition:

The CryptoServer is (not in *defect* state but) in any FIPS error state, i. e. the *GetState* command (see [CSFIPS-AdmGuide]) answers with '**FIPS error state = ON**'.[3]

What to do:

1. Perform the Reset command or power-cycle the CryptoServer.
2. Check if the module is still in FIPS error state by performing the GetState command. If no, you are ready. If yes, go to step 3.
3. Remove the power from the CryptoServer for at least 30 seconds. Power on the CryptoServer again.
4. Check if the module is still in FIPS error state by performing the GetState command. If no, you are ready. If yes, go to step 5.
5. The CryptoServer has to be erased completely and later to be personalized again. This task can only be performed by a user who is allowed to assume the Administrator role. Thus please ask an Administrator for help.

[3] If the CryptoServer is in *defect* state (i. e. the *GetState* command returns **'state = defect'**), please contact the manufacturer/Utimaco.

4.4 How to Enter FIPS-Mode: Setup and First Personalization of a CryptoServer

If you receive a new CryptoServer from Utimaco, the module will be in personalization mode and no FIPS firmware modules are loaded yet. Thus prior to start operating the CryptoServer in FIPS-mode, a personalization process has to be performed. This personalization process can furthermore be necessary

- after a physical alarm has been occurred to the CryptoServer,
- after the CryptoServer has been cleared on purpose (e. g. to quit certain FIPS error states).

This process cannot be performed by a *Cryptographic User* but only by an administrator:

The process of personalizing a CryptoServer can only be performed by the CryptoServer's System Administrator.

4.5 How to Clear the CryptoServer

It may be useful respectively necessary to clear all data inside of a CryptoServer for example in the following situations:

- You want to securely clear all secret data inside a CryptoServer.
- Emergency case only! You have lost your customer-individual administrator keys and want to regain an administrable CryptoServer system.

At the end of this clearance process, the CryptoServer will be in personalization mode.

Be aware that all data stored on the CryptoServer will be lost, in particular all cryptographic keys!

At the end of this clearance process the help of an *Administrator* is needed, to reset the CryptoServer from alarm state. Since additionally only the CryptoServer's system administrator is able to load new firmware afterwards and to set-up the CryptoServer again, the clearance process should only be performed by the system administrator. Therefore a detailed description of this process is given in the Administrator's Guide [CSFIPS-AdmGuide].

5 Administrative Services

The *CryptoServer Administration Tool* (CSADM) is a command line utility designed for being called from the command line or in a batch file.It offers various functions to execute administrative commands on the CryptoServer. In addition it contains utility functions processed without a connection to a CryptoServer (e. g. change PIN of smart card).

- The administration tool CSADM is mainly created to support the CryptoServer's administrators. For this purpose, it offers many administrative services which have to be authenticated by a user that has assumed the Administrator role.
- A user who is allowed to assume the Cryptographic User role can use the administration tool CSADM to perform non-security relevant administrative services like e. g. requests for status and logging information, or to change his/her user token (RSA key/password), or to change the PIN of his/her smart card (if he/she uses the RSA signature authentication mechanism).
- [CSFIPS-AdmGuide] gives a detailed description of installation and usage of CSADM. The cryptographic user can execute all commands which can be authenticated by a cryptographic user (e.g. the *changeUserXXX* commands) or which need no authentication on the cryptoserver at all.
- You will need a **PIN-Pad** with integrated smart card reader to perform the administrative commands which have to be authenticated only if you use the *RSA Signature* authentication mechanism.

The CSADM cannot be used for the performance of any cryptographic services!

The following list gives an overview about all CSADM commands which are available to a Cryptographic User (see [CSFIPS-AdmGuide] for a complete list and a detailed description of all commands):

Basic Commands

Help	If called without any parameter, this command shows a list of all available CSADM commands. If the command name is given as a parameter, specific help will be provided.
PrintError	This command displays the corresponding error message text to an error code.
Version	This command shows the version of the CSADM.

CryptoServer Driver Commands

Reset	This command performs a hardware reset (like Power Off-On) of the CryptoServer on PCIe level.
ResetToBL	This command will reset the CryptoServer and get it into *boot loader mode.*
Restart	This command will reset/restart the CryptoServer and get it into *operational mode.*
GetInfo	This command retrieves information from the PCIe driver of the CryptoServer.

SetTimeout	With this command the maximum time that the driver waits for a response of the CryptoServer can be changed.

Commands for CryptoServer's Administration

GetState	This command returns the status and mode of the CryptoServer, its temperature, alarm state, error indicators, hardware information and set-up information.
GetBattState	This command can be used to show the state of the two batteries.
StartOS	The regular set of firmware modules (*.msc) is started.
RecoverOS	The recovery set of firmware modules (*.sys) is started.
ListFiles	This command lists all files stored on the CryptoServer.
GetTime	This command returns the CryptoServer's system time.
ListModulesActive	This function returns a list with information about all firmware modules which are currently active
GetBootLog	GetBootLog retrieves a log file which contains log messages made during the CryptoServer's boot process.
GetAuditConfig	(CSe only) The configuration for generation of the audit logfile is displayed.
GetAuditLog	The content of the audit logfile is displayed.
MemInfo	This function returns information about the current memory usage of the CryptoServer.
Test	With this command a communication test with the CryptoServer is executed.
Clear (to factory defaults)	The Clear command erases all sensitive data from the CryptoServer.

Commands for User Management

ListUser	This command lists all existing users from the 'user.db' database.
ShowAuthState	CryptoServer's current authentication state is displayed.
ChangeUserRSASign	With this command a user using the authentication mechanism 'RSA-Signature' changes his RSA Key.
ChangeUserHMACPwd	With this command a user with the authentication mechanism 'HMAC Password' changes his password.
GenKey	This command generates a key file which e.g. can be used as RSA user key.
SaveKey	This command reads a key from one storage medium and stores it to another storage medium.
BackupKey	This command reads a key from a key file, splits it into two 'XOR' parts and saves the parts on two smartcards for back-up matters.

CopyBackupCard	This command copies one XOR-half of a key-backup (backup of user's authentication key) from one smartcard to another.
GetCardInfo	The *GetCardInfo* command reads information about keys eventually stored on a smartcard.
ChangePassword	This command changes the password of an encrypted key file.
ChangePin	This command changes the PIN of a given smart card.

Command Authentication

LogonSign	With this command a user with RSA Signature authentication mechanism opens an authenticated Secure Messaging session for the given command.
LogonPass	With this command a user with HMAC Password authentication mechanism opens an authenticated Secure Messaging session for the given command.
AuthRSASign	With this command a user with the authentication mechanism 'RSA Signature' authenticates a single command.
AuthHMACPwd	With this command a user with the authentication mechanism 'HMAC Password' authenticates a single command.
SessionDH	This command opens a session for Secure Messaging using the Diffie-Hellman key agreement.

Administration of the CryptoServer LAN

CSLGetConnections	All current connections to the CryptoServer LAN are listed.
CSLScanDevices	The *CSLScanDevices* command returns the internal configuration of one or more CryptoServer LAN.
CSLGetVersion	The version number of the TCP-Server ('csxlan'-daemon) is shown.
CSLGetLogFile	With this command the log file from the CryptoServer LAN can be retrieved
CSLGetConfigFile	With this command the configuration file ('/etc/csxlan.conf') of the CryptoServer LAN can be retrieved.
CSLPutConfigFile	This command imports a new configuration file into a CryptoServer LAN.
CSLGetTime	This command reads the local system time of the CryptoServer LAN box.
CSLGetSerial	This command reads and outputs the serial number of the CryptoServer LAN box
CSLGetLoad	This command reads and outputs the work load of the CryptoServer PCI/PCIe card in percent.
ListPINPadApps	*ListPinPadApps* shows all commands that have been registered as PIN PAD application.

Miscellaneous Commands

Cmd	The *Cmd* command provides a generic command interface.
CmdFile	The *CmdFile* command provides a generic command interface. Unlike the *Cmd* command it reads the input data from a file.
CSTerm	This command retrieves messages from a serial port and displays them on the screen.
Sleep	The *Sleep* command delays the further command processing by the time given.

6 Data Concept and Definitions

Several services in the next chapters work on ***CXI Objects*** which denote a *Key*, a *Configuration Object* or a *Storage Object*:

- A ***Key*** is a cryptographic key, key pair or generic secret to be used with a specific cryptographic algorithm (DES, AES, RSA, DSA/DH/DH_PKCS (CSe only), ECDSA/ECDH, Generic Secrets to be used for HMAC calculation),
- A ***Storage Object*** may contain a certificate, domain parameters or other data which shall be stored in the CryptoServer's key table. Storage Objects can only be used to store information securely within the CryptoServer; they cannot be used for any evaluation or calculation within the CryptoServer.
- A ***Configuration Object*** contains a list of configuration properties. The following properties exist:

Property	***Description***
ALLOW_GROUPS	If *true* local (group specific) configuration objects are allowed, otherwise only the global configuration object may be used (default: *false*). This value may only be set by an Administrator with the permission mask of '20000000'.
CHECK_VALIDITY_ PERIOD	If *true* the start and end date of the validity period is checked before a key is used (default: *false*). If the key's property list doesn't contain a start / end date, the beginning / ending of the validity period will not be checked regardless of this configuration property.
AUTH_PLAIN	Required permission mask needed to export or import plain text keys (default: 0000002; in FIPS mode plain text key export is always blocked).
WRAP_POLICY	If *true* the algorithm strength of the wrapping key is checked and has to be greater or equal than the strength of the wrapped key. If *false* the strength of the wrapping key is not checked (default: *false*).
AUTH_KEYM	Permission mask of the *Key Manager* (default: 00000002). May be reconfigured to '00000020'.

The services *OpenKey, ListKeys, GetKeyProperty, SetKeyProperty, DeleteKey, InitKeyGroup, BackupKey* and *RestoreKey* work on all types of CXI Objects. The services *CreateObject* and *CopyObject* only work on Keys and on Storage Objects, not on Configuration Objects. All other services listed in this section only work on Keys, neither on Configuration Objects nor on Storage Objects.

Each *Security Officer, Key Manager* and *User* (and Cryptographic User) as well as every CXI Object may be assigned to a ***Key Group:***

- A CXI Object is called ***Global*** if assigned to no Key Group; it is called ***Local*** if it is assigned to a Key Group. In general Local CXI Objects can only be accessed by users which are assigned to the same Key Group; Global Objects can be seen by all users.

- A Key or Storage Object is called ***Assigned*** to a user if both belong to the same Key Group, or if the Object is Global.
- Operators may be assigned to multiple Key Groups by using wildcards within the ‘Key Group’ attribute.
- *Key Managers* and *Users* can only work with Assigned Keys and Storage Objects.
- *Security Officers* can only work on Local Configuration Objects (Configuration Object whose Key Group matches the *Security Officer*’s Key Group), and they can set the TRUSTED attribute of every Assigned wrapping key.
- *Administrators* are responsible for the Global Configuration Object.
- A Local Configuration Object only applies to the assigned Key Group; the attributes of the Global Configuration Object apply to all Global Objects, and to all Local Objects with no Local configuration value defined.

  This attribute ‘ALLOW_GROUPS’ only exists in the Global Configuration Object; it does not exist in a Local Configuration Object.

A ***Key Handle*** denotes a hash value over Key Group, key name and key specifier and is needed to identify a CXI Object in succeeding commands:

- The three values ‘Key Group’, ‘key name’ and ‘key specifier’ must be unique within the CryptoServer.
- Configuration Objects neither have key name nor key specifier; they are identified by the Key Group: There exists exactly one Global Configuration Object; for each Key Group there exists exactly one Local Configuration Object.
- All Keys and Storage Objects must be assigned to a key name and/or to a key specifier.

A ***Key Blob*** encapsulates a key or other CXI Object and may contain the public & private or secret key. In FIPS mode private and secret key parts within a Key Blob must always be encrypted.

Different types of Key Blobs are defined:

- ***Simple Blob, MS CNG Blob (“Bcrypt”) and MS CSP Blob (“Legacy”)****:* Encapsulate *Keys* and may be used as input or output parameter for the Export Key and Import Key function (see later on). They encapsulate the key components and additional parameters in different formats; the secret and private key components are encrypted with another Key (key encryption key).
- ***PKCS#11 Blob****:* Encapsulates *Keys* and is used as input or output parameter for the Wrap Key and Unwrap Key function (see later on). It encapsulates the key components and additional parameters as required by PKCS#11 (see [PKCS#11]); the secret and private key components are encrypted with another Key (key encryption key).
- ***Backup Blob:*** This format is used to handle CXI Object backups and external keys. It contains an encapsulated property list, the key components (Keys only) and a check value (MAC calculated with Master Backup Key (MBK)). The secret and private key components are encrypted with the Master Backup Key MBK.

The cryptographic services as listed in the next sections are available as follows:

- *Users* can execute the services *ListKeys, OpenKey, GetKeyProperty, GenerateDSAParam (CSe only), GenerateRandom, Crypt, Sign, Verify, ComputeHash* and *AgreeSecret*.

- *Key Managers* can execute all listed services except *InitKeyGroup, GenerateRandom, Crypt, Sign* and *Verify*.
- By configuration both user roles *Key Manager* and *User* may be grouped together to one user role called '*Cryptographic User*' (this is the default configuration)*. Cryptographic Users* can assume the *Key Manager* and the *User* role: They can execute all *Key Manager* and all *User* services, i.e. all listed services except *InitKeyGroup*.
- *Administrators* can execute services working on (Global) Configuration Objects:

  ListKeys, OpenKey, GetKeyProperty, SetKeyProperty BackupKey, RestoreKey, and DeleteKey. OpenKey and *GetKeyProperty* work on Local and Global Configuration Objects*.*
- *Security Officers* can execute services working on Local Configuration Objects:

  ListKeys, OpenKey, GetKeyProp, SetKeyProp, BackupKey, RestoreKey, DeleteKey, and *InitKeyGroup. InitKeyGroup* may delete all Assigned Local CXI Objects. Additionally *Security Officers* may access any CXI Objects with the services *OpenKey and GetKeyProp*, and they can use the service *'SetKeyProp'* on keys to set the key attribute 'TRUSTED'.

See [CSCXI] for a detailed description of all services and access rules.

7 Configuration and Key Management Services

The *PKCS#11 Administration Tool* (*p11toolv2*) is a command line utility designed for being called from the command line or in a batch file. It offers various functions to execute PKCS#11 typical key management or configuration commands on the CryptoServer.

The PKCS#11 administration tool p11toolv2 is mainly created to support the CryptoServer's Security Officers and Cryptographic Users or Key Managers when executing PKCS#11 typical tasks:

- An operator who is allowed to assume the **Security Officer** role can use p11toolv2 to set up and display Key Goup (slot) specific configuration values and to generate or delete the PKCS#11 user.
- An operator who is allowed to assume the **Administrator** role can use p11toolv2 to set up and display global (not Key Group/slot specific) configuration values and to generate or delete the PKCS#11 Security Officers.
- With default configuration (configuration attribute 'AUTH_KEYM' is set to '00000002') every operator with permission mask 00000002' (or higher) is allowed to assume the ***Cryptographic User*** role (***User*** and ***Key Manager*** role) and can use the PKCS#11 administration tool *p11toolv2* to execue key management services like key generation or key backup and restore, and to display the current configuration setting.
- If the configuration attribute 'AUTH_KEYM' is set to '00000020' key management services can only be executed by dedicated ***Key Managers*** (Users with a permission mask of '00000020' or higher). In this case with *p11toolv2* the PKCS#11 standard slot ***User*** can only list available keys and storage objects and display the current configuration settings.

[CSP11-ToolGuide] gives a detailed description of installation and usage of p11toolv2.

You will need a **PIN-Pad** with integrated smart card reader to perform the commands which have to be authenticated only if you use the RSA Signature authentication mechanism.

Remarks:

- A 'slot' according to PKCS#11 corresponds to Key Group 'SLOT_<nnnn>'.
- The security officer SO for slot <nnnn> according to PKCS#11 corresponds to user 'SO_<nnnn>' with permission mask '00000200' (e.g. Security Officer 'SO_0001' for slot 1 rsp. Key Group 'SLOT_0001').
- The slot user according to PKCS#11 for slot <nnnn> corresponds to user 'USER_<nnnn>' with permission mask '00000002' (e.g. user 'USER_0001' for slot 1 rsp. Key Group 'SLOT_0001').
- Users with different user names, Key Groups or permission masks (e.g. Key Managers with permission mask '00000020') have to be configured by an Administrator using the appropriate CSADM commands as described in [CSFIPS-AdmGuide].

The following list gives an overview about all *p11toolv2* commands which are available to the several user roles (see [CSP11-ToolGuide] for a complete list and a detailed description of all commands):

Basic Commands (no authenticaton required)

Command	*Description*	*Authenticated by*[4]
Help	If called without any parameter, this command shows a list of all available *p11toolv2* commands. If the command name is given as a parameter, specific help will be provided.	none
PrintError	This command displays the corresponding error message text to an error code.	none
Version	This command shows the version of the *p11toolv2*.	none
ListSlots	This command displays a list of all slots in the system.	none
GetInfo	This command displays general information about the CryptoServer PKCS#11 library.	none
GetSlotInfo	This command displays information about a specific slot (Key Group 'SLOT_<nnnn>').	none
GetTokenInfo	This command displays information about a specific CryptoServer (in failover mode: about the currently active CryptoServer).	none
ListConfig	This command displays a list of all configuration attributes.	none
GetLocalConfig	This command displays the value of the local (host) configuration attribute.	none
GetBackupInfo	This command displays information about a given backup file.	none

General Commands (valid authentication for several user roles required)

LoginUser	This command logs a standard PKCS#11 slot user (USER_<nnnn>) into the CryptoServer.	CU/U/KM
Login	This generic command logs any operator into the CryptoServer.	any
SetPIN	This command changes the password of the PKCS#11 standard slot Security Officer, (Cryptographic) User or Key Manager.	SO or CU/U/KM
ListObjects	This command displays a list of available keys and storage objects.	SO, CU/U/KM or none
GetGlobalConfig	This command displays the value of the global	any

[4] None: no authentication required
Any: valid authentication for any user role required
SO: Security Officer
CU/U: Cryptographic User or User
KM: Key Manager

	configuration attribute with the given name.	
GetSlotConfig	This command displays the value of a slot configuration attribute.	SO; CU/U/KM

Administrator commands (valid Administrator authentication required)

InitToken (first execution per slot/group)	This command initializes a slot by generating the PKCS#11 standard slot Security Officer.	Administrator
SetGlobalConfig	This command sets the value of a global configuration attribute.	Administrator
DeleteSO	This command deletes the the PKCS#11 standard slot Security Officer.	Administrator

SO commands (valid Security Officer authentication required)

LoginSO	This command logs the standard PKCS#11 Security Officer (SO) into the CryptoServer.	SO
InitToken (repeated execution per slot/group)	This command re-initializes a slot by deleting all slot users and data (keys, storage objects and configuration settings).	SO
InitPIN	This command generates the standard PKCS#11 User in a specific slot (group) with a given password.	SO
SetSlotConfig	This command sets the value of a slot configuration attribute.	SO
BackupConfig	This command creates a backup of the slot configuration object.	SO
RestoreConfig	This command restores the slot configuration object from the given configuration backup file.	SO

Key Management commands (valid Key Manager authentication required)

DeleteObjects	This command deletes specified keys or storage objects.	CU/KM
ImportP12	This command imports an X509 certificate, a public or a private key.	CU/KM
ImportCert	This command imports an X509 certificate and a public key.	CU/KM
ExportCert	This command exports a certificate.	CU/KM
GenerateKeyPair	This command generates a public/private key pair.	CU/KM
GenerateKey	This command generates a secret key or set of domain parameters.	CU/KM

BackupInternalKeys	This command creates a backup of all available internal[5] keys within a slot.	CU/KM
BackupExternalKeys	This command creates a backup of all available external[5] keys within the slot.	CU/KM
RestoreInternalKeys	This command restores all keys from the given key backup file to the internal[5] key store.	CU/KM
RestoreExternalKeys	This command restores all keys from the given key backup file to the external[5] key store.	CU/KM

The tool, its installation and usage is described in detail in [CSP11-ToolGuide].

[5] Internal keys are stored within the CryptoServer; external keys are stored in a key database not located within the CryptoServer, e.g. on the host.

8 Cryptographic Services

In this chapter it will be given a survey about how a *Cryptographic User* may develop his own applications that use the cryptographic services that are offered by the CXI firmware module of the CryptoServer.

The CXI firmware module offers key management and cryptographic services to the external world by providing a byte stream interface called **external CXI Interface**. It can be attached by a host application directly or by one of the CryptoServer APIs that sends/receives byte streams to/from the CryptoServer's physical interface.

Utimaco offers different libraries as interface to the cryptographic services. These libraries can be used by the host application to access cryptographic services in a comfortable way (see chapter 8.4). Although the usage of these APIs is recommended, basically the CryptoServer can be used without API. In this case the application itself has to create the command and parse the answer byte streams. Therefore [CSCXI] describes the CryptoServer byte stream interface in detail.

All cryptographic services can only be performed if authenticated by a *Cryptographic User* and within a *Secure Messaging* session. Therefore, in order to use the external CXI interface correctly, the introductory chapters of the document [CSCXI] have to be read first. There it will be explained

- how a connection to the CryptoServer is to be opened respectively closed,
- how an authenticated Secure Messaging session has to be opened (and closed), and
- how Secure Messaging is used within a secure session.

In FIPS mode the Cryptographic User is responsible to make sure that no 16 bytes TDES session key is used to encrypt more than 2^{20} blocks of data. [6]

[6] This is only relevant for an Se because 16 bytes TDES session keys are not available on a CSe.

8.1 Administrative Functions

In this chapter the external functions for administrative services for Cryptographic Users, Key Managers and Users which are offered by the CXI firmware module are listed. For details of the syntax and usage of these functions, as well as for background information about the CryptoServer security concept, the respective chapters of the [CSCXI] document have to be consulted.

8.1.1 Verify Genuineness

This function allows the user to check whether the CryptoServer was modified or replaced.

Therefore the function creates a signature over the CXI firmware module's version number and the given challenge value and returns it to the application. The signature is calculated with the private part of the Local Personalization Key that is created on the first start of the CXI firmware module and is therefore specific for each CryptoServer.

8.1.2 Get CXI Info

This function returns some information about the CXI firmware module (e.g. version number of CXI firmware module and fill level of the database).

8.1.3 Get Personalization Key

This function returns the public part of the Local Personalization Key (NIST-P256 based ECDSA key).

8.2 Key Management Functions

In this chapter the external functions for key management which are offered by the CXI firmware module are listed. For details of the syntax and usage of these functions, as well as for background information about the CryptoServer's key management concept, the respective chapters of the [CSCXI] document have to be consulted.

8.2.1 List Keys

This function returns the property lists of all CXI Objects stored in the internal key table of the CryptoServer which match the given search criteria and the requesting user is allowed to see:

- Each authenticated *Administrator* is allowed to see the Global Configuration Object
- Each authenticated *Security Officer* is allowed to see 'his/her' Local Configuration Object(s).
- Each authenticated *Key Manager* and *User* is allowed to see all Assigned Keys and Storage Objects.

This list may also be empty, e.g. if no operator is authenticated, or if no Key Group matches.

For every such CXI Object the following data are output:

- the Key Handle,
- the key name (if set),
- the Key Group (if set),
- the key specifier (if set);
- the key algorithm (DES, AES, RSA, ECDSA, ECDH, DSA, DH, DH_PKCS (CSe only), RAW, X509, X509_ATT),
- size of the key (if set)
- and the key type (public key, private key, secret key)

With this function

- *Key Managers* and *Users* can list all Assigned Keys and Storage Objects,
- *Security Officers* can list Local Configuration Object(s) and
- *Administrators* can list the Global Configuration Object.

8.2.2 Open Key

This function opens an existing CXI Objectand returns a Key Handle which allows identifying the CXI Objectin further commands, or a Backup Blob if requested. A template containing the name and optionally Key Group and specifier is given as input data.

With this function

- *Key Managers, Users* and *Security Officers* can open all (Assigned) CXI Objects, and
- *Administrators* can open all Configuration Objects.

8.2.3 Get Key Property

This function returns one or more properties of a given CXI Object. Input data are

- a CXI Object (in form of a Key Handle or Backup Blob) whose properties are to be inquired,
- and a list of properties to be inquired.

Output data is the filled property list containing the actual property values.

With this function

- *Key Managers, Users* and *Security Officers* can access all (Assigned) CXI Objects, and
- *Administrators* can access all Configuration Objects.

8.2.4 Set Key Property

This function sets one or more properties for the given CXI Object. Input data is the CXI Object whose properties are to be set (as Key Handle or as Backup Blob), and a list of property values to be set.

Output data is the modified CXI Object (Backup Blob if input was given as Backup Blob, Key Handle otherwise).

With this function

- *Key Managers* and *Users* can update Assigned Keys and Storage Objects,
- *Security Officers* can update Local Configuration Objects and the 'TRUSTED' attribute of each Assigned Key Object, and
- *Administrators* can update the Global Configuration Object.

8.2.5 Generate Key

This function creates a new key (or key pair) according to the given property list. The generated key will either be stored internally (and the corresponding Key Handle will be returned) or the key is not stored in the key table but an MBK encrypted Backup Blob is returned.

The following properties **must** be given as input data:

- algorithm (DES, AES, RSA, ECDSA, ECDH, DSA, DH, DH_PKCS (CSe only), RAW)
- DES, AES, RSA: key size in bits. In FIPS mode only the following key sizes are allowed:
    - DES: 112 and 168 bits (16 or 24 bytes)
    - AES: 128, 192 or 256 bits
    - RSA: no modulus sizes less than 1024 bits
    - Generic Secret (Algorithm 'RAW'): $8 < \text{len} < 1024$
- Key type

Further properties **may** be given as input data, e.g.:

- Key Group, name and specifier

- export permission (allowed, allowed in plain, deny MBK encrypted backup)
- key usage (encryption, decryption, signature, verification, key derivation, key encryption and key decryption)
- key label
- key generation date
- key certificate

The following **flags** may be given as input data:

- overwrite already existing key
- don’t store key on CryptoServer but return MBK encrypted Backup Blob
- don’t store key permanently (i.e. key will be deleted on restart of CryptoServer)

The following parameters must be given depending on the given algorithm:

RSA:

- use hardware (real) or deterministic (pseudo) random number generator (in FIPS mode, always deterministic RNG is used)
- public exponent (optional, default: ‘010001’)
- key generation mode (create ‘probable’ prime values according to Miller-Rabin’s algorithm / create proven prime values according to Maurer’s algorithm / use ANSI X9.31 prime generation algorithm)
- Length difference of primes p and q in bits (default 0)

ECDSA, ECDH:

- use hardware (real) or deterministic (pseudo) random number generator (in FIPS mode, always deterministic RNG is used)
- curve name (in FIPS-mode only elliptic curves P-192, P-224, P-256, P-384, P-521, K-163, K-233, K-283, K-409, K-571, B-163, B-233, B-283, B-409 or B-571 as specified in FIPS 186-2 Appendix 6 are allowed)
- public key format (compressed, uncompressed or hybrid form)

DSA, DH, DH_PKCS (CSe only):

- use hardware (real) or deterministic (pseudo) random number generator (in FIPS mode, always deterministic RNG is used)
- domain parameters p, q and g (in FIPS mode: |p| >= 1024, |q| >= 160, g optional for algorithm DH_PKCS)
- key generation mode (create ‘probable’ prime values according to Miller-Rabin’s algorithm / create proven prime values according to Maurer’s algorithm)

In FIPS mode, key generation relies on the CryptoServer’s deterministic random number generator that is compliant with FIPS 186-2, Appendix 3.1 [FIPS 186-2].

Each generated DES key (random number) will be checked for weakness and padded to odd parity to a size of 8, 16 or 24 bytes (FIPS mode: 16 or 24 bytes).

This function is available for *Key Managers* only.

8.2.6 Generate Key Pair

This function generates a new asymmetric key pair and stores the two key parts in two different Key Objects (with different Key Handles). In difference to function ‘Generate Key’ no symmetric algorithm can be given, two different property lists for each key part are given as input parameters, and two different Key Handles or Backup Blobs may be returned.

This function is available for *Key Managers* only.

8.2.7 Derive Key

This function derives a DES or AES key or a Generic Secret from a base key (DES, AES, ECDH, DH_PKCS or DH).

Input parameters are storage flags and property list for the derived key, the base key (as Key Handle or Backup Blob), the key derivation function KDF and data as needed by chosen KDF (e.g. chaining mode, initialization vector, data to be encrypted / hashed / XOPRed / appended / prepended, hash algorithm, public key, key to be concatenated (Key Handle or Backup Blob), offset size).

The derived key is either stored in the internal database or returned as an MBK encrypted *Backup Blob*.

This function is available for *Key Managers* only.

8.2.8 Generate DSA Param (CSe only)

This function generates DSA parameters p, q and g of given length. If the CryptoServer is in FIPS mode only parameter lengths |p| >= 1024 and |q| >= 160 are allowed.

This function is available for *(Cryptographic) Users and Key Managers.*

8.2.9 Generate DSA Param PQ (CSe only)

This function generates DSA parameters p and q of given length. If the CryptoServer is in FIPS mode only parameter lengths |p| >= 1024 and |q| >= 160 are allowed.

This function is available for *(Cryptographic) Users and Key Managers.*

8.2.10 Generate DSA Param G (CSe only)

This function generates DSA parameter g from given p and q. If the CryptoServer is in FIPS mode only parameter lengths |p| >= 1024 and |q| >= 160 are allowed.

This function is available for *(Cryptographic) Users and Key Managers.*

8.2.11 Create Object

This function generates a new Key or Storage Object in the database. All required properties must be given in the property list.

This function is available for *Key Managers* only.

8.2.12 Copy Object

This function copies a Key Object or a Storage Object. A key template may be given that contains a list of properties which should be added to the original properties or replace existing properties. The copied Object is either stored in the internal database or returned as an MBK encrypted *Backup Blob*.

This function is available for *Key Managers* only.

8.2.13 Wrap Key

This function exports a key from the internal key table of the CryptoServer encrypted with a key encryption key (KEK) and formatted according to PKCS#11 (see [PKCS#11]). The key to be exported must have the attribute “Exportable”. The KEK must have the attribute “WRAP”.

Data input parameters have to be the key IDs or Backup Blobs of both keys, padding and chaining mode, prefix and postfix data and Initialization Vector (optional; if wrapped key is AES or DES only).

If the encryption affords randomly generated padding bytes the function calls the deterministic random number generator which will be checked by the *Continuous RNG Test*, see above.

The function returns the wrapped key and its key type, key size, and key attributes (Key Encryption Key, exportable key).

This function is available for *Key Managers* only.

8.2.14 Unwrap Key

This function imports an encrypted key into the internal key table of the CryptoServer. The key encryption key (KEK) must have the attribute “UNWRAP”.

Input parameters have to be the Key Handle or Backup Blob of the KEK, the wrapped key (as PKCS#11 Blob), key properties to be set for the unwrapped key, and the wanted key storage mode.

This function is available for *Key Managers* only.

8.2.15 Export Key

This function exports a key from the internal key table of the CryptoServer.

Input parameter is

- the Key Handle or (MBK encrypted) Backup Blob to be exported,
- the key type to be exported: private/public/secret,
- the Key Blob format
- optionally for public keys; in FIPS mode mandatory for private and secret keys to be exported: the key encryption key (with key usage attribute ‘WRAP’) to be used to encrypt the secret key component and the chaining and padding mode .

The function returns the key to be exported including key algorithm, key size and secret/private and/or public part (private or secret key encrypted with key encryption key) encoded as Key Blob in the requested format.

When used on internal keys the key (pair) remains stored in the key table.

The key to be exported must have the attribute “Exportable”. The key encryption key must have the attribute “Key Encryption Key”.

This function is available for *Key Managers* only.

8.2.16 Import Key

This function imports a key into the internal key table of the CryptoServer. Input parameters are

- the key type (public, private, secret),
- the Key Blob type,
- key attributes,
- key storage mode (volatile/non-volatile storage, overwrite/don't overwrite existing key with given Key Handle, do/do not store key on CryptoServer but return MBK encrypted Backup Blob),
- optional: the key encryption key to be used to decrypt and the chaining and padding mode used to encrypt the private/secret key component (key attribute 'UNWRAP'),
- and the key (optionally encrypted), including algorithm and key size encoded in a Key Blob.

The imported key and its attributes will be stored in the CryptoServer's internal key table (if internal storage is wanted). In case of external storage the MBK encrypted key is output (Backup Blob format), in case of internal storage a Key Handle will be output.

The key encryption key (KEK) must have the attribute "Key Encryption Key".

In FIPS-mode only ECDSA keys on elliptic curves that are specified and recommended in FIPS 186-2 are allowed to be imported (see chapter 10).

This function is available for *Key Managers* only.

8.2.17 Backup Key

This function returns a Backup Blob which is encrypted with the Master Backup Key (MBK).

Input parameter is the Key Handle of a CXI Object stored in the internal key table.

Output parameter is the requested Backup Blob containing

- all assigned properties,
- for keys: the secret key (encrypted) or the public and private (encrypted) key parts
- and a check value (an MBK based MAC over all preceding output parameters).

With this function *Key Managers* can back up Assigned Keys and Storage Objects, *Security Officers* can back up Local Configuration Objects, and *Administrators* can back up the Global Configuration Object.

8.2.18 Restore Key

This function imports a Backup Blob which is encrypted with the Master Backup Key (MBK).

Input parameters are

- command flags indicating the storage mode (existing CXI Object shall be overwritten, CXI Object shall not be stored within the CryptoServer but shall be returned as MBK encrypted Backup Blob, key shall be stored permanently or not - in the latter case it will be deleted on restart of the CryptoServer),

- backed up CXI Object to be restored (MBK encrypted Backup Blob, including the check value),
- and optionally a list of CXI Object properties which shall be set for the restored CXI Object .

If the decryption with the MBK was successful and the check value could be verified, the additional properties will be added to the unwrapped Key Object if given. The unwrapped CXI Object will then either be stored in the CryptoServer's internal key table under the respective Key Handle together with its properties, or an MBK encrypted Backup Blob will be generated if requested. Output parameter is the newly generated Backup Blob if requested or the Key Handle of the stored CXI Object otherwise.

With this function *Key Managers* can restore Assigned Keys and Storage Objects, *Security Officers* can restore Local Configuration Objects, and *Administrators* can restore the Global Configuration Object.

8.2.19 Delete Key

This function removes a CXI Object from the internal key table of the CryptoServer. Input data is the respective Key Handle or key template. There is no output data.

With this function

- *Key Managers* can delete Assigned Keys and Storage Objects,
- *Security Officers* can delete Local Configuration Objects, and
- *Administrators* can delete the Global Configuration Object.

8.2.20 Init Key Group

This function removes all CXI Objects from the internal key table of the CryptoServer which belong to the given Key Group and which are assigned to the authenticated *Security Officer*. Input data is the respective Key Handle or key template optionally containing the Key Group.

This service is available to *Security Officers* only, there is no output data.

8.3 Cryptographic Functions

In this chapter the external cryptographic functions which are offered by the CXI firmware module are listed. These are services for symmetric (DES, AES) and asymmetric (RSA, ECDH, ECDSA) cryptography, hashing and random number generation. For details of the syntax, algorithm and usage the respective chapters of the [CSCXI] document have to be consulted.

8.3.1 Crypt Data

This function encrypts or decrypts data with a DES or AES key in OFB, ECB or CBC mode. In FIPS mode RSA keys are not allowed for data de- or encryption.

Input parameters have to be

- the key given as Key Handle or Backup Blob (if data shall be decrypted the key to be used must have the key usage attribute "DECRYPT"; if data shall be decrypted the key to be used must have the key usage attribute "CRYPT"),
- operation mode (*encrypt* or *decrypt*),
- chaining mode (OFB, ECB or CBC),
- OFB and CBC mode only: initialization vector (optional),
- padding mode and hash algorithm if required,
- and the data to be encrypted or decrypted.

The command returns the encrypted or decrypted data (and, in OFB and CBC mode, the final chaining vector).

This function is available for *(Cryptographic) Users* only.

In FIPS mode the Cryptographic User is responsible to make sure that no 16 bytes TDES key is used to encrypt more than 2^{20} blocks of data. [7]

8.3.2 Sign Data

This function calculates

- for asymmetric algorithms: a signature for a given hash value, or
- for symmetric algorithms: a message authentication code (MAC).

Input parameters have to be

[7] This is only relevant for the Se series because encryption with a 16 bytes TDES key is blocked on the CSe series.

- the signing or MAC key as Key Handle or Backup Blob (the key to be used must have the key usage property “SIGN”),
- hashing algorithm (in FIPS-mode, only the SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 hashing algorithms can be used),
- the padding algorithm and parameters,
- MAC calculation only: chaining mode and initialization vector to be used for chaining (optional),
- and the hash value to be signed or data to be MACed.

The command returns the calculated signature or MAC. If random bytes are needed in FIPS mode always the DRNG will be used.

For large data to be MACed the data can be split into several blocks and the function can be called for each block separately. The calculated MAC must be used as IV for the MAC calculation of the next block.

This function is available for *(Cryptographic) Users* only*.*

8.3.3 Verify Signature

This function verifies a signature or MAC. Input parameters have to be

- the signature key or MAC key as Key Handle or Backup Blob (the key must have the attribute “VERIFY”),
- the padding algorithm,
- hashing algorithm (in FIPS mode, only the SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 hashing algorithms can be used),
- MAC only: chaining mode and initialization vector to be used for chaining (optional),
- hash value and corresponding signature (if asymmetric key is given) or MACed data and corresponding MAC (if symmetric key is given),
- and the signature (if asymmetric key is given) or MAC (if symmetric key is given) which shall be verified.

Successful completion of the command means that the signature has been verified successfully.

MAC verification: If chaining is used (for large MACed data the data can be split into several blocks and the function can be called for each block consequently) the input parameter ‘corresponding MAC’ is left empty (for all but the last chaining block) and the calculated ‘preliminary’ MAC is output in encrypted form which has to be used as (encrypted) initialization vector for the next chaining block.

This function is available for *(Cryptographic) Users* only*.*

8.3.4 Compute Hash

This function computes a hash or HMAC value over given data or key components with a chosen algorithm. If the CryptoServer is in FIPS-mode, only the SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 hashing algorithms or HMAC can be used.

Input parameters are the data to be hashed or HMACed, the hashing/HMAC algorithm, the HMAC key (optional), a flag if multipart hash calculation shall be done and (optionally) Hash digest information, returned on a previous call of this function, to be used to continue a multipart hash calculation.

The data to be hashed may either be given in plain, or in the format as output by the ‘Agree Secret’ function (i.e. MBK encrypted and with check value).

Optionally additional data may be input that, if given, will be prepended or appended to the data before hash calculation:.

The **hash** is calculated over the concatenation of the following items:	The **HMAC** is calculated over concatenation of the following items:
• Data to be prepended • Data • Key components of secret key • Data to be appended	• Data to be prepended • Data • Data to be appended

Any missing item or item with a length of zero will be skipped.

Output parameter is the calculated hash or HMAC value, or the Hash digest information to be used to continue a multipart hash calculation if requested.

This function is available for *(Cryptographic) Users and Key Managers.*

8.3.5 Agree Secret

This function derives a shared secret from a public and a private EC key agreement key (key usage type ‘DERIVE’) according to [TR-03111] chapter 4.3.1.

Input parameters are the private part of key 1 and the public part of key 2 (given as Key Handle or Backup Blob) for the key agreement. Output is the calculated shared secret encrypted with the CryptoServer’s MBK (Master Backup Key), and an integrity check value.

This function is available for *(Cryptographic) Users and Key Managers.*

8.3.6 Generate Random Number

This function generates a random number of arbitrary length. If the CryptoServer is in FIPS-mode, always the deterministic RNG is used. This DRNG is based on the SHA-512 algorithm as transition function and compliant to [NIST SP 800-90].

This function is available for *(Cryptographic) Users* only.

8.4 CryptoServer APIs

Different API libraries are available for different cryptographic interfaces. With all of them it is possible either to access a CryptoServer plugged in a slot of the local computer or to access a CryptoServer LAN over TCP/IP:

CXI API (Java) :	The CXI API provides a general purpose Java interface to be used with the CryptoServer firmware module CXI. To use this interface the application must have been linked against the CXI library file 'CryptoServerCXI.jar'.
CXI API (C++):	The CXI API provides a general purpose C++ interface to be used with the CryptoServer firmware module CXI. To use this interface the file 'cxi.h' has to be included and the application has to be linked against the CXI library file ‘cxi.dll’.

CXI API (.NET):	The CryptoServer Core Interface for .NET is a programming interface to access the external functions of the CryptoServer's firmware module CXI from a Microsoft .NET application running on a host (CXI.NET).
JCE:	JCE is a cryptographic extension to Sun's Java framework. It defines a unified interface between an application and a cryptographic device. With this concept a Java application must not know about specific drivers to access the CryptoServer directly. In order to use the CryptoServer JCE provider the CryptoServer JCE provider must be installed on the host system.
OpenSSL:	The API for the CryptoServer, which is used by the ENGINE interface from OpenSSL, is called "CryptoServer OpenSSL ENGINE next generation API", in short "cs_oenga". This API can be loaded dynamically during runtime from the ENGINE as well as it can be linked into the static version of the OpenSSL tools.
EKM:	Actually the Utimaco EKM Provider supports: • Microsoft SQL Server 2008 (with SP 1) on Windows Server 2008 • Microsoft SQL Server 2008 R2 on Windows Server 2008 R2
CSP[8]:	The Utimaco CryptoServer CSP implements full RSA provider functionality and is used with the CryptoServer firmware module CXI. To use this C interface the Utimaco's CSP library (cs2csp.dll) must be installed and registered on the host computer.
CNG[9]:	The Utimaco CryptoServer Key Storage Provider implements a CNG key storage interface and is used with the CryptoServer firmware module CXI. To use this C interface the Utimaco's CNG library (cs2cng.dll) must be installed and registered on the host computer.
PKCS#11:	PKCS#11 is a general purpose Public Key Cryptography Standard developed by RSA Security [PKCS#11] which defines an interface between an application and a cryptographic device. The CryptoServer provides a PKCS#11 interface. To use this interface a dedicated firmware package including the CXI firmware module must be loaded into the CryptoServer and the PKCS#11 application must be linked against Utimaco's CryptoServer PKCS#11 library or it must be able to load the specific shared library (DLL/so). Utimaco's CryptoServer PKCS#11 library is also able to support more than one CryptoServer devices for each application. The CryptoServer is able to handle up to 256 parallel PKCS#11 sessions or applications per CryptoServer.

Respective API libraries are available for various operating systems. They are described

[8] CSP (Cryptographic Service Provider) is a general purpose cryptography standard developed by Microsoft. At the top side it defines an cryptographic interface to be used by applications (CryptoAPI), on the bottom side it defines an interface to be used by manufacturers in order to integrate their cryptographic hardware. With this concept the application must not know about specific drivers to access cryptographic hardware directly. Please refer to Microsoft's MSDN web pages for a detailed specification of the CSP functionality.

[9] CNG (Crypto Next Generation) is a new cryptographic interface, which has been introduced on Vista and Windows Server 2008. It offers updated cryptographic algorithms and is intended as long term replacement of CSP. For now CSP is still supported on Vista and Windows Server 2008. Please refer to Microsoft's MSDN web pages for a detailed specification of the CNG functionality.

in the corresponding documentation which can be found on the Product CD which is delivered with the CryptoServer.

9 Troubleshooting

9.1 Check Operativeness and State of CryptoServer

This section deals with the situation where it is not known whether the CryptoServer works at all, and in which mode/state it is. The following steps should systematically be performed to check CryptoServer's operativeness and, if possible, to get the CryptoServer working again.

In some cases it will be necessary to ask an *Administrator* for help.

Precondition:

If the CryptoServer is installed on your local computer, make sure that the PCIe Driver is running and that the Administration Tool CSADM is installed. See [CSPCIe-InstallManual] or [CSLAN-InstallManual] for help.

What to do?

(1) Perform the *GetState* command (see [CSFIPS-AdmGuide]). Compare answers with state indicators as listed in chapter 4.2:

State indicator	Result	Explanation/Reason/Adjustment
1.	No answer	Dead state or power down mode Reset or power-cycle the CryptoServer.
2.	state not INITIALIZED	CryptoServer is not correctly initialized or even defect. ⇒ Please get in contact with manufacturer/Utimaco.

3.1.1	state = INITIALIZED,
	ALARM = off,
	mode = BL Mode,
	but CryptoServer is not in FIPS mode
		You may analyze the problem using the *ListModulesActive* (see (2)) and *GetBootLog* commands: Check if all necessary firmware modules are present and successfully initialized (see chapter 11 for a complete list). After that the CryptoServer should be set-up and personalized again, see 4.4.

3.1.2.1	state = INITIALIZED
	mode = Maintenance mode
	alarm = ON
		An alarm has occurred (and is possibly physically still present)
		⇒ See chapter 9.2 for alarm treatment.

3.1.2.2	mode : Maintenance mode state : INITIALIZED alarm : OFF CryptoServer is not in FIPS mode	Only the backup set of system firmware modules (*.sys) has been started (see 3.2.2.2). No cryptographic services are available. CryptoServer is in personalization mode or not correctly initialized in FIPS mode. ⇒ Re-initialize the CryptoServer, see 4.4

State indicator	Result	Explanation/Reason/Adjustment
3.1.3	mode : Operational state : INITIALIZED alarm : OFF but CryptoServer is not in FIPS mode	CryptoServer is in personalization mode or not correctly initialized in FIPS mode. ⇒ Re-initialize the CryptoServer, see 4.4.
3.2.1, 3.2.2	state INITIALIZED, FIPS mode = ON, CryptoServer is in FIPS error state, ALARM = off	Quit error state, see 4.3.
3.2.3	state = INITIALIZED, FIPS mode = ON, CryptoServer is not in FIPS error state	Everything is ok. End.
	Error B9011xxx, B9015xxx, B9016xxx, B9017xxx or B9021xxx until B9024xxx	CryptoServer's PCIe carrier card does not react. ⇒ Try a restart (3).
	other errors: B901xxxx or B902xxxx	No connection to CryptoServer / CryptoServer LAN, communication problem, wrong host or device name, problem with network. ⇒ Check parameters. ⇒ Perform 'ping' at CryptoServer LAN. ⇒ Check state/configuration of the TCP daemon on the CryptoServer LAN.

(2) Perform the *Reset* command (see [CSFIPS-AdmGuide]). Wait a minute, then perform a *GetState* command.

Result	Explanation / Reason / Adjustment
no error	Ok ⇒ back to (1)
any error	⇒ power-cycle the CryptoServer (3)

(3) Remove the power from the CryptoServer for at least 30 seconds and power-up the CryptoServer again. Then perform a *GetState* command.

Result	Explanation / Reason / Adjustment
no error	Ok ⇒ back to (1)
Any error	hardware problem ⇒ please contact manufacturer/Utimaco

9.2 Alarm Treatment

If an alarm has occurred to the CryptoServer, this will be announced with the *GetState* command (**alarm = ON**). If *GetState* additionally answers with **'Alarm is present'** then the alarm is physically still present. But it is also possible that in the meantime the alarm cause has been removed (this would be indicated as **'Alarm has occurred'**).

An alarm can be triggered on the CryptoServer as a result of the following reasons:

- Power is too low
- Power is too high
- Temperature too high (> 66°C)
- Temperature too low (< -13°C)
- No reaction from the sensory controller
- Power down (Sensory controller without power): The sensory controller (having its own battery power supply) has been down: If this power supply is missing (e.g. battery empty) the sensory controller starts up with this alarm set.
- Invalid (corrupted) Master Key $\mathbf{K_{CS}}$ (this usually occurs in case of an empty battery)
- External Erase is executed (manually by a short-circuit of the 'External Erase' pins on the PCIe-card)

CryptoServer CSe additionally triggers an alarm in the following cases:

- destruction of the outer foil
- destruction of the inner foil

In any case, for security reasons, all sensitive data that have been stored on the CryptoServer, in particular all cryptographic keys are deleted after an alarm and the CryptoServer leaves FIPS mode.

After any alarm the CryptoServer remains in personalization mode (i. e. it has left FIPS mode) and in *maintenance mode*. Even if the alarm cause can be removed, only a *System Administrator*, i. e. a user who is allowed to perform the *ResetAlarm* command, is now able to reset the pending alarm state and to set-up the CryptoServer again.

If any alarm has occurred, please ask the CryptoServer's System Administrator for help.

10 Built-in Elliptic Curves

The CryptoServer offers a collection of elliptic curves (to be used by firmware module CXI) which can be used e.g. for ECDSA key generation. Each curve, given by its elliptic curve domain parameters, can be identified by a name.

The following table lists all built-in elliptic curves domain parameters that are available in FIPS mode.

Name(s)	**Size**	**Defined in:**
NIST-P192 / secp192r1	192	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-P224 / secp224r1	224	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-P256 / secp256r1	256	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-P384 / secp384r1	384	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-P521 / secp521r1	521	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-K163 / sect163k1	163	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-B163 / sect163r2	163	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-K233 / sect233k1	233	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-B233 / sect223r1	233	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-K283 / sect283k1	283	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-B283 / sect283r1	283	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-K409 / sect409k1	409	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-B409 / sect409r1	409	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-K571 / sect571k1	571	[FIPS 186-2], [ANSI-X9.62], [SEC2]
NIST-B571 / sect571r1	571	[FIPS 186-2], [ANSI-X9.62], [SEC2]

11 Appendix: Mandatory Firmware Modules

The following firmware modules (in MTC format) are mandatory in FIPS-mode and have to be loaded by the CryptoServer's *System Administrator* during the setup and personalization process (see 4.4):

Module	File	Version Number	
		CryptoServer Se	CryptoServer CSe
ADM	adm.msc	3.0.11.4	3.0.14.0
AES	aes.msc	1.3.1.1	1.3.4.0
ASN1	asn1.msc	1.0.3.3	1.0.3.3
CMDS	cmds.msc	3.1.1.2	3.2.0.4
CXI	cxi.msc	2.1.3.2	2.1.7.3
DB	db.msc	1.1.2.5	1.1.2.6
DSA (CSe only)	dsa.mtc	-	1.2.2.1
ECA	eca.msc	1.1.3.2	1.1.5.2
ECDSA	ecdsa.msc	1.1.2.0	1.1.6.0
FIPS140	fips140.msc	3.0.2.0	4.0.3.0
HASH	hash.msc	1.0.8.0	1.0.9.0
HCE (Se only)	hce.msc	1.0.1.1	-
LNA	lna.msc	1.2.0.1	1.2.2.0
MBK	mbk.msc	2.2.4.0	2.2.4.2
SMOS	smos.msc	3.1.2.1	4.4.5.0
UTIL	util.msc	3.0.2.0	3.0.2.0
VDES	vdes.msc	1.0.5.0	1.0.9.0
VRSA	vrsa.msc	1.1.2.0	1.1.7.1

The listed firmware modules are approved in the context of the FIPS 140-2 validation process of the CryptoServer. These firmware modules have to be loaded if the CryptoServer shall run in FIPS mode.

If the CryptoServer is run with other firmware than the above listed MTC modules, or if the listed firmware is used in other versions, the CryptoServer can not considered to be in certified FIPS mode!

If the FIPS Mode Control Module FIPS140 is loaded but the other loaded firmware modules are not identical with the above listed firmware modules the CryptoServer is NOT in validated FIPS mode, but most of the restrictions implemented for FIPS mode are nevertheless applied. In particular all restrictions for the usage of non-approved cryptographic algorithms and key lengths are applied. This mode is indicated as "FIPS restrictions applied" (command getstate*, see [CSFIPS-AdmGuide]).*

The FIPS140-2 approved version of the boot loader firmware is

- version 3.0.3.0 for a CryptoServer Se, and
- version 4.0.5.0 for a CryptoServer CSe.

The boot loader firmware module is loaded by Utimaco during the CryptoServer's production process and cannot be changed or deleted by the customer.

12 References

No.	Title / Company	Doc.-No.
[ANSIX9.31]	ANSI X9.31-1998: Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA) / American Bankers Association, 1998	
[CSCXI]	CryptoServer Cryptographic Service Interface – Firmware Module CXI – Interface Specification / Utimaco IS GmbH	2008-0009
[CSFIPS-AdmGuide]	CryptoServer - Administrator's Guide for CryptoServer Se/CSe in FIPS Mode / Utimaco IS GmbH	2011-0002
[CSP11-ToolGuide]	CryptoServer Manual for CryptoServer PKCS#11 Administration Tool Release 2	2012-0004
[CSPCIe-InstallManual]	CryptoServer PCIe – Operating & Installation Manual – Se-Series / Utimaco IS GmbH	M010-0004-en
	CryptoServer PCIe – Operating & Installation Manual – CSe-Series / Utimaco IS GmbH	t.b.d
[CSLAN-InstallManual]	CryptoServer LAN – Operating Manual – Se-Series / Utimaco IS GmbH	M010-0006-en
	CryptoServer LAN – Operating Manual – CSe-Series / Utimaco IS GmbH	t.b.d
[FIPS 186-2]	FIPS PUB 186-2: Digital Signature Standard (DSS) / National Institute of Standards and Technology (NIST), January 2000	
[PKCS#1]	PKCS#1: RSA Cryptography Standard v2.1, 14th June 2002 / RSA Laboratories, http://www.rsasecurity.com/rsalabs/pkcs	
[PKCS#3]	PKCS#3: Diffie-Hellman Key Agreement Standard v1.4, 1st November 1993 / RSA Laboratories, http://www.rsasecurity.com/rsalabs/pkcs	
[PKCS#11]	PKCS#11: Cryptographic Token Interface Standard v2.20, 28th June 2004 / RSA Laboratories, http://www.rsa.com/rsalabs/node.asp?id=2133	
[FIPS140-2]	FIPS PUB 140-2, Security Requirements for Cryptographic Modules / National Institute of Standards and Technology (NIST), May 2001	
[NIST SP 800-90]	NIST Special Publication 800-90: Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised) / National Institute of Standards and Technology (NIST), March 2007	
[TR-03111]	TR-03111 Technical Guideline TR-03111 – Elliptic Curve Cryptography,; Version 1.11, April 2009 / Bundesamt für Sicherheit in der Informationstechnik (BSI)	

No.	Title / Company	Doc.-No.
[ANSI-X9.62]	ANS X9.62-2005: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA) / ANSI (American National Standards Institute)	
[SEC2]	SEC2: Recommended Elliptic Curve Domain Parameters – Certicom Research – September 20, 2000, Version 1.0	