CryptoServer CS

User's Guide
for CryptoServer in FIPS-Mode

Utimaco Safeware AG

Transaction Security

Imprint

Utimaco Safeware AG
Transaction Security
Germanusstraße 4
52080 Aachen

Phone ++49 (0)241 / 1696-200

Telefax ++49 (0)241 / 1696-222

Internet www.utimaco.de

E-Mail info.sp@aachen.utimaco.de

Document Number 2004-0005

Version 2.0.0

Status Released

Date 18th December 2006

Authors Dr. rer. nat. Gesa Ott
Dipl. Ing. Sven Kaltschmidt
Dipl. Inf. Rainer Herbertz

Table of Contents

1 Introduction

This document gives comprehensive guidelines on the cryptographic usage of Utimaco's hardware security module **CryptoServer CS** (CryptoServer) if run in FIPS-mode. It should be read carefully by all persons who are allowed to assume the role of a *Cryptographic User* 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 not be performed by any Cryptographic User *but only by persons who are allowed to assume the* Administrator *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].

The CryptoServer consists of a small computer unit which is mounted on a PCI carrier card. This computer unit is the "heart" of the hardware security module: to protect it against attacks, e. g. hostile attempts to read out data, it is encapsulated by metal shells, a special tamper detection foil and potting material. The CryptoServer's physical interface for communication is given over the PCI interface and two serial interfaces.

In FIPS-mode, the CryptoServer also has a well-defined external software interface. The CryptoServer's software concept is modular: the CryptoServer's firmware consists of encapsulated software parts, called *firmware modules*, each of them having a well-defined external and/or internal interface. Some of these firmware modules offer internal services or administrative services, other offer cryptographic services. Single firmware modules can be loaded, replaced or deleted by the customer, but only after a special form of authentication which has to be done by an *Administrator*. A *Cryptographic User* is allowed to use the offered cryptographic services, but again only after an appropriate authentication. In both cases 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, *Roles for Operators*, the two different roles that can be assumed towards the CryptoServer are outlined. These are the *Administrator* and the *Cryptographic User* roles respectively.
- 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 and its execution.
- In chapter 6, *Cryptographic Commands Offered by CSI Library*, a survey about the cryptographic commands that are offered by the CryptoServer and which can be used via the cryptographic C-library CSI-Lib will be given.
- Chapter 7 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 8 lists all firmware modules that are mandatory for a CryptoServer in approved FIPS-mode.
- A reference list is provided in chapter 9.

2 Roles for Operators

A CryptoServer in FIPS-mode knows two different roles for operators:

1. **Administrator role**

   An *Administrator* is allowed to perform administrative services for firmware module management (load/replace/delete firmware module) and user management (add or delete user) on the CryptoServer. He is not allowed to perform cryptographic services!

   See section 2.1 below.

2. **Cryptographic User role**

   A *Cryptographic User* is allowed to perform cryptographic services like en-/decryption, signing/verifying data, hashing and random number generation. He is not allowed to perform security-relevant administrative services!

   See section 2.2 below.

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

In this document guidance for a *Cryptographic User* is given. The administrator'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.

2.1 Tasks for an Administrator

Main task of the **CryptoServer's system administrator** is the management of the CryptoServer's firmware modules, user management and the initial process of first personalization of the CryptoServer in order to enter FIPS-mode.

*Main tool for this and other administrative tasks is the **Initialization Key**, a cryptographic RSA key which is stored e. g. on smart card or floppy. This key is used to authenticate most security-relevant administrative commands. The customer alone bears responsibility for the Initialization Key!*

Usually, if the concerned CryptoServer system is not a test system, Utimaco does not know the *Initialization Key*; in particular, Utimaco does not have a back-up copy of the key.

Because of the importance and the power of the *Initialization Key*, it is highly recommended to generate a back-up copy of the key and to store the key(s) very carefully, e. g. in a safe. Only few authorized persons, among these the CryptoServer's system administrator(s), should be given access to it.

2.2 Tasks for a Cryptographic User

In order to use the CryptoServer PCI card (extention board), a PCI driver for your operating system must be properly installed in a computer. Refer to the document [CSInstall-Manual] for installation instructions of the hardware and driver software of the CryptoServer.

Then a *Cryptographic User* may

1. Retrieve status information from the CryptoServer using the administration tool CSADM.
2. Develop own applications that use cryptographic services.

The following illustration shows the configuration of a system using the CryptoServer PCI-card:

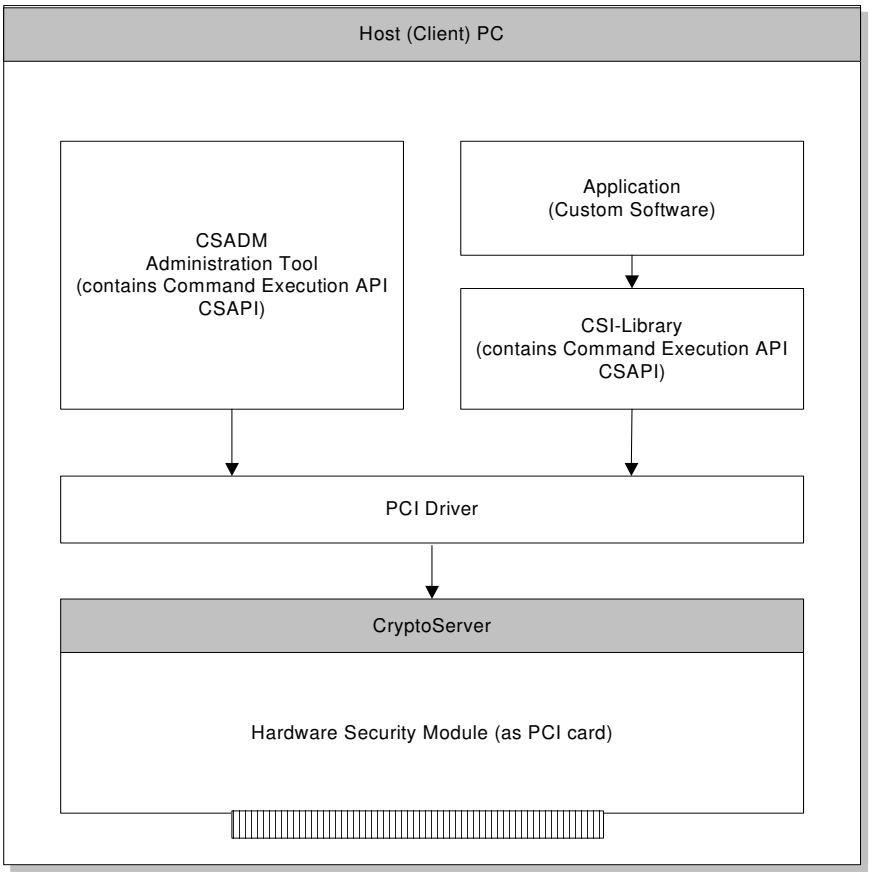

Illustration 2-1: Command Execution

Utimaco provides the following host software for the CryptoServer:

1. A PCI driver.
2. The administration tool CSADM.
3. C-library "CSI-Lib" as interface to the cryptographic services.

The PCI driver is available for Linux and Windows operating systems. It will be used by the CSADM tool and the CSI-Lib for the communication to the CryptoServer's PCI interface. For details see [CSInstall-Manual].

As a standard application the CryptoServer *Administration Tool* CSADM provides any kind of basic administration like file download or deletion, 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 also be used by a *Cryptographic User*. Furthermore a *Cryptographic User* can change his own authentication data (password or RSA key).

The CSADM tool is available for Linux and Windows operating systems. In chapter 5 the usage of the CSADM tool will be explained, including a detailed description of all CSADM commands that are available for a *Cryptographic User*.

An application on the host PC can use the *Cryptographic Services Interface Library* CSI-Lib to execute cryptographic services on a CryptoServer. The library is available for the operating systems Linux and Windows. It has an interface for the programming language C. The library consists of the following components:

	Linux operating system	Windows operating system
C header file	`cs_csi.h`	`cs_csi.h`
C library	`cs_csi.dll` `cs_csi.lib`	`libcs_csi.a`

To use this library a program has to be written that includes the header file and is linked to the library file. Before the program can use any cryptographic services is has to call the C functions '`cs_open()`' and the three '`cs_push_...()`' functions, to open a connection to the CryptoServer. Then an authentication has to be performed using one of the functions '`cs_login_pwd()`' or '`cs_login_sign()`'. After successful authentication any other functions of the interface can be used to perform cryptographic services. A detailed description of the C interface can be found in the document [CSCSI].

In order to do an authentication with the login functions the *Cryptographic User* must have an account on the CryptoServer consisting of a user name and either a password or a RSA key. Accounts must be created by an *Administrator* (see 2.1 above).

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 encased in a hard opaque commercial grade metal case which contains a tamper response envelope around the module: All hardware components of the cryptographic module (including the Central Processing Unit, all memory chips, Real Time Clock and hardware noise generator for seeding the random number generator) are located on a printed circuit board and encapsulated by metal shells, a special tamper detection envelope (which is a special foil bearing a flexible printed circuit with a serpentine geometric pattern of conductors) and potting material. This hard, opaque enclosure defines the cryptographic boundary of the module as illustrated in the picture below.

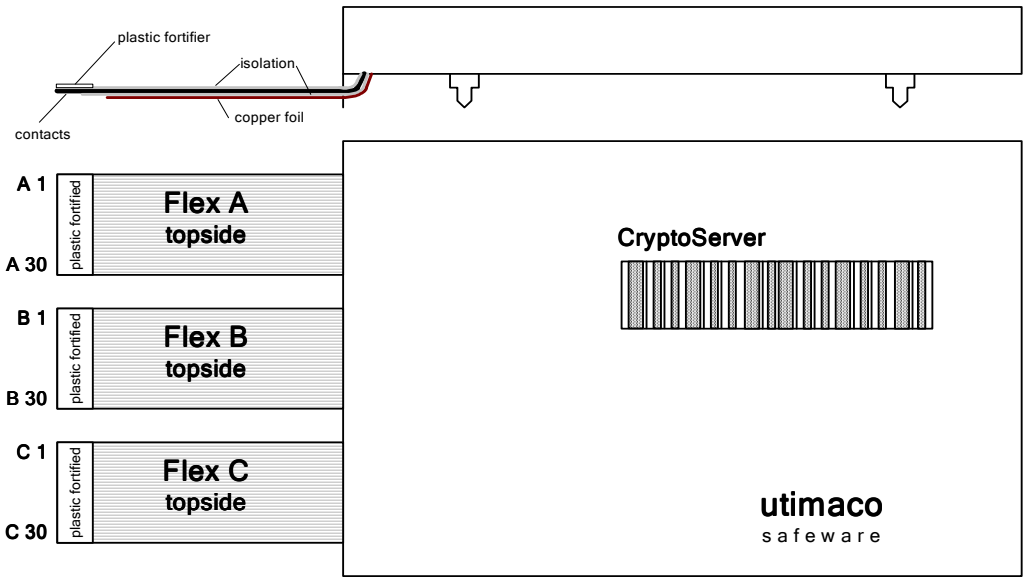

Illustration 3-1 – CryptoServer CS (cryptographic boundary)

If attacked, the security foil will trigger an alarm which will immediately erase all internally stored firmware and data (see 7.2).

To enable communication with a host, this encapsulated cryptographic module is mounted on a carrier card which supports a PCI interface and two serial interfaces (RS232). The connection between the cryptographic module and the carrier card is done by the three ribbon cables shown in figure 1. The following picture shows the cryptographic module CryptoServer mounted on a PCI carrier card:

Illustration 3-2 – CryptoServer mounted on the PCI carrier card

For the rest of this document, *CryptoServer* will denote the hardware security module (in a narrower sense) mounted on the PCI carrier card.

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**. It follows a more explicit description.

After any reset (power-up or hardware reset) the CryptoServer starts with the program code located in the *boot flash* device. This is the *boot loader firmware code*. The boot loader will do the first necessary start procedures like various tests and initialization steps:

After a hardware reset the CPU provided boot strap loader copies the first 1024 bytes of the boot loader code from the boot flash into the processor's internal memory (IRAM) at address Null. As first action, the boot loader now deletes the rest IRAM and therewith all sensitive data stored there. Afterwards the boot loader copies its rest code from the boot flash device into the IRAM and checks the integrity of the boot loader code by comparing the stored checksum (CRC) with the recalculated checksum value. Then various self-tests and initialization steps will be performed, like erasure and test of the SD-RAM,

initialization and test of the random number generator, initialization of the flash file system, alarm handling (if an alarm is present) or extreme temperature handling if necessary (power-down of the processor if it exceeds the critical limit, in this case the boot process is stopped, see section 3.4).

In error free case, at the end of this process the operating system SMOS will be started automatically. No boot loader command can be performed as long as no error has occurred.

In error case the following will happen:

- If during this boot phase a *FIPS error* is found (if for instance the power-up test for the CryptoServer's deterministic random number generator fails, or if the integrity check for the SMOS module fails), the CryptoServer remains in boot loader mode and enters the ***Boot Loader Error* state**: This means that the boot loader remains active and a time window for command is opened where commands for requesting status and loggin information can be performed. See 5.4. The fact that the boot loader is still active will be indicated as *boot loader mode and initialized* or *defect state* (depending on the error that is found) (both indicated over the *GetState* command, see 4.2 for state and error indicators).
- If during this phase an *alarm* is found, the CryptoServer also remains in boot loader mode but leaves FIPS-mode. All data that have been stored on the CryptoServer, in particular all cryptographic keys and all firmware modules, are deleted after an alarm. See 7.2 for further explanations, possible alarm reasons and alarm handling.

If no FIPS Error is found and if there is no alarm present 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, the part of the boot phase which is controlled by SMOS starts. This means that SMOS performs further tests and initialization steps, like initialization of the memory and the flash file system, and initialization of the serial and PCI interfaces. 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 mode, 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 test has failed (DES, AES or RSA), or any firmware module that is mandatory in FIPS-mode can not be successfully initialized (e. g. because its software integrity check fails). In *OS Error* state all started software remains active, but only the non-security relevant commands (i. e. those commands that have not to be authenticated) can be performed. See 5.4.

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 one of the serial interfaces, usually the *COM1 interface*. If the flash file system contains a file named "\FLASH\swap.com" then the messages are redirected to the *COM2 interface*. To watch these messages a terminal (e. g. a PC running a terminal program) has to be connected to the serial line of the CryptoServer, using the following interface settings: 115200 baud, 8 bits, no parity.

For every error or warning that occurs during the start-up of SMOS, an appropriate message is output. Additionally the log messages contain a list of all firmware modules that have been found by SMOS and whether these modules could have been started successfully or not.
If the boot process is stopped due to a fatal error, connecting a terminal to the serial line may be the only way to get information about the problem.

If no fatal error occurs during the boot phase (i. e. if all basic firmware modules – necessary for communication: SMOS, ADM, UTIL, CMDS - can be started successfully) the log messages can be retrieved later using the administration command *GetBootLog* (see 5.4.5).

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

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*, 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 **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 8 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** *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 **normal operational state** (i. e. 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**, but in **Boot Loader Error state** (i. e. a FIPS error has been noticed during the first phase of the CryptoServer's boot process). This includes that the CryptoServer is in **boot loader mode**. Only basic status request services are available.
- CryptoServer is in **FIPS-mode**, but in **OS error state** (i. e. a FIPS error has been noticed when the operating system was already up). This includes that the OS is still active; the CryptoServer is thus necessarily in **operational mode**. But only non-sensitive services that have not to be authenticated (like status request services) are available.
- CryptoServer is in **personalization mode** (i. e. not in FIPS-mode!) and in **boot loader mode** (i. e. the boot loader is active). In this mode the CryptoServer is usually ready for the setup and personalization process, see 4.4, but this mode can also occur after an alarm has happened. In any case the CryptoServer can only be handled by an *Administrator*.
- CryptoServer is in **personalization mode** (i. e. not in FIPS-mode!) and in **operational mode** (i. e. the operating system SMOS is already loaded and active). This state and mode usually occurs in the second phase of the personalization process and is thus only relevant for *Administrators*, 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, CSI) 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**. Such an external interface can then be attached by the **CSAPI (CryptoServer Application Programming Interface)** (or another host application) that sends/receives byte streams to/from the CryptoServer's physical interfaces (PCI interface), see below. 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.

CSAPI is a software running on the host (see page 10, *Illustration 2-1: Command Execution*) which has the job to generate a C-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 C-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 (CSAPI output) is left for the host application software. This has to be done pursuant to the appropriate module specification. As explained in chapter 2.2, for a CryptoServer in FIPS-mode, with the respective cryptographic firmware modules loaded, Utimaco provides the following host software:

- For the external interface of the "administrative" firmware modules (CMDS, ADM) this job is done by the *CryptoServer Administration Tool CSADM* (which attaches on the CSAPI, see *Illustration 2-1*). This command line utility will be described in chapter 5.
- As interface to the cryptographic services, Utimaco provides the C-library *Cryptographic Services Interface Library* (CSI-Lib). An application on the host PC can use the CSI-Lib to execute cryptographic services on a CryptoServer. See chapter 6.

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 PCI interface via the PCI 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 two user roles predefined:

1) Users who are allowed to assume the ***Administrator*** role.
   These users are allowed to perform all commands for CryptoServer administration 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 ***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, ...).

Moreover, every user has the choice between two mechanisms of authentication: either via password (protected against being eavesdropped by a hash 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. **SHA-1 Hashed Password Authentication:**

   For this mechanism a 16 bytes long operator *password* will be used. First the host demands an 8 bytes random value from the CryptoServer. Then the host calculates the SHA-1 hash value over this random value, the password and the command data block. It transfers this hash value to the CryptoServer which recalculates and checks the hash 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 can not 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.
- *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.

The CryptoServer provides the appropriate commands to create or delete a user or to change his/her authentication token (data), see section 5.5. 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.

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

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

These users must have the *user permission* '22000000' (or higher). All commands for CryptoServer administration 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.

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

   These users must have the *user permission* '00000020' (or higher). All external functions realized by the firmware module CSI which offer cryptographic services (like encryption/decryption, MAC calculation, hashing, ...) and key management functions (like key generation, key import/export, ...) can only be performed after a *Cryptographic User's* authentication, i. e. when authentication state '00000020' (or higher) has been reached.

5) Additionally it is possible to create users which can assume both *Administrator* and *Cryptographic User* role, i. e. with user permission '22000020' (or higher).

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 and 1 (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* or *Cryptographic User*) is selected based on the user name of the operator.

A user with user name ADMIN who is authorized to assume the Administrator *role is always present in the CryptoServer.*

One first user with user name ADMIN and with the user permission '22000000' to assume the *Administrator* role is always present in the CryptoServer. The authentication mechanism of this predefined user ADMIN is 'RSA Signature Authentication' with the CryptoServer's private *Initialization Key* (see section 2.1) The user ADMIN cannot be changed or deleted in the user database.

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 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 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 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 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*

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.

- *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. 5.5.2 and 5.5.3), 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 C-library CSI-Lib, see [CSCSI] section 2.

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	An alarm is triggered and the CryptoServer enters *power down mode*.
–13°C to 5°C	The CryptoServer enters power down mode.
5°C to 58°C	Normal operation.
58°C to 66°C	The CryptoServer enters power down mode.
above 66°C	An alarm is triggered and the CryptoServer enters power down mode.

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 temperature inside the inner case 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 temperature log file which can be read out with the administration command *GetTempLog*.

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 the range between –13°C and 66°C, the CryptoServer is completely cleared and 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.
Most of these administration tasks can only be performed by the CryptoServer's System Administrator ADMIN, i. e. the operator who is allowed to use the CryptoServer's Initialization Key.

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 PCI plug-in card on the host PC, and for physical security advices, see [CSInstall-Manual].

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 5.4.1).

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

CryptoServer's state/mode	Required Indicator	Remarks
CryptoServer is in *FIPS-mode*.	*GetState* command returns `FIPS mode = ON`	
CryptoServer is in *personalization mode*.	Line '`FIPS mode = ON`' is missing in answer of *GetState* command	A CryptoServer is defined to be in personalization mode if it is not in FIPS-mode, see 3.2.2.

CryptoServer is in *Boot Loader Error State*.	*GetState* command returns
	`state = INITIALIZED (0x00040004)`
	`FIPS mode = ON`
	`FIPS error state (0xb0070039)`
	`temp = …`
	or
	`state = DEFECT (0x00000001)`
	`temp = …`
		The number behind the '`FIPS error state`' indicator is for error analysis and serves here just as an example.
		(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.)
		A CryptoServer in *defect* state has found a defect boot loader code or file system and can thus not decide yet if it is in FIPS-mode or not.

CryptoServer's state/mode	Required Indicator	Remarks

CryptoServer is in *OS Error State*.	*GetState* command returns
	`state = OPERATIONAL (0x00040005)`
	`FIPS mode = ON`
	`FIPS error state (0xb0830025)`
		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 is for error analysis and serves here just as an example.
		(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.)

CryptoServer is in alarm state.	*GetState* command returns `alarm = ON`	If alarm is given, the CryptoServer is automatically in *initialized* state and in personalization mode, i. e. it has left FIPS-mode. (The alarm has cleared all data and firmware modules.) The detailed alarm reasons can be seen from the '`sens = (…)`' text which follows the '`alarm = ON`' line. See example below. See 7.2 for alarm treatment.
No alarm is given.	*GetState* command returns `alarm = OFF`	

CryptoServer's state/mode	Required Indicator	Remarks
CryptoServer is in *dead* state or power down mode.	CryptoServer does not answer to *GetState* command.	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 operational temperature range from +5°C to +58°C. In this case the module has to be cooled down before a successful reset.)

The state indicators can occur in various (but not all) combinations (see also 3.2.2.3). For a better understanding, here for every mode/state combinations that can occur at the customer's site, one example of the output of the *GetState* command is given. Variations can only occur in the specific error reasons or alarm reasons.

Examples:

***GetState* command answers with …**	**Explanation**

`mode = Operational Mode`
`state = OPERATIONAL (0x00040005)`
`FIPS mode = ON`
`temp = 34,5 [C]`
`alarm = OFF`
`bl_ver = (…)`
`(…)`
	CryptoServer is in **FIPS-mode** and **normal operational state** (i. e. no FIPS error has occurred):
	All administrative and cryptographic services are available.

`mode = Bootloader Mode`
`state = INITIALIZED (0x00000004)`
`temp = 35,5 [C]`
`alarm = OFF`
`bl_ver = (…)`
`(…)`
	CryptoServer is in **personalization mode** (i. e. not in FIPS mode!) and in **boot loader mode** (i. e. the boot loader is active).
	The CryptoServer is ready for the personalization process, see 4.4. This process can only be performed by the CryptoServer's *System Administrator*.

***GetState* command answers with …**	**Explanation**

`mode = Operational Mode`
`state = OPERATIONAL (0x00000005)`
`temp = 35,0 [C]`
`alarm = OFF`
`bl_ver = (…)`
`(…)`
	CryptoServer is in **personalization mode** (i. e. not in FIPS mode!) and in **operational mode** (i. e. the operating system SMOS is already loaded and active).
	This state and mode usually occurs in the second phase of the personalization process, see 4.4, and is thus only of relevance for an *Administrator*.

`mode = Bootloader Mode`
`state = INITIALIZED (0x00040004)`
`FIPS mode = ON`
`FIPS error state (0xb0070039)`
`temp = 34,5 [C]`
`alarm = OFF`
`bl_ver = (…)`
`(…)`
	CryptoServer is in **FIPS-mode**, but in **Boot Loader error state** (i. e. a FIPS error has been noticed during the first phase of the CryptoServer's boot process).
	For error analysis, the error number behind `FIPS error state` indicates which specific FIPS error has been occurred.
	Only basic status request services are available. For leaving the FIPS error state, see 4.3.

`mode = Operational Mode`
`state = OPERATIONAL (0x00040005)`
`FIPS mode = ON`
`FIPS error state (0xb0830025)`
`temp = 34,5 [C]`
`alarm = OFF`
`bl_ver = (…)`
`(…)`
	CryptoServer is in **FIPS-mode**, but 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 state and operational mode).
	For error analysis, the error number behind `FIPS error state` indicates which specific FIPS error has been occurred.
	Only non-sensitive services that have not to be authenticated (like status request services) are available. For leaving the FIPS error state, see 4.3.

***GetState* command answers with …**	**Explanation**

`mode = Bootloader Mode`
`state = INITIALIZED (0x00027f84)`
`temp = 35,0 [C]`
`alarm = ON`
`sens = 027f`
`- Alarm has occurred`
`- external Erase is executed`
 	
`bl_ver = (…)`
`(…)`
	CryptoServer is in **personalization mode** (i. e. not in FIPS-mode!) since an **alarm has occurred**. (As always after an alarm has occurred, the module is in initialized state and boot loader mode: The alarm has cleared all data and firmware modules.)
	
	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`').
	
	No command can be performed before the alarm is set back. See section 7.2 for alarm treatment.

`mode = Bootloader Mode`
`state = DEFECT (0x00000001)`
`temp = …`
	The reason for a CryptoServer to be in *defect* state is either defect boot loader code or a defect file system. Therefore the module itself cannot 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.

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 chapter 5.4.1) answers with '**FIPS error state = ON**'.[1]

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.

[1] 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 CS from Utimaco, the module will be in personalization mode and no 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 the CryptoServer's system administrator:

The process of personalizing a CryptoServer can only be performed by the CryptoServer's System Administrator*, i. e. by an Administrator who is allowed to use the CryptoServer's* Initialization Key *as authentication token.*
Other Administrators are not able to setup and personalize the CryptoServer!

4.5 How to Enter Personalization Mode and 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.
- You want to change the *Initialization Key*.
- You want to leave FIPS-mode, e. g. in order to load and use other (non-FIPS) software.
- The CryptoServer is not workable, e. g. because of buggy basic firmware modules.

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 except for the Initialization Key*.*

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, with the help of the *Initialization Key*, 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 The CryptoServer Administration Tool CSADM

The *CryptoServer Administration Tool* (CSADM) is a command line utility designed for being called from the command line or in a batch file. As described in 2.2, 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 loggin 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).*

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

Depending on your authentication mechanism, you will need a **PIN-Pad** with integrated smart card reader to perform the administrative commands which have to be authenticated. In this case, the following requirements have to be made for the CSADM:

PC-Hardware:

- no special requirements as far as memory and CPU performance is concerned.
- if you use the *RSA Signature* authentication mechanism: one free serial port to connect the PIN-Pad (smart card reader with keyboard and display).

PC-Software:

- Windows NT or
- Winows XP or
- Windows 2000/2003 or
- Linux.

Installation and usage of the CSADM will be explained in the next chapter.

Sections 5.2 - 5.5 give detailed descriptions of all CSADM commands that are available for a *Cryptographic User*.

5.1 Usage of CSADM

5.1.1 Installation of the CSADM

The installation of the Administration Tool CSADM on the PC is quite simple.

Installation under Windows:

- Copy the file 'csadm.exe' to a well-chosen directory.
- Add this directory to the 'PATH' environment variable to be able to call the Administration Tool from any other directory.
- If you do not want to set the 'Dev=' parameter with each execution of a CSADM command: It is possible to set an environment variable CRYPTOSERVER (e. g. with the value ‚PCI:0') which sets the CryptoServer address permanently (unless a 'Dev=' parameter is explicitly set for a specific command, see 5.1.2).

Installation under Linux:

- Copy the executable file 'csadm' to a well-chosen bin directory.
- If you do not want to set the 'Dev=' parameter with each execution of a CSADM command: It is possible to set an environment variable CRYPTOSERVER (e. g. with the value ‚/dev/cs2') which sets the CryptoServer address permanently (unless a 'Dev=' parameter is explicitly set for a specific command, see 5.1.2).

5.1.2 Syntax of the CSADM

The syntax of a CSADM command is according to the following scheme:

csadm [Dev=...] #param1[=...] #param2[=...] ... #command1[=...] #command2[=...] ...

Note:

- *Parameters and commands are processed from left to right.*
- *If a subsequent command requires to set a parameter or to run another command prior to its own execution, it will have to be entered rightmost.*
- *Expressions in brackets are optional.*
- *Some parameters or commands require an assigned value ('=...'), some do not.*
- *Some commands use a default value if none is given ('[=...]').*

Examples:

- csadm GetState
- csadm ListFiles
- csadm ListFiles=*.mtc
- csadm Dev=PCI:/dev/cryptoserver0 GetTime
- csadm Dev=PCI:0 AuthSHA1Pwd=paul,swordfish ChangeUserSHA1Pwd=paul,sesame

Every command running on CryptoServer requires the 'Dev=' parameter (device parameter) which sets CryptoServer's address. Possible values are:

Address	Description
`PCI:/dev/cryptoserver0`	local CryptoServer No. 1 in a Linux system.
`PCI:0`	local CryptoServer No. 1 in a Windows system.
`PCI:1`	local CryptoServer No. 2 in a Windows system.
`...`	...

If the environment variable CRYPTOSERVER is set according to the above mentioned syntax, the 'Dev=' parameter can be skipped (see also 5.1.1).
Using the 'Dev=' parameter overrides the CRYPTOSERVER environment variable in any case for the specific command

5.1.3 Key Specifiers

Some of the CSADM commands use a private or public RSA key to sign a command or a file or to verify a signature. CSADM can handle RSA keys in two different ways:

(1) RSA keys stored in a file (in plain text)

(2) RSA keys stored on a smart card

If a command needs a public key, it can be read from a file or from a smart card. If a command needs a private key, it can either be read from a file, or a smart card can be used to calculate the signature. In the latter case the key will not be read out of the smart card. A PIN has to be entered via the PIN-Pad to enable the smart card to generate signatures.

For security reasons private RSA keys should normally be used only from smart cards. In a test environment, where the private RSA key does not need to be kept secret, it may be useful to store the keys in files.

*For all commands that use a RSA key a **key specifier** has to be given in the command syntax. A key specifier is either*

- *a filename of a key file or*
- *a smart card specifier.*

A smart card specifier always starts with a colon and consists of 3 strings separated by colons (e. g. `:cs2:cp8:COM1`):

1. The first string identifies the type of the smart card.
2. The second string identifies the type of the smart card reader.
3. The last string is the name of the serial device the reader is connected to.

Smart card types supported at the moment are the following types:

Identifier	Smart card type
`cs2`	CryptoServer smart card (using TCOS).
`usa`	Utimaco PKI card (using TCOS).
`cos`	CardOS smart card.
`nkey`	Telesec NetKey Card.

At the moment supported reader types are:

Identifier	Smart card reader type
`cp8`	Ingenico / Bull SafePad.
`cm8`	Omnikey CardMan 8630.
`acr`	Advanced Card System ACR80

Key specifier examples:

Key specifier	Description
`C:\my_keys\initprv.key`	Key file.
`:cs2:cp8:COM1`	Key from a CryptoServer smart card using an Ingenico SafePad connected to `COM1` of a windows PC.
`:cos:cm8:/dev/ttyS0`	Key from a CardOS smart card using a CardMan 8630 reader connected to `ttyS0` of a Unix machine.

5.2 Basic Commands

These basic commands are CSADM internal functions.

No connection to a CryptoServer will be established.

5.2.1 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.

Syntax	csadm Help csadm Help=#command
Parameter	#command specific command of the CSADM
Example	csadm Help=ListFiles
Output	**List File(s) from FLASH / SYS / NVRAM Directory** **syntax:** **csadm ListFiles[=(FLASH\ | SYS\ | NVRAM\)#pattern]**
Note	Not all of the commands that are listed as answer to the 'Help' request can be executed by a *Cryptographic User*. Some commands can only be executed by a person who has authenticated for the *Administrator* role.

5.2.2 PrintError

This command displays the corresponding error message text to an error code. CSADM has a built-in list with all standard error messages of the CryptoServer, PCI-Driver and host-API (CSAPI). Error messages for the CSI library software (see 6) are not included in this list and will therefore not displayed.

Syntax	csadm PrintError=#errorcode
Parameter	#errorcode error code (hexadecimal).
Example	csadm PrintError=B901306F
Output	**Error B901306F** **CryptoServer API LINUX** **can't get connection** **errno = 111**

5.2.3 Version

This command shows the version of the CSADM.

Syntax	csadm Version
Output	**CryptoServer Administration Utility Ver. 1.0.5**

5.3 CryptoServer Driver Commands

The commands in this section are directed to the PCI driver of the CryptoServer.

The CryptoServer may be in any state or mode. No authentication is required.

5.3.1 Reset

This command performs a hardware reset (like Power Off-On) of the CryptoServer on PCI level.

Syntax	csadm [Dev=#device] Reset
Parameter	#device device specifier (see 5.1.2)
Output	none on success or error message

5.3.2 GetInfo

This command retrieves information from the PCI driver of the CryptoServer. It shows the driver version, PCI slot number, interrupt number, battery state, timeout, state of the error correction and the contents of the PCI registers.

The GetInfo *command will be executed even if the CryptoServer does not respond to any command (even the* GetState *command). In some cases the information provided this way helps to identify CryptoServer's low level problems. The interpretation of the output requires extensive knowledge of the CryptoServer and is therefore not explained in more detail.*

Please prepare this output in case of a support request!

Syntax	csadm [Dev=#device] GetInfo
Parameter	#device device specifier (see 5.1.2)
Output	**vers 1.0.4.0** **slot 11** **irq 10** **batt ok** **timeout 600000** **tx idle** **rx idle** **txrt 0 0** **rxrt 0 0** **mbr 00100021 00000020** **bar0 00000000 00ABF000 00 Master Write Address0** **bar0 00000000 00000020 08 MW Count Status0 / MW Transfer Count0** **bar0 4B525950 544F4B4F 10 Master Write Address1** **bar0 00000000 00000000 18 MW Count Status1 / MW Transfer Count1** **bar0 00000000 000F0100 20 misc** **bar0 0E040000 00000000 28 misc** **bar0 00000008 00000000 30 I2O** **bar0 00000000 00000000 38 reserved** **bar0 FFFFFFFF FFFFFFFF 40 I2O** **bar0 00000000 00AC2000 48 Master Read Address0 / Chain Desc. Start Address0** **bar0 00000000 00000010 50 Master Read Count Status0 / Master Read Transfer Count0** **bar0 00000000 00000000 58 Master Read Address1 / Chain Desc. Start Address1** **bar0 00000000 00000000 60 Master Read Count Status1 / Master Read Transfer Count1** **bar0 08080808 08080808 68 misc** **bar0 00400020 00000020 70 PCI Outgoing MailBox Register Host->CS2** **bar0 00100021 000000XX 78 PCI Incomming MailBox Register CS2->Host** **...**
Note	The command is used only for diagnostic purposes in error case!

5.4 Commands for CryptoServer's Administration

The commands described in this chapter offer non-security relevant administrative services for the CryptoServer like status requests or loggin information.

The five commands

- GetState
- GetAlarmLog
- GetTempLog
- GetTimeLog
- GetBootLog

which request status information will be executed in normal operational state (i. e. no FIPS error has occurred) as well as in any FIPS error *state (*Boot Loader Error *state in boot loader mode as well as* OS Error *state in operational mode).*

These commands will even be executed if the CryptoServer is not in FIPS-mode.

The remaining commands

- GetTime
- ListFiles
- ListModulesActive
- MemInfo
- Test

can be executed in normal operational state (i. e. no FIPS error has occurred) as well as in OS Error *state. (In both cases the CryptoServer is in operational mode.)*

These commands will not be executed if the CryptoServer is in Boot Loader Error *state. In personalization mode only* GetTime *and* ListFiles *will be executed.*

None of the commands in this chapter have to be authenticated.

5.4.1 GetState

This command returns the status and mode of the CryptoServer (see 3.2.2), its temperature, alarm state, error indicators, hardware information and set-up information.

The GetState *command is responded by the CryptoServer in any mode and state (in FIPS-mode as well as in personalization mode, and in normal operational state as well as in any* FIPS error *state) and therefore should be executed as first diagnostic measure in case of problems.*
Please prepare the output of GetState *in case of a support request.*

Syntax	csadm [Dev=#device] GetState
Parameter	#device device specifier (see 5.1.2)
Required State	any
Output	Example for CryptoServer in FIPS-mode and normal operational state: **mode = Operational Mode** **state = OPERATIONAL (0x00040005)** **FIPS mode = ON** **temp = 47,5 [C]** **alarm = OFF** **bl_ver = 01000500** **hw_ver = 01000200** **UID = f4000006 fa1ee701** **adm1 = 5554494d 41434f20 43533030 30303036 | UTIMACO CS000006** **adm2 = 5376656e 27730000 00000000 00000000 | Sven's** **adm3 = 496e6974 2d446576 2d312d4b 65790000 | Init-Dev-1-Key** Example for CryptoServer in FIPS-mode and Boot Loader Error state: **mode = Bootloader Mode** **state = INITIALIZED (0x00040004)** **FIPS mode = ON** **FIPS error state (0xb0070039)** **temp = 47,5 C** **alarm = OFF** **bl_ver = 01000500** **(…) = (…)**

See chapter 4.2 for further examples.

The displayed fields have the following meaning:

field	***description***	
mode	*operational mode* or *bootloader mode, see 3.2.2.* This mode has not to be confused with FIPS-mode (respectively personalization mode), these modes can occur independently from each other!	
state	state of the CryptoServer (*defect, manufactured, produced, initialized* or *operational*; see also *3.2.2*) At the customer's site only the CryptoServer's *initialized* and *operational* states will normally occur. If the CryptoServer is found to be in any other state, the manufacturer/Utimaco Safeware AG has to be contacted.	
FIPS mode	**ON**: The CryptoServer is in FIPS-mode. If the module is not in FIPS-mode but in personalization mode, this line is left out!	
FIPS error state	If this indicator is returned, the CryptoServer is in FIPS error state. The number in brackets behind the '**FIPS error state**' indicator serves for error analysis; here just an example is given. If no FIPS error has occurred (i. e. the module is not in FIPS error state), this line is left out!	
temp	temperature of the CryptoServer (see also 3.4):	
	< -13°C	sensory triggers a temperature alarm -> all user data on the CryptoServer will be cleared!
	< 5°C	during start-up or reset: ■ a new entry is put into the log file ‚temp.log' ■ CryptoServer is set into *power down mode* (see 3.2.2) in running operation: ■ module will be reset
	5°C-58°C	normal operation (CryptoServer ready to work)
	> 58°C	during start-up or reset: ■ a new entry is put into the log file ‚temp.log' ■ CryptoServer is set into *power down mode* (see 3.2.2) in running operation: ■ module will be reset
	> 66°C	sensory triggers a temperature alarm -> all user data on the CryptoServer will be cleared!

field	*description*
alarm	either **ON** or **OFF**, if alarm is **ON** the following reasons are possible and will be shown in case of an alarm state: ■ Power is too low ■ Power is too high ■ Temperature too high ■ Temperature too low ■ Outer foil is broken ■ Inner foil is broken ■ Invalid Master Key (this usually occurs in case of an empty battery) ■ External Erase is executed (manually by a short-circuit of the corresponding pins on the PCI-card) In addition it is shown if the alarm reason is still present (e. g. foil is still broken) or if it has been removed in the meantime (e. g. empty battery has been replaced). In the first case, '**alarm is present**' will be displayed, in the second case, '**alarm has occurred**' will be displayed.
version of the boot loader	boot loader version has to be 2.0.2.4 in certified FIPS-mode
version of the hardware	hardware version has to be 2.0.2.0 in certified FIPS-mode
UID	8 bytes long, unique Unit Identifier of CryptoServer's processor (as a hardware property)
adm1	16 bytes long, unique serial number of the CryptoServer which is assigned during the production process by Utimaco Safeware AG.
adm2	16 bytes long field which is assigned by Utimaco Safeware AG with the first loading of the *Initialization Key*. Usually the customer's name is registered here.
adm3	16 bytes long field which can be freely assigned with every change of the *Initialization Key* in the CryptoServer's personalization phase (performed by the CryptoServer's system administrator). Usually here the name of the actual *Initialization Key* is stored.

5.4.2 ListFiles

This command lists all files stored on the CryptoServer.

The following directories are available on the CryptoServer:

Directory	Component	Size	Description
SYS	flash device	416 KBytes	**System directory**. The initial administration keys, the boot loader's configuration file, log files and the back-up copies of the some basic firmware modules are stored here.
FLASH	flash device	15,5 MBytes	**Working directory**. The regular set of firmware modules and any other kind of application data (databases, configuration or log files, ...) is stored here.
NVRAM	NV-RAM	512 KBytes	**Non-volatile directory**. The NV-RAM is not used in FIPS-mode.

Syntax	csadm [Dev=#device] ListFiles[=#pattern]
Parameters	#device device specifier (see 5.1.2) #pattern file name pattern (search mask, wildcards '*' can be used)
Examples	■ csadm ListFiles ■ csadm ListFiles=exmp.mtc ■ csadm ListFiles=*.mtc ■ csadm ListFiles=SYS\smos.mtc ■ csadm ListFiles=*
Required State	*initialized* (if in personalization mode) or *operational* (here also in *OS Error* state available)

Output

```
SYS\Init.KeyRsaPub          168      -
SYS\MdlSg.KeyRsaPub         168      -
SYS\Prod.KeyRsaPub          168      -
SYS\adm.mtc                 30920    ADM     0x87 1.0.2.0 Administration Module
SYS\bl.ini                  72       -
SYS\cmds.mtc                49504    CMDS    0x83 1.0.5.0 Command Scheduler
SYS\smos.mtc                92240    SMOS    0x00 1.0.3.7 SMOS
SYS\util.mtc                32088    UTIL    0x86 1.0.6.0 Utility Module
                  8 files   20532    8 bytes
FLASH\CERT.db               4105     -
FLASH\RSA.db                18593    -
FLASH\adm.mtc               30920    ADM     0x87 1.0.2.0 Administration Module
```

	FLASH\asn1.mtc	**12088**	**ASN1**	**0x91 1.0.1.0 Asn1 Module**
	FLASH\cmds.mtc	**49504**	**CMDS**	**0x83 1.0.5.0 Command Scheduler**
	FLASH\db.mtc	**37288**	**DB**	**0x88 1.0.1.1 Database module**
	FLASH\hash.mtc	**36136**	**HASH**	**0x89 1.0.1.0 Hash Module**
	FLASH\lna.mtc	**47360**	**LNA**	**0x8e 1.0.3.0 LNA**
	FLASH\smos.mtc	**92240**	**SMOS**	**0x00 1.0.3.7 SMOS**
	FLASH\time.log	**14**	**-**	
	FLASH\user.db	**60**	**-**	
	FLASH\util.mtc	**32104**	**UTIL**	**0x86 1.0.6.0 Utility Module**
	FLASH\vdes.mtc	**26944**	**VDES**	**0x81 1.0.0.3 DES Module**
	FLASH\vrsa.mtc	**44648**	**VRSA**	**0x84 1.0.3.3 RSA Module**
	16 files	**504988**	**bytes**	

Note

- Depending on the ‘pattern’ parameter, information about a dedicated file, a group of files or all files will be returned regarding the CryptoServer*’s* ‘FLASH\’, ‘SYS\’ and ‘NVRAM\’ directory.
- If no pattern is given, all files are listed.
- A wildcard (‘*’) can be used.
- The CryptoServer’s file system is case sensitive!
- If a directory does not contain any file, it will not be displayed.
- ‘ListFiles’ displays the following information:
  - path\filename
  - file size

  If the file is a firmware module, additionally the following information will be displayed:
  - abbreviation (module’s short name)
  - module ID (FC)
  - version number
  - long name

5.4.3 GetTime

This command returns the CryptoServer's system time.

Syntax	csadm [Dev=#device] GetTime
Parameter	#device device specifier (see 5.1.2)
required state	*initialized* (if in personalization mode) or *operational* (here also in *OS Error* state available)
Output	**Date: 06.11.2004 Time: 15:35:49.400**
Note	The system clock of the CryptoServer has a resolution of 1/1000 second (one millisecond).

5.4.4 ListModulesActive

This function returns a list with information about all firmware modules which are currently active (i. e. loaded and started by the operating system SMOS, see below), giving their module-ID, abbreviated name, version number and their respective initialization level.

If a module cannot be started or fully initialized, the GetBootLog *command (see next chapter 5.4.5) provides more detailed information about the reason.*

Syntax	csadm [Dev=#device] ListModulesActive
Parameters	#device device specifier (see 5.1.2)
required state	*operational* (also in *OS Error* state available)
Output	**0 SMOS 1.0.3.7 OS_MDL_INIT_OK** **1 FIPS140 1.0.0.2 OS_MDL_INIT_OK** **65 CSI 1.0.0.3 OS_MDL_INIT_OK** **81 VDES 1.0.0.3 OS_MDL_INIT_OK** **83 CMDS 1.0.5.0 OS_MDL_INIT_OK** **84 VRSA 1.0.3.3 OS_MDL_INIT_OK** **86 UTIL 1.0.6.0 OS_MDL_INIT_OK** **87 ADM 1.0.2.0 OS_MDL_INIT_OK** **88 DB 1.0.1.1 OS_MDL_INIT_OK** **89 HASH 1.0.1.0 OS_MDL_INIT_OK** **8b AES 1.0.0.0 OS_MDL_INIT_OK** **8e LNA 1.0.3.0 OS_MDL_INIT_OK** **91 ASN1 1.0.1.1 OS_MDL_INIT_OK**
Note	■ Possible module initialization levels are 0 MDL_INIT_NONE 1 MDL_INIT_INTERNAL 2 MDL_INIT_DEP_OK 3 MDL_INIT_OK 4 MDL_INIT_FAILED (see below for the meaning of this levels) ■ If for a module no entry is found, the module is not loaded. ■ After loading or replacing a firmware module, the CryptoServer has to be restarted (see *Reset* command chapter 5.3.1) before the firmware module becomes active.

During the boot process of the CryptoServer, the operating system SMOS starts, after its own initialization, all other firmware modules. The module start is a complex process: Every module that is found and started by SMOS inside the flash file will run through the above given states of initialization, in case of success ending with the MDL_INIT_OK state.

The meaning of the initialization levels is the following:

Initialization level	Description
MDL_INIT_NONE	The firmware module is present but the initialization of the module has not been started yet. This entry will be made by the OS when loading the module into the SD-RAM.
MDL_INIT_INTERNAL	The module has finished its internal initialization (first step of initialization). "Internal" means all initialization tasks that can be done without using services from other firmware modules like memory allocation, FIFO creation, global data initialization, etc.
MDL_INIT_DEP_OK	The module has successfully completed the check of dependencies on other modules (second step of initialization). "Dependencies on other modules" means that the module is dependent on services provided by other modules in order to run correctly. Example: the CSI (Cryptographic Services Interface) module requires the VDES (DES functionality) module to do its work.
MDL_INIT_OK	This is the highest possible level: **The initialization of the module is completed** (third step of initialization). Services provided by this module are available; other modules can call internal services that are provided by this module.
MDL_INIT_FAILED	**Module initialization failed**. Services provided by this module are not available.

5.4.5 GetBootLog

GetBootLog retrieves a log file which contains log messages made by the operating system and other firmware modules during the CryptoServer's boot process. The boot log is held in memory and is not written into a file. In this way the content of the previous boot log file is cleared every time the operating system starts. The size of the boot log is limited, that is why it contains only log messages made during the boot phase.

Log messages can be viewed externally by connecting a PC's serial port with a crossed serial cable to the CryptoServer's serial port 1 (at the bracket of the PCI-card).
The terminal software has to be configured as follows:

- *baudrate: 115200*
- *databits: 8*
- *parity: none*
- *stopbits: 1*

Syntax	csadm [Dev=#device] GetBootLog
Parameter	#device device specifier (see 5.1.2)
Required State	*initialized* or *operational* (any mode or state)
Output	**SMOS Ver. 2.0.0.6 started** **module 0x83 (CMDS) initialized successfully** **module 0x89 (HASH) initialized successfully** **module 0x86 (UTIL) initialized successfully** **module 0x8e (LNA) initialized successfully** **module 0x81 (VDES) initialized successfully** **module 0x87 (ADM) initialized successfully** **module 0x84 (VRSA) initialized successfully** **module 0x91 (ASN1) initialized successfully** **module CSI can't find module AES (00000000)** **FATAL: module 0x65 (CSI) initialization failed (err = b065fe01)**

5.4.6 GetAlarmLog

The content of the 'alarm.log' file is displayed. Every new alarm is taken down on this log file by the boot loader. If no alarm has occurred since CryptoServer's production, the 'alarm.log' file does not exist.

This command is responded by the CryptoServer in any mode and state (FIPS mode or personalization mode; operational state or FIPS error state).

Syntax	csadm [Dev=#device] GetAlarmLog
Parameter	#device device specifier (see 5.1.2)
Required State	*initialized* or *operational* (any mode or state)
Output	**No. Date Time Sens** **----------** **0 22.03.2002 19:06:35 0x027f : ext_Erase** **1 22.03.2002 19:07:12 0x02f7 : Out_foil** If no file 'alarm.log' exists an error message is returned.
Note	The following (combination of) alarms are possible: ■ Pow_low Power is too low ■ Pow_high Power is too high ■ Temp_high Temperature too high ■ Temp_low Temperature too low ■ Out_foil Outer foil is broken ■ In_foil Inner foil is broken ■ ext_Erase External Erase was executed (manually) ■ inval_MK Invalid (corrupted) CryptoServer Master Key $\mathbf{K_{CS2}}$ (this occurs in case of an empty battery) See chapter 7.2 how to deal with an alarm.

5.4.7 GetTempLog

The content of the 'temp.log' file is displayed.

A new entry is taken down on this log file if the CryptoServer's temperature exceeds the range between 5°C and 58°C during its starting-up process (i. e. in case of power-on, after the occurrence of an alarm or after the execution of *Reset*). If the temperature has never been out of range since CryptoServer's last personalization, the file 'temp.log' does not exist.

If the CryptoServer's temperature is lower than 5°C or higher than 58°C the CryptoServer will no longer respond to any command. If it is restarted in this temperature state it will be put into power-down mode and then, after cooling down, must be restarted in order to return to the normal mode.
Exceeding this temperature range does not inevitably lead to an alarm (and CryptoServer's clearing as a result). A temperature alarm does only occur in case of exceeding the range between –13°C and 66°C. In this case an Administrator has to be called for help.

This command can be performed by the CryptoServer in any mode and state (FIPS mode or personalization mode; operational state or any FIPS error state).

See chapter 3.4 for more details about the CryptoServer's temperature treatment.

Syntax	csadm [Dev=#device] GetTempLog
Parameter	#device device specifier (see 5.1.2)
Required State	*initialized* or *operational* (any mode or state)
Output	**No. Date Time Temp Low High** **----------** **0 22.03.2002 19:06:35 58,5 5,0 58,0** **1 22.03.2002 19:07:12 59,0 5,0 58,0** Within one line, the **Temp** temperature gives the measured temperature, whereas **Low** and **High** give the limits of the normal temperature range. If the file 'temp.log' does not exist, an error message will be returned. This means that the temperature has not been out of range since CryptoServer's personalization.

5.4.8 GetTimeLog

The content of the file 'time.log' is displayed.

A new entry is taken down on this log file any time the CryptoServer's clock is set. If the clock was not set since CryptoServer's last personalization, the 'time.log' file does not exist.

This command can be performed by the CryptoServer in any mode and state (FIPS-mode or personalization mode; operational state or any FIPS error state).

Syntax	csadm [Dev=#device] GetTimeLog
Parameter	#device device specifier (see 5.1.2)
required state	*initialized* or *operational* (any mode or state)
Output	**No. date time new date new time** **----------** **0 22.10.2002 18:24:57 22.10.2002 18:24:31** **1 25.10.2002 09:45:13 25.10.2002 09:44:59** **2 05.11.2002 13:14:27 05.11.2002 13:14:01** If the 'time.log' file does not exist an error message is returned.

5.4.9 MemInfo

This function returns information about the current memory usage of the CryptoServer.

The desired directory ('SYS\', 'FLASH\' or 'NVRAM\') may be passed as command parameter. If none of the above directories is given, memory information about all directories will be returned.

Syntax	csadm [Dev=#device] MemInfo[=#dir]
Parameters	#device device specifier (see 5.1.2) #dir directory on the CryptoServer ('SYS', 'FLASH' or 'NVRAM')
Example	csadm MemInfo=SYS
Required State	*operational* (also in *OS Error* state available)
Output	**SYS** **max_size = 425984** **used_size = 225280** **free_size = 199680** **available_size = 200704**
Note	■ max_size: maximum size of the directory ■ used_size: currently used size ■ free_size: currently free size regarding only already formatted blocks. This value is returned only for diagnostic purposes. It does only show a part of the space available for the user. ■ available_size: size available for the user regarding already free blocks as well as unused space which could be freed if needed

5.4.10 Test

With this command a communication test with the CryptoServer is executed. It sends series of test patterns to the CryptoServer, which echoes the received command. On the host side the received answer is compared with the command originally sent.

Syntax	csadm [Dev=#device] Test=[#datalength,]#loopcount
Parameter	#device device specifier (see 5.1.2) #datalength length [bytes] of the used pattern. If this parameter is omitted, 2048 bytes will be used as default value. #loopcount number of execution cycles.
Examples	■ csadm Test=16,10000 ■ csadm Test=1000000
required state	*operational* (also in *OS Error* state available)
Output	**- Random Block Test** **data length: 16** **loop count : 10000** **10000** **execution completed in 832 ms** **data throughput: 192307 bytes/s** **transaction time: 0.083200 ms**
Note	■ If no data length is given, a 'walking zero' test is performed with a block length of 2048 bytes. ■ If a data length is given, a random data pattern will be generated by the host PC to perform this test. ■ The maximum data length is 256 kBytes ■ A little time is needed to create the test patterns for each loop. A pure benchmark test leads to slightly better results ;-)

5.5 Commands for User Management

For a CryptoServer in FIPS-mode, security relevant commands have to be authenticated either by an user who has assumed the ***Administrator*** role, or by an user who has assumed the ***Cryptographic User*** role (see chapter 3.3.2 for the details). In dependency on the command the user therefore has to be fitted out with certain permissions.

New users can be set up on the CryptoServer by adding them to the user database which is hosted on the CryptoServer. This can only be done by an *Administrator* with a respective *AddUser* command. During this process the user name, authentication mechanism, authentication data (RSA key or password) and FIPS-mode-specific flags will be stored. User name, authentication mechanism and flags are fixed from this point on and cannot be changed later. Only an *Administrator* can delete a registered user from the CryptoServer's user database.

But, any user (*Administrator* or *Cryptographic User*) can change his authentication token if wanted (RSA key or password). The respective command has to be authenticated by the user himself. These commands are described in this chapter. Furthermore any user can request a list with information about all registered users.

Additionally CSADM offers a function *ChangePIN* which changes the PIN of a given smartcard. This function can be used by any user (*Administrator* or *Cryptographic User*) who uses a RSA key on a smart card as authentication token. It which works solely on the PC without connection to the CryptoServer, only a PIN-Pad has to be connected.

5.5.1 ListUser

This command lists all existing users from the 'user.db' database. In FIPS-mode, these are all users that are allowed to authenticate a command either as *Cryptographic User* or as *Administrator*.

Syntax	csadm [Dev=#device] ListUser
Parameter	#device device specifier (see 5.1.2)
Required State	*operational,* not available in any FIPS error state
Output	**Name** **Permission** **Mechanism** **Flags** **ADMIN** **22000000** **RSA sign** **no_login + sma** **paula** **22000000** **sha-1 passwd** **no_login + sma** **paul** **00000020** **sha-1 passwd** **no_login + sma** **test** **22000020** **sha-1 passwd** **no_login + sma**
Note	■ The initial user 'ADMIN' (*System Administrator* as the owner of the *Initialization Key*) will always be displayed (even if the database 'user.db' is not yet present) and cannot be deleted. ■ For a description of the user's properties (*name, permission, mechanism, flags*) see below.

Property	***Description***
Name	user name, up to 8 characters (A..Z, a..z, 0..9)
Permission	In FIPS mode the following permissions are possible: 1) Users who shall be allowed to assume the ***Administrator* role** must have the **user permission '22000000'** (or higher). 2) Users who shall be allowed to assume the ***Cryptographic User* role** must have the **user permission '00000020'** (or higher).
Mechanism	■ RSA signature (either with a smart card or a key file, communication runs over the host) ■ SHA-1 hashed password See 3.3.2 for more details about the mechanisms and their advantages / disadvantages.
Flags	■ no_login: user has to authenticate each (sensitive) command separately (i. e. no static login is allowed) ■ sma: user may open a secure messaging session if he/she authenticates the command (see 3.3.3). In FIPS-mode, both flags have to be set.

See 3.3.2 for more details about the CryptoServer's authentication concept.

5.5.2 ChangeUserRSASign

With this command a user using the authentication mechanism 'RSA-Signature' changes his RSA Key.

The user needs a new RSA key pair (on smart card or as key file) as well as his/her old RSA Key.

A user is not allowed to change his authentication mechanism, permission or flags.

Syntax	csadm [Dev=#device] AuthRSASign=#user,#keyspec1 … ..... ChangeUserRSASign=#user,#keyspec2
Parameter	#device device specifier (see 5.1.2) #user existing user name (identical in both cases!) #keyspec1 private part of the user's old RSA key (key specifier for smart card or key file, see 5.1.3) #keyspec2 public part of the user's new RSA key (key specifier for smart card or key file, see 5.1.3)
Example	csadm AuthRSASign=paul,:cs2:cp8:COM1 … … ChangeUserRSASign=paul,:cs2:cp8:COM1
Required State	*operational,* not available in any FIPS error state
Output	none on success or error message
Note	• The user has to be already present in the user database. • The command has to be authenticated by the existing user himself, see below. This will be done via the command part 'AuthRSASign=…'. • If (one of) the user's RSA key is stored on a smart card, the user will be prompted at the PIN-Pad to insert his smart card. The PIN-Pad has to be connected to a serial line of the computer where the CSADM tool is running. • If both user's RSA keys (old and new) are stored on a smart card, watch the PIN-Pad's display and... 1. insert the smart card containing the user's new RSA key at first, 2. insert the smart card for command authentication (user's old RSA key) after that. You will be prompted at the PIN-Pad to enter the PIN of the smart card.

Command authentication with 'RSA-Signature' authentication mechanism:

The 'AuthRSASign=#user,#keyspec1' command part serves for authentication of the *ChangeUserRSASign* command.

In detail, the following steps are performed during authentication:

1. A challenge value is requested from the CryptoServer.
2. Command data and challenge value are signed (according to PKCS#1) using the private part of the user's old RSA key.
3. Command data and signature are sent to the CryptoServer.
4. The CryptoServer verifies the signature using the public part of the user's (old) RSA key from its user database.
5. If the verification has been successful, the CryptoServer will execute the *ChangeUserRSASign* command.

5.5.3 ChangeUserSHA1Pwd

With this command a user with the authentication mechanism 'SHA1 hashed Password' changes his password.

A user is not allowed to change his authentication mechanism, permission or flags.

Syntax	csadm [Dev=#device] AuthSHA1Pwd=#user,#password1 … ... SessionDH=1024 ChangeUserSHA1Pwd=#user,#password2
Parameter	#device device specifier (see 5.1.2) #user existing user name (identical in both cases!) #password1 for hidden password entry: string 'ask' (see below); otherwise: user's old password (up to 16 characters) #password2 for hidden password entry: string 'ask' (see below); otherwise: user's new password (length between 6 and 16 characters)
Examples	*Example for hidden password entry (mandatory in FIPS-mode):* csadm AuthSHA1Pwd=paul,ask ChangeUserSHA1Pwd=paul,ask *Example for direct password entry (not recommended except for test purposes):* csadm AuthSHA1Pwd=paul,swordfish ChangeUserSHA1Pwd=paul,sesame
required state	*operational,* not available in any FIPS error state
Output	none on success or error message
Note	• The user has to be already present in the user database. • The command has to be authenticated by the existing user himself, see below. This will be done via the command part 'AuthSHA1Pwd=…'. • The usage of hidden password entry is mandatory in FIPS-mode. See below. • The command has to be performed with Secure Messaging: The command part 'SessionDH=1024' opens a session for Secure Messaging. See below.

In FIPS-mode it is mandatory to use passwords of at least 6 characters length.

Command authentication with 'SHA1 hashed Password' authentication mechanism:

The 'AuthSHA1Pwd=#user,#password1' command part serves for authentication of the *ChangeUserSHA1Pwd* command.

In detail, the following steps are performed during authentication:

1. A challenge value is requested from the CryptoServer.
2. A SHA-1 hash value is calculated over the user's (old) password, the command data and the challenge value.
3. Command data and hash value are sent to the CryptoServer.
4. The CryptoServer re-calculates the SHA-1 hash via the user's (old) password (taken from the user database), the command data and the challenge value and compares it with the given hash.
5. If the comparison has been successful, the CryptoServer will execute the *ChangeUserSHA1Pwd* command

Note: The (old) password will never be transmitted in clear to the CryptoServer but only in a SHA1-hashed form.

Hidden password entry:

If the (old or new) password is entered directly within the CSADM command line, it is possible to read the password on the monitor which is connected to the PC on which CSADM is running.

To avoid the password being reflected as plaintext on the monitor, the CryptoServer offers the possibility for hidden password entry:
If hidden password entry is used, the CSADM will ask for the password separately and hide the entrance on the monitor by the display of default characters.

For hidden password entry, instead of the passwords first the string 'ask' has to be entered respectively. Then CSADM will, before starting to process the command, return and prompt for the passwords separately. If at this point the passwords will be entered over the keyboard, they will not be reflected in clear on the monitor, but be hidden by the display of default characters.

Example:

csadm AuthSHA1Pwd=paul,ask ChangeUserSHA1Pwd=paul,ask

Enter Passphrase: *xxxxxxxxx* *(enter old password at this point)*

Enter New Passphrase: *xxxxxxxxx* *(enter new password at this point)*

Repeat New Passphrase: *xxxxxxxxx* *(repeat new password)*

For a CryptoServer in FIPS-mode, the usage of hidden password entry is mandatory.

Secure Messaging:

With Secure Messaging, commands to and from the CryptoServer will be encrypted and integrity protected using a Triple DES session key which was exchanged when opening the session. See also 3.3.3 for a description of CryptoServer's Secure Messaging concept.

The *ChangeUserSHA1Pwd* command is to be performed with Secure Messaging:

The command part 'SessionDH=1024' opens a session for Secure Messaging, using the Diffie-Hellman key agreement according to [PKCS#3], with size of the Diffie-Hellman parameters (prime) 1024 bits. The session will be closed automatically when the command execution finishes and CSADM ends.

If the ChangeUserSHA1Pwd *command was not performed with Secure Messaging, the new password would be transmitted in clear to the CryptoServer. Therefore for this command the usage of Secure Messaging is mandatory.*

5.5.4 ChangePin

This command changes the PIN of a given smart card. A PIN-Pad including smart card reader must be used for this command.

This command is executed locally without a connection to a CryptoServer. Therefore the state or mode of any underlying CryptoServer is irrelevant for the command execution, and no authentication at any CryptoServer is necessary.

Syntax	csadm ChangePIN=#keyspec
Parameter	#keyspec specifier for smart card type, reader type and serial port (see 5.1.3)
Example	csadm ChangePIN=:cs2:cp8:COM1
Output	none on success or error message
Note	The PIN-Pad has to be connected to a serial line of that computer where the CSADM tool is running.

Watch the PIN-Pad's display:

- *enter old PIN first,*
- *enter new PIN then and*
- *confirm new PIN.*

6 Cryptographic Commands Offered by CSI Library

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 CryptoServer.

As explained in chapter 2.2, Utimaco offers a C-library “CSI-Lib” as interface to the cryptographic services. This C-library will run on the host where the CryptoServer is installed.

All cryptographic services can only be performed if authenticated by a *Cryptographic User*, and it is strongly recommended to perform these services only within a *Secure Messaging* session. Therefore, in order to use the CSI-Lib correctly, the introductory chapters of the document [CSCSI] 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.

The respective commands and syntax for this will be detailed on the C-interface level as well as on the CryptoServer byte stream interface level in [CSCSI] sections 2 and 3.

6.1 Key Management Functions

In this chapter the external functions for key management which are offered by the CSI library 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 [CSCSI] document have to be consulted.

6.1.1 Generate DES Key

The function `cs_generateDESkey( )` generates a Triple-DES key with an effective size of 112 or 168 bits (adjusted to odd parity to a length of 16 or 24 bytes). The generated key is stored into the internal key table of the CryptoServer.

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].

See [CSCSI] section 5.1 for the syntax and a detailed description of this function.

6.1.2 Generate AES Key

The function `cs_generateAESkey( )` generates an AES key with a size of 128, 192 or 256 bits. The generated key is stored into the internal key table of the CryptoServer.

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].

See [CSCSI] section 5.2 for the syntax and a detailed description of this function.

6.1.3 Generate RSA Key

The function `cs_generateRSAkey( )` generates a RSA key pair with a given size and public exponent. The generated key is stored into the internal key table of the CryptoServer.

If the CryptoServer is in FIPS mode, no key sizes less than 1024 bits are allowed. For the key generation probable prime generation using ANSI X9.31 algorithm will be used. For the underlying randon number generation the CryptoServer's deterministic random number generator is used.

See [CSCSI] section 5.3 for the syntax and a detailed description of this function.

6.1.4 Generate ECDSA Key (SFC = 0x17)

The function `cs_generateECDSAkey( )` generates an ECDSA key pair for a given elliptic curve. The key pair is intended solely for signature generation and verification (and

cannot be used for data encryption/decryption). The generated key is stored in the internal key table of the CryptoServer.

In FIPS-mode only ECDSA keys on those elliptic curves that are specified and recommended in FIPS 186-2 (curves P-192, P-224, P-256, P-384, P-521 from [FIPS 186-2], Appendix 6.1) can be generated.

See [CSCSI] section 5.4 for the syntax and a detailed description of this function.

6.1.5 Wrap Key (Export Key)

The function `cs_wrap_key( )` exports a key from the internal key table of the CryptoServer encrypted with a key encryption key (KEK).

The key to be exported must have the attribute “Exportable”. The key encryption key must have the attribute “Key Encryption Key”. It is not possible to export a RSA key encrypted with a RSA key, or to use an ECDSA key as key encryption key.

See [CSCSI] section 5.5 for the syntax and a detailed description of this function.

6.1.6 Unwrap Key (Import Key)

The function `cs_unwrap_key( )` imports an encrypted key, unwraps it and stores the key in the CryptoServer’s internal key table.

The key encryption key (KEK) must have the attribute “Key Encryption Key”. It is not possible to import a RSA key encrypted with a RSA key, or to use an ECDSA key as 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 (curves P-192, P-224, P-256, P-384, P-521 from [FIPS 186-2], Appendix 6.1).

See [CSCSI] section 5.6 for the syntax and a detailed description of this function.

6.1.7 Export Public RSA Key

The function `cs_export_pubkey( )` exports a public RSA key from the internal key table of the CryptoServer.

See [CSCSI] section 5.7 for the syntax and a detailed description of this function.

6.1.8 Export Public ECDSA Key (SFC = 0x16)

The function `cs_key_ecdsa_pub_export( )` exports a public ECDSA key from the internal key table of the CryptoServer.

See [CSCSI] section 5.8 for the syntax and a detailed description of this function.

6.1.9 List Keys

The function `cs_list_keys( )` outputs a list of the info data of all keys stored in the CryptoServer's internal key table. Key info data are the ID, type, size and attributes of the respective key.

See [CSCSI] section 5.9 for the syntax and a detailed description of this function.

6.1.10 Delete Key

The function `cs_delete_key( )` removes a key from the internal key table of the CryptoServer.

See [CSCSI] section 5.10 for the syntax and a detailed description of this function.

6.1.11 Import Clear Key

This function imports a clear key into the internal key table of the CryptoServer. In FIPS mode only the import of a public RSA or public ECDSA key is available.

See [CSCSI] section 5.11 for the syntax and a detailed description of this function.

6.2 Cryptographic Functions

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

6.2.1 DES Encryption in ECB Mode

The function `cs_DEScryptECB( )` encrypts or decrypts data with a Triple-DES key in ECB mode (key size 16 or 24 bytes). The key to be used must not have the attribute “Key Encryption Key”.

See [CSCSI] section 7.1 for the syntax and a detailed description of this function.

6.2.2 DES Encryption in CBC Mode

The function `cs_DEScryptCBC( )` encrypts or decrypts data with a Triple-DES key in CBC mode (key size 16 or 24 bytes). The key to be used must not have the attribute “Key Encryption Key”.

See [CSCSI] section 7.2 for the syntax and a detailed description of this function.

6.2.3 AES Encryption in ECB Mode

The function `cs_AEScryptECB( )` encrypts or decrypts data with an AES key in ECB mode. The key to be used must not have the attribute “Key Encryption Key”. Possible AES key sizes are 128, 192 or 256 bits.

See [CSCSI] section 7.3 for the syntax and a detailed description of this function.

6.2.4 AES Encryption in CBC Mode

The function `cs_AEScryptCBC( )` encrypts or decrypts data with a AES key in CBC mode. The key to be used must not have the attribute “Key Encryption Key”. Possible AES key sizes are 128, 192 or 256 bits.

See [CSCSI] section 7.4 for the syntax and a detailed description of this function.

6.2.5 DES MAC Calculation

The function `cs_DEScalc_mac( )` calculates a MAC using a Triple-DES key (CBC-MAC, with TDES keys of size 16 or 24 bytes). The key to be used must not have the attribute "Key Encryption Key".

See [CSCSI] section 7.5 for the syntax and a detailed description of this function.

6.2.6 Sign Data with RSA

The function `cs_RSAsign( )` calculates a RSA signature for a given hash value, according to either PKCS#1 or ANSI X9.31 ([PKCS#1], [ANSIX9.31]).

The key to be used must not have the attribute "Key Encryption Key", and its key size must be at least 1024 bits. As hashing algorithm SHA-1, SHA-224, SHA-256, SHA-384 or SHA-512 can be used.

See [CSCSI] section 8.1 for the syntax and a detailed description of this function.

6.2.7 Verify RSA Signature

The function `cs_RSAverify( )` verifies the RSA signature (according to either PKCS#1 or ANSI X9.31, see [PKCS#1], [ANSIX9.31]) over a given hash value.

The key to be used for verification must not have the attribute "Key Encryption Key", and its key size must be at least 1024 bits. As underlying hashing algorithms SHA-1, SHA-224, SHA-256, SHA-384 or SHA-512 are allowed.

See [CSCSI] section 8.2 for the syntax and a detailed description of this function.

6.2.8 Sign Data with ECDSA

The function `cs_ECDSAsign( )` calculates a ECDSA signature for a given hash value or given data (in the latter case the function first calculates the hash value over the given data).

Only ECDSA keys on those elliptic curves that are specified and recommended in FIPS 186-2 are allowed (curves P-192, P-224, P-256, P-384, P-521 from [FIPS 186-2], Appendix 6.1). Additionally the key to be used must not have the attribute "Key Encryption Key". As hashing algorithm SHA-1, SHA-224, SHA-256, SHA-384 or SHA-512 can be used.

See [CSCSI] section 8.3 for the syntax and a detailed description of this function.

6.2.9 Verify ECDSA Signature

The function `cs_ECDSAverify( )` verifies an ECDSA signature over a given hash value or over given data.

Only ECDSA keys on those elliptic curves that are specified and recommended in FIPS 186-2 are allowed (curves P-192, P-224, P-256, P-384, P-521 from [FIPS 186-2], Appendix 6.1). Additionally the key to be used for verification must not have the attribute “Key Encryption Key”. As hashing algorithm SHA-1, SHA-224, SHA-256, SHA-384 or SHA-512 are allowed.

See [CSCSI] section 8.4 for the syntax and a detailed description of this function.

6.2.10 Compute Hash

The function `cs_hash( )` computes a hash value over given data with a chosen hashing algorithm (SHA-1, SHA-224, SHA-256, SHA-384 or SHA-512).

See [CSCSI] section 9.1 for the syntax and a detailed description of this function.

6.2.11 Generate Random Number

The function `cs_gen_rnd( )` generates a random number of wanted size. For this the CryptoServer’s deterministic random number generator (which is implemented according to FIPS 186-2 Appendix 3.1 [FIPS 186-2], based on SHA-1 as transition function, and which complies with ANSI X9.31, Appendix A.2.1 and A.2.2 [ANSIX9.31]) will be used.

See [CSCSI] section 9.2 for the syntax and a detailed description of this function.

7 Troubleshooting

7.1 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 PCI Driver is running and that the Administration Tool CSADM is installed. See [CSInstall-Manual] for help.

What to do?

1. Perform the *GetState* command (see 5.4.1).

Result	**Explanation / Reason / Adjustment**
state = OPERATIONAL, FIPS mode = ON, CryptoServer is not in FIPS error state	Everything is ok. End.
state = OPERATIONAL, but CryptoServer is not in FIPS-mode	You may analyze the problem using the *ListModulesActive* and *GetBootLog* commands: Check if all necessary firmware modules are present and successfully initialized (see chapter 8 for a complete list). After that the CryptoServer should be set-up and personalized again ⇒ please ask an *Administrator* for help.
state = OPERATIONAL, FIPS mode = ON, CryptoServer is in FIPS error state	CryptoServer is in (FIPS) OS Error state. Try to quit error state, see 4.3.
state = INITIALIZED, ALARM = off, but CryptoServer is not in FIPS-mode	CryptoServer is in personalization mode or not correctly initialized in FIPS-mode. ⇒ please ask an *Administrator* for help.

state = INITIALIZED, ALARM = off, FIPS mode = ON, CryptoServer is in FIPS error state	CryptoServer is in (FIPS) Boot Loader Error state. Try to quit error state, see 4.3.
state = INITIALIZED, ALARM = ON	An alarm has occurred (and is possibly physically still present) ⇒ see chapter 7.2 for alarm treatment
state neither INITIALIZED nor OPERATIONAL	CryptoServer is not correctly initialized or even defect ⇒ please get in contact with manufacturer/Utimaco.
Error B9011xxx, B9015xxx, B9016xxx, B9017xxx or B9021xxx until B9024xxx	CryptoServer's PCI carrier card does not react ⇒ try a reset (2)
other errors: B901xxxx or B902xxxx	No connection to CryptoServer: communication problem, wrong host or device name, problem with network. ⇒ check parameters
No answer from CryptoServer	⇒ try a reset (2).

2. Perform the *Reset* command (see 5.3.1). 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

7.2 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)
- Outer foil is broken
- Inner foil is broken
- Invalid (corrupted) Master Key $\mathbf{K_{CS2}}$ (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 PCI-card)

In any case, for security reasons, all data that have been stored on the CryptoServer, in particular all cryptographic keys and all firmware modules, are deleted after an alarm. No cryptographic services are available.

Some alarm reasons can be removed (e. g. exchange low battery or cool down high tempera¬ture), these are called temporary alarms. Other alarm reasons cannot be removed, these are called permanent alarms. Permanent alarms occur e. g. in case of a damage of the inner or outer tamper protecting foil, whereas usually all other possible alarms are temporary.

After any alarm the CryptoServer remains in personalization mode (i. e. it has left FIPS mode) and in global state *initialized*. Even if the alarm cause can be removed, only CryptoServer's *System Administrator*, i. e. a user who is allowed to use CryptoServer's *Initialization Key*, 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.*

8 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):

- SMOS: file 'smos.mtc', version 2.0.0.6
- CMDS: file 'cmds.mtc', version 1.0.9.0
- UTIL: file 'util.mtc', version 2.0.0.3
- ADM: file 'adm.mtc', version 2.0.0.2
- DB: file 'db.mtc', version 1.0.1.2
- VDES: file 'vdes.mtc', version 1.0.1.1
- VRSA: file 'vrsa.mtc', version 1.0.5.4
- AES: file 'aes.mtc', version 1.0.3.0
- HASH: file 'hash.mtc', version 1.0.4.1
- LNA: file 'lna.mtc', version 1.0.4.0
- ASN1: file 'asn1.mtc', version 1.0.3.2
- ECA: file 'eca.mtc', version 1.0.0.0
- ECDSA: file 'ecdsa.mtc', version 1.0.0.0
- CSI: file 'csi.mtc', version 1.2.1.1
- FIPS140: file 'fips140.mtc', version 2.0.0.1

The listed firmware modules are approved in the context of the FIPS 140-2 validation process of the CryptoServer CS. 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!

The FIPS140-2 approved version of the boot loader firmware is version 2.0.2.4. The boot loader firmware module is loaded by Utimaco during the CryptoServer's production process and cannot be changed or deleted by the customer.

9 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	
[CSCSI]	CryptoServer Cryptographic Service Interface – Firmware Module CSI – Interface Specification / Utimaco Safeware AG	2004-0003
[CSFIPS-AdmGuide]	CryptoServer CS - Administrator's Guide for CryptoServer in FIPS Mode / Utimaco SafewareAG	2004-0002
[CSInstall-Manual]	CryptoServer – Installation Manual / Utimaco SafewareAG	2003-0007
[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	